AI & MACHINE LEARNING

A friendly path from first ideas to practical models and responsible AI

LEARNSAPPEAL | ORIGINAL COURSE

Learn what AI can do, how machine learning works, and how to build a small project.
No advanced mathematics is required to begin. The ideas start with examples and everyday language, then gradually introduce equations, Python, and model evaluation.

Before you start
Basic computer use is enough for the first modules. Later coding examples assume introductory Python. If you are new to Python, work through a Python introduction course alongside the coding sections. Every new term is explained before it is used.

Course map: Foundations (Modules 1-6), followed by applied practice (Modules 7-12). Includes guided examples, self-checks with answers, three projects, and a short glossary.

How to learn from this course

Go in order. Write down one example from your own life for every new concept. When you reach code, run it and change one input at a time. Do not worry if the equations look unfamiliar: each one is explained in words and paired with a concrete example.

What you will be able to do. Explain the difference between AI and machine learning; describe how examples teach a model; prepare simple data; train a basic classifier; recognize overfitting; choose sensible evaluation measures; understand the purpose of neural networks and language models; and plan a responsible small AI project.

Use the answer key well. Try the questions first, even if your answer is incomplete. The solutions are short explanations, not the only acceptable wording. For projects, focus on asking a clear question and checking whether the data can answer it.

Table of contents

How to learn from this course 2
1. What do AI and machine learning mean? 3
2. Data, examples, and features 3
3. How a model learns 4
4. A first model in Python 4
5. Check whether a model is any good 5
6. Overfitting and better data habits 6
7. Trees, ensembles, and feature engineering 6
8. Unsupervised learning and grouping 7
9. Neural networks and modern AI 7
10. Train, improve, and compare models 8
11. Responsible use, fairness, and privacy 9
12. Put a model into a real workflow 9
Guided coding lab: classify a flower 10
Project 1: a course catalog helper 10
Project 2: predict bike demand 10
Project 3: review an AI text assistant 11
Glossary 11
Keep learning 11

FOUNDATIONS

1. What do AI and machine learning mean?

Artificial intelligence

Artificial intelligence (AI) is a broad name for computer systems that perform tasks we associate with abilities such as recognizing patterns, understanding language, making recommendations, or planning. Some AI follows rules written by people. Other AI learns patterns from examples.

Machine learning

Machine learning (ML) is one way to make AI. Instead of writing every rule by hand, we give an algorithm examples and a goal. It adjusts an internal model so its answers become more useful on similar examples. A model is the learned pattern; an algorithm is the recipe used to learn it.

Generative AI

Generative AI creates new content such as text, images, audio, or code. A language model learns patterns in text and uses the prompt plus context to produce a likely continuation. It can be useful and fluent while still making things up, so important answers need checking.

A simple example

A spam filter is trained with emails marked spam or not spam. It learns clues such as words, sender patterns, and links. It then estimates whether a new email looks like examples from either group. The filter is not thinking like a person; it is applying patterns learned from data.

Check your understanding

- How is machine learning different from writing every rule by hand?
- Is a language model guaranteed to tell the truth?
- What is the model in a spam filter?

Suggested answers

1. It learns patterns from examples instead of relying only on rules explicitly written by a person.

2. No. Check important claims against reliable evidence.

3. The learned system that uses email clues to estimate the class of a new message.

2. Data, examples, and features

Rows and columns

A dataset is often a table. Each row is one example, such as one house or one email. Each column is a property recorded about that example. We call an input property a feature. A target is the answer we want to predict, such as the sale price or whether an email is spam.

Good questions need good labels

A label is the target value provided for a training example. Labels might come from human review, measurements, or records. If labels are inconsistent, outdated, or biased, a model can learn those problems. Before modeling, write down what each label means and how it was collected.

Training, validation, and test examples

Training examples teach the model. Validation examples help compare choices during development. Test examples are held back to estimate performance on new cases after decisions are finished. If examples from the same person or time period appear in both training and test, results may look better than real use.

