AIM learning

# Machine Learning with Python

## Expected skill level: Intermediate

This material is intended for someone who may know a bit of Python, but for whom machine learning is new. The guide is targeted for Windows, using Python virtual environments and Jupyter Notebook.

## What does this learning package include?

During the exercise, you will do four things:

1. Create a folder for the Python project
2. Create and activate a virtual environment
3. Install required libraries
4. Run three very simple machine learning examples in Jupyter Notebook

The algorithms used are:

| Algorithm | What is it used for? |
|---|---|
| Linear regression | Predict a numerical value |
| KNN classifier | Classify based on nearest examples |
| Decision tree | Perform classification using simple rules |

## Before starting

Make sure Python is installed. Open the Windows command prompt and check:

```
python –version
```

If this shows, for example, Python version 3.11.x or Python 3.12.x, everything is fine. Otherwise, Python is probably not installed correctly, or it is not included in the Windows PATH settings.

## Creating the project folder

Create a new folder for the machine learning exercise. Type in the command line:

```
mkdir ml-beginning
cd ml-beginning
```

The `mkdir` command creates a new folder, and the `cd` command moves into that folder. This way, all the files for the exercise will be placed in the `ml-beginning` folder.

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

AIM learning

# Creating a Virtual Environment

## What is a virtual environment?

A Python virtual environment (`venv`) is a tool for projects where you can install libraries that are available only inside that environment, not on the entire computer. This is good practice when different projects require their own libraries or different versions of them.

## Creating a virtual environment

In the project folder, type:

```
python -m venv ml_venv
```

This creates a new folder called `ml_venv` inside the project folder, which provides a separate Python environment for the project.

---

# Activating the virtual environment on

The virtual environment must be activated each time you start working on this project in a new command prompt window. Open Command Prompt and run:

```
ml_venv\Scripts\activate
```

This activates the virtual environment, and all commands you run will take place inside it.

---

# Installing Libraries

When the virtual environment is active, install the required libraries (this may take a moment):

```
pip install numpy pandas matplotlib scikit-learn jupyter
```

Libraries are essentially pre-built tools:

| Library | What is it used for |
|---|---|
| numpy | Numerical computing and arrays |
| pandas | Working with tabular data |
| matplotlib | Plotting charts and graphs |
| scikit-learn | Building machine learning models |
| jupyter | Running notebooks in a browser |

You can verify the installation with:

```
pip list
```

In the list, you should see at least numpy, pandas, matplotlib, scikit-learn, and jupyter.

---

Co-funded by the European Union

Economic Development Centre
Kokkola Karleby
centria University of Applied Sciences
kpedu

# Starting Jupyter Notebook

Jupyter is a Python development environment where it is easy to navigate folders, and notebooks allow running code in small chunks. It keeps results in memory, like a program still running. This works especially well for machine learning tasks, where some computations may take a long time. This way, you don’t always need to manually save intermediate results.

When you are still in the project folder and the virtual environment is active, run:

```
jupyter notebook
```

This will open the Jupyter Notebook view in your browser.

Create a new notebook by selecting:

```
New -> Python 3
```

Give the notebook a name, for example:

```
first-machine-learning.ipynb
```

# How to Use a Notebook

A notebook consists of cells. A cell can contain:

- Python-code
- Text
- Images
- Tables
- Output

Run a code cell by pressing (or the play triangle):

```
Shift + Enter
```

Try your first code

```
print("Hello machine learning!")
```

Run the cell with **Shift + Enter**. If you see the output, the notebook is working.

Next, test that the libraries work:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

print("Libraries are working!")
```

Co-funded by the European Union

Economic Development Centre
Kokkola Karleby
centria University of Applied Sciences
kpedu

# Basic Idea of Machine Learning

In machine learning, you give the computer example data, and the model's task is to learn some kind of relationship from that data. For example:

| Input | Thing to predict |
|---|---|
| Apartment size | Apartment price |
| Flower measurements | Flower species |
| Email content | Is it spam? |

A simple workflow usually looks like this:

1. Take the data
2. Split the data into training data and test data
3. Train the model using the training data
4. Ask the model to make predictions on the test data
5. Evaluate how well the model performed

Important Terms:

| Sana | Selitys |
|---|---|
| Data | Examples that the model learns from |
| Feature | One input variable (column), e.g. apartment size |
| Target / Label / Class | The thing being predicted, e.g. flower species |
| Malli | The algorithm/program that learns from data |
| Training data | Data used to train the model |
| Test data | Data used to evaluate how well the model predicts new data |

# Example 1: Linear Regression

Regression has multiple "variants" that aim to predict a numeric value, usually from continuous, trend-like data. Linear regression is the simplest case, where the goal is to fit a straight line that best describes the data.

In this example, we create a small dataset ourselves. The idea is:

> *The larger the value of x, the larger the value of y.*

This could resemble a situation where apartment size affects price.

## Creating the Data

Run in the notebook:

```
import numpy as np
import matplotlib.pyplot as plt

# Create a random number generator so results stay the same each run
rng = np.random.RandomState(42)

# X is the input variable (feature)
X = 10 * rng.rand(50, 1)
```

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

```
# y is the value to predict (target)
# y depends on X, with some random noise added
y = 2 * X.squeeze() + 3 + rng.randn(50) * 2

plt.scatter(X, y)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Example data")
plt.show()
```

## Training the Model

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Create the model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

Three things happen here:

1. The data is split into two parts (train and test)
2. A linear regression model is created
3. The model is trained using the training data

## Making Predictions

```
y_pred = model.predict(X_test)

print(y_pred)
```

`y_pred` contains the predictions made by the model.

## Visualizing the Results

```
plt.scatter(X_test, y_test, label="Actual values")
plt.scatter(X_test, y_pred, label="Model predictions")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.title("Actual values vs. Predictions")
plt.show()
```

## Simple Evaluation

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)
```

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

```
print("Mean absolute error:", mae)
```

The mean absolute error (MAE) tells roughly how much the model's prediction differs from the true value on average. The meaning of this number depends heavily on the scale of the case.

