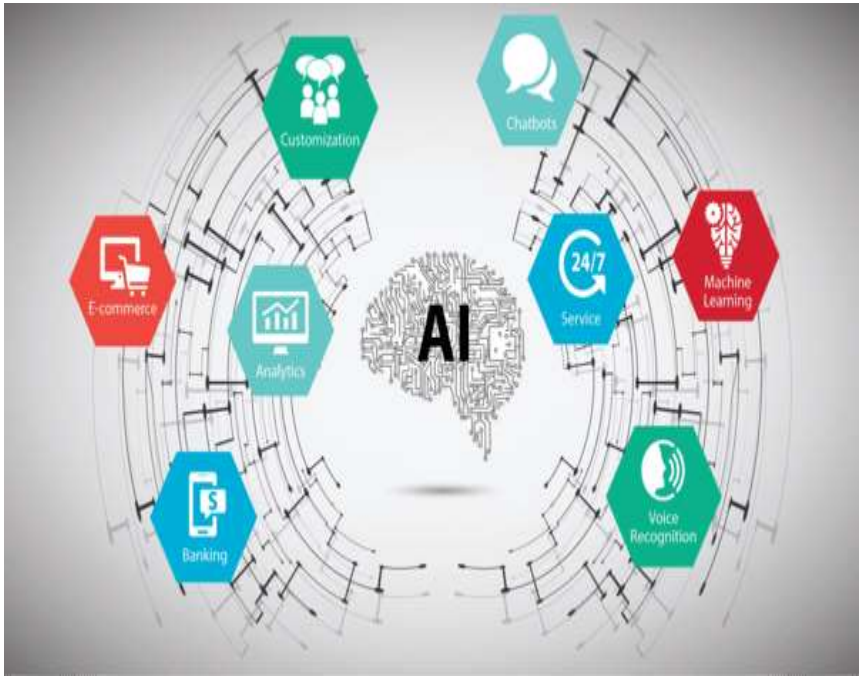
Chapter 6:AI developement tools



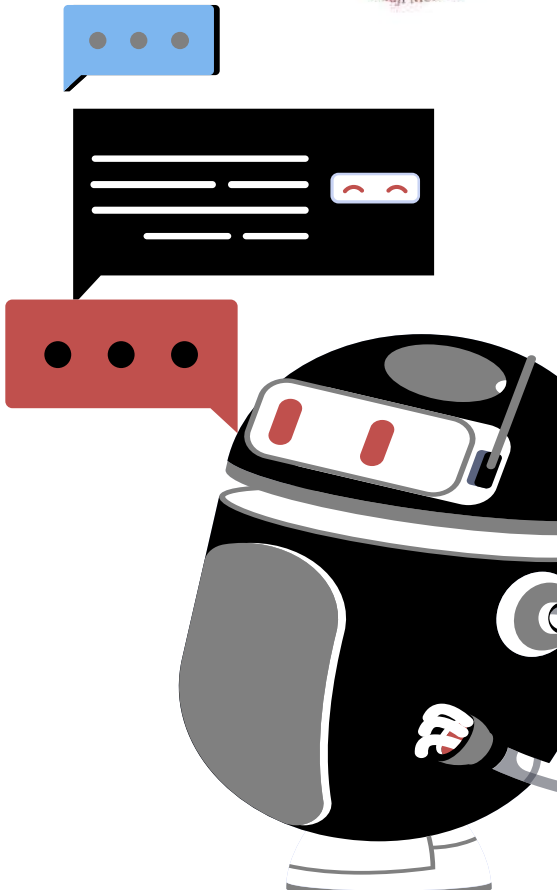


Table of contents

Introduction:

- In this chapter, we will explore some of the most widely used tools and libraries in AI development. These tools are essential for data manipulation, visualization, and machine learning model development.
- Google Colab
- Jupyter
- NumPy
- Matplotlib
- SciKit-Learn
- Pandas

Introduction:

Python programming concepts for data analysis, using popular libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn. By the end of this course, you'll be comfortable with basic Python programming, handling data structures like NumPy arrays and Pandas DataFrames, and visualizing data with Matplotlib. Additionally, you'll gain an understanding of key tools used in AI development, including Google Colab and Jupyter Notebooks.

