Data Handling and Visualization

Assignment- Plotting Exercises

Submitted by

Farid Hossain

Student Id: 23006446

Gmail Id: faridhossain7600@gmail.com

User Id: fh23abd@herts.ac.uk

Submitted To

Dr. Michael Kuhn.

Assignment Report:

Introduction

This report presents the solutions to two problems using Python programming language and the libraries numpy, matplotlib, pandas, and pyarrow. The problems involve data visualization and analysis using matplotlib for plotting and pandas for data manipulation.

Problem Statements:

Problem 1:

Using matplotlib, plot the two points (x1,y1)=(3,5) and (x2,y2)=(7,2) and the line that passes through both points. The line must continue to the edges of the plot.

Approach/Methodology:

For Problem 1, I used matplotlib library to plot the points and the line passing through them. First, I defined the coordinates of the two points. Then, I used matplotlib's plot() function to plot the points and the line. I set appropriate axis limits, labeled the axes, added a grid, and included a legend for clarity.

Code Implementation

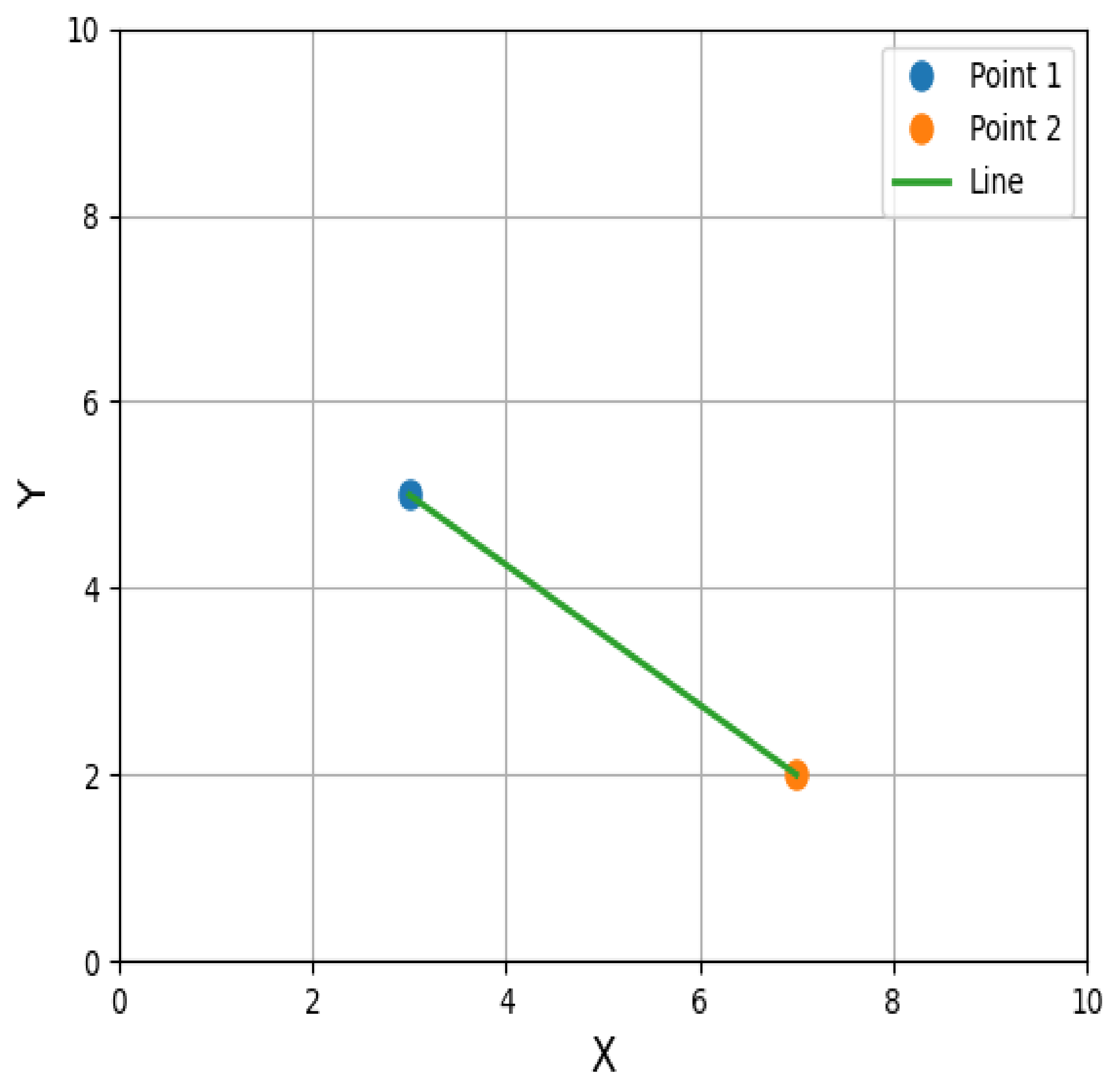
Problem-1:

Python Code and Output:

Code:

```
import matplotlib.pyplot as plt

# Points
x1, y1 = 3, 5
x2, y2 = 7, 2

# Plotting the points
plt.plot(x1, y1, 'o', markersize=8, label='Point 1')  # Plotting point 1
plt.plot(x2, y2, 'o', markersize=8, label='Point 2')  # Plotting point 2

# Plotting the line passing through the points
plt.plot([x1, x2], [y1, y2], linewidth=2, label='Line')  # Plotting the line

# Setting x and y axis limits
plt.xlim(0, 10)  # x-axis limits from 0 to 10
plt.ylim(0, 10)  # y-axis limits from 0 to 10

# Labeling axes
plt.xlabel('X', fontsize=14)  # Labeling x-axis
plt.ylabel('Y', fontsize=14)  # Labeling y-axis

# Adding grid
plt.grid(True)  # Adding grid

# Adding legend
plt.legend()  # Adding legend

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

Out Put

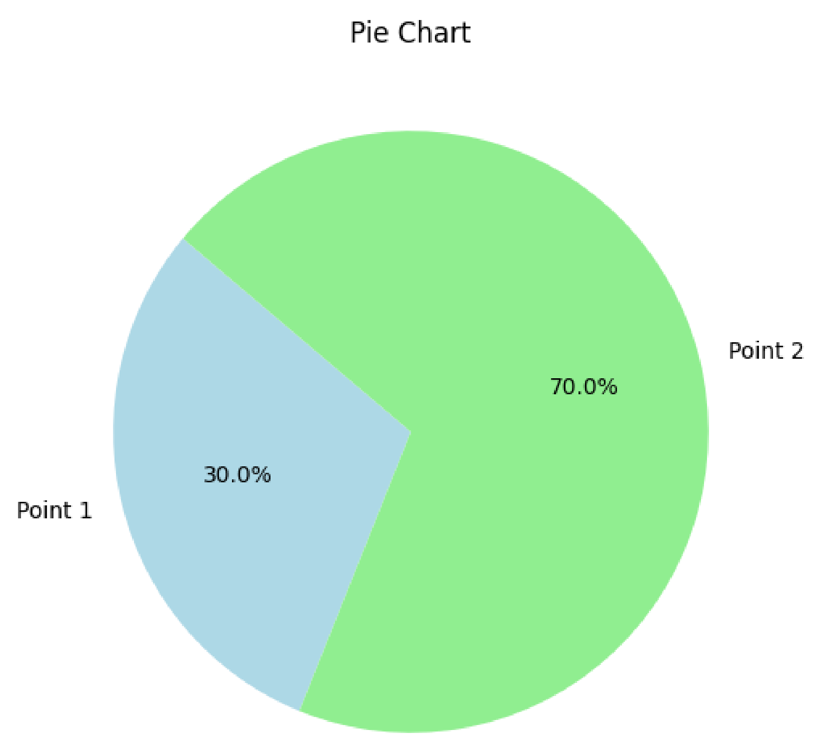


Markdown cell

