AI & Machine Learning Mastery

Complete 300-Page Syllabus, Code Labs, Exams & Projects

Zero to Advanced - Python, ML, Deep Learning, NLP, GenAI, RAG, Agents, MLOps

Course Promise

This PDF is arranged topic-wise. Every module contains clear theory, practical workflow, runnable code, output preview screen, mistakes, mini project, exam questions, revision checklist, and portfolio direction.

Who Should Study This

Beginners, frontend/backend developers moving into AI, students, working professionals, and founders who want to understand full AI product development from data to deployment.

What You Will Build

10 portfolio projects, one final capstone, model APIs, AI tutor logic, RAG workflow, dashboards, ML reports, and interview-ready GitHub repositories.

Main Languages & Tools

Python, SQL, JavaScript/TypeScript, R, C++/Rust concepts, Jupyter, NumPy, Pandas, scikit-learn, deep learning frameworks, FastAPI, Docker, GitHub.

Copyright & Use

Copyright © 2026 AiWorldAcademy. Website: www.aiworldacademy.in. This material is prepared for AiWorldAcademy learning use. Do not copy, sell, or republish without permission.

Complete Course Table of Contents

30 modules x 10 structured pages = 300 pages. Use this as a full AI/ML learning workbook.

01. AI & Machine Learning Orientation (pp. 1-10)
Understand AI, ML, deep learning, career paths, and the complete learning map.

02. Languages & Tools Used in AI/ML (pp. 11-20)
Know which programming languages, libraries, platforms, and file formats are used in real AI work.

03. Python Programming for AI (pp. 21-30)
Master the Python basics needed for data science, ML pipelines, and AI apps.

04. Python Data Structures & OOP (pp. 31-40)
Use lists, dictionaries, classes, and reusable objects to structure AI projects.

05. Environment, Git, GitHub & Notebooks (pp. 41-50
Set up a professional AI development environment and version-controlled workflow.

06. Mathematics: Linear Algebra for ML (pp. 51-60)
Learn vectors, matrices, dot products, norms, and the math used inside models.

07. Probability & Statistics for ML (pp. 61-70)
Build the statistical foundation needed for data analysis, evaluation, and model decisions.

08. NumPy & Pandas for Data Handling (pp. 71-80)
Load, inspect, filter, group, transform, and save datasets for ML work.

09. Data Visualization & Storytelling (pp. 81-90)
Turn raw numbers into charts and explanations that business teams can understand.

10. Data Cleaning & Feature Engineering (pp. 91-100
Prepare messy real-world data so machine learning models can learn correctly.

11. Machine Learning Workflow with scikit-learn (pp
Learn the standard pipeline: problem framing, split, train, evaluate, improve, save.

12. Regression Models (pp. 111-120)
Predict continuous values such as price, score, demand, revenue, or completion time.

13. Classification Models (pp. 121-130)
Predict categories such as spam/not spam, pass/fail, churn/no churn, fraud/safe.

16. Recommendation Systems (pp. 151-160)
Build systems that recommend courses, videos, notes, products, or content.

17. Time Series Forecasting (pp. 161-170)
Predict values over time such as traffic, sales, student activity, or server load.

18. Neural Network Fundamentals (pp. 171-180)
Understand neurons, layers, activations, loss, and backpropagation intuition.

19. Deep Learning with PyTorch/TensorFlow (pp. 181-
Learn how modern deep-learning training loops and model layers are structured.

20. Computer Vision (pp. 191-200)
Process images and build image classification, detection, and visual AI workflows.

21. Natural Language Processing (pp. 201-210)
Work with text data for sentiment, classification, search, chatbots, and document intelligence.

22. Transformers & Embeddings (pp. 211-220)
Understand attention, embeddings, semantic search, and transformer-based AI systems.

23. Generative AI & Large Language Models (pp. 221-
Learn how LLMs generate text, use prompts, follow context, and support AI products.

24. Prompt Engineering (pp. 231-240)
Write prompts that produce structured, accurate, testable, and useful AI responses.

25. RAG & Vector Databases (pp. 241-250)
Ground AI answers in documents using retrieval augmented generation.

26. Fine-tuning, PEFT & Custom Models (pp. 251-260)
Understand when and how to adapt models using datasets, fine-tuning, LoRA, and evaluation.

27. AI Agents & Tool Calling (pp. 261-270)
Design AI systems that choose tools, call functions, remember context, and complete tasks safely.

28. MLOps, APIs & Deployment (pp. 271-280)
Deploy models as reliable services with monitoring, versioning, security, and automation.

Study Method

For every module: read theory, run the code, compare your output with the output preview screen, complete the mini project, attempt the exam, then update your GitHub repo.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01: AI & Machine Learning Orientation

Understand AI, ML, deep learning, career paths, and the complete learning map.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- explain AI terms clearly
- identify ML problem types
- plan a complete project
- set up weekly practice

Topic Map

- AI vs ML vs deep learning
- supervised learning
- unsupervised learning
- generative AI
- data lifecycle
- model lifecycle
- AI career roles
- course workflow

Tools & Languages

- Python
- Jupyter
- VS Code
- GitHub
- Google Colab/local notebook

Deliverable for This Module

Create a one-page AI career roadmap showing target role, required skills, tools, and 90-day practice plan.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Understand AI, ML, deep learning, career paths, and the complete learning map.
Important topics: AI vs ML vs deep learning, supervised learning, unsupervised learning, generative AI, data lifecycle, model lifecycle, AI career roles.
Practical skill goal: explain AI terms clearly; identify ML problem types; plan a complete project; set up weekly practice.
Tools/languages used here: Python, Jupyter, VS Code, GitHub, Google Colab/local notebook.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- AI vs ML vs deep learning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- supervised learning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- unsupervised learning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- generative AI: know the meaning, why it matters, when to use it, and one real example from an AI product.
- data lifecycle: know the meaning, why it matters, when to use it, and one real example from an AI product.
- model lifecycle: know the meaning, why it matters, when to use it, and one real example from an AI product.
- AI career roles: know the meaning, why it matters, when to use it, and one real example from an AI product.
- course workflow: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

AI & Machine Learning Orientation is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on AI vs ML vs deep learning, supervised learning, unsupervised learning, generative AI, data lifecycle. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_01_ai_and_machine_learning_
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 01

```
# file: 01_ai_ml_warmup.py
course = "AI & Machine Learning Mastery"
modules = 30
projects = 10
print(f"Welcome to {course}")
print(f"Roadmap: {modules} modules + {projects} portfolio projects")
print("Goal: learn theory, run code, evaluate models, deploy AI apps")
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 01_ai_ml_warmup.py
Welcome to AI & Machine Learning Mastery
Roadmap: 30 modules + 10 portfolio projects
Goal: learn theory, run code, evaluate models, deploy AI apps
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a one-page AI career roadmap showing target role, required skills, tools, and 90-day practice plan.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 01 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain AI vs ML vs deep learning in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python, Jupyter, VS Code?
3. What mistake can happen while using supervised learning and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about AI & Machine Learning Orientation in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02: Languages & Tools Used in AI/ML

Know which programming languages, libraries, platforms, and file formats are used in real AI work.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- choose the right language
- read common file formats
- understand model-serving stack
- explain frontend + backend + ML separation

Topic Map

- Python for ML
- R for statistics
- SQL for data
- JavaScript/TypeScript for AI web apps
- C/C++/Rust for speed
- Java/Scala for big data
- Bash for automation
- YAML/Dockerfile for deployment

Tools & Languages

- Python
- R
- SQL
- JavaScript
- TypeScript
- C++
- Rust
- Docker
- YAML

Deliverable for This Module

Prepare a table of AI/ML languages with use cases: data, modeling, UI, deployment, optimization, and automation.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Know which programming languages, libraries, platforms, and file formats are used in real AI work.
Important topics: Python for ML, R for statistics, SQL for data, JavaScript/TypeScript for AI web apps, C/C++/Rust for speed, Java/Scala for big data, Bash for automation.
Practical skill goal: choose the right language; read common file formats; understand model-serving stack; explain frontend + backend + ML separation.
Tools/languages used here: Python, R, SQL, JavaScript, TypeScript, C++, Rust, Docker, YAML.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- Python for ML: know the meaning, why it matters, when to use it, and one real example from an AI product.
- R for statistics: know the meaning, why it matters, when to use it, and one real example from an AI product.
- SQL for data: know the meaning, why it matters, when to use it, and one real example from an AI product.
- JavaScript/TypeScript for AI web apps: know the meaning, why it matters, when to use it, and one real example from an AI product.
- C/C++/Rust for speed: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Java/Scala for big data: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Bash for automation: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Languages & Tools Used in AI/ML is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on Python for ML, R for statistics, SQL for data, JavaScript/TypeScript for AI web apps, C/C++/Rust for speed. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_02_languages_and_tools_used
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 02

```
# file: 02_ai_language_map.py
language_map = {
    "Python": "core ML, deep learning, notebooks",
    "SQL": "query data from databases",
    "JavaScript/TypeScript": "AI dashboards and web UI",
    "C++/Rust": "fast inference and performance modules",
    "YAML/Dockerfile": "MLOps deployment configuration"
}
for lang, use in language_map.items():
    print(f"{lang:22} -> {use}")
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 02_ai_language_map.py
Python                 -> core ML, deep learning, notebooks
SQL                    -> query data from databases
JavaScript/TypeScript  -> AI dashboards and web UI
C++/Rust               -> fast inference and performance modules
YAML/Dockerfile        -> MLOps deployment configuration
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Languages & Tools Used in AI/ML, debug these first: Python for ML, R for statistics, SQL for data, JavaScript/TypeScript for AI web apps, C/C++/Rust for speed. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Prepare a table of AI/ML languages with use cases: data, modeling, UI, deployment, optimization, and automation.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain Python for ML in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python, R, SQL?
3. What mistake can happen while using R for statistics and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Languages & Tools Used in AI/ML in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 02 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain Python for ML with one example.
☐ I can explain R for statistics with one example.
☐ I can explain SQL for data with one example.
☐ I can explain JavaScript/TypeScript for AI web apps with one example.
☐ I can explain C/C++/Rust for speed with one example.
☐ I can explain Java/Scala for big data with one example.
☐ I can explain Bash for automation with one example.
☐ I can explain YAML/Dockerfile for deployment with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Python Programming for AI. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03: Python Programming for AI

Master the Python basics needed for data science, ML pipelines, and AI apps.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- write clean functions
- debug syntax/runtime errors
- read CSV/text data
- create reusable scripts

Topic Map

- variables
- data types
- operators
- conditions
- loops
- functions
- modules
- exceptions
- file reading