Features must exist at prediction time

Suppose we predict whether a package will arrive late at the moment it is shipped. A feature recorded after delivery cannot honestly help with that prediction. It leaks future information. Ask when each feature is known, not only whether it correlates with the answer.

Check your understanding

- In a table of house sales, what could be a feature and what could be the target?
- Why keep test examples separate until the end?
- What is data leakage?

Suggested answers

1. Size could be a feature; sale price could be the target.
2. To estimate how well the chosen model works on examples it did not learn from.
3. Using information in training or evaluation that would not be available in the real prediction setting.

3. How a model learns

Prediction and error

A model makes a prediction, compares it with the known answer during training, and gets an error score. A loss function is a way to measure that error. Training changes the model so its average loss on training examples becomes smaller.

A gentle view of linear regression

Imagine predicting a taxi fare from trip distance. A simple model might say: fare = starting fee + (price per kilometer x distance). Training finds values for the starting fee and price per kilometer that fit past trips reasonably well. This is linear regression.

Classification

Classification predicts a category, such as spam/not spam or plant type. Some models also produce a probability-like score. A threshold turns a score into an action. If a bank wants to review risky transactions, it may choose a threshold based on how many cases staff can inspect.

The training loop in simple terms

The model starts with adjustable settings called parameters. It predicts, measures errors, and nudges parameters to reduce those errors. This repeats many times. The method for nudging parameters is optimization. You can understand the process before learning the calculus behind it.

Check your understanding

- What does a loss function measure?
- Is predicting a fare or a spam label regression or classification?
- What is a threshold used for?

Suggested answers

1. How far predictions are from the known answers according to a chosen scoring rule.
2. Fare is regression; spam is classification.
3. It turns a model score into a category or decision.

4. A first model in Python

Python ingredients

A common workflow uses a table library to inspect data and a machine-learning library to split, train, and evaluate. X is the input data; y is the target. fit learns from examples. predict applies what was learned to new examples.

A tiny readable example

This example creates a training set with short messages and labels. A text vectorizer converts words into numbers because most algorithms work with numbers. A logistic regression classifier learns a simple decision boundary. Real projects need more examples and careful evaluation.

Why a pipeline helps

A pipeline keeps data preparation and the model together. This makes it easier to apply the same steps at training and prediction time. It also helps prevent accidentally learning preprocessing settings from the test examples.

Small experiments

Change one thing at a time: add examples, try different text, or compare a baseline. Keep a note of what changed and what result you observed. A high score on a handful of toy examples does not mean a system is ready for real use.

Check your understanding

- What do fit and predict mean?
- Why does text need to be converted for many models?
- Why is a toy example not proof of real-world quality?

Suggested answers

1. fit learns patterns from examples; predict applies the learned model to new inputs.
2. Many algorithms expect numeric features.
3. It is too small and simple to represent the variety and mistakes found in real data.

5. Check whether a model is any good

Start with a baseline

A baseline is a simple reference model. For a classification task, it might always predict the most common class. For a price task, it might always predict the average price. A new model should improve on a useful baseline in ways that matter.

Accuracy and its limits

Accuracy is the share of predictions that are correct. If only 1 out of 100 transactions is fraudulent, a model that always says not fraud has 99% accuracy but catches none of the fraud. The number alone hides the important failure.

Precision and recall

Precision asks: of the items we flagged, how many were truly positive? Recall asks: of all positive items, how many did we find? In a medical screening example, missing a real case may be costly, so recall may matter. For a noisy fraud alert team, precision may matter too.

A fair comparison

Evaluate models on the same untouched test set and report more than one useful metric. Also inspect errors. Are mistakes concentrated in one type of example or user group? Can a human safely review uncertain cases?

Check your understanding

- Why can accuracy be misleading when positive cases are rare?
- What does high recall mean?
- Why inspect examples the model gets wrong?