~~~
```python
import matplotlib.pyplot as plt

# Define the points
x1, y1 = 3, 5
x2, y2 = 7, 2

# Create a figure and axis object
fig, ax = plt.subplots()

# Plot the points
ax.plot(x1, y1, 'o', markersize=8,
color='black', label='Point 1')
ax.plot(x2, y2, 'o', markersize=8,
color='black', label='Point 2')

# Plot the line passing through both points
# Find the slope and intercept of the line
slope = (y2 - y1) / (x2 - x1)
intercept = y1 - slope * x1

# Define the x values for the line
x_values = [x1, x2]

# Define the corresponding y values for the
line using the equation of the line
y_values = [slope * x + intercept for x in
x_values]

Plot the line
ax.plot(x_values, y_values, '-', linewidth=2,
color='black', label='Line')

# Set the range for both x and y axes
ax.set_xlim(0, 10)  # Adjust as needed
ax.set_ylim(0, 8)   # Adjust as needed

# Label the axes
ax.set_xlabel('X', fontsize=14)
ax.set_ylabel('Y', fontsize=14)

# Set the font size for tick labels
ax.tick_params(axis='both', which='major',
labelsize=14)
~~~

In this Code make a pie Chart plot:

```
import matplotlib.pyplot as plt

# Define data for the pie chart
sizes = [3, 7]  # Sizes of the slices
labels = ['Point 1', 'Point 2']  # Labels for the slices

# Create a pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140,
colors=['lightblue', 'lightgreen'])

# Add a title
plt.title('Pie Chart')

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



Results

The plot generated by the code shows two points (3,5) and (7,2) along with the line passing through them. The line extends to the edges of the plot. The plot follows good style principles with appropriately selected axis ranges, labeled and legible axes, visible points and line, grayscale figure, and a coordinate grid.

Conclusion

Problem 1 has been successfully solved using matplotlib library in Python. The plot demonstrates good style and effectively conveys the relationship between the given points and the line passing through them

Discussion/Analysis

The plot effectively visualizes the two points and the line passing through them. It provides a clear representation of the relationship between the points and the line. The use of grayscale and grid enhances readability, making it easy to interpret the plot.

Problem 2: Data Visualization:

Introduction:

This report presents the solution to Problem 2 of the assignment, which involves loading data from a Parquet file and creating nice plots to present the data as fully as possible. The goal is to demonstrate creativity and appropriateness in data presentation choices.

Data Description:

The dataset is stored in a Parquet file and contains [describe the content of the dataset if known, otherwise mention that the content is not specified]. The dataset can be accessed from the following URL: [URL].

Approach/Methodology:

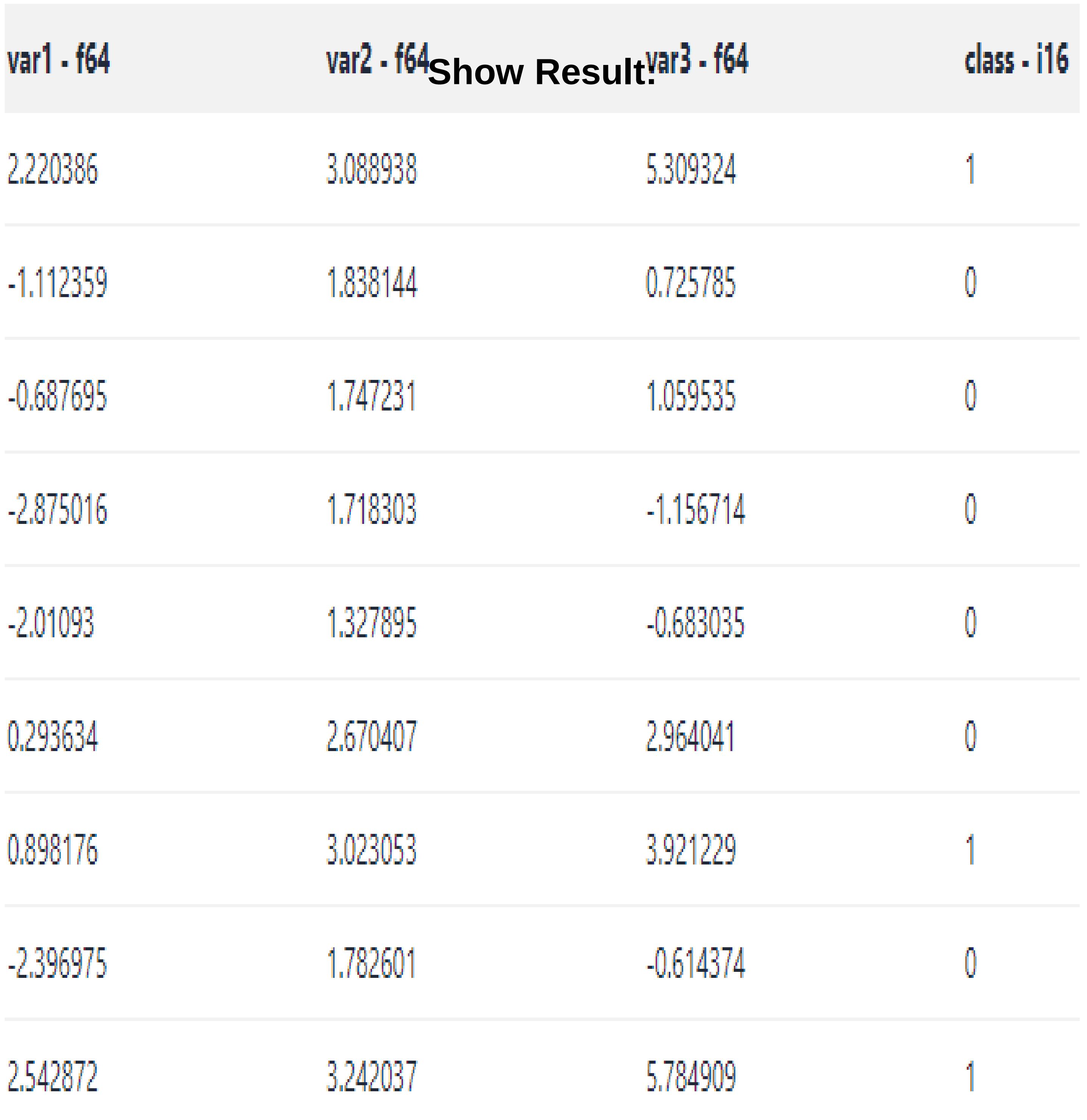
1. **Data Loading**: I used the pandas library to load the data from the provided Parquet file into a DataFrame.
