SBS4115 Fundamentals of AI & Data Analytics

Practical Applications and Project Work I

Lecturer: Ir Dr Kelvin K. W. Siu
email: kelvinsiu@thei.edu.hk

Department of Construction,
Environment and Engineering

Intended Learning Outcomes

- By the end of this lecture, you will be able to…
  - Perform data analysis of open data from open source of Hong Kong government

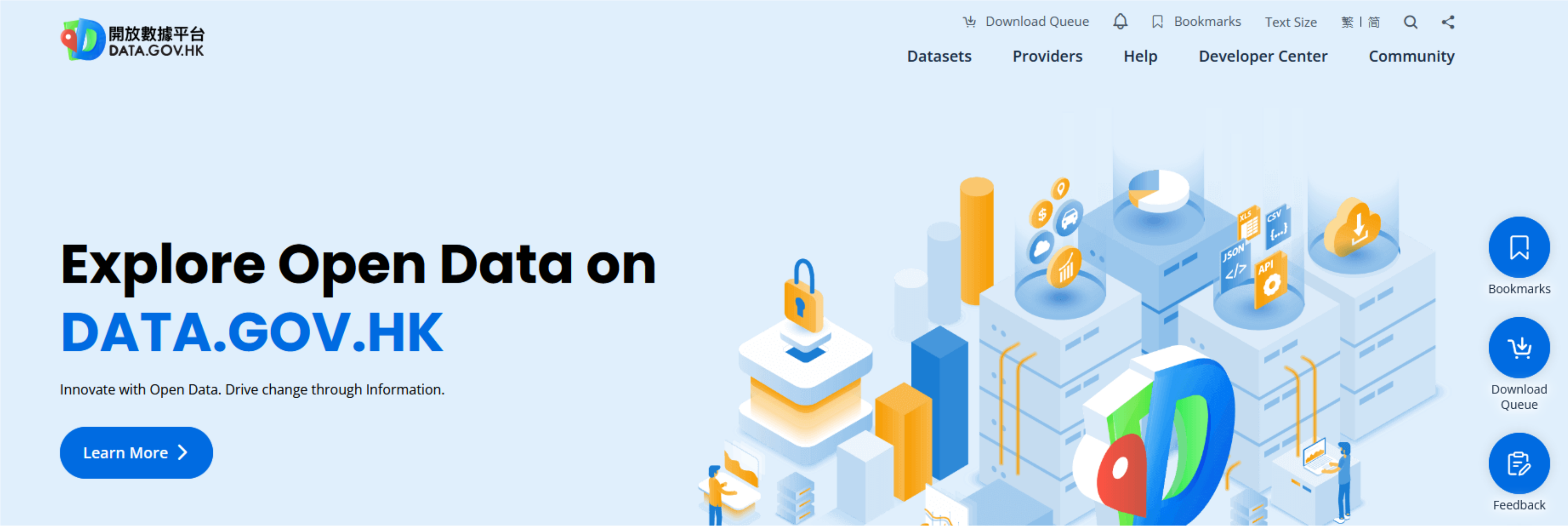
Assignment

The assignment is about data analytics.

You can treat it as an individual project.

Please select any dataset from the governments open data platform.

(https://data.gov.hk/en/)



Explore Open Data on DATA.GOV.HK

Innovate with Open Data. Drive change through Information.

Assignment

After selecting a dataset, perform analysis using the Python skills taught in the class. You are encouraged to do more with skills not mentioned in the notes.

Save your work in the form of **.ipynb format (or .py format)**

Submit both your analysis file and the simplified .csv data file to **Moodle by 25 November 2024**, remember to state your full name and student number in your email. (Please make sure your code can be run with no problems.)

The baseline score is 70, you will get extra marks if you can do more analysis using skills not mentioned in the notes.

Downloading data

Select any dataset from the governments open data platform. (https://data.gov.hk/en/)

e.g. air pressure data from the observatory

Simple analysis by plotting graphs

First, we need to import the file

```
import pandas as pd

pressuredata = pd.read_csv(r"C:\Users\User user\Desktop\daily_HKA_MSLP_ALL.csv")

print(pressuredata)
```

What do you get?

