Introduction to Artificial Intelligence using Python

Week 2 of 4

Anass B. El-Yaagoubi

STAT - CEMSE, King Abdullah University of Science and Technology

KAUST Academy: AI Summer School

Thuwal, July 9th, 2023

Course Overview

- Goal of this course is to introduce you to artificial intelligence (AI).
- Learn how to implement various AI algorithms in Python.
- Four weeks program.
  1. Python programming
  2. Data wrangling and visualization
  3. Machine learning
  4. Neural networks and deep learning
- AI project in parallel the course

Overview of week 2:

I Introduction to NumPy

II Introduction to Pandas

III More about Pandas

IV Visualization with Matplotlib

V More advanced visualization

Day 1: Introduction to NumPy

- Multidimensional Arrays.
- Numpy Functions.
- Concatenation (Horizontal and Vertical).
- Vectorized Operations.
- Linear Algebra module.
- Random module.
- Loading and Saving NumPy Arrays.

Multidimensional Arrays

```
# Example 1
import numpy as np
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
print(array_2d)

# Example 2
reshaped_arr = np.reshape(array_2d, (2, 6))
print(reshaped_arr)
```

Numpy Functions

```
# Example 1
my_arr = np.array([1, 2, 3, 4, 5])
arr_mean = np.mean(my_arr)
arr_std = np.std(my_arr)
arr_sum = np.sum(my_arr)
print(arr_mean, arr_std, arr_sum)

# Example 2
min_val = np.min(my_arr)
max_val = np.max(my_arr)
print(min_val, max_val)

# What does axis parameter stand for?
```

Concatenation (Horizontal and Vertical)

```
# Example 1
my_arr1 = np.array([1, 2, 3])
my_arr2 = np.array([4, 5, 6])
hstack_result = np.hstack((my_arr1, my_arr2))
print(hstack_result)

# Example 2
vstack_result = np.vstack((my_arr1, my_arr2))
print(vstack_result)

# What is the difference?
```

Vectorized Operations

```
# Example 1
my_arr3 = np.array([1, 2, 3])
my_arr4 = np.array([4, 5, 6])
add_result = my_arr3 + my_arr4
sub_result = my_arr3 - my_arr4
mul_result = my_arr3 * my_arr4
div_result = my_arr3 / my_arr4
print(add_result, sub_result, mul_result, div_result)

# Example 2
exp_result = np.exp(my_arr3)
log_result = np.log(my_arr4)
sqrt_result = np.sqrt(my_arr3)
print(exp_result, log_result, sqrt_result)
```

NumPy Linear Algebra Module

```
# Example 1
my_matrix = np.array([[1, 2], [3, 4]])
mat_mul_result1 = np.linalg.matrix_power(my_matrix, n=2)
mat_mul_result2 = my_matrix @ my_matrix
print(mat_mul_result1)
print(mat_mul_result2)

# Example 2
matrix_inv = np.linalg.inv(my_matrix)
print(matrix_inv)
```

NumPy Random Module

```
# Generate a random float between 0 and 1
random_float = np.random.random()
print("Random Float:", random_float)

# Generate an array of random integers
random_integers = np.random.randint(low=1, high=10, size=5)
print("Random Integers:", random_integers)

# Generate a random array from a normal distribution
random_normal = np.random.normal(loc=0, scale=1, size=(3, 3))
print("Random Normal Array:")
print(random_normal)
```

Loading and Saving NumPy Arrays

```
# Example 1
data = np.array([1, 2, 3, 4, 5])
np.save('data.npy', data)

# Example 2
loaded_data = np.load('data.npy')
print(loaded_data)
```