Tools & Languages

- Python 3.11+
- venv
- pip
- VS Code
- Jupyter

Deliverable for This Module

Build a Python student-score analyzer that reads marks, normalizes them, and prints pass/risk labels.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Master the Python basics needed for data science, ML pipelines, and AI apps.
Important topics: variables, data types, operators, conditions, loops, functions, modules.
Practical skill goal: write clean functions; debug syntax/runtime errors; read CSV/text data; create reusable scripts.
Tools/languages used here: Python 3.11+, venv, pip, VS Code, Jupyter.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- variables: know the meaning, why it matters, when to use it, and one real example from an AI product.
- data types: know the meaning, why it matters, when to use it, and one real example from an AI product.
- operators: know the meaning, why it matters, when to use it, and one real example from an AI product.
- conditions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- loops: know the meaning, why it matters, when to use it, and one real example from an AI product.
- functions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- modules: know the meaning, why it matters, when to use it, and one real example from an AI product.
- exceptions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- file reading: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Python Programming for AI is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on variables, data types, operators, conditions, loops. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_03_python_programming_for_a
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 03

```
# file: 03_python_ai_basics.py
def normalize_score(score, max_score=100):
    if score < 0 or score > max_score:
        raise ValueError("score out of range")
    return score / max_score

scores = [75, 88, 92]
normalized = [normalize_score(s) for s in scores]
print("raw:", scores)
print("normalized:", normalized)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 03_python_ai_basics.py
raw: [75, 88, 92]
normalized: [0.75, 0.88, 0.92]
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Python Programming for AI, debug these first: variables, data types, operators, conditions, loops. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build a Python student-score analyzer that reads marks, normalizes them, and prints pass/risk labels.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain variables in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python 3.11+, venv, pip?
3. What mistake can happen while using data types and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Python Programming for AI in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 03 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain variables with one example.
☐ I can explain data types with one example.
☐ I can explain operators with one example.
☐ I can explain conditions with one example.
☐ I can explain loops with one example.
☐ I can explain functions with one example.
☐ I can explain modules with one example.
☐ I can explain exceptions with one example.
☐ I can explain file reading with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Python Data Structures & OOP. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04: Python Data Structures & OOP

Use lists, dictionaries, classes, and reusable objects to structure AI projects.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- model data cleanly
- write classes for datasets/models
- avoid repeated code
- organize folders

Topic Map

- lists
- tuples
- sets
- dictionaries
- comprehensions
- classes
- objects
- dataclasses
- project structure

Tools & Languages

- Python
- dataclasses
- typing
- pytest basics

Deliverable for This Module

Create reusable Python classes for DatasetInfo, ModelConfig, and TrainingResult.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Use lists, dictionaries, classes, and reusable objects to structure AI projects.
Important topics: lists, tuples, sets, dictionaries, comprehensions, classes, objects.
Practical skill goal: model data cleanly; write classes for datasets/models; avoid repeated code; organize folders.
Tools/languages used here: Python, dataclasses, typing, pytest basics.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- lists: know the meaning, why it matters, when to use it, and one real example from an AI product.
- tuples: know the meaning, why it matters, when to use it, and one real example from an AI product.
- sets: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dictionaries: know the meaning, why it matters, when to use it, and one real example from an AI product.
- comprehensions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- classes: know the meaning, why it matters, when to use it, and one real example from an AI product.
- objects: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dataclasses: know the meaning, why it matters, when to use it, and one real example from an AI product.
- project structure: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Python Data Structures & OOP is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on lists, tuples, sets, dictionaries, comprehensions. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_04_python_data_structures_a
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 04

```
# file: 04_model_config.py
from dataclasses import dataclass

@dataclass
class ModelConfig:
    name: str
    learning_rate: float
    epochs: int

config = ModelConfig("baseline_classifier", 0.01, 20)
print(config)
print("model:", config.name, "epochs:", config.epochs)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 04_model_config.py
ModelConfig(name='baseline_classifier', learning_rate=0.01, epochs=20)
model: baseline_classifier epochs: 20
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Python Data Structures & OOP, debug these first: lists, tuples, sets, dictionaries, comprehensions. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create reusable Python classes for DatasetInfo, ModelConfig, and TrainingResult.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain lists in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python, dataclasses, typing?
3. What mistake can happen while using tuples and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Python Data Structures & OOP in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 04 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain lists with one example.
- ☐ I can explain tuples with one example.
- ☐ I can explain sets with one example.
- ☐ I can explain dictionaries with one example.
- ☐ I can explain comprehensions with one example.
- ☐ I can explain classes with one example.
- ☐ I can explain objects with one example.
- ☐ I can explain dataclasses with one example.
- ☐ I can explain project structure with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Environment, Git, GitHub & Notebooks. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05: Environment, Git, GitHub & Notebooks

Set up a professional AI development environment and version-controlled workflow.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- create venv
- install packages
- push project to GitHub
- document experiments

Topic Map

- virtual environments
- pip requirements
- Jupyter notebooks
- Git commits
- GitHub repo
- README files
- notebook hygiene
- project reproducibility

Tools & Languages

- venv
- pip
- Git
- GitHub
- Jupyter
- VS Code

Deliverable for This Module

Create a clean GitHub repository with README, requirements.txt, notebooks/, src/, data/, reports/ folders.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Set up a professional AI development environment and version-controlled workflow.
Important topics: virtual environments, pip requirements, Jupyter notebooks, Git commits, GitHub repo, README files, notebook hygiene.
Practical skill goal: create venv; install packages; push project to GitHub; document experiments.
Tools/languages used here: venv, pip, Git, GitHub, Jupyter, VS Code.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- virtual environments: know the meaning, why it matters, when to use it, and one real example from an AI product.
- pip requirements: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Jupyter notebooks: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Git commits: know the meaning, why it matters, when to use it, and one real example from an AI product.
- GitHub repo: know the meaning, why it matters, when to use it, and one real example from an AI product.
- README files: know the meaning, why it matters, when to use it, and one real example from an AI product.
- notebook hygiene: know the meaning, why it matters, when to use it, and one real example from an AI product.
- project reproducibility: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Environment, Git, GitHub & Notebooks is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on virtual environments, pip requirements, Jupyter notebooks, Git commits, GitHub repo.
For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_05_environment,_git,_github
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 05

```
# terminal commands
python -m venv .venv
.venv\Scripts\activate   # Windows PowerShell example
pip install numpy pandas scikit-learn matplotlib
pip freeze > requirements.txt
git init
git add .
git commit -m "start ai ml course project"
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
Created virtual environment .venv
Installed packages: numpy, pandas, scikit-learn, matplotlib
Created requirements.txt
Git repository initialized
Commit created: start ai ml course project
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Environment, Git, GitHub & Notebooks, debug these first: virtual environments, pip requirements, Jupyter notebooks, Git commits, GitHub repo. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a clean GitHub repository with README, requirements.txt, notebooks/, src/, data/, reports/ folders.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain virtual environments in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: venv, pip, Git?
3. What mistake can happen while using pip requirements and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Environment, Git, GitHub & Notebooks in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 05 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain virtual environments with one example.
☐ I can explain pip requirements with one example.
☐ I can explain Jupyter notebooks with one example.
☐ I can explain Git commits with one example.
☐ I can explain GitHub repo with one example.
☐ I can explain README files with one example.
☐ I can explain notebook hygiene with one example.
☐ I can explain project reproducibility with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Mathematics: Linear Algebra for ML. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06: Mathematics: Linear Algebra for ML

Learn vectors, matrices, dot products, norms, and the math used inside models.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- represent data as matrices
- compute dot products
- understand weights and features
- read model equations

Topic Map

- scalars
- vectors
- matrices
- dot product
- matrix multiplication
- transpose
- identity matrix
- norms
- linear transformations

Tools & Languages

- NumPy
- Jupyter
- graph paper thinking

Deliverable for This Module

Create a feature-weight calculator for predicting a simple student performance score.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Learn vectors, matrices, dot products, norms, and the math used inside models.
Important topics: scalars, vectors, matrices, dot product, matrix multiplication, transpose, identity matrix.
Practical skill goal: represent data as matrices; compute dot products; understand weights and features; read model equations.
Tools/languages used here: NumPy, Jupyter, graph paper thinking.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- scalars: know the meaning, why it matters, when to use it, and one real example from an AI product.
- vectors: know the meaning, why it matters, when to use it, and one real example from an AI product.
- matrices: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dot product: know the meaning, why it matters, when to use it, and one real example from an AI product.
- matrix multiplication: know the meaning, why it matters, when to use it, and one real example from an AI product.
- transpose: know the meaning, why it matters, when to use it, and one real example from an AI product.
- identity matrix: know the meaning, why it matters, when to use it, and one real example from an AI product.
- norms: know the meaning, why it matters, when to use it, and one real example from an AI product.
- linear transformations: know the meaning, why it

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Mathematics: Linear Algebra for ML is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on scalars, vectors, matrices, dot product, matrix multiplication. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_06_mathematics:_linear_alge
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 06

```
# file: 06_linear_algebra.py
import numpy as np
features = np.array([2, 3, 5])
weights = np.array([0.4, 0.2, 0.1])
score = np.dot(features, weights)
print("features:", features)
print("weights:", weights)
print("weighted score:", round(score, 2))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 06_linear_algebra.py
features: [2 3 5]
weights: [0.4 0.2 0.1]
weighted score: 1.9
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Mathematics: Linear Algebra for ML, debug these first: scalars, vectors, matrices, dot product, matrix multiplication. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a feature-weight calculator for predicting a simple student performance score.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain scalars in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: NumPy, Jupyter, graph paper thinking?
3. What mistake can happen while using vectors and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Mathematics: Linear Algebra for ML in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 06 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain scalars with one example.
☐ I can explain vectors with one example.
☐ I can explain matrices with one example.
☐ I can explain dot product with one example.
☐ I can explain matrix multiplication with one example.
☐ I can explain transpose with one example.
☐ I can explain identity matrix with one example.
☐ I can explain norms with one example.
☐ I can explain linear transformations with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Probability & Statistics for ML. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07: Probability & Statistics for ML

Build the statistical foundation needed for data analysis, evaluation, and model decisions.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- summarize datasets
- detect spread and outliers
- interpret correlation
- avoid statistical mistakes

Topic Map

- mean
- median
- variance
- standard deviation
- probability
- conditional probability
- distributions
- correlation
- sampling

Tools & Languages

- Python statistics
- NumPy
- Pandas

Deliverable for This Module

Analyze a marks dataset and report average, spread, top performers, weak areas, and outliers.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Build the statistical foundation needed for data analysis, evaluation, and model decisions.
Important topics: mean, median, variance, standard deviation, probability, conditional probability, distributions.
Practical skill goal: summarize datasets; detect spread and outliers; interpret correlation; avoid statistical mistakes.
Tools/languages used here: Python statistics, NumPy, Pandas.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- mean: know the meaning, why it matters, when to use it, and one real example from an AI product.
- median: know the meaning, why it matters, when to use it, and one real example from an AI product.
- variance: know the meaning, why it matters, when to use it, and one real example from an AI product.
- standard deviation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- probability: know the meaning, why it matters, when to use it, and one real example from an AI product.
- conditional probability: know the meaning, why it matters, when to use it, and one real example from an AI product.
- distributions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- correlation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- sampling: know the meaning, why it matters, when to

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Probability & Statistics for ML is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on mean, median, variance, standard deviation, probability. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_07_probability_and_statisti
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 07

```
# file: 07_statistics.py
import statistics as stats
scores = [55, 64, 72, 72, 81, 90, 95]
print("mean:", round(stats.mean(scores), 2))
print("median:", stats.median(scores))
print("stdev:", round(stats.stdev(scores), 2))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 07_statistics.py
mean: 75.57
median: 72
stdev: 14.44
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Probability & Statistics for ML, debug these first: mean, median, variance, standard deviation, probability. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Analyze a marks dataset and report average, spread, top performers, weak areas, and outliers.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain mean in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python statistics, NumPy, Pandas?
3. What mistake can happen while using median and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Probability & Statistics for ML in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 07 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain mean with one example.
☐ I can explain median with one example.
☐ I can explain variance with one example.
☐ I can explain standard deviation with one example.
☐ I can explain probability with one example.
☐ I can explain conditional probability with one example.
☐ I can explain distributions with one example.
☐ I can explain correlation with one example.
☐ I can explain sampling with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: NumPy & Pandas for Data Handling. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08: NumPy & Pandas for Data Handling

Load, inspect, filter, group, transform, and save datasets for ML work.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- load tabular data
- clean columns
- group results
- prepare model-ready datasets

Topic Map

- arrays
- Series
- DataFrame
- read_csv
- head/info/describe
- filtering
- groupby
- missing values
- exporting

Tools & Languages

- NumPy
- Pandas
- CSV
- Excel basics

Deliverable for This Module

Create a course-progress CSV and analyze completion percentage by student, course, and week.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Load, inspect, filter, group, transform, and save datasets for ML work.
Important topics: arrays, Series, DataFrame, read_csv, head/info/describe, filtering, groupby.
Practical skill goal: load tabular data; clean columns; group results; prepare model-ready datasets.
Tools/languages used here: NumPy, Pandas, CSV, Excel basics.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- arrays: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Series: know the meaning, why it matters, when to use it, and one real example from an AI product.
- DataFrame: know the meaning, why it matters, when to use it, and one real example from an AI product.
- read_csv: know the meaning, why it matters, when to use it, and one real example from an AI product.
- head/info/describe: know the meaning, why it matters, when to use it, and one real example from an AI product.
- filtering: know the meaning, why it matters, when to use it, and one real example from an AI product.
- groupby: know the meaning, why it matters, when to use it, and one real example from an AI product.
- missing values: know the meaning, why it matters, when to use it, and one real example from an AI product.
- exporting: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

NumPy & Pandas for Data Handling is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on arrays, Series, DataFrame, read_csv, head/info/describe. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_08_numpy_and_pandas_for_dat
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 08