Suggested answers

1. A model can predict the common negative class every time and still miss all positives.
2. It finds a large share of the true positive cases.
3. Errors can reveal data problems, blind spots, or groups that need separate attention.

6. Overfitting and better data habits

Learning the examples too closely

Overfitting happens when a model learns details that work on its training examples but do not carry over to new ones. A model that memorizes practice answers may perform poorly on a new exam. Compare training performance with validation performance to spot warning signs.

Ways to reduce overfitting

Use more representative examples, simplify the model, limit its complexity, or stop training when validation performance stops improving. Regularization is a family of techniques that discourages unnecessarily complicated models. These methods help, but they cannot fix a fundamentally unrepresentative dataset.

Missing values and unusual values

Real datasets often have blanks, typos, and extreme values. Decide what a blank means before filling it. A median can be a sensible simple fill for a numeric field, but missingness may itself carry information. Always fit fill rules using training data only.

A practical checklist

Check duplicates, units, label consistency, privacy, and whether features are known at prediction time. Keep a copy of the original data. Write down the split method, preprocessing, and random seed so a future run can be compared.

Check your understanding

- What is overfitting in everyday language?
- Does regularization fix incorrect labels?
- Why should fill values be learned from training data only?

Suggested answers

1. Memorizing quirks of the practice examples instead of learning patterns that transfer.

2. No. Fix label quality and understand its source.

3. Using test data to calculate fills would let information from the evaluation set influence training.

APPLIED PRACTICE

7. Trees, ensembles, and feature engineering

Decision trees

A decision tree asks a sequence of questions, such as whether income is above a cutoff. Trees can represent nonlinear patterns and are easy to picture. A very deep tree can memorize examples, so limits on depth or leaf size help control complexity.

Forests and boosting

A random forest combines many trees to make a more stable prediction. Gradient boosting builds a sequence of small models, each trying to improve on earlier mistakes. These methods often work well with table-shaped business data, but still need thoughtful validation and tuning.

Feature engineering

Feature engineering means choosing or constructing useful inputs. A date might become day of week, month, or time since the last event. Categories can be turned into indicator columns. Add only features that make sense, are available when needed, and do not reveal the answer.

Compare fairly

Try a simple linear model and a tree-based model. Use the same training and validation plan. More complex is not automatically better: compare quality, speed, stability, explanation needs, and maintenance cost.

Check your understanding

- How does a decision tree make a prediction?
- What is one difference between a forest and boosting?
- Name one rule for a good feature.

Suggested answers

1. By following learned questions or splits until it reaches an answer.
2. A forest combines trees in parallel; boosting adds models in sequence to correct earlier errors.
3. It is useful, valid at prediction time, and does not leak the target.

8. Unsupervised learning and grouping

Learning without target labels

Unsupervised learning looks for structure in examples that do not come with a target answer. It can help organize customers, summarize many measurements, or find unusual records. A discovered pattern still needs a human interpretation.

Clustering

Clustering groups examples that look similar according to a chosen distance. K-means is a common method that groups points around centers. It is sensitive to feature scale and tends to favor round groups. The group labels are not automatically meaningful categories.

Dimensionality reduction

A dataset with many columns can be hard to view. Principal component analysis (PCA) combines columns into a smaller set of directions that preserve much of the overall variation. This may make data easier to visualize, but it can also discard information useful for a specific prediction.

Anomaly detection

Anomaly detection assigns unusualness scores to examples. A rare score is not automatically fraud or an error. It could be a legitimate new pattern. Decide how a person will review alerts and measure false alarms as well as discoveries.

Check your understanding

- Does a cluster name explain itself?
- Why scale features before many distance-based methods?
- Is an anomaly score the same as a probability of fraud?

Suggested answers

1. No. A person must inspect what the group represents and whether it is useful.
2. Otherwise a large-unit feature can dominate the distance calculation.
3. No. It indicates unusualness unless specifically calibrated to a probability.