1

Google colab

(AI)

Google colab :

- Google Colab (short for Colaboratory) is a free, cloud-based platform that allows you to write and execute Python code in a Jupyter notebook environment. It is particularly popular for AI and machine learning projects because it provides free access to GPUs and TPUs, making it easier to train complex models.
- Key Features:
- Free access to GPUs and TPUs.
- Pre-installed libraries like TensorFlow, PyTorch, and Keras.

Google colab :

- Easy sharing and collaboration.
- Integration with Google Drive for saving and loading notebooks.

Use Cases:

1. Prototyping machine learning models.
2. Running data analysis and visualization.
3. Collaborating on AI projects with team members

2

Jupyter

(AI)

Jupyter :

Jupyter is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It supports multiple programming languages, but it is most commonly used with Python.

Jupyter :

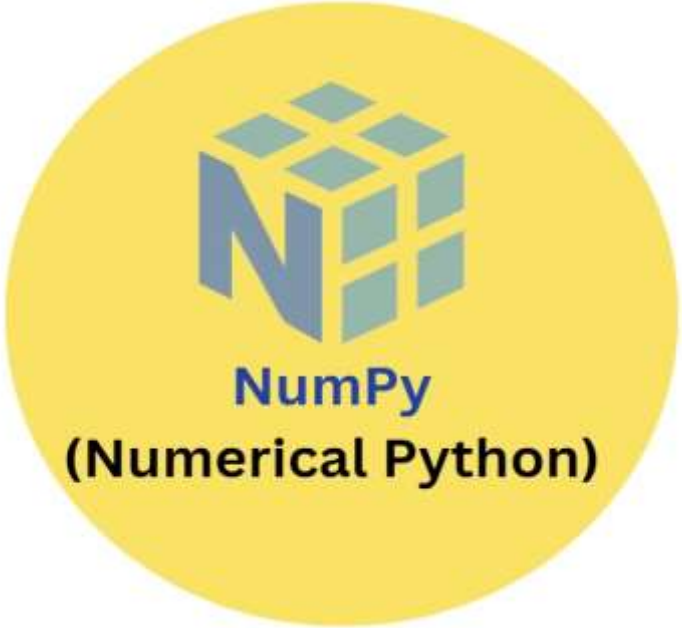
- Key Features:
- Interactive coding environment.
- Support for Markdown and LaTeX for documentation.
- Easy integration with data science libraries.
- Ability to export notebooks in various formats (HTML, PDF, etc.).

Jupyter :

- Use Cases:
- Data cleaning and preprocessing.
- Exploratory data analysis (EDA).
- Prototyping and testing machine learning models.

2

NumPy



NumPy:

NumPy (Numerical Python) is a fundamental library for scientific computing in Python. It provides support for arrays, matrices, and many mathematical functions to operate on these data structures.

- Efficient array operations.
- Broadcasting and vectorization for performance optimization.
- Integration with other libraries like SciPy, Pandas, and Matplotlib

NumPy:

- import numpy as np # import the library numpy
- array = np.array([1, 2, 3, 4, 5]) # Create a 1D array
- matrix = np.array([[1, 2], [3, 4]]) # Create a 2D array
- This imports the NumPy library and gives it the alias np. NumPy is a powerful library for numerical computations in Python.
- Why np? Using np as an alias is a convention in the Python community to make the code shorter and easier to read.

NumPy:

- print(array)
- output:
- [1 2 3 4 5]
- example :
- import numpy as np
- # Create an array of random temperatures in Celsius between -10 and 40
- temperatures = np.random.uniform(-10, 40, 5)
- print(temperatures)

NumPy:

- np.random.uniform(): Generates random values.
- np.array(): Creates an array.
- np.mean(), np.median(): Statistical operations.