```
# file: 08_pandas_intro.py
import pandas as pd

data = {"name": ["Asha", "Ravi", "Sara"], "hours": [2, 5, 7], "score": [60,
78, 91]}
df = pd.DataFrame(data)
print(df)
print("average score:", df["score"].mean())
print("high practice students:")
print(df[df["hours"] >= 5][["name", "score"]])
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 08_pandas_intro.py
   name  hours  score
0  Asha      2     60
1  Ravi      5     78
2  Sara      7     91
average score: 76.33333333333333
high practice students:
   name  score
1  Ravi     78
2  Sara     91
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For NumPy & Pandas for Data Handling, debug these first: arrays, Series, DataFrame, read_csv, head/info/describe. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a course-progress CSV and analyze completion percentage by student, course, and week.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain arrays in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: NumPy, Pandas, CSV?
3. What mistake can happen while using Series and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about NumPy & Pandas for Data Handling in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 08 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain arrays with one example.
- ☐ I can explain Series with one example.
- ☐ I can explain DataFrame with one example.
- ☐ I can explain read_csv with one example.
- ☐ I can explain head/info/describe with one example.
- ☐ I can explain filtering with one example.
- ☐ I can explain groupby with one example.
- ☐ I can explain missing values with one example.
- ☐ I can explain exporting with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Data Visualization & Storytelling. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09: Data Visualization & Storytelling

Turn raw numbers into charts and explanations that business teams can understand.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- choose right chart
- label charts properly
- spot patterns visually
- explain insights

Topic Map

- line charts
- bar charts
- histograms
- scatter plots
- labels
- legends
- chart mistakes
- storytelling
- dashboard thinking

Tools & Languages

- Matplotlib
- Pandas plotting
- Jupyter

Deliverable for This Module

Build a 4-chart learning analytics report: completion, score, practice hours, and improvement trend.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Turn raw numbers into charts and explanations that business teams can understand.
Important topics: line charts, bar charts, histograms, scatter plots, labels, legends, chart mistakes.
Practical skill goal: choose right chart; label charts properly; spot patterns visually; explain insights.
Tools/languages used here: Matplotlib, Pandas plotting, Jupyter.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- line charts: know the meaning, why it matters, when to use it, and one real example from an AI product.
- bar charts: know the meaning, why it matters, when to use it, and one real example from an AI product.
- histograms: know the meaning, why it matters, when to use it, and one real example from an AI product.
- scatter plots: know the meaning, why it matters, when to use it, and one real example from an AI product.
- labels: know the meaning, why it matters, when to use it, and one real example from an AI product.
- legends: know the meaning, why it matters, when to use it, and one real example from an AI product.
- chart mistakes: know the meaning, why it matters, when to use it, and one real example from an AI product.
- storytelling: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dashboard thinking: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Data Visualization & Storytelling is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on line charts, bar charts, histograms, scatter plots, labels. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_09_data_visualization_and_s
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 09

```
# file: 09_visualization.py
import matplotlib.pyplot as plt
weeks = [1, 2, 3, 4]
accuracy = [0.62, 0.70, 0.76, 0.82]
plt.plot(weeks, accuracy, marker="o")
plt.title("Model Accuracy by Week")
plt.xlabel("Week")
plt.ylabel("Accuracy")
plt.savefig("accuracy_chart.png")
print("saved: accuracy_chart.png")
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 09_visualization.py
saved: accuracy_chart.png
Preview: a line chart increasing from 0.62 to 0.82 across week 1 to week
4.
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Data Visualization & Storytelling, debug these first: line charts, bar charts, histograms, scatter plots, labels. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build a 4-chart learning analytics report: completion, score, practice hours, and improvement trend.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain line charts in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Matplotlib, Pandas plotting, Jupyter?
3. What mistake can happen while using bar charts and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Data Visualization & Storytelling in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 09 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain line charts with one example.
- ☐ I can explain bar charts with one example.
- ☐ I can explain histograms with one example.
- ☐ I can explain scatter plots with one example.
- ☐ I can explain labels with one example.
- ☐ I can explain legends with one example.
- ☐ I can explain chart mistakes with one example.
- ☐ I can explain storytelling with one example.
- ☐ I can explain dashboard thinking with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Data Cleaning & Feature Engineering. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10: Data Cleaning & Feature Engineering

Prepare messy real-world data so machine learning models can learn correctly.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- clean missing data
- create useful features
- encode text categories
- avoid leakage

Topic Map

- missing values
- duplicates
- outliers
- encoding categories
- scaling
- feature creation
- data leakage
- train/test split safety

Tools & Languages

- Pandas
- scikit-learn preprocessing
- NumPy

Deliverable for This Module

Clean a student dataset with missing hours, duplicate users, text categories, and engineered progress score.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Prepare messy real-world data so machine learning models can learn correctly.
Important topics: missing values, duplicates, outliers, encoding categories, scaling, feature creation, data leakage.
Practical skill goal: clean missing data; create useful features; encode text categories; avoid leakage.
Tools/languages used here: Pandas, scikit-learn preprocessing, NumPy.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- missing values: know the meaning, why it matters, when to use it, and one real example from an AI product.
- duplicates: know the meaning, why it matters, when to use it, and one real example from an AI product.
- outliers: know the meaning, why it matters, when to use it, and one real example from an AI product.
- encoding categories: know the meaning, why it matters, when to use it, and one real example from an AI product.
- scaling: know the meaning, why it matters, when to use it, and one real example from an AI product.
- feature creation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- data leakage: know the meaning, why it matters, when to use it, and one real example from an AI product.
- train/test split safety: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Data Cleaning & Feature Engineering is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on missing values, duplicates, outliers, encoding categories, scaling. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_10_data_cleaning_and_featur
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 10

```
# file: 10_feature_engineering.py
import pandas as pd
df = pd.DataFrame({"hours": [2, None, 5], "level": ["basic", "pro", "basic"]})
df["hours"] = df["hours"].fillna(df["hours"].median())
df["is_pro"] = (df["level"] == "pro").astype(int)
df["practice_score"] = df["hours"] * 10
print(df)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 10_feature_engineering.py
   hours  level  is_pro  practice_score
0    2.0  basic       0            20.0
1    3.5    pro       1            35.0
2    5.0  basic       0            50.0
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Data Cleaning & Feature Engineering, debug these first: missing values, duplicates, outliers, encoding categories, scaling. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Clean a student dataset with missing hours, duplicate users, text categories, and engineered progress score.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain missing values in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Pandas, scikit-learn preprocessing, NumPy?
3. What mistake can happen while using duplicates and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Data Cleaning & Feature Engineering in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 10 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain missing values with one example.
☐ I can explain duplicates with one example.
☐ I can explain outliers with one example.
☐ I can explain encoding categories with one example.
☐ I can explain scaling with one example.
☐ I can explain feature creation with one example.
☐ I can explain data leakage with one example.
☐ I can explain train/test split safety with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Machine Learning Workflow with scikit-learn. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11: Machine Learning Workflow with scikit-learn

Learn the standard pipeline: problem framing, split, train, evaluate, improve, save.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- train baseline models
- build repeatable pipeline
- compare experiments
- save models

Topic Map

- problem framing
- features and labels
- train/test split
- fit/predict
- pipeline
- baseline model
- metrics
- model persistence

Tools & Languages

- scikit-learn
- joblib
- Pandas
- NumPy

Deliverable for This Module

Build a baseline pass/fail classifier from practice hours, attendance, quiz score, and assignment score.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Learn the standard pipeline: problem framing, split, train, evaluate, improve, save.
Important topics: problem framing, features and labels, train/test split, fit/predict, pipeline, baseline model, metrics.
Practical skill goal: train baseline models; build repeatable pipeline; compare experiments; save models.
Tools/languages used here: scikit-learn, joblib, Pandas, NumPy.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- problem framing: know the meaning, why it matters, when to use it, and one real example from an AI product.
- features and labels: know the meaning, why it matters, when to use it, and one real example from an AI product.
- train/test split: know the meaning, why it matters, when to use it, and one real example from an AI product.
- fit/predict: know the meaning, why it matters, when to use it, and one real example from an AI product.
- pipeline: know the meaning, why it matters, when to use it, and one real example from an AI product.
- baseline model: know the meaning, why it matters, when to use it, and one real example from an AI product.
- metrics: know the meaning, why it matters, when to use it, and one real example from an AI product.
- model persistence: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Machine Learning Workflow with scikit-learn is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on problem framing, features and labels, train/test split, fit/predict, pipeline. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_11_machine_learning_workflo
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 11

```
# file: 11_ml_workflow.py
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X = [[1], [2], [3], [4], [5], [6]]
y = [0, 0, 0, 1, 1, 1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
model = LogisticRegression().fit(X_train, y_train)
pred = model.predict(X_test)
print("pred:", pred.tolist())
print("accuracy:", accuracy_score(y_test, pred))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 11_ml_workflow.py
pred: [0, 0]
accuracy: 0.5
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Machine Learning Workflow with scikit-learn, debug these first: problem framing, features and labels, train/test split, fit/predict, pipeline. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build a baseline pass/fail classifier from practice hours, attendance, quiz score, and assignment score.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain problem framing in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: scikit-learn, joblib, Pandas?
3. What mistake can happen while using features and labels and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Machine Learning Workflow with scikit-learn in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 11 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain problem framing with one example.
☐ I can explain features and labels with one example.
☐ I can explain train/test split with one example.
☐ I can explain fit/predict with one example.
☐ I can explain pipeline with one example.
☐ I can explain baseline model with one example.
☐ I can explain metrics with one example.
☐ I can explain model persistence with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Regression Models. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12: Regression Models

Predict continuous values such as price, score, demand, revenue, or completion time.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- train regression models
- evaluate error
- interpret coefficients
- improve features

Topic Map

- linear regression
- multiple regression
- polynomial features
- regularization
- MAE
- MSE
- RMSE
- R2 score
- residuals

Tools & Languages

- scikit-learn LinearRegression
- Ridge
- Lasso
- metrics

Deliverable for This Module

Predict final course score from study hours, practice tasks, attendance, and quiz average.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Predict continuous values such as price, score, demand, revenue, or completion time.
Important topics: linear regression, multiple regression, polynomial features, regularization, MAE, MSE, RMSE.
Practical skill goal: train regression models; evaluate error; interpret coefficients; improve features.
Tools/languages used here: scikit-learn LinearRegression, Ridge, Lasso, metrics.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- linear regression: know the meaning, why it matters, when to use it, and one real example from an AI product.
- multiple regression: know the meaning, why it matters, when to use it, and one real example from an AI product.
- polynomial features: know the meaning, why it matters, when to use it, and one real example from an AI product.
- regularization: know the meaning, why it matters, when to use it, and one real example from an AI product.
- MAE: know the meaning, why it matters, when to use it, and one real example from an AI product.
- MSE: know the meaning, why it matters, when to use it, and one real example from an AI product.
- RMSE: know the meaning, why it matters, when to use it, and one real example from an AI product.
- R2 score: know the meaning, why it matters, when to use it, and one real example from an AI product.
- residuals: know the meaning, why it matters, when to

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Regression Models is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on linear regression, multiple regression, polynomial features, regularization, MAE. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_12_regression_models
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 12