Remove inconsistent format

Remove rows with inconsistent format, and save the file as: daily_HKA_MSLP_ALLa.csv

Try again:

```
import pandas as pd

pressuredata = pd.read_csv(r"C:\Users\User
user\Desktop\daily_HKA_MSLP_ALLa.csv")

print(pressuredata)
```

Simple analysis by plotting graphs

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the data into a DataFrame
df = pd.read_csv(r"C:\Users\User user\Desktop\daily_HKA_MSLP_ALLa.csv")

# Combine the Year, Month, and Day columns to create a datetime index
df['Date'] = pd.to_datetime(df[['Year', 'Month', 'Day’]])

# Set the Date column as the index,the original DataFrame is modified, and no new DataFrame is returned.
df.set_index('Date', inplace=True)

# Convert the 'Value' column to numeric and handle errors
df['Value'] = pd.to_numeric(df['Value'], errors='coerce')
```

Simple analysis by plotting graphs

```
# Drop rows with NaN values
df.dropna(subset=['Value'], inplace=True)

# Plot the data
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Value'], marker='o’)
plt.title('Value over Time’)
plt.xlabel('Date’)
plt.ylabel('Value’)
```

Simple analysis by plotting graphs

```
# Set custom y-axis intervals
y_min, y_max = df['Value'].min(), df['Value'].max()

# Adjust the interval as needed
plt.yticks(range(int(y_min), int(y_max), 10))

# Adjust y-axis labels for better readability
plt.yticks(fontsize=10)
plt.xticks(rotation=45, ha='right', fontsize=10)
```

Simple analysis by plotting graphs

```
# Add grid for better visualization
plt.grid(True)

# Add padding to avoid clipping of labels
plt.tight_layout()
plt.show()
```

Calculate different parameters

```
mean_value = df['Value'].mean()
median_value = df['Value'].median()

# mode() returns a Series, so we take the first element, even if there are multiple modes, this ensures that you get the first one.
mode_value = df['Value'].mode()[0]

std_value = df['Value'].std()