4

MatPlotLib

(AI)

MatPlotLib:

- Matplotlib is a plotting library for Python that provides a wide variety of static, animated, and interactive visualizations. It is highly customizable and integrates well with other data science libraries.
- Support for a wide range of plot types (line, bar, scatter, histogram, etc.).
- Customizable plots with labels, legends, and annotations.
- Integration with Jupyter notebooks for online plotting

MatPlotLib :

```
import matplotlib.pyplot as plt
# Sample data: Months and average temperatures
months = ['January', 'February', 'March', 'April']
avg_temperatures = [5, 7, 12, 16]
# Line plot of temperatures over the months
plt.plot(months, avg_temperatures)
plt.xlabel('Months')
plt.ylabel('Average Temperature (°C)')
plt.title('Average Temperatures Over the Year')
plt.show()
```

MatPlotLib :

Key Functions:

plt.plot(): Line plot.

plt.bar(), plt.scatter(): Bar chart and scatter plot.

plt.show(): Displays the plot.

5

SciKits Learn

SciKits Learn :

- SciKit-Learn is one of the most popular machine learning libraries in Python, offering a wide range of algorithms for classification, regression, clustering, and more.

Key Features:

- Supervised and unsupervised learning algorithms.
- Tools for model evaluation and validation.
- Easy integration with Pandas and NumPy.

SciKits Learn:

- Example:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
# Dummy data for linear regression
X = [[1], [2], [3], [4], [5]]
y = [2, 4, 6, 8, 10]
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

SciKits Learn :

```
# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
# Predict the test data
predictions = model.predict(X_test)
print(predictions)
```

SciKits Learn :

- Key Functions:
- train_test_split(): Splits data into training and test sets.
- LinearRegression(): Linear regression model.
- model.fit(), model.predict(): Model training and prediction.

SciKits Learn :

- Data Setup:
- We have the following data:
- Features (X): [[1], [2], [3], [4], [5]]
- Target (y): [2, 4, 6, 8, 10]
- This is a simple linear relationship where y is exactly double the value of X. So, for any given value of X, the value of y should be 2 * X.

SciKits Learn :

- Step 1: Splitting the Data
- Using train_test_split with test_size=0.2, we randomly split the data into training and testing sets. Since the dataset is very small, there is only one data point in the test set and four in the training set. The actual split will be random, but here is an example of how the data might be split:

SciKits Learn :

Training Data (X_train, y_train):

X_train = [[3], [5], [2], [4]]

y_train = [6, 10, 4, 8]

Test Data (X_test, y_test):

X_test = [[1]]

y_test = [2]

SciKits Learn :

- Step 2: Training the Model
- The LinearRegression model will learn the relationship between the independent variable X (features) and the dependent variable y (target). It will compute the intercept and slope of the best-fit line.
- The expected values for the slope (β1) and intercept (β0) are:
- Slope (β1): 2 (since y = 2 * X in the dataset).
- Intercept (β0): 0 (since the line passes through the origin in this case).

SciKits Learn :

- Step 3: Making Predictions
- After the model is trained, we can make predictions on the test set. In this case, X_test = [[1]], and the model should predict a y value close to 2 (because y = 2 * 1).
- Predictions: [2.]
- The model predicts y = 2.0 when X = 1, which is exactly what we expect because the relationship in the dataset is linear with a slope of 2 and an intercept of 0.