```
# file: 12_regression.py
from sklearn.linear_model import LinearRegression
X = [[1], [2], [3], [4], [5]]
y = [50, 55, 65, 70, 80]
model = LinearRegression().fit(X, y)
print("slope:", round(model.coef_[0], 2))
print("prediction for 6 hours:", round(model.predict([[6]])[0], 2))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 12_regression.py
slope: 7.5
prediction for 6 hours: 88.0
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Regression Models, debug these first: linear regression, multiple regression, polynomial features, regularization, MAE. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Predict final course score from study hours, practice tasks, attendance, and quiz average.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain linear regression in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: scikit-learn LinearRegression, Ridge, Lasso?
3. What mistake can happen while using multiple regression and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Regression Models in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 12 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain linear regression with one example.
- ☐ I can explain multiple regression with one example.
- ☐ I can explain polynomial features with one example.
- ☐ I can explain regularization with one example.
- ☐ I can explain MAE with one example.
- ☐ I can explain MSE with one example.
- ☐ I can explain RMSE with one example.
- ☐ I can explain R2 score with one example.
- ☐ I can explain residuals with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Classification Models. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13: Classification Models

Predict categories such as spam/not spam, pass/fail, churn/no churn, fraud/safe.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- train classifiers
- interpret predictions
- choose class metrics
- handle imbalance

Topic Map

- logistic regression
- KNN
- decision tree
- random forest
- SVM basics
- naive Bayes
- class imbalance
- thresholds

Tools & Languages

- scikit-learn classifiers
- classification_report
- confusion_matrix

Deliverable for This Module

Create a certificate-eligibility classifier using attendance, assignments, test score, and project status.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Predict categories such as spam/not spam, pass/fail, churn/no churn, fraud/safe.
Important topics: logistic regression, KNN, decision tree, random forest, SVM basics, naive Bayes, class imbalance.
Practical skill goal: train classifiers; interpret predictions; choose class metrics; handle imbalance.
Tools/languages used here: scikit-learn classifiers, classification_report, confusion_matrix.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- logistic regression: know the meaning, why it matters, when to use it, and one real example from an AI product.
- KNN: know the meaning, why it matters, when to use it, and one real example from an AI product.
- decision tree: know the meaning, why it matters, when to use it, and one real example from an AI product.
- random forest: know the meaning, why it matters, when to use it, and one real example from an AI product.
- SVM basics: know the meaning, why it matters, when to use it, and one real example from an AI product.
- naive Bayes: know the meaning, why it matters, when to use it, and one real example from an AI product.
- class imbalance: know the meaning, why it matters, when to use it, and one real example from an AI product.
- thresholds: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Classification Models is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on logistic regression, KNN, decision tree, random forest, SVM basics. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_13_classification_models
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 13

```
# file: 13_classification.py
from sklearn.tree import DecisionTreeClassifier
X = [[2, 40], [3, 50], [6, 80], [7, 92]]
y = [0, 0, 1, 1]  # 1 = certificate ready
model = DecisionTreeClassifier(random_state=1).fit(X, y)
print("student A:", model.predict([[5, 75]])[0])
print("student B:", model.predict([[2, 45]])[0])
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 13_classification.py
student A: 1
student B: 0
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Classification Models, debug these first: logistic regression, KNN, decision tree, random forest, SVM basics. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a certificate-eligibility classifier using attendance, assignments, test score, and project status.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain logistic regression in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: scikit-learn classifiers, classification_report, confusion_matrix?
3. What mistake can happen while using KNN and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Classification Models in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 13 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain logistic regression with one example.
☐ I can explain KNN with one example.
☐ I can explain decision tree with one example.
☐ I can explain random forest with one example.
☐ I can explain SVM basics with one example.
☐ I can explain naive Bayes with one example.
☐ I can explain class imbalance with one example.
☐ I can explain thresholds with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Model Evaluation & Validation. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14: Model Evaluation & Validation

Measure model quality correctly and avoid fake accuracy.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- select metrics
- read confusion matrix
- validate safely
- compare models fairly

Topic Map

- accuracy
- precision
- recall
- F1 score
- ROC-AUC
- cross-validation
- confusion matrix
- overfitting
- underfitting

Tools & Languages

- sklearn.metrics
- cross_val_score
- train_test_split

Deliverable for This Module

Evaluate two certificate classifiers and write which one is safer for real student decisions.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Measure model quality correctly and avoid fake accuracy.
Important topics: accuracy, precision, recall, F1 score, ROC-AUC, cross-validation, confusion matrix.
Practical skill goal: select metrics; read confusion matrix; validate safely; compare models fairly.
Tools/languages used here: sklearn.metrics, cross_val_score, train_test_split.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- accuracy: know the meaning, why it matters, when to use it, and one real example from an AI product.
- precision: know the meaning, why it matters, when to use it, and one real example from an AI product.
- recall: know the meaning, why it matters, when to use it, and one real example from an AI product.
- F1 score: know the meaning, why it matters, when to use it, and one real example from an AI product.
- ROC-AUC: know the meaning, why it matters, when to use it, and one real example from an AI product.
- cross-validation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- confusion matrix: know the meaning, why it matters, when to use it, and one real example from an AI product.
- overfitting: know the meaning, why it matters, when to use it, and one real example from an AI product.
- underfitting: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Model Evaluation & Validation is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on accuracy, precision, recall, F1 score, ROC-AUC. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_14_model_evaluation_and_val
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 14

```
# file: 14_evaluation.py
from sklearn.metrics import confusion_matrix, classification_report
y_true = [1, 0, 1, 1, 0, 0]
y_pred = [1, 0, 0, 1, 0, 1]
print(confusion_matrix(y_true, y_pred))
print(classification_report(y_true, y_pred, zero_division=0))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 14_evaluation.py
[[2 1]
 [1 2]]
precision/recall/F1 report printed for classes 0 and 1.
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Model Evaluation & Validation, debug these first: accuracy, precision, recall, F1 score, ROC-AUC. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Evaluate two certificate classifiers and write which one is safer for real student decisions.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain accuracy in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: sklearn.metrics, cross_val_score, train_test_split?
3. What mistake can happen while using precision and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Model Evaluation & Validation in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 14 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain accuracy with one example.
☐ I can explain precision with one example.
☐ I can explain recall with one example.
☐ I can explain F1 score with one example.
☐ I can explain ROC-AUC with one example.
☐ I can explain cross-validation with one example.
☐ I can explain confusion matrix with one example.
☐ I can explain overfitting with one example.
☐ I can explain underfitting with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Unsupervised Learning. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15: Unsupervised Learning

Find hidden patterns without labels using clustering and dimensionality reduction.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- segment users
- reduce dimensions
- interpret clusters
- detect unusual behavior

Topic Map

- clustering
- K-Means
- DBSCAN idea
- PCA
- t-SNE/UMAP idea
- anomaly detection
- customer/student segmentation

Tools & Languages

- KMeans
- PCA
- StandardScaler
- silhouette score

Deliverable for This Module

Cluster students into learning groups: beginner support, consistent learner, and advanced achiever.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Find hidden patterns without labels using clustering and dimensionality reduction.
Important topics: clustering, K-Means, DBSCAN idea, PCA, t-SNE/UMAP idea, anomaly detection, customer/student segmentation.
Practical skill goal: segment users; reduce dimensions; interpret clusters; detect unusual behavior.
Tools/languages used here: KMeans, PCA, StandardScaler, silhouette score.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- clustering: know the meaning, why it matters, when to use it, and one real example from an AI product.
- K-Means: know the meaning, why it matters, when to use it, and one real example from an AI product.
- DBSCAN idea: know the meaning, why it matters, when to use it, and one real example from an AI product.
- PCA: know the meaning, why it matters, when to use it, and one real example from an AI product.
- t-SNE/UMAP idea: know the meaning, why it matters, when to use it, and one real example from an AI product.
- anomaly detection: know the meaning, why it matters, when to use it, and one real example from an AI product.
- customer/student segmentation: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Unsupervised Learning is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on clustering, K-Means, DBSCAN idea, PCA, t-SNE/UMAP idea. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_15_unsupervised_learning
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 15

```
# file: 15_kmeans.py
from sklearn.cluster import KMeans
X = [[1, 2], [1, 3], [8, 8], [9, 8], [2, 2], [8, 9]]
model = KMeans(n_clusters=2, random_state=1, n_init=10).fit(X)
print("labels:", model.labels_.tolist())
print("centers:", model.cluster_centers_.round(2).tolist())
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 15_kmeans.py
labels: [1, 1, 0, 0, 1, 0]
centers: [[8.33, 8.33], [1.33, 2.33]]
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Unsupervised Learning, debug these first: clustering, K-Means, DBSCAN idea, PCA, t-SNE/UMAP idea. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Cluster students into learning groups: beginner support, consistent learner, and advanced achiever.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain clustering in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: KMeans, PCA, StandardScaler?
3. What mistake can happen while using K-Means and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Unsupervised Learning in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 15 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain clustering with one example.
☐ I can explain K-Means with one example.
☐ I can explain DBSCAN idea with one example.
☐ I can explain PCA with one example.
☐ I can explain t-SNE/UMAP idea with one example.
☐ I can explain anomaly detection with one example.
☐ I can explain customer/student segmentation with one example.
☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.

Practical Checklist

☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Recommendation Systems. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16: Recommendation Systems

Build systems that recommend courses, videos, notes, products, or content.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- build simple recommender
- rank content
- handle new users
- explain recommendations

Topic Map

- content-based recommendation
- collaborative filtering
- cosine similarity
- user-item matrix
- ranking
- cold start
- feedback loops

Tools & Languages

- NumPy
- Pandas
- cosine_similarity

Deliverable for This Module

Create a course recommender that suggests the next lesson based on skills and completed modules.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Build systems that recommend courses, videos, notes, products, or content.
Important topics: content-based recommendation, collaborative filtering, cosine similarity, user-item matrix, ranking, cold start, feedback loops.
Practical skill goal: build simple recommender; rank content; handle new users; explain recommendations.
Tools/languages used here: NumPy, Pandas, cosine_similarity.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- content-based recommendation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- collaborative filtering: know the meaning, why it matters, when to use it, and one real example from an AI product.
- cosine similarity: know the meaning, why it matters, when to use it, and one real example from an AI product.
- user-item matrix: know the meaning, why it matters, when to use it, and one real example from an AI product.
- ranking: know the meaning, why it matters, when to use it, and one real example from an AI product.
- cold start: know the meaning, why it matters, when to use it, and one real example from an AI product.
- feedback loops: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Recommendation Systems is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on content-based recommendation, collaborative filtering, cosine similarity, user-item matrix, ranking. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_16_recommendation_systems
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 16

```
# file: 16_recommender.py
from sklearn.metrics.pairwise import cosine_similarity
courses = [[1,0,1], [1,1,0], [0,1,1]]
student_interest = [[1,0,1]]
scores = cosine_similarity(student_interest, courses)[0]
print("similarity scores:", [round(s, 2) for s in scores])
print("best course index:", scores.argmax())
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 16_recommender.py
similarity scores: [1.0, 0.5, 0.5]
best course index: 0
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Recommendation Systems, debug these first: content-based recommendation, collaborative filtering, cosine similarity, user-item matrix, ranking. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a course recommender that suggests the next lesson based on skills and completed modules.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain content-based recommendation in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: NumPy, Pandas, cosine_similarity?
3. What mistake can happen while using collaborative filtering and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Recommendation Systems in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 16 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain content-based recommendation with one example.
☐ I can explain collaborative filtering with one example.
☐ I can explain cosine similarity with one example.
☐ I can explain user-item matrix with one example.
☐ I can explain ranking with one example.
☐ I can explain cold start with one example.
☐ I can explain feedback loops with one example.
☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.