9. Neural networks and modern AI

Neural network building blocks

A neural network passes numbers through layers of adjustable connections. An activation function adds a bend or nonlinearity so the network can learn more than a straight-line relationship. Training adjusts the connections to reduce errors.

Images, text, and embeddings

Convolutional networks use small filters that scan images for local patterns. Language models split text into tokens and represent tokens as vectors called embeddings. An embedding is a learned numeric representation that can place related items near one another.

Transformers and attention

Transformers use attention to let each token combine information from other tokens in a sequence. This helps the model use context. A large language model is commonly trained to predict the next token over many examples of text. The prediction skill can support summarization, drafting, and question answering, but does not guarantee factual accuracy.

Prompting and retrieval

A prompt supplies a task and context. Retrieval adds relevant documents to the context before asking a model to answer. This can help ground responses in a selected collection, but retrieval quality, permissions, and citations still need to be checked.

Check your understanding

- What does an activation function add to a neural network?
- What does attention help a transformer do?
- Does retrieval guarantee an answer is correct?

Suggested answers

1. Nonlinearity, which lets stacked layers learn more varied patterns.
2. Use information from other tokens in the sequence.
3. No. It can provide better evidence, but retrieval and generated answers can both fail.

10. Train, improve, and compare models

A validation loop

Build a training pipeline, choose a validation split, fit the model, and calculate metrics. Tune settings such as tree depth or regularization using validation data. Keep the final test data untouched until you have selected a model and a decision rule.

Cross-validation

Cross-validation repeats training and validation across several folds. It reduces dependence on one lucky split. If examples are related by person or time, ordinary random folds can still leak; choose group-aware or time-aware folds instead.

Hyperparameters

Parameters are learned from examples; hyperparameters are choices set before or around training. Examples include tree depth, number of trees, and regularization strength. Search a small sensible range first and record every run. A larger search is not useful when the evaluation design is flawed.

Reproducibility

Save the data definition, split strategy, feature list, code version, package versions, random seed, metric results, and selected model. Reproducibility helps explain why a later model behaves differently and makes deployment safer.

Check your understanding

- Which data should be used to tune hyperparameters?
- Why can random cross-validation be wrong for a time prediction task?
- Name three details to record for a reproducible experiment.

Suggested answers

1. Training and validation data, never the sealed final test set.

2. It may train on future records and validate on past records, unlike real deployment.

3. Split strategy, data version, features, code/package version, seed, metrics, model artifact.

11. Responsible use, fairness, and privacy

Start from a human need

Ask what problem the model should help with, who benefits, who could be harmed, and what happens when it is wrong. Sometimes a checklist, search tool, or clearer process is a better solution than machine learning.

Fairness and uneven errors

A model can work differently across groups because the training examples or labels differ. Compare error rates and outcomes across relevant groups when it is appropriate and lawful. Different fairness measures may conflict, so the choice needs context and consultation, not a single magic number.

Privacy and security

Collect only the data needed for a clear purpose. Protect personal information, limit access, and decide how long it is kept. Treat user-provided text and model-generated text as data that may be wrong or unsafe. Do not send private information to a service unless you understand the applicable settings and terms.

Human review and documentation

For consequential decisions, provide a meaningful review or appeal path. Document intended use, test results, known limitations, data sources, and a responsible owner. A system should have a plan for mistakes, not only a demonstration of success.

Check your understanding

- What should you ask before deciding to use machine learning?
- Why compare errors across groups?
- What should documentation include?

Suggested answers

1. Whether the problem is well-defined and whether a model is better than a simpler solution.

2. A strong overall score can hide worse results for some people.

3. Purpose, data, evaluation, limits, owner, and what to do when the system is wrong.

12. Put a model into a real workflow

A model is one part of a system

A deployed system also includes data collection, preprocessing, model files, application code, monitoring, and people using the result. The training and prediction steps must apply the same feature definitions. Package preprocessing with the model so a new example is handled consistently.