print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Standard Deviation: {std_value}")
```

Calculate different parameters

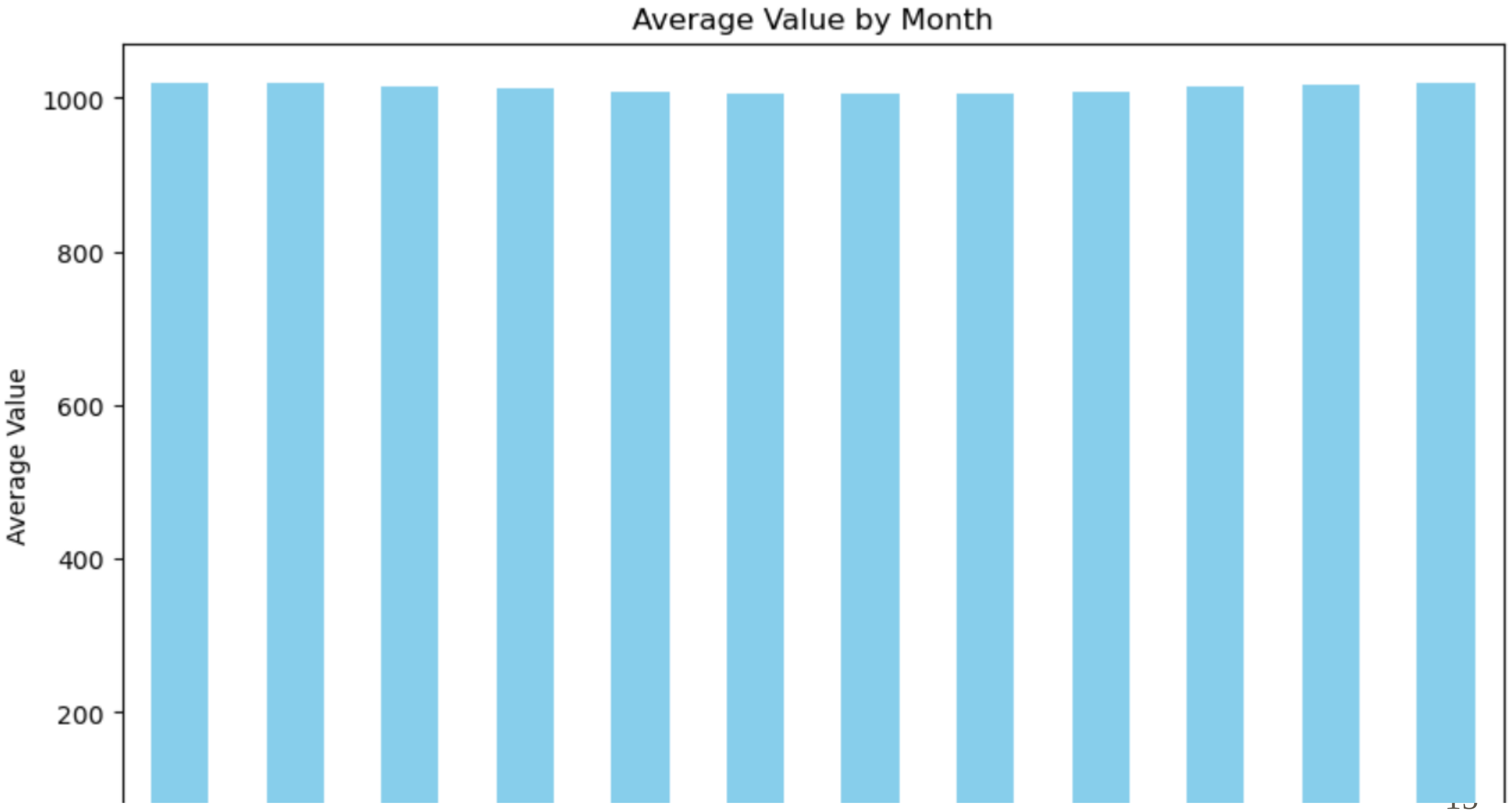
```
max_value = df['Value'].max()
min_value = df['Value'].min()

print(f"Maximum Value: {max_value}")
print(f"Minimum Value: {min_value}")
```

Using bar chart to analyze the data

```
# Group by Month and calculate the average Value
monthly_avg = df.groupby(df.index.month)['Value'].mean()