Practical Checklist

☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Time Series Forecasting. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17: Time Series Forecasting

Predict values over time such as traffic, sales, student activity, or server load.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- create time features
- plot trends
- build baseline forecast
- validate by time

Topic Map

- time index
- trend
- seasonality
- moving average
- lag features
- forecast horizon
- walk-forward validation
- forecast errors

Tools & Languages

- Pandas
- Matplotlib
- scikit-learn
- statsmodels concept

Deliverable for This Module

Forecast weekly course traffic and identify peak days for live-class scheduling.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Predict values over time such as traffic, sales, student activity, or server load.
Important topics: time index, trend, seasonality, moving average, lag features, forecast horizon, walk-forward validation.
Practical skill goal: create time features; plot trends; build baseline forecast; validate by time.
Tools/languages used here: Pandas, Matplotlib, scikit-learn, statsmodels concept.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- time index: know the meaning, why it matters, when to use it, and one real example from an AI product.
- trend: know the meaning, why it matters, when to use it, and one real example from an AI product.
- seasonality: know the meaning, why it matters, when to use it, and one real example from an AI product.
- moving average: know the meaning, why it matters, when to use it, and one real example from an AI product.
- lag features: know the meaning, why it matters, when to use it, and one real example from an AI product.
- forecast horizon: know the meaning, why it matters, when to use it, and one real example from an AI product.
- walk-forward validation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- forecast errors: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Time Series Forecasting is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on time index, trend, seasonality, moving average, lag features. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_17_time_series_forecasting
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 17

```
# file: 17_time_series.py
import pandas as pd
traffic = pd.Series([100, 120, 130, 125, 150, 160])
rolling = traffic.rolling(window=3).mean()
print("traffic:", traffic.tolist())
print("3-day moving avg:", rolling.round(1).tolist())
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 17_time_series.py
traffic: [100, 120, 130, 125, 150, 160]
3-day moving avg: [nan, nan, 116.7, 125.0, 135.0, 145.0]
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Time Series Forecasting, debug these first: time index, trend, seasonality, moving average, lag features. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Forecast weekly course traffic and identify peak days for live-class scheduling.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain time index in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Pandas, Matplotlib, scikit-learn?
3. What mistake can happen while using trend and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Time Series Forecasting in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 17 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain time index with one example.
☐ I can explain trend with one example.
☐ I can explain seasonality with one example.
☐ I can explain moving average with one example.
☐ I can explain lag features with one example.
☐ I can explain forecast horizon with one example.
☐ I can explain walk-forward validation with one example.
☐ I can explain forecast errors with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Neural Network Fundamentals. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18: Neural Network Fundamentals

Understand neurons, layers, activations, loss, and backpropagation intuition.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- explain neural nets
- calculate simple forward pass
- read training logs
- avoid learning-rate mistakes

Topic Map

- perceptron
- weights
- bias
- activation
- forward pass
- loss function
- gradient descent
- learning rate
- epochs

Tools & Languages

- NumPy
- PyTorch/TensorFlow concept
- Jupyter

Deliverable for This Module

Create a hand-calculated neural network worksheet for a two-feature prediction.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Understand neurons, layers, activations, loss, and backpropagation intuition.
Important topics: perceptron, weights, bias, activation, forward pass, loss function, gradient descent.
Practical skill goal: explain neural nets; calculate simple forward pass; read training logs; avoid learning-rate mistakes.
Tools/languages used here: NumPy, PyTorch/TensorFlow concept, Jupyter.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- perceptron: know the meaning, why it matters, when to use it, and one real example from an AI product.
- weights: know the meaning, why it matters, when to use it, and one real example from an AI product.
- bias: know the meaning, why it matters, when to use it, and one real example from an AI product.
- activation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- forward pass: know the meaning, why it matters, when to use it, and one real example from an AI product.
- loss function: know the meaning, why it matters, when to use it, and one real example from an AI product.
- gradient descent: know the meaning, why it matters, when to use it, and one real example from an AI product.
- learning rate: know the meaning, why it matters, when to use it, and one real example from an AI product.
- epochs: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Neural Network Fundamentals is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on perceptron, weights, bias, activation, forward pass. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_18_neural_network_fundament
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 18

```
# file: 18_neural_forward.py
import math
x = [0.8, 0.4]
w = [0.6, -0.2]
b = 0.1
z = x[0]*w[0] + x[1]*w[1] + b
sigmoid = 1 / (1 + math.exp(-z))
print("z:", round(z, 3))
print("activation:", round(sigmoid, 3))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 18_neural_forward.py
z: 0.5
activation: 0.622
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Neural Network Fundamentals, debug these first: perceptron, weights, bias, activation, forward pass. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a hand-calculated neural network worksheet for a two-feature prediction.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain perceptron in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: NumPy, PyTorch/TensorFlow concept, Jupyter?
3. What mistake can happen while using weights and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Neural Network Fundamentals in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 18 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain perceptron with one example.
- ☐ I can explain weights with one example.
- ☐ I can explain bias with one example.
- ☐ I can explain activation with one example.
- ☐ I can explain forward pass with one example.
- ☐ I can explain loss function with one example.
- ☐ I can explain gradient descent with one example.
- ☐ I can explain learning rate with one example.
- ☐ I can explain epochs with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Deep Learning with PyTorch/TensorFlow. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19: Deep Learning with PyTorch/TensorFlow

Learn how modern deep-learning training loops and model layers are structured.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- read deep learning code
- create training loop
- track loss
- save model checkpoint

Topic Map

- tensors
- datasets
- dataloaders
- layers
- optimizers
- loss
- training loop
- validation loop
- saving checkpoints

Tools & Languages

- PyTorch or TensorFlow
- NumPy
- GPU/CPU basics

Deliverable for This Module

Write a training-log simulator, then replace it later with a real PyTorch/TensorFlow model.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Learn how modern deep-learning training loops and model layers are structured.
Important topics: tensors, datasets, dataloaders, layers, optimizers, loss, training loop.
Practical skill goal: read deep learning code; create training loop; track loss; save model checkpoint.
Tools/languages used here: PyTorch or TensorFlow, NumPy, GPU/CPU basics.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- tensors: know the meaning, why it matters, when to use it, and one real example from an AI product.
- datasets: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dataloaders: know the meaning, why it matters, when to use it, and one real example from an AI product.
- layers: know the meaning, why it matters, when to use it, and one real example from an AI product.
- optimizers: know the meaning, why it matters, when to use it, and one real example from an AI product.
- loss: know the meaning, why it matters, when to use it, and one real example from an AI product.
- training loop: know the meaning, why it matters, when to use it, and one real example from an AI product.
- validation loop: know the meaning, why it matters, when to use it, and one real example from an AI product.
- saving checkpoints: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Deep Learning with PyTorch/TensorFlow is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on tensors, datasets, dataloaders, layers, optimizers. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_19_deep_learning_with_pytor
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 19

```
# file: 19_training_loop_pseudocode.py
for epoch in range(1, 4):
    train_loss = 1.0 / epoch
    val_loss = 1.2 / epoch
    print(f"epoch={epoch} train_loss={train_loss:.3f}
val_loss={val_loss:.3f}")
print("checkpoint saved: model_epoch_3.pt")
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 19_training_loop_pseudocode.py
epoch=1 train_loss=1.000 val_loss=1.200
epoch=2 train_loss=0.500 val_loss=0.600
epoch=3 train_loss=0.333 val_loss=0.400
checkpoint saved: model_epoch_3.pt
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Deep Learning with PyTorch/TensorFlow, debug these first: tensors, datasets, dataloaders, layers, optimizers. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Write a training-log simulator, then replace it later with a real PyTorch/TensorFlow model.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain tensors in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: PyTorch or TensorFlow, NumPy, GPU/CPU basics?
3. What mistake can happen while using datasets and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Deep Learning with PyTorch/TensorFlow in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 19 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain tensors with one example.
- ☐ I can explain datasets with one example.
- ☐ I can explain dataloaders with one example.
- ☐ I can explain layers with one example.
- ☐ I can explain optimizers with one example.
- ☐ I can explain loss with one example.
- ☐ I can explain training loop with one example.
- ☐ I can explain validation loop with one example.
- ☐ I can explain saving checkpoints with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Computer Vision. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20: Computer Vision

Process images and build image classification, detection, and visual AI workflows.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- load images
- preprocess pixels
- explain CNN layers
- evaluate image model

Topic Map

- pixels
- channels
- image resize
- augmentation
- CNN concept
- classification
- object detection idea
- OCR idea
- visual datasets

Tools & Languages

- Pillow
- OpenCV concept
- PyTorch/TensorFlow vision libraries

Deliverable for This Module

Build an image preprocessing notebook that resizes, normalizes, and previews sample training images.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Process images and build image classification, detection, and visual AI workflows.
Important topics: pixels, channels, image resize, augmentation, CNN concept, classification, object detection idea.
Practical skill goal: load images; preprocess pixels; explain CNN layers; evaluate image model.
Tools/languages used here: Pillow, OpenCV concept, PyTorch/TensorFlow vision libraries.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- pixels: know the meaning, why it matters, when to use it, and one real example from an AI product.
- channels: know the meaning, why it matters, when to use it, and one real example from an AI product.
- image resize: know the meaning, why it matters, when to use it, and one real example from an AI product.
- augmentation: know the meaning, why it matters, when to use it, and one real example from an AI product.
- CNN concept: know the meaning, why it matters, when to use it, and one real example from an AI product.
- classification: know the meaning, why it matters, when to use it, and one real example from an AI product.
- object detection idea: know the meaning, why it matters, when to use it, and one real example from an AI product.
- OCR idea: know the meaning, why it matters, when to use it, and one real example from an AI product.
- visual datasets: know the meaning, why it matters,

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Computer Vision is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on pixels, channels, image resize, augmentation, CNN concept. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_20_computer_vision
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 20

```
# file: 20_image_info.py
from PIL import Image
img = Image.new("RGB", (224, 224), color="white")
print("mode:", img.mode)
print("size:", img.size)
print("channels:", len(img.getbands()))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 20_image_info.py
mode: RGB
size: (224, 224)
channels: 3
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Computer Vision, debug these first: pixels, channels, image resize, augmentation, CNN concept. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build an image preprocessing notebook that resizes, normalizes, and previews sample training images.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain pixels in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Pillow, OpenCV concept, PyTorch/TensorFlow vision libraries?
3. What mistake can happen while using channels and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Computer Vision in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 20 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain pixels with one example.
- ☐ I can explain channels with one example.
- ☐ I can explain image resize with one example.
- ☐ I can explain augmentation with one example.
- ☐ I can explain CNN concept with one example.
- ☐ I can explain classification with one example.
- ☐ I can explain object detection idea with one example.
- ☐ I can explain OCR idea with one example.
- ☐ I can explain visual datasets with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Natural Language Processing. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21: Natural Language Processing

Work with text data for sentiment, classification, search, chatbots, and document intelligence.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- clean text
- convert text to features
- train text classifier
- explain NLP pipeline

Topic Map

- tokenization
- normalization
- stopwords
- bag of words
- TF-IDF
- sentiment
- text classification
- named entities
- text cleaning