Think about:

- Are the predictions close to the true data points?
- What would happen if there were more random variation (noise) in the data?

# Example 2: KNN Classification

KNN (k-nearest neighbors) is a machine learning algorithm where a data point is classified based on its surrounding neighbors. You choose a value k, which is the number of neighbors considered. The data point is then assigned to the majority class among those neighbors. In other words, instead of predicting a number, we now predict a category (class).

In this example, we use the built-in Iris dataset, which is a common dataset used in machine learning. It contains measurements of flowers, such as petal length and width. The model's task is to determine the species of the flower based on these measurements.

## Loading the Data

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()

X = iris.data
y = iris.target

print(iris.feature_names)
print(iris.target_names)
```

In this dataset:

- X contains the flower measurements (features)
- y contains the correct flower species (labels)

## Viewing the Data as a Table

```
df = pd.DataFrame(X, columns=iris.feature_names)
df["target"] = y

df.head()
```

df.head() shows the first rows of the table.

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

## Splitting the Data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

## Training the KNN Model

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Here, n_neighbors=3 means the model looks at the 3 closest neighbors when making a decision.

## Making Predictions

```
y_pred = knn.predict(X_test)

print(y_pred)
print(y_test)
```

The first line shows the model's predictions.

The second line shows the correct answers.

## Measuring Performance

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

If the result is, for example 0.966, it means that about 96.6% of the test samples were classified correctly.

Think about:

- How many predictions were correct?
- What happens if you change n_neighbors=3 to 1 or 10?

# Example 3: Decision Tree

In simple terms, a decision tree tries to create rules like this:

```
If the petal length is small, the flower is likely setosa.
Otherwise, ask the next question.
```

A decision tree is a good model for beginners because its behavior is easy to understand and visualize.

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

## Training the Model

Use the same Iris dataset as in the previous example. If you started from a new notebook, reuse the code for data loading and preprocessing steps from the KNN example.

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)
```

`max_depth=3` refers to the depth of the tree, meaning it can make at most three consecutive splits. This makes the tree simple and easier to understand (you'll see this clearly after visualization).

## Making Predictions

```
y_pred = tree.predict(X_test)

print(y_pred)
print(y_test)
```

The first line shows the model's predictions.

The second line shows the true values.

## Evaluation

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

## Visualizing the Decision Tree

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plot_tree(
    tree,
    feature_names=iris.feature_names,
    class_names=iris.target_names,
    filled=True
)
plt.show()
```

The plot shows the decisions the tree makes at each step.

Think about:

- What does the first decision (split) ask?
- Is the idea of a decision tree easier to understand than KNN?

Co-funded by the European Union

Economic Development Centre
Kokkola Karleby
centria University of Applied Sciences
kpedu

# What Should You Take Away from This Exercise?

After this material, the most important thing is not to memorize all the code, but to understand the basic idea behind machine learning.

The key concept is this general workflow, which appears in many machine learning models:

```
1. Data
2. Train-test split
3. Model creation
4. Model training
5. Predictions
6. Evaluation
```

# Yhteenveto algoritmeista

| Algorithm | Use case | One-sentence explanation |
|---|---|---|
| Linear regression | Predicting numeric values | Fits the best possible straight line through the data |
| KNN | Classification | Assigns a class based on the nearest examples |
| Decision Tree | Classification | Makes decisions using learned rules |

# Final Assignment

At the end of your notebook, write a short summary in your own words.

For example, answer these questions:

1. What is the difference between training data and test data?
2. What does fit() do?
3. What does predict() do?
4. Which of the three models was the easiest to understand?
5. Which step felt the most difficult?
6. Did you try changing different values (parameters) of the algorithms? How did they affect the results?

You can easily export your notebook from Jupyter: File -> Save and Export Notebook As -> HTML.
Many other formats are also useful but may require additional extensions.

# Finished!

Now you have learned the basics of how machine learning and AI applications work in practice. Although more complex models, such as neural networks, are harder to understand, it is still possible to build simple implementations of them in a straightforward way.

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu

Language models were used in the creation of this material, but the content has been strongly edited and refined. These tools are excellent aids for learning and can help you progress further in the subject.

If you are interested in learning more, good next steps include data cleaning and handling missing values, implementing a simple multilayer perceptron (e.g. for handwritten digit recognition, and hyperparameter tuning. The field of machine learning is broad and offers a lot to explore. Welcome to the journey!

Co-funded by the European Union

Economic Development Centre Kokkola Karleby

centria University of Applied Sciences

kpedu