# Plot the bar chart
plt.figure(figsize=(10, 6))
monthly_avg.plot(kind='bar', color='skyblue')
plt.title('Average Value by Month')
plt.xlabel('Month')
plt.ylabel('Average Value')
plt.xticks(rotation=0)
plt.show()
```

Using bar chart to analyze the data

The variations can be observed but it is not clear enough



Using bar chart to analyze the data

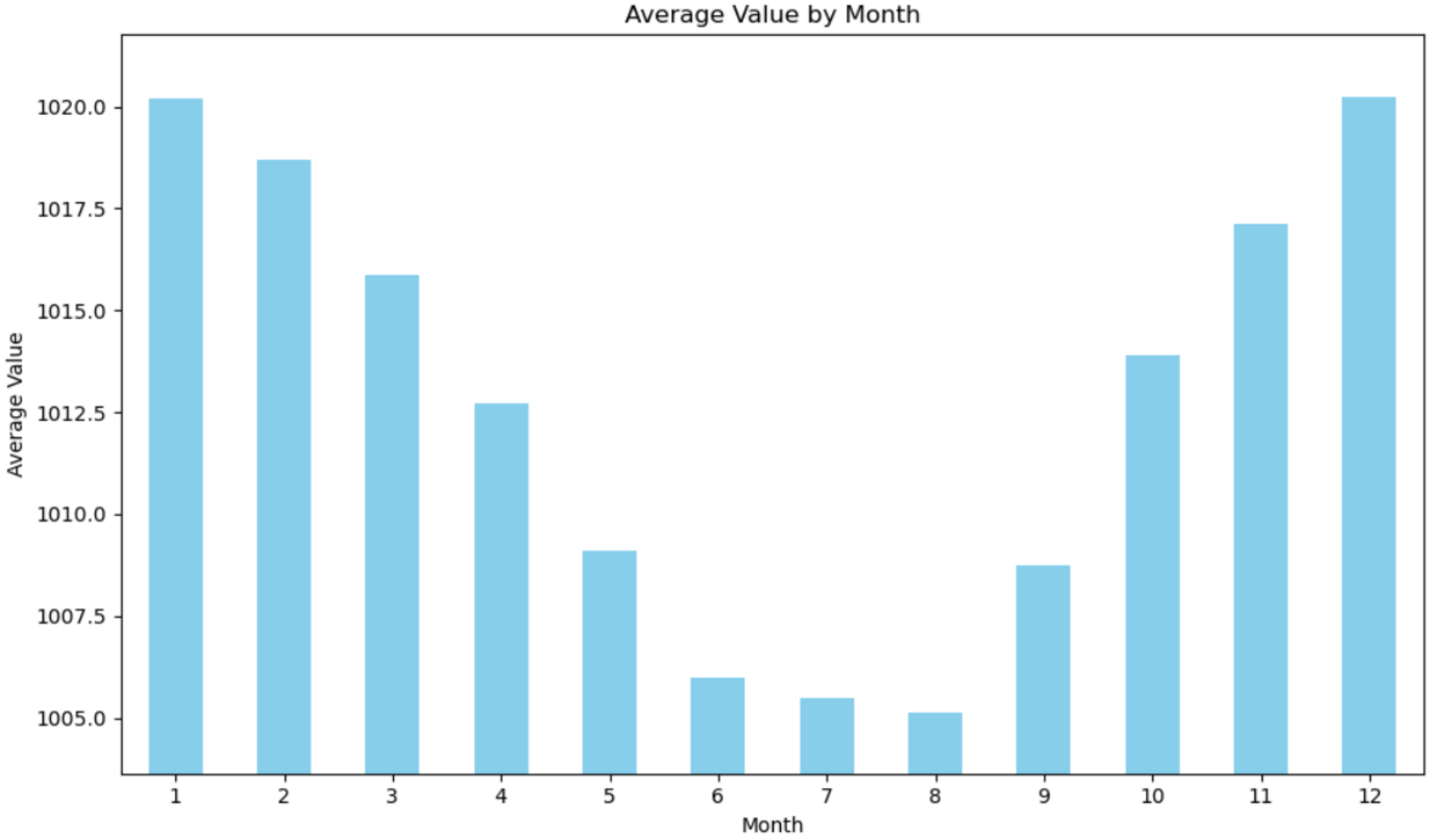
```
# Plot the bar chart
plt.figure(figsize=(10, 6))
monthly_avg.plot(kind='bar', color='skyblue’)
plt.title('Average Value by Month’)
plt.xlabel('Month’)
plt.ylabel('Average Value’)
```

Using bar chart to analyze the data

```
# Set custom y-axis limits to amplify differences
y_min, y_max = monthly_avg.min(), monthly_avg.max()

# Adjust the limits as needed
plt.ylim(y_min - (y_max - y_min) * 0.1, y_max + (y_max - y_min) * 0.1)

# Adjust y-axis labels for better readability
plt.yticks(fontsize=10)
plt.xticks(rotation=0, fontsize=10)

# Add padding to avoid clipping of labels
plt.tight_layout()
plt.show()
```

Using bar chart to analyze the data

The variations can be observed clearly



Outputting summary

```
# Get the statistical summary of the 'Value' column

summary = df['Value'].describe()
print(summary)
```

Measures of Association

- We have visualized the data, now we are going to investigate the associations between different data. (remember what you have learned in lecture 8)

Measures of Association

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns  #It’s a Python visualization library built on top of Matplotlib

# Read the data into a DataFrame
df = pd.read_csv(r"C:\Users\User user\Desktop\daily_HKA_MSLP_ALLa.csv")
```

Measures of Association

```
# Combine the Year, Month, and Day columns to create a datetime index
df['Date'] = pd.to_datetime(df[['Year', 'Month', 'Day’]])

# Set the Date column as the index
df.set_index('Date', inplace=True) #The original DataFrame is modified, and no new DataFrame is returned.

# Convert all columns to numeric and handle errors
df = df.apply(pd.to_numeric, errors='coerce’)
```

Note that the above line of code is used to convert all columns in the DataFrame df to numeric data types. It handles any errors that occur during the conversion process by coercing non-numeric values to NaN (Not a Number).