Tools & Languages

- Python strings
- scikit-learn text features
- regex

Deliverable for This Module

Create a text classifier that labels student messages as doubt, feedback, complaint, or appreciation.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Work with text data for sentiment, classification, search, chatbots, and document intelligence.
Important topics: tokenization, normalization, stopwords, bag of words, TF-IDF, sentiment, text classification.
Practical skill goal: clean text; convert text to features; train text classifier; explain NLP pipeline.
Tools/languages used here: Python strings, scikit-learn text features, regex.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- tokenization: know the meaning, why it matters, when to use it, and one real example from an AI product.
- normalization: know the meaning, why it matters, when to use it, and one real example from an AI product.
- stopwords: know the meaning, why it matters, when to use it, and one real example from an AI product.
- bag of words: know the meaning, why it matters, when to use it, and one real example from an AI product.
- TF-IDF: know the meaning, why it matters, when to use it, and one real example from an AI product.
- sentiment: know the meaning, why it matters, when to use it, and one real example from an AI product.
- text classification: know the meaning, why it matters, when to use it, and one real example from an AI product.
- named entities: know the meaning, why it matters, when to use it, and one real example from an AI product.
- text cleaning: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Natural Language Processing is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on tokenization, normalization, stopwords, bag of words, TF-IDF. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_21_natural_language_process
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 21

```
# file: 21_nlp_basics.py
text = "ASFI helps students learn AI and Machine Learning."
tokens = text.lower().replace(".", "").split()
print("tokens:", tokens)
print("token count:", len(tokens))
print("contains ai:", "ai" in tokens)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 21_nlp_basics.py
tokens: ['asfi', 'helps', 'students', 'learn', 'ai', 'and', 'machine',
'learning']
token count: 8
contains ai: True
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Natural Language Processing, debug these first: tokenization, normalization, stopwords, bag of words, TF-IDF. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a text classifier that labels student messages as doubt, feedback, complaint, or appreciation.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain tokenization in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python strings, scikit-learn text features, regex?
3. What mistake can happen while using normalization and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Natural Language Processing in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 21 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain tokenization with one example.
- ☐ I can explain normalization with one example.
- ☐ I can explain stopwords with one example.
- ☐ I can explain bag of words with one example.
- ☐ I can explain TF-IDF with one example.
- ☐ I can explain sentiment with one example.
- ☐ I can explain text classification with one example.
- ☐ I can explain named entities with one example.
- ☐ I can explain text cleaning with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Transformers & Embeddings. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22: Transformers & Embeddings

Understand attention, embeddings, semantic search, and transformer-based AI systems.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- compare semantic similarity
- explain embeddings
- design search flow
- understand transformer limits

Topic Map

- word embeddings
- sentence embeddings
- cosine similarity
- attention intuition
- transformer blocks
- semantic search
- vector space
- context length

Tools & Languages

- sentence-transformers concept
- NumPy
- vector databases

Deliverable for This Module

Create a semantic search demo that finds the most related course note for a student question.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Understand attention, embeddings, semantic search, and transformer-based AI systems.
Important topics: word embeddings, sentence embeddings, cosine similarity, attention intuition, transformer blocks, semantic search, vector space.
Practical skill goal: compare semantic similarity; explain embeddings; design search flow; understand transformer limits.
Tools/languages used here: sentence-transformers concept, NumPy, vector databases.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- word embeddings: know the meaning, why it matters, when to use it, and one real example from an AI product.
- sentence embeddings: know the meaning, why it matters, when to use it, and one real example from an AI product.
- cosine similarity: know the meaning, why it matters, when to use it, and one real example from an AI product.
- attention intuition: know the meaning, why it matters, when to use it, and one real example from an AI product.
- transformer blocks: know the meaning, why it matters, when to use it, and one real example from an AI product.
- semantic search: know the meaning, why it matters, when to use it, and one real example from an AI product.
- vector space: know the meaning, why it matters, when to use it, and one real example from an AI product.
- context length: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Transformers & Embeddings is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on word embeddings, sentence embeddings, cosine similarity, attention intuition, transformer blocks. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_22_transformers_and_embeddi
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 22

```
# file: 22_embedding_similarity.py
import numpy as np
def cosine(a, b):
    return float(np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b)))
query = np.array([0.9, 0.1, 0.3])
doc1 = np.array([0.8, 0.2, 0.2])
doc2 = np.array([0.1, 0.9, 0.4])
print("doc1 similarity:", round(cosine(query, doc1), 3))
print("doc2 similarity:", round(cosine(query, doc2), 3))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 22_embedding_similarity.py
doc1 similarity: 0.991
doc2 similarity: 0.422
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Transformers & Embeddings, debug these first: word embeddings, sentence embeddings, cosine similarity, attention intuition, transformer blocks. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a semantic search demo that finds the most related course note for a student question.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain word embeddings in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: sentence-transformers concept, NumPy, vector databases?
3. What mistake can happen while using sentence embeddings and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Transformers & Embeddings in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 22 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain word embeddings with one example.
☐ I can explain sentence embeddings with one example.
☐ I can explain cosine similarity with one example.
☐ I can explain attention intuition with one example.
☐ I can explain transformer blocks with one example.
☐ I can explain semantic search with one example.
☐ I can explain vector space with one example.
☐ I can explain context length with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Generative AI & Large Language Models. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23: Generative AI & Large Language Models

Learn how LLMs generate text, use prompts, follow context, and support AI products.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- design LLM flow
- identify hallucination risk
- build safe prompts
- separate model vs app logic

Topic Map

- language models
- tokens
- context window
- temperature
- top-p
- system/user messages
- hallucination
- grounding
- chat UX

Tools & Languages

- local LLM concept
- Ollama/vLLM concept
- FastAPI integration idea

Deliverable for This Module

Create a safe AI tutor prompt template with role, context, task, output format, and refusal rules.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Learn how LLMs generate text, use prompts, follow context, and support AI products.
Important topics: language models, tokens, context window, temperature, top-p, system/user messages, hallucination.
Practical skill goal: design LLM flow; identify hallucination risk; build safe prompts; separate model vs app logic.
Tools/languages used here: local LLM concept, Ollama/vLLM concept, FastAPI integration idea.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- language models: know the meaning, why it matters, when to use it, and one real example from an AI product.
- tokens: know the meaning, why it matters, when to use it, and one real example from an AI product.
- context window: know the meaning, why it matters, when to use it, and one real example from an AI product.
- temperature: know the meaning, why it matters, when to use it, and one real example from an AI product.
- top-p: know the meaning, why it matters, when to use it, and one real example from an AI product.
- system/user messages: know the meaning, why it matters, when to use it, and one real example from an AI product.
- hallucination: know the meaning, why it matters, when to use it, and one real example from an AI product.
- grounding: know the meaning, why it matters, when to use it, and one real example from an AI product.
- chat UX: know the meaning, why it matters, when to

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Generative AI & Large Language Models is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on language models, tokens, context window, temperature, top-p. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_23_generative_ai_and_large_
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 23

```
# file: 23_llm_prompt_template.py
def build_prompt(question, context):
    return f"""You are an AI tutor.
Use only this context:
{context}
Question: {question}
Answer clearly with steps."""
print(build_prompt("What is regression?", "Regression predicts continuous
values."))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 23_llm_prompt_template.py
You are an AI tutor.
Use only this context:
Regression predicts continuous values.
Question: What is regression?
Answer clearly with steps.
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Generative AI & Large Language Models, debug these first: language models, tokens, context window, temperature, top-p. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create a safe AI tutor prompt template with role, context, task, output format, and refusal rules.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain language models in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: local LLM concept, Ollama/vLLM concept, FastAPI integration idea?
3. What mistake can happen while using tokens and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Generative AI & Large Language Models in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 23 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain language models with one example.
☐ I can explain tokens with one example.
☐ I can explain context window with one example.
☐ I can explain temperature with one example.
☐ I can explain top-p with one example.
☐ I can explain system/user messages with one example.
☐ I can explain hallucination with one example.
☐ I can explain grounding with one example.
☐ I can explain chat UX with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Prompt Engineering. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24: Prompt Engineering

Write prompts that produce structured, accurate, testable, and useful AI responses.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- write structured prompts
- debug weak prompts
- create rubrics
- design prompt test cases

Topic Map

- role prompts
- task prompts
- context blocks
- examples
- constraints
- output formats
- rubrics
- prompt testing
- failure cases

Tools & Languages

- prompt templates
- JSON output
- evaluation checklist

Deliverable for This Module

Build 20 prompt templates for AI tutoring, code review, lesson generation, and quiz creation.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Write prompts that produce structured, accurate, testable, and useful AI responses.
Important topics: role prompts, task prompts, context blocks, examples, constraints, output formats, rubrics.
Practical skill goal: write structured prompts; debug weak prompts; create rubrics; design prompt test cases.
Tools/languages used here: prompt templates, JSON output, evaluation checklist.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- role prompts: know the meaning, why it matters, when to use it, and one real example from an AI product.
- task prompts: know the meaning, why it matters, when to use it, and one real example from an AI product.
- context blocks: know the meaning, why it matters, when to use it, and one real example from an AI product.
- examples: know the meaning, why it matters, when to use it, and one real example from an AI product.
- constraints: know the meaning, why it matters, when to use it, and one real example from an AI product.
- output formats: know the meaning, why it matters, when to use it, and one real example from an AI product.
- rubrics: know the meaning, why it matters, when to use it, and one real example from an AI product.
- prompt testing: know the meaning, why it matters, when to use it, and one real example from an AI product.
- failure cases: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Prompt Engineering is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on role prompts, task prompts, context blocks, examples, constraints. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_24_prompt_engineering
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 24

```
# file: 24_prompt_builder.py
prompt = {
    "role": "AI ML tutor",
    "task": "explain overfitting",
    "format": ["definition", "example", "fixes", "quiz"],
    "level": "beginner"
}
for key, value in prompt.items():
    print(key, "=>", value)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 24_prompt_builder.py
role => AI ML tutor
task => explain overfitting
format => ['definition', 'example', 'fixes', 'quiz']
level => beginner
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Prompt Engineering, debug these first: role prompts, task prompts, context blocks, examples, constraints. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build 20 prompt templates for AI tutoring, code review, lesson generation, and quiz creation.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain role prompts in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: prompt templates, JSON output, evaluation checklist?
3. What mistake can happen while using task prompts and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Prompt Engineering in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 24 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain role prompts with one example.
- ☐ I can explain task prompts with one example.
- ☐ I can explain context blocks with one example.
- ☐ I can explain examples with one example.
- ☐ I can explain constraints with one example.
- ☐ I can explain output formats with one example.
- ☐ I can explain rubrics with one example.
- ☐ I can explain prompt testing with one example.
- ☐ I can explain failure cases with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: RAG & Vector Databases. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25: RAG & Vector Databases

Ground AI answers in documents using retrieval augmented generation.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- design RAG flow
- chunk documents
- retrieve relevant context
- reduce hallucination

Topic Map

- RAG pipeline
- chunking
- embedding
- vector search
- retrieval
- reranking
- context assembly
- citations
- answer generation

Tools & Languages

- FAISS concept
- pgvector concept
- Chroma concept
- FastAPI

Deliverable for This Module

Build a small RAG knowledge base from AI course notes and answer questions using retrieved chunks.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Ground AI answers in documents using retrieval augmented generation.
Important topics: RAG pipeline, chunking, embedding, vector search, retrieval, reranking, context assembly.
Practical skill goal: design RAG flow; chunk documents; retrieve relevant context; reduce hallucination.
Tools/languages used here: FAISS concept, pgvector concept, Chroma concept, FastAPI.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- RAG pipeline: know the meaning, why it matters, when to use it, and one real example from an AI product.
- chunking: know the meaning, why it matters, when to use it, and one real example from an AI product.
- embedding: know the meaning, why it matters, when to use it, and one real example from an AI product.
- vector search: know the meaning, why it matters, when to use it, and one real example from an AI product.
- retrieval: know the meaning, why it matters, when to use it, and one real example from an AI product.
- reranking: know the meaning, why it matters, when to use it, and one real example from an AI product.
- context assembly: know the meaning, why it matters, when to use it, and one real example from an AI product.
- citations: know the meaning, why it matters, when to use it, and one real example from an AI product.
- answer generation: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