Day 2: Introduction to Pandas

- Series
- DataFrames
- Data manipulation
- Loading and saving

import pandas as pd

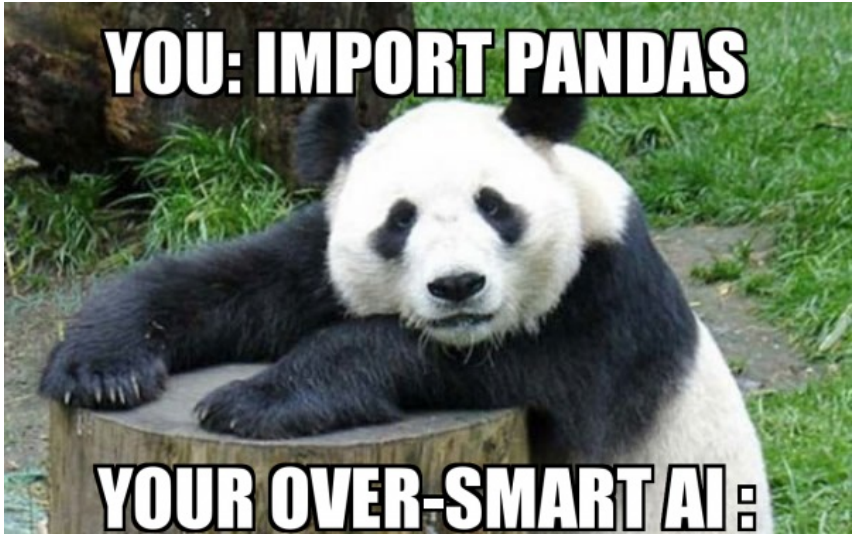

Fig. 1: Be careful when you import pandas in you project.

Series

```python
# Example 1: Creating a Series
import pandas as pd

temperatures = [35.5, 36.1, 37.2, 34.8, 35.9]
series = pd.Series(temperatures, name='Jeddah Temperatures')
print(series)

# Example 2: Accessing Elements in a Series
humidity = [60, 65, 70, 75, 80]
series = pd.Series(humidity, name='Jeddah Humidity Levels')
print(series[1])
```

DataFrames

```python
# Example 1: Creating a DataFrame
import pandas as pd

data = {'Name': ['Ali', 'Fatima', 'Mohammed'],
        'Age': [25, 30, 35],
        'Country': ['Morocco', 'Saudi Arabia', 'Egypt']}
df = pd.DataFrame(data)
print(df)

# Example 2: Accessing Data in a DataFrame
name_column = df['Name']
age_column = df['Age']
print(name_column)
print(age_column)
```

Data Manipulation: Basic Pandas Functions

```
# Example 1: Filtering cities with population larger than 3 million.
data = {'City': ['JED', 'RUH', 'JAZ', 'TAB', 'MAK', 'MAD'],
        'Population': [3500, 7000, 600, 500, 2000, 1500]}
df = pd.DataFrame(data)
filtered_df = df[df['Population'] > 3000]
print(filtered_df)

# Example 2: Average population per city
city_mean_population = df['Population'].mean()
print(city_mean_population)
```

Loading and Saving DataFrames

```
# Example 1: Loading Data from CSV
df = pd.read_csv('data.csv')
print(df)

# Example 2: Saving DataFrame to CSV
data = {'Name': [],
        'Age': [],
        'Salary': [],
        'Country': []}
df = pd.DataFrame(data)
df.to_csv('output.csv', index=False)
```

Day 3: More about Pandas

- Handling Missing Data
- Concatenating DataFrames
- Merging DataFrames
- Transforming Data

Handling Missing Data - Dropping Rows

```
import pandas as pd

data = {'Name': ['Ali', 'Fatima', 'Mohammed'],
        'Age': [25, None, 35],
        'AI Score': [80, None, 95]}
df = pd.DataFrame(data)

df = df.dropna()
print(df)
```

Handling Missing Data - Filling Missing Values

```
import pandas as pd

data = {'Name': ['Ali', 'Fatima', 'Mohammed'],
        'Age': [25, None, 35],
        'AI Score': [80, 90, None]}
df = pd.DataFrame(data)

df = df.fillna(method='ffill')
print(df)

# What does the option method='ffill' do?
```

Concatenating DataFrames

```
import pandas as pd

df1 = pd.DataFrame({'Name': ['Ali', 'Fatima'],
                    'Age': [25, 30],
                    'AI Score': [80, 90]})
df2 = pd.DataFrame({'Name': ['Mohammed', 'Layla'],
                    'Age': [35, 28],
                    'Experience': [5, 3]})

concatenated_df = pd.concat([df1, df2], ignore_index=True)
print(concatenated_df)
```