Monitoring after launch

Watch for missing fields, unusual feature values, changes in prediction rates, slow responses, and user complaints. When reliable labels arrive, compare real outcomes with predictions. A change in incoming data is a reason to investigate, not automatic proof that a model must be retrained.

Small safe launches

Start with a limited rollout or a human-in-the-loop trial. Keep a simple fallback and a rollback plan. Assign an owner who can respond to alerts. Record issues and use them to update tests and documentation.

Final project: course recommendation helper

Build a simple tool that recommends courses from a small catalog. Begin with category filters or transparent rules. If you have enough labeled examples, compare a model that predicts a learner's chosen topic. Explain why each recommendation appeared, let users change preferences, and measure whether recommendations are useful.

Check your understanding

- What needs to stay consistent between training and prediction?
- Is data drift always a reason to retrain immediately?
- Why start with a limited launch?

Suggested answers

1. The feature definitions and preprocessing steps.

2. No. Investigate the change and its effect first.

3. To find problems with less impact and keep a fallback available.

Guided coding lab: classify a flower

This short example uses a small dataset included with scikit-learn. It trains a simple model and reports accuracy on examples held out during the split. It is a practice workflow, not a claim that accuracy is the right metric for every problem.

TRY IT IN PYTHON

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=7, stratify=y)

model = make_pipeline(
    StandardScaler(),
    LogisticRegression(max_iter=1000))
model.fit(X_train, y_train)
predicted = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, predicted))
```

What is happening? The split reserves some rows for a final check. The pipeline learns scaling from training rows and applies it consistently. Logistic regression learns a boundary between flower classes. Try changing random_state or replacing the model, then compare results. Do not use this tiny dataset to make decisions about real plants.

Project 1: a course catalog helper

Make a small catalog with course title, topic, level, and description. Start with filters a learner can understand. Add a recommendation rule such as matching selected topics. Ask a few learners whether results are useful, record feedback, and revise the design. Include clear empty and no-match states.

Project 2: predict bike demand

Use a public or toy dataset with date, weather, and rental count. Predict demand with a mean baseline and a linear model. Hold out later dates for testing, plot errors over time, and explain which features are known before the prediction is made. Treat the model as a planning aid, not a guarantee.

Project 3: review an AI text assistant

Create a set of 20 sample questions for a study assistant. Label which answers need a source, which should say they do not know, and which contain sensitive information. Try prompts and retrieval examples, record errors, and propose a human review and privacy plan. Do not use private learner data in the prototype.

Glossary

Term	Meaning
AI	A broad field of systems that perform tasks associated with intelligent behavior.
Feature	An input property used by a model.
Label / target	The answer a supervised model learns to predict.
Loss	A score describing prediction error during training.
Model	A learned pattern used to produce predictions or content.
Overfitting	Learning training-example quirks that do not transfer well.
Precision	Share of flagged positive cases that are actually positive.
Recall	Share of all positive cases that were found.
Training data	Examples used to learn model parameters.
Validation data	Examples used to compare models and settings.
Test data	Held-back examples used for a final evaluation.
Threshold	A cutoff that turns a model score into an action.

Keep learning

These official learning materials provide alternate explanations and help with practical follow-up. Check current library documentation before running examples because software versions change.

- Google for Developers. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/
- scikit-learn. Getting Started and model pipelines. https://scikit-learn.org/stable/getting_started
- Vaswani et al. (2017). Attention Is All You Need. https://arxiv.org/abs/1706.03762
- NIST. Artificial Intelligence Risk Management Framework 1.0. https://doi.org/10.6028/NIST.AI.100-1

You have completed the course when you can:
Explain the basic idea of AI and machine learning; identify features and targets; split data carefully; train and compare a simple model; describe a model's errors and limits; and outline how to use an AI system with privacy, human review, and monitoring in mind.