RAG & Vector Databases is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on RAG pipeline, chunking, embedding, vector search, retrieval. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_25_rag_and_vector_databases
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 25

```
# file: 25_simple_rag.py
docs = ["Regression predicts numbers.", "Classification predicts classes.",
"KMeans finds clusters."]
query = "How do I predict a number?"
match = [d for d in docs if "predicts numbers" in d]
context = match[0] if match else "No context found"
print("retrieved context:", context)
print("answer: Use regression when the target is a continuous number.")
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 25_simple_rag.py
retrieved context: Regression predicts numbers.
answer: Use regression when the target is a continuous number.
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For RAG & Vector Databases, debug these first: RAG pipeline, chunking, embedding, vector search, retrieval. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build a small RAG knowledge base from AI course notes and answer questions using retrieved chunks.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain RAG pipeline in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: FAISS concept, pgvector concept, Chroma concept?
3. What mistake can happen while using chunking and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about RAG & Vector Databases in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 25 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain RAG pipeline with one example.
- ☐ I can explain chunking with one example.
- ☐ I can explain embedding with one example.
- ☐ I can explain vector search with one example.
- ☐ I can explain retrieval with one example.
- ☐ I can explain reranking with one example.
- ☐ I can explain context assembly with one example.
- ☐ I can explain citations with one example.
- ☐ I can explain answer generation with one example.

Practical Checklist

- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Fine-tuning, PEFT & Custom Models. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26: Fine-tuning, PEFT & Custom Models

Understand when and how to adapt models using datasets, fine-tuning, LoRA, and evaluation.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- prepare training data
- identify fine-tuning cases
- create eval prompts
- avoid bad data

Topic Map

- fine-tuning need
- dataset quality
- instruction data
- JSONL format
- train/validation split
- LoRA concept
- overfitting risk
- evaluation set

Tools & Languages

- JSONL
- Hugging Face concept
- PEFT/LoRA concept

Deliverable for This Module

Prepare a 50-row instruction dataset for an AI tutor and define an evaluation checklist before training.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Understand when and how to adapt models using datasets, fine-tuning, LoRA, and evaluation.
Important topics: fine-tuning need, dataset quality, instruction data, JSONL format, train/validation split, LoRA concept, overfitting risk.
Practical skill goal: prepare training data; identify fine-tuning cases; create eval prompts; avoid bad data.
Tools/languages used here: JSONL, Hugging Face concept, PEFT/LoRA concept.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- fine-tuning need: know the meaning, why it matters, when to use it, and one real example from an AI product.
- dataset quality: know the meaning, why it matters, when to use it, and one real example from an AI product.
- instruction data: know the meaning, why it matters, when to use it, and one real example from an AI product.
- JSONL format: know the meaning, why it matters, when to use it, and one real example from an AI product.
- train/validation split: know the meaning, why it matters, when to use it, and one real example from an AI product.
- LoRA concept: know the meaning, why it matters, when to use it, and one real example from an AI product.
- overfitting risk: know the meaning, why it matters, when to use it, and one real example from an AI product.
- evaluation set: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Fine-tuning, PEFT & Custom Models is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on fine-tuning need, dataset quality, instruction data, JSONL format, train/validation split. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_26_fine-tuning,_peft_and_cu
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 26

```
# file: 26_training_data_format.py
example = {
    "instruction": "Explain precision and recall",
    "input": "beginner student",
    "output": "Precision checks correctness of positive predictions; recall
checks how many positives were found."
}
print(example)
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 26_training_data_format.py
{'instruction': 'Explain precision and recall', 'input': 'beginner
student', 'output': 'Precision checks correctness of positive predictions;
 recall checks how many positives were found.'}
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Fine-tuning, PEFT & Custom Models, debug these first: fine-tuning need, dataset quality, instruction data, JSONL format, train/validation split. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Prepare a 50-row instruction dataset for an AI tutor and define an evaluation checklist before training.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain fine-tuning need in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: JSONL, Hugging Face concept, PEFT/LoRA concept?
3. What mistake can happen while using dataset quality and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Fine-tuning, PEFT & Custom Models in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 26 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain fine-tuning need with one example.
☐ I can explain dataset quality with one example.
☐ I can explain instruction data with one example.
☐ I can explain JSONL format with one example.
☐ I can explain train/validation split with one example.
☐ I can explain LoRA concept with one example.
☐ I can explain overfitting risk with one example.
☐ I can explain evaluation set with one example.
☐ I ran the code example and compared the output screen.

Practical Checklist

☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: AI Agents & Tool Calling. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27: AI Agents & Tool Calling

Design AI systems that choose tools, call functions, remember context, and complete tasks safely.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- design tool schema
- route tasks
- validate tool outputs
- avoid unsafe automation

Topic Map

- agent loop
- planner
- tool router
- function calling
- memory
- state
- guardrails
- human approval
- audit logs

Tools & Languages

- Python functions
- FastAPI tools
- JSON schema concept
- Redis memory concept

Deliverable for This Module

Build an AI learning assistant design that can search lessons, generate quizzes, and explain code safely.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Design AI systems that choose tools, call functions, remember context, and complete tasks safely.
Important topics: agent loop, planner, tool router, function calling, memory, state, guardrails.
Practical skill goal: design tool schema; route tasks; validate tool outputs; avoid unsafe automation.
Tools/languages used here: Python functions, FastAPI tools, JSON schema concept, Redis memory concept.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- agent loop: know the meaning, why it matters, when to use it, and one real example from an AI product.
- planner: know the meaning, why it matters, when to use it, and one real example from an AI product.
- tool router: know the meaning, why it matters, when to use it, and one real example from an AI product.
- function calling: know the meaning, why it matters, when to use it, and one real example from an AI product.
- memory: know the meaning, why it matters, when to use it, and one real example from an AI product.
- state: know the meaning, why it matters, when to use it, and one real example from an AI product.
- guardrails: know the meaning, why it matters, when to use it, and one real example from an AI product.
- human approval: know the meaning, why it matters, when to use it, and one real example from an AI product.
- audit logs: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

AI Agents & Tool Calling is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on agent loop, planner, tool router, function calling, memory. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_27_ai_agents_and_tool_calli
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 27

```
# file: 27_agent_router.py
def route_tool(user_text):
    text = user_text.lower()
    if "weather" in text:
        return "weather_tool"
    if "course" in text:
        return "course_search_tool"
    return "chat_response"

for msg in ["find AI course", "hello", "weather today"]:
    print(msg, "->", route_tool(msg))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 27_agent_router.py
find AI course -> course_search_tool
hello -> chat_response
weather today -> weather_tool
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For AI Agents & Tool Calling, debug these first: agent loop, planner, tool router, function calling, memory. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Build an AI learning assistant design that can search lessons, generate quizzes, and explain code safely.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain agent loop in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: Python functions, FastAPI tools, JSON schema concept?
3. What mistake can happen while using planner and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about AI Agents & Tool Calling in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 27 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

☐ I can explain agent loop with one example.
☐ I can explain planner with one example.
☐ I can explain tool router with one example.
☐ I can explain function calling with one example.
☐ I can explain memory with one example.
☐ I can explain state with one example.
☐ I can explain guardrails with one example.
☐ I can explain human approval with one example.
☐ I can explain audit logs with one example.

Practical Checklist

☐ I ran the code example and compared the output screen.
☐ I completed the mini project and wrote a README.
☐ I committed the work to GitHub.
☐ I attempted written exam and coding practice.
☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: MLOps, APIs & Deployment. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28: MLOps, APIs & Deployment

Deploy models as reliable services with monitoring, versioning, security, and automation.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- serve a model API
- create request/response schema
- log predictions
- plan monitoring

Topic Map

- model serving
- FastAPI
- Streamlit/Gradio
- Docker
- CI/CD
- monitoring
- model versioning
- latency
- scaling
- rollback

Tools & Languages

- FastAPI
- Docker
- Uvicorn
- joblib
- GitHub Actions concept

Deliverable for This Module

Deploy a model API with FastAPI, add health route, predict route, README, and sample request screenshots.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Deploy models as reliable services with monitoring, versioning, security, and automation.
Important topics: model serving, FastAPI, Streamlit/Gradio, Docker, CI/CD, monitoring, model versioning.
Practical skill goal: serve a model API; create request/response schema; log predictions; plan monitoring.
Tools/languages used here: FastAPI, Docker, Uvicorn, joblib, GitHub Actions concept.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- model serving: know the meaning, why it matters, when to use it, and one real example from an AI product.
- FastAPI: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Streamlit/Gradio: know the meaning, why it matters, when to use it, and one real example from an AI product.
- Docker: know the meaning, why it matters, when to use it, and one real example from an AI product.
- CI/CD: know the meaning, why it matters, when to use it, and one real example from an AI product.
- monitoring: know the meaning, why it matters, when to use it, and one real example from an AI product.
- model versioning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- latency: know the meaning, why it matters, when to use it, and one real example from an AI product.
- scaling: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

MLOps, APIs & Deployment is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on model serving, FastAPI, Streamlit/Gradio, Docker, CI/CD. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_28_mlops,_apis_and_deployme
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 28

```
# file: 28_fastapi_predict.py
# pip install fastapi uvicorn
from fastapi import FastAPI
app = FastAPI()

@app.get("/predict")
def predict(hours: float):
    score = min(100, 40 + hours * 8)
    return {"hours": hours, "predicted_score": round(score, 2)}

# run: uvicorn 28_fastapi_predict:app --reload
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
Browser preview after run:
GET /predict?hours=6
{
  "hours": 6.0,
  "predicted_score": 88.0
}
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For MLOps, APIs & Deployment, debug these first: model serving, FastAPI, Streamlit/Gradio, Docker, CI/CD. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Deploy a model API with FastAPI, add health route, predict route, README, and sample request screenshots.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain model serving in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: FastAPI, Docker, Uvicorn?
3. What mistake can happen while using FastAPI and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about MLOps, APIs & Deployment in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 28 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain model serving with one example.
- ☐ I can explain FastAPI with one example.
- ☐ I can explain Streamlit/Gradio with one example.
- ☐ I can explain Docker with one example.
- ☐ I can explain CI/CD with one example.
- ☐ I can explain monitoring with one example.
- ☐ I can explain model versioning with one example.
- ☐ I can explain latency with one example.
- ☐ I can explain scaling with one example.

Practical Checklist

- ☐ I can explain rollback with one example.
- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Responsible AI, Security & Privacy. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29: Responsible AI, Security & Privacy

Build trustworthy AI with fairness, privacy, safety, auditability, and secure deployment.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- identify AI risks
- write safety checks
- protect user data
- evaluate harmful outputs

Topic Map

- bias
- fairness
- privacy
- PII handling
- model attacks
- prompt injection
- data poisoning
- rate limiting
- audit logs
- human oversight