Linear regression: example

- from sklearn.linear_model import LinearRegression
- from sklearn.model_selection import train_test_split
- import numpy as np
- # Example dataset
- experience = np.array([1, 2, 3, 4, 5]).reshape(-1, 1) # Feature
- salary = np.array([30000, 35000, 40000, 45000, 50000]) # Target

Linear regression: example

- # Split data
- X_train, X_test, y_train, y_test = train_test_split(experience, salary, test_size=0.2)
- # Create and train model
- model = LinearRegression()
- model.fit(X_train, y_train)
- # Predict
- predictions = model.predict(X_test)

Linear regression: example

- # Coefficient and Intercept
- print(f"Slope (Coefficient): {model.coef_[0]}")
- print(f"Intercept: {model.intercept_}")

5

Pandas

Pandas :

- Pandas is an essential library for data manipulation and analysis, particularly with structured data. It provides DataFrame and Series objects for handling various types of datasets, from CSVs to SQL queries.
- Key Features:
- Handling and manipulation of structured data.
- Powerful operations such as groupby, pivot, and merge.
- Handling missing data and time series data.

Pandas :

```
import pandas as pd
# Create a DataFrame
data = {'Name': ['John', 'Anna', 'Peter'],
 'Age': [28, 24, 35],
'City': ['New York', 'Paris', 'Berlin']}
df = pd.DataFrame(data)
# Display the DataFrame
print(df)
```

Pandas :

Display descriptive statistics

print(df.describe())

Key Functions:

pd.DataFrame(): Creates a DataFrame.

df.groupby(): Group data by a specific column.

df.describe(): Summary statistics for numerical columns.

Manipulate dataframe:

```
# Select a specific column
ages = df['Age']
# Filter rows where Age > 30
older_than_30 = df[df['Age'] > 30]
# Create a new column
df['Age_in_10_years'] = df['Age'] + 10
# Sort the dataset by Age
sorted_df = df.sort_values(by='Age')
```

Manipulate dataframe:

- # Group by City and calculate the mean Age

grouped = df.groupby('City')['Age'].mean()

Drop missing values

- df_cleaned = df.dropna()
- # Fill missing values
- df_filled = df.fillna({'Age': df['Age'].mean()})

Manipulate dataframe:

```
import pandas as pd
import numpy as np
# Create a DataFrame with missing values
df = pd.DataFrame({
 'Name': ['Alice', 'Bob', np.nan, 'David'],
 'Age': [24, np.nan, 22, 32],
 'City': ['New York', 'Paris', 'London', np.nan]})
```

Manipulate dataframe:

- # Fill missing values
- df_filled = df.fillna({'Name': 'Unknown', 'Age': df['Age'].mean(), 'City': 'Unknown'})
- print(df_filled)

Reading a Dataset with Pandas

```
import pandas as pd
# Read a dataset from a CSV file
df = pd.read_csv('sample_data.csv')
# Display the first 5 rows
print(df.head())
# Display basic information about the dataset
print(df.info())
# Display basic statistics
print(df.describe())
```

Reading a Dataset with Pandas

read_csv() reads the CSV file.

head() shows the first few entries.

info() gives a summary (columns, types, missing values).

describe() gives mean, std, min, max, etc

Example of linear regression

```
import pandas as pd
# Reading a dataset from a CSV file
# Example file: students_scores.csv
# Columns: 'Hours_Studied', 'Test_Score', 'Passed'
df = pd.read_csv('C:\\Users\\hp\\Desktop\\mémoire licence\\students_scores.csv')
# Display the first few rows
print(df.head())
# Display dataset information
```

Example of linear regression

```
print(df.info())
# Check for missing values
print(df.isnull().sum())
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
# Select features and target variable
```

Example of linear regression

```
X = df[['Hours_Studied']]  # Independent variable
y = df['Test_Score']       # Dependent variable
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# Create and train the model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

Example of linear regression

```
# Make predictions
y_pred = linear_model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")
# Visualize the regression line
plt.scatter(X, y, color='blue')
plt.plot(X, linear_model.predict(X), color='red')
```

Example of linear regression

```
plt.xlabel('Hours Studied')
plt.ylabel('Test Score')
plt.title('Linear Regression: Hours Studied vs Test Score')
plt.show()
```