Machine Learning Training Module

A Comprehensive Guide for Beginners

Prepared by xAI Training Team May 07, 2025

Downloadable PDF version available for training purposes.

**Contents**

1. [Introduction to Machine Learning](#_bookmark0) 2
   1. [Why Study Machine Learning?](#_bookmark1) 2
   2. [Setting Up the Environment](#_bookmark2) 2
2. [Supervised Learning](#_bookmark3) 2
   1. [Linear Regression Example](#_bookmark4) 2
3. [Classification](#_bookmark5) 3
   1. [Logistic Regression Example](#_bookmark6) 3
4. [Unsupervised Learning](#_bookmark7) 3
   1. [K-Means Clustering Example](#_bookmark8) 3
5. [Model Evaluation](#_bookmark9) 4
   1. [Cross-Validation Example](#_bookmark10) 4
6. [Feature Engineering](#_bookmark11) 4
   1. [Standard Scaling Example](#_bookmark12) 5
7. [Decision Trees](#_bookmark13) 5
   1. [Decision Tree Classifier Example](#_bookmark14) 5
8. [Neural Networks (Introduction)](#_bookmark15) 5
   1. [Simple Neural Network Example](#_bookmark16) 6
9. [Hyperparameter Tuning](#_bookmark17) 6
   1. [Grid Search Example](#_bookmark18) 6
10. [Conclusion](#_bookmark19) 7
11. [References](#_bookmark20) 7

1

# Introduction to Machine Learning

Machine Learning (ML) is a field of artificial intelligence that enables systems to learn from data and improve over time. This module introduces ML concepts— supervised and unsupervised learning, model evaluation, and more—with Python examples using scikit-learn.

## Why Study Machine Learning?

* + - **Automation**: Enables predictive and decision-making systems.
    - **Applications**: Used in healthcare, finance, and autonomous systems.
    - **Innovation**: Drives advancements in AI and data science.

## Setting Up the Environment

Install Python and libraries like scikit-learn, numpy, and pandas. Use a Jupyter Notebook or IDE like PyCharm. Install dependencies with:

pip install scikit-learn numpy pandas matplotlib

1

# Supervised Learning

Supervised learning uses labeled data to predict outcomes.

## Linear Regression Example

Predict a continuous variable using linear regression.

**import** numpy as np

**from** sklearn.linear\_model **import** LinearRegression

**import** matplotlib.pyplot as plt

# Sample data

X = np.array([[1], [2], [3], [4], [5]])

y = np.array([2, 4, 5, 4, 5])

# Train model

model = LinearRegression() model.fit(X, y)

# Predict

y\_pred = model.predict(X)

# Plot

plt.scatter(X, y, color=’blue’, label=’Data’) plt.plot(X, y\_pred, color=’red’, label=’Fit’) plt.xlabel(’X’)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

plt.ylabel(’y’) plt.legend()

plt.savefig(’linear\_regression.png’)

21

22

## Explanation:

* LinearRegression: Fits a linear model.
* model.fit: Trains the model on input data.
* plt.savefig: Saves the plot as an image.

# Classification

Classification predicts discrete labels.

## Logistic Regression Example

Classify data using logistic regression.

**from** sklearn.datasets **import** load\_iris

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.model\_selection **import** train\_test\_split

# Load data

iris = load\_iris()

X, y = iris.data, iris.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size

=0.2, random\_state=42)

# Train model

model = LogisticRegression(max\_iter=200) model.fit(X\_train, y\_train)

# Evaluate

accuracy = model.score(X\_test, y\_test)

**print**(f”Accuracy: {accuracy:.2f}”)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

# Unsupervised Learning

Unsupervised learning finds patterns in unlabeled data.

## K-Means Clustering Example

Group data into clusters.