Merging DataFrames

```
import pandas as pd

df1 = pd.DataFrame({'Name': ['Ali', 'Fatima', 'Mohammed'],
                    'Age': [25, 30, 35],
                    'AI Score': [80, 90, 85]})
df2 = pd.DataFrame({'Name': ['Layla', 'Mohammed', 'Ibrahim'],
                    'Experience': [5, 7, 3],
                    'Salary': [70000, 80000, 60000]})

merged_df = pd.merge(df1, df2, on='Name', how='inner')
print(merged_df)

# What is the meaning of on='Name' and how='inner'?
```

Transforming Data

```
import pandas as pd

data = {
'SaleAmount': [100, 200, 150, 300, 250, 120],
'CountryCode': ['SA', 'SA', 'AE', 'AE', 'SA', 'AE'],
'City': ['Riyadh', 'Jeddah', 'Dubai', 'Abu Dhabi', 'Riyadh', 'Dubai']
}

df = pd.DataFrame(data)

# Computing the mean sales
grouped_df = df.groupby(['City', 'CountryCode']).mean()

print(grouped_df)
```

Day 4: Visualization with Matplotlib

- Why data visualization matters?
- How to make line plots?
- How to make scatter plots?
- How to make histograms?
- How to customize a plot?

Why data visualization matters?

- Visual representation of data simplifies complex information and reveals patterns, trends, and relationships.
- Data visualization aids decision-making, promotes data exploration, and facilitates effective communication of insights.
- Simplifies complex datasets into intuitive visuals, enhancing comprehension and interpretation.
- Tools like Matplotlib offer customization for creating impactful visualizations tailored to specific needs.

Line Plots with Matplotlib

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

# Sample data
x = np.linspace(0, 10, 20)
y = np.random.randint(5, 15, 20)

# Creating a line plot
plt.plot(x, y)

# Display the plot
plt.show()
```

Scatter Plots with Matplotlib

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

# Sample data
n = 100
x = np.random.randn(n)
y = np.random.randn(n)

# Creating a scatter plot
plt.scatter(x, y)

# Display the plot
plt.show()
```

Histograms with Matplotlib

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

# Sample data
data = np.random.binomial(100, .1, 10000)

# Creating a histogram
plt.hist(data, bins=20)

# Display the plot
plt.show()
```

Customization with Matplotlib

```
# Sample data
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
# Creating a plot with customizations
plt.plot(x, y1, label='sin(x)', color='blue', linestyle='--')
plt.plot(x, y2, label='cos(x)', color='red', linestyle='-.')
plt.legend()
# Customizing the plot
plt.title("Customized Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
# Display the plot
plt.show()
```

Day 5: More advanced visualization

- Subplots.
- Visualization in 2D: Heatmaps.
- Visualization in 3D: Scatter plots.

Subplots with Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
# Sample data
x = np.linspace(0, 2 * np.pi, 100)
y1, y2 = np.sin(x), np.cos(x)
# Creating subplots
plt.subplot(211)
plt.plot(x, y1, label='sin(x)') # Can we see the label?
plt.title('Sine function')
plt.subplot(212)
plt.plot(x, y2, label='cos(x)') # Can we see the label?
plt.title('Cosine function')
# Display the subplots
plt.tight_layout()
plt.show()
```

Heatmaps with Matplotlib

```
# Sample data
data = np.random.rand(10, 10)

# Creating a heatmap
plt.imshow(data, cmap='hot')
# Adding colorbar
plt.colorbar()

# Customizing the plot
plt.title('Heatmap')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Display the heatmap
plt.show()
```

3D Scatter Plots with Matplotlib

```
# Sample data
np.random.seed(1)
n = 100
x, y, z = np.random.rand(n), np.random.rand(n), np.random.rand(n)
colors = np.random.rand(n)
# Creating a 3D scatter plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c=colors, cmap='viridis')
# Customizing the plot
ax.set_title('3D Scatter Plot')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
# Display the 3D scatter plot
plt.show()
```