Measures of Association

```
# Drop columns with all NaN values
df.dropna(axis=1, how='all', inplace=True)

# Drop rows with NaN values
df.dropna(inplace=True)

# Calculate the correlation matrix
correlation_matrix = df.corr()
print(correlation_matrix)
```

Measures of Association

Explanation

df.dropna(): This function is used to remove missing data (NaN values) from a DataFrame.

axis=1: This specifies that the operation should be performed on columns. If axis=0, it would operate on rows.

how='all': This parameter determines the condition for dropping. 'all' means that only columns where all values are NaN will be dropped. If it were 'any', columns with any NaN values would be dropped.

inplace=True: This modifies the original DataFrame directly, rather than returning a new DataFrame with the changes.

Measures of Association

```
# Export the correlation matrix to a CSV file
correlation_matrix.to_csv(r"C:\Users\User user\Desktop\correlation_matrix.csv")

# Visualize the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm’)
plt.title('Correlation Matrix’)
plt.show()
```

Measures of Association

Explanation

```
plt.figure(figsize=(10, 8))
```

Purpose: This line creates a new figure for plotting.figsize=(10, 8): This sets the size of the figure to 10 inches wide and 8 inches tall.

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm’)
```

Purpose: This line creates a heatmap using the Seaborn library.

correlation_matrix: This is the data being plotted.

In this case, it's the correlation matrix of your DataFrame.

Measures of Association

annot=True: This adds annotations to the heatmap, displaying the correlation values within each cell. This makes it easier to see the exact correlation values at a glance.

cmap='coolwarm': This sets the color map for the heatmap. The 'coolwarm' color map ranges from cool colors (blue) to warm colors (red), which helps in visualizing the strength and direction of correlations. Positive correlations are shown in warm colors, while negative correlations are shown in cool colors.

Further improve the observation

By observation we see that the correlation between month and the pressure value is highest compared to that between other data. But it seems that the magnitude is unusually small.

What’s the reason behind?

Further improve the observation

We can try to find out the correlation between months and pressure values but only consider the months from January to June.

Try this:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Read the data into a DataFrame
df = pd.read_csv(r"C:\Users\User user\Desktop\daily_HKA_MSLP_ALLa.csv")
```

Further improve the observation

```
# Combine the Year, Month, and Day columns to create a datetime index
df['Date'] = pd.to_datetime(df[['Year', 'Month', 'Day’]])

# Set the Date column as the index
df.set_index('Date', inplace=True)

# Convert all columns to numeric and handle errors
df = df.apply(pd.to_numeric, errors='coerce’)

# Drop columns with all NaN values
df.dropna(axis=1, how='all', inplace=True)

# Drop rows with NaN values
df.dropna(inplace=True)
```

Further improve the observation

```
# Extract the month from the date
df['Month'] = df.index.month

# Filter the data to include only the months from January to June
df_jan_to_jun = df[df['Month'].isin([1, 2, 3, 4, 5, 6])]

# Group by month and calculate the mean value for each month
monthly_avg_jan_to_jun = df_jan_to_jun.groupby('Month')['Value'].mean()

# Calculate the correlation between the month and the average value for January to June
correlation_jan_to_jun = monthly_avg_jan_to_jun.corr(pd.Series([1, 2, 3, 4, 5, 6],
index=monthly_avg_jan_to_jun.index))

print(f"Correlation between month and value (January to June): {correlation_jan_to_jun}")
```

Checklist

- Can you:
  1. Perform data analysis of open data from open source of Hong Kong government?