Tools & Languages

- privacy checklist
- logging
- validation
- moderation concept

Deliverable for This Module

Create an AI safety checklist for student data, prompts, model outputs, logging, and deployment.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Build trustworthy AI with fairness, privacy, safety, auditability, and secure deployment.
Important topics: bias, fairness, privacy, PII handling, model attacks, prompt injection, data poisoning.
Practical skill goal: identify AI risks; write safety checks; protect user data; evaluate harmful outputs.
Tools/languages used here: privacy checklist, logging, validation, moderation concept.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- bias: know the meaning, why it matters, when to use it, and one real example from an AI product.
- fairness: know the meaning, why it matters, when to use it, and one real example from an AI product.
- privacy: know the meaning, why it matters, when to use it, and one real example from an AI product.
- PII handling: know the meaning, why it matters, when to use it, and one real example from an AI product.
- model attacks: know the meaning, why it matters, when to use it, and one real example from an AI product.
- prompt injection: know the meaning, why it matters, when to use it, and one real example from an AI product.
- data poisoning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- rate limiting: know the meaning, why it matters, when to use it, and one real example from an AI product.
- audit logs: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Responsible AI, Security & Privacy is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on bias, fairness, privacy, PII handling, model attacks. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_29_responsible_ai,_security
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 29

```
# file: 29_safety_filter.py
def clean_input(text):
    blocked = ["password", "secret key", "api key"]
    lowered = text.lower()
    if any(word in lowered for word in blocked):
        return "Blocked: sensitive information detected"
    return "Accepted: safe to process"

print(clean_input("my API key is 123"))
print(clean_input("explain regression"))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 29_safety_filter.py
Blocked: sensitive information detected
Accepted: safe to process
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Responsible AI, Security & Privacy, debug these first: bias, fairness, privacy, PII handling, model attacks. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Mini Project

This converts the module into a portfolio-ready practical task.

Project Task

Create an AI safety checklist for student data, prompts, model outputs, logging, and deployment.

Acceptance Criteria

- Create a clear problem statement.
- Use at least one concept from this module.
- Write runnable code or notebook.
- Include input sample and expected output.
- Add screenshot/output preview or saved chart/API response.
- Write a README explaining result, limitation, and next improvement.
- Push code to GitHub using a meaningful commit message.

Submission Files

- README.md with explanation
- src/ or notebook file
- sample input data or generated demo data
- output screenshot or output text file
- short conclusion
- GitHub commit link
- one improvement idea

Real-World Upgrade Idea

Upgrade this project by connecting it with an AIWorldAcademy-style student learning workflow: course data, progress data, assistant response, dashboard, API, or certificate decision.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain bias in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: privacy checklist, logging, validation?
3. What mistake can happen while using fairness and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Responsible AI, Security & Privacy in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 29 Revision Checklist

Before moving forward, confirm every item below is complete.

Knowledge Checklist

- ☐ I can explain bias with one example.
- ☐ I can explain fairness with one example.
- ☐ I can explain privacy with one example.
- ☐ I can explain PII handling with one example.
- ☐ I can explain model attacks with one example.
- ☐ I can explain prompt injection with one example.
- ☐ I can explain data poisoning with one example.
- ☐ I can explain rate limiting with one example.
- ☐ I can explain audit logs with one example.

Practical Checklist

- ☐ I can explain human oversight with one example.
- ☐ I ran the code example and compared the output screen.
- ☐ I completed the mini project and wrote a README.
- ☐ I committed the work to GitHub.
- ☐ I attempted written exam and coding practice.
- ☐ I wrote one interview answer for this module.

Revision Notes

Best concept I learned:
Error I fixed:
Output I generated:
Where this is used in real AI products:
Next improvement I will try:

_____
_____
_____

Next Step

Next module: Capstone Portfolio, Projects & Interview Prep. Keep your code clean because the next topic will build on this foundation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30: Capstone Portfolio, Projects & Interview Prep

Combine every skill into real portfolio projects, final exams, and job-ready preparation.

Theory | Code Lab | Output Preview | Mini Project | Exam | Portfolio

Learning Outcomes

- plan end-to-end AI project
- document results
- present demos
- prepare for interviews

Topic Map

- capstone planning
- 10 projects
- GitHub portfolio
- README
- demo video
- resume bullets
- interview questions
- final certificate checklist

Tools & Languages

- GitHub
- PDF report
- deployed demo
- resume
- LinkedIn

Deliverable for This Module

Complete the final AI learning platform capstone with data, ML model, API, UI, report, and deployment.

How to Study This Module

1) Read the concepts. 2) Type the code yourself. 3) Run the command. 4) Compare the output screen. 5) Fix errors. 6) Commit code to GitHub. 7) Attempt exam and mini project.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Topic-Wise Knowledge Map

This page explains what must be known before moving to the code lab.

What You Must Understand

Core focus: Combine every skill into real portfolio projects, final exams, and job-ready preparation.
Important topics: capstone planning, 10 projects, GitHub portfolio, README, demo video, resume bullets, interview questions.
Practical skill goal: plan end-to-end AI project; document results; present demos; prepare for interviews.
Tools/languages used here: GitHub, PDF report, deployed demo, resume, LinkedIn.
Output standard: code must run, result must be saved or shown, and student must explain what the result means.
Portfolio habit: keep clean folders, clear README, screenshots, output notes, and final conclusion for every module.

Topic-Wise Definitions

- capstone planning: know the meaning, why it matters, when to use it, and one real example from an AI product.
- 10 projects: know the meaning, why it matters, when to use it, and one real example from an AI product.
- GitHub portfolio: know the meaning, why it matters, when to use it, and one real example from an AI product.
- README: know the meaning, why it matters, when to use it, and one real example from an AI product.
- demo video: know the meaning, why it matters, when to use it, and one real example from an AI product.
- resume bullets: know the meaning, why it matters, when to use it, and one real example from an AI product.
- interview questions: know the meaning, why it matters, when to use it, and one real example from an AI product.
- final certificate checklist: know the meaning, why it matters, when to use it, and one real example from an AI product.

Professional Workflow

- Start with a small dataset or example.
- Write the simplest possible code that runs.
- Check output shape, values, and meaning.
- Add validation and error handling.
- Document result in README with screenshot/output.
- Only then improve model, UI, or deployment.

Notebook/Code Run Rule

Every practical page must include: problem statement, input data, code, command to run, expected output, explanation, mistake checks, and a final student task.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Theory Notes

Concepts are written in practical language so students can connect theory with code.

Detailed Explanation

Capstone Portfolio, Projects & Interview Prep is an important part of the AI/ML pipeline because it helps convert raw learning into working systems. The aim is not only to memorize definitions but to understand how the concept is used in real projects.
In this module, focus on capstone planning, 10 projects, GitHub portfolio, README, demo video. For each topic, write one definition, one real use case, one mistake, and one small code experiment.
A professional AI engineer always separates data, code, model, evaluation, deployment, and documentation. This habit makes projects maintainable and production-ready.
Do not jump directly to advanced models. First create a baseline, measure it, understand the errors, then improve step by step. This is the most reliable learning and production method.

Why This Matters

- It teaches the concept behind model behavior.
- It connects theory with measurable output.
- It builds interview-ready explanations.
- It prevents copy-paste learning and weak debugging.
- It helps create clean portfolio documentation.

When You Use It

- Use this knowledge when selecting tools.
- Use it while cleaning and preparing data.
- Use it during model training and evaluation.
- Use it when explaining results to non-technical users.
- Use it when deploying and monitoring AI systems.

Student Notes Box

Write your own notes here: definition, one example, one formula/command if any, one common mistake, and one practical use case.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Practical Workflow

Follow this runbook before writing the full project code.

Step-by-Step Lab Flow

1. Create folder: module_30_capstone_portfolio,_proj
2. Create README.md with problem statement and expected result.
3. Create a small Python file or notebook and keep data small at first.
4. Run code after every change. Never write 100 lines without running.
5. Print important values: shape, sample row, metric, prediction, or output file path.
6. Save screenshot or output copy inside reports/output_notes.md.
7. Commit working version to GitHub with a clear commit message.

Folder Structure

module/
data/
notebooks/
src/
reports/
screenshots/
README.md
requirements.txt

Quality Checklist

- Code runs without hidden manual changes
- Output matches expected preview
- README explains input and output
- No secret keys or private data committed
- Screenshots/output notes are attached
- Result includes one limitation and one improvement

Professional Habit

The best AI/ML learners do not only run models. They document the data, explain the output, test edge cases, and write exactly what decision the model supports.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Code Running Example

Type this code manually, run it, then compare with the output preview on the next page.

Runnable Code Example - Module 30

```
# file: 30_project_tracker.py
projects = ["student score predictor", "course recommender", "RAG tutor"]
completed = [True, False, False]
for name, done in zip(projects, completed):
    status = "done" if done else "pending"
    print(f"{name}: {status}")
print("complete projects:", sum(completed), "/", len(projects))
```

How to Run

1) Save the code with the filename shown. 2) Open terminal in the same folder. 3) Activate your environment if needed. 4) Run: python filename.py. 5) Compare terminal output with expected output.

Before Run Check

- Python installed
- Required package installed
- File name is correct
- Terminal path is correct
- No copy-paste indentation error

After Run Check

- Output visible
- No error traceback
- Result meaning understood
- Screenshot/output note saved
- Code committed to GitHub

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Output Preview Screen

Use this page like a screenshot reference after running your code.

Expected Output / Preview Screen

```
$ python 30_project_tracker.py
student score predictor: done
course recommender: pending
RAG tutor: pending
complete projects: 1 / 3
```

How to Read This Output

- The output proves that the code executed successfully.
- Read every printed line and connect it with the input and logic.
- If your result is different, check package version, random_state, file path, or indentation.
- For model outputs, focus on interpretation, not only the number.
- Save your own output screenshot in the project folder before moving ahead.

Student Output Attachment Area

Attach or paste your own terminal screenshot, notebook output, chart preview, API response, or browser preview here when using this workbook physically or digitally.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Common Mistakes & Debugging

Debugging is a core AI/ML skill. Use this page when code or output is not working.

Common Mistakes

- Running the command from the wrong folder.
- Package not installed inside the active environment.
- Copy-paste indentation error in Python code.
- Expecting the same random output without setting random_state or seed.
- Confusing training accuracy with real validation performance.
- Not checking data shape, missing values, or target column.
- Using production/private data without privacy checks.
- Committing large datasets, model files, or secret keys to GitHub.

Fix / Prevention

- Print current working directory and file path.
- Run pip show package-name in the active environment.
- Format code and check indentation line by line.
- Set random_state=42 for reproducible examples.
- Use train/test split and proper validation metrics.
- Print df.head(), df.info(), df.isna().sum() before modeling.
- Remove private information and use mock/demo data for learning.
- Use .gitignore for data/, models/, .env, and cache files.

Debug Rule

When stuck, do not randomly change code. Read the exact error line, search the variable name, print intermediate values, and make one change at a time.

Module-Specific Debug Focus

For Capstone Portfolio, Projects & Interview Prep, debug these first: capstone planning, 10 projects, GitHub portfolio, README, demo video. Explain the root cause in your own words before asking for help.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Final 10 Portfolio Projects

Complete these 10 projects to become job-ready and certificate-ready.

Projects 1 to 5

1. Student Score Prediction System - regression model + dashboard + report
2. Certificate Eligibility Classifier - classification model + fairness checklist
3. AI Course Recommender - content-based recommendation from student profile
4. Study Doubt Classifier - NLP text classifier for student support routing
5. Learning Analytics Dashboard - charts, insights, and action plan

Projects 6 to 10

6. RAG AI Tutor - retrieve course notes and answer with context
7. Image Preprocessing Lab - computer vision dataset cleaning and preview
8. Traffic/Enrollment Forecasting - time series baseline forecast
9. FastAPI Model Deployment - model API with docs and sample requests
10. AI Safety Monitor - prompt/input checker, audit log, and risk report

Submission Standard for Each Project

- Problem statement and dataset/source explanation
- Clean notebook or src/ Python code
- Screenshot/output preview or API/browser result
- Metrics or success criteria
- README with installation and run commands
- GitHub repository and demo link where possible
- Short limitation and next improvement

Capstone Requirement

Choose one project and upgrade it into a full capstone: data cleaning, ML model, evaluation, API deployment, simple UI/dashboard, safety checklist, final PDF report, and presentation.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Module 30 Exam & Practice

Attempt without looking at notes first. Then check your code and explanations.

Written Exam Questions

1. Explain capstone planning in one simple sentence and one real AI use case.
2. Which tool/language from this module is most important and why: GitHub, PDF report, deployed demo?
3. What mistake can happen while using 10 projects and how will you prevent it?
4. Write the input, process, and output for the code lab in this module.
5. Describe one metric, screenshot, output, or proof that shows your task worked.
6. Write one interview answer about Capstone Portfolio, Projects & Interview Prep in 4 lines.

Coding Practice Task

- Create a new file from scratch.
- Use the module code as reference but change the input values.
- Print a clear output label before every result.
- Add one error check or validation.
- Save output screenshot or paste terminal result.

Viva / Interview Questions

- What problem does this module solve?
- Which data is required?
- What output proves success?
- What can go wrong?
- How will you improve it in production?

Passing Criteria

Pass this module when you can explain the concept, run the code without help, understand the output, fix one error, and submit the mini project with README and screenshot.

 Copyright © 2026 AiWorldAcademy. All rights reserved. Educational use only.

Final Certificate & Interview Readiness Checklist

Use this as the last page before issuing the AI & Machine Learning certificate.

Certificate Eligibility

☐ Completed all 30 modules and revised core theory.
☐ Ran code labs and saved output screenshots/previews.
☐ Completed all module mini projects or equivalent practice tasks.
☐ Completed at least 10 final portfolio projects or selected capstone path.
☐ Created GitHub repositories with clean README files.
☐ Prepared one deployed demo or local demo video.
☐ Understood Python, SQL, data cleaning, ML, DL, NLP, GenAI, RAG, Agents, and MLOps basics.
☐ Prepared resume bullets for AI/ML projects.
☐ Practiced written exam, coding round, and viva questions.
☐ Accepted responsible AI, privacy, and safety guidelines.

Final Interview Topics

- Explain AI vs ML vs deep learning vs generative AI.
- Explain supervised vs unsupervised learning with examples.
- Explain train/test split, overfitting, and evaluation metrics.
- Explain regression, classification, clustering, NLP, computer vision, and RAG.
- Explain how to deploy an ML model using an API.
- Explain AI safety, privacy, bias, and prompt injection risk.

Copyright Notice

Copyright © 2026 AiWorldAcademy. Website: www.aiworldacademy.in. This course PDF is for AiWorldAcademy educational use. Unauthorized selling, copying, republication, or commercial redistribution is not permitted.