1

**from** sklearn.cluster **import** KMeans

**import** matplotlib.pyplot as plt

# Sample data

X = np.array([[1, 2], [1.5, 1.8], [5, 8], [8, 8], [1, 0.6], [9,

11]])

# Train model

kmeans = KMeans(n\_clusters=2, random\_state=42) kmeans.fit(X)

# Plot clusters

plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels\_, cmap=’viridis’)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_ [:, 1], s=200, c=’red’, marker=’X’)

plt.xlabel(’Feature 1’)

plt.ylabel(’Feature 2’) plt.savefig(’kmeans\_clustering.png’)

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

# Model Evaluation

Evaluate models to ensure reliability.

## Cross-Validation Example

Use cross-validation to assess model performance.

**from** sklearn.datasets **import** load\_iris

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.ensemble **import** RandomForestClassifier

# Load data

iris = load\_iris()

X, y = iris.data, iris.target

# Train model

model = RandomForestClassifier(random\_state=42) scores = cross\_val\_score(model, X, y, cv=5)

# Print results

**print**(f”Cross-validation scores: {scores}”)

**print**(f”Average accuracy: {scores.mean():.2f}”)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# Feature Engineering

Feature engineering improves model performance by transforming data.

## Standard Scaling Example

Normalize features using StandardScaler.

**from** sklearn.preprocessing **import** StandardScaler

**import** numpy as np

# Sample data

X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])

# Scale features

scaler = StandardScaler() X\_scaled = scaler.fit\_transform(X)

# Print results **print**(”Scaled data:”) **print**(X\_scaled)

1

2

3

4

5

6

7

8

9

10

11

12

13

# Decision Trees

Decision trees make decisions based on feature splits.

## Decision Tree Classifier Example

Classify data using a decision tree.

**from** sklearn.datasets **import** load\_iris

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.model\_selection **import** train\_test\_split

# Load data

iris = load\_iris()

X, y = iris.data, iris.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size

=0.2, random\_state=42)

# Train model

model = DecisionTreeClassifier(random\_state=42) model.fit(X\_train, y\_train)

# Evaluate

accuracy = model.score(X\_test, y\_test)

**print**(f”Accuracy: {accuracy:.2f}”)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

# Neural Networks (Introduction)

Neural networks model complex patterns.

## Simple Neural Network Example

Use scikit-learn’s MLPClassifier for a basic neural network.

**from** sklearn.neural\_network **import** MLPClassifier

**from** sklearn.datasets **import** load\_iris

**from** sklearn.model\_selection **import** train\_test\_split

# Load data

iris = load\_iris()

X, y = iris.data, iris.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size

=0.2, random\_state=42)

# Train model

model = MLPClassifier(hidden\_layer\_sizes=(10,), max\_iter=1000, random\_state=42)

model.fit(X\_train, y\_train)

# Evaluate

accuracy = model.score(X\_test, y\_test)

**print**(f”Accuracy: {accuracy:.2f}”)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

# Hyperparameter Tuning

Optimize model performance by tuning hyperparameters.

## Grid Search Example

Use GridSearchCV to find the best parameters.

**from** sklearn.model\_selection **import** GridSearchCV

**from** sklearn.svm **import** SVC

**from** sklearn.datasets **import** load\_iris

# Load data

iris = load\_iris()

X, y = iris.data, iris.target

# Define parameter grid

param\_grid = {’C’: [0.1, 1, 10], ’kernel’: [’linear’, ’rbf’]}

# Perform grid search

model = GridSearchCV(SVC(), param\_grid, cv=5) model.fit(X, y)

# Print results

**print**(f”Best parameters: {model.best\_params\_}”)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

**print**(f”Best score: {model.best\_score\_:.2f}”)

# Conclusion

This module covers machine learning fundamentals—supervised and unsuper- vised learning, model evaluation, feature engineering, and neural networks. Prac- tice these examples and explore advanced topics like deep learning with Tensor- Flow or PyTorch.

# References

* Scikit-learn Documentation: [https://scikit-learn.org](https://scikit-learn.org/)
* Machine Learning, Tom Mitchell