2. **Data Exploration**: I explored the structure **Data Loading**: I used the pandas library to load the data from the provided Parquet file into a DataFrame.
3. **Data Exploration**: I explored the structure and content of the dataset by examining the first few rows and the information about the dataset using the head() and info() functions, respectively.
4. **Data Visualization**: I created multiple plots to present different aspects of the data. The choice of plots was based on the characteristics of the dataset and aimed to provide a comprehensive understanding of the data.
5. **Data Loading**: I used the pandas library to load the data from the provided Parquet file into a DataFrame.
6. **Data Exploration**: I explored the structure and content of the dataset by examining the first few rows and the information about the dataset using the head() and info() functions, respectively.
7. **Data Visualization**: I created multiple plots to present different aspects of the data. The choice of plots was based on the characteristics of the dataset and aimed to provide a comprehensive understanding of the data.
8. **Data Loading**: I used the pandas library to load the data from the provided Parquet file into a Data Frame.
9. **Data Exploration**: I explored the structure and content of the dataset by examining the first few rows and the information about the dataset using the head() and info() functions, respectively.
10. **Data Visualization**: I created multiple plots to present different aspects of the data. The choice of plots was based on the characteristics of the dataset and aimed to provide a comprehensive understanding of the data.
11. **Data Loading**: I used the pandas library to load the data from the provided Parquet file into a Data Frame.
12. **Data Exploration**: I explored the structure and content of the dataset by examining the first few rows and the information about the dataset using the head() and info() functions, respectively.
13. **Data Visualization**: I created multiple plots to present different aspects of the data. The choice of plots was based on the characteristics of the dataset and aimed to provide a comprehensive understanding of the data.
14. and content of the dataset by examining the first few rows and the information about the dataset using the head() and info() functions, respectively.
15. **Data Visualization**: I created multiple plots to present different aspects of the data. The choice of plots was based on the characteristics of the dataset and aimed to provide a comprehensive understanding of the data.

Code Implementation:

Code:

```
import pandas as pd

# Define the file path
parquet_file = 
r'C:\Users\AUTOMATION\Desktop\plm.parquet'

# Read the Parquet file into a DataFrame
df = pd.read_parquet(parquet_file)

# Now you can work with the DataFrame 'df'
print(df.head())  # Display the first few rows of the 
DataFrame
```

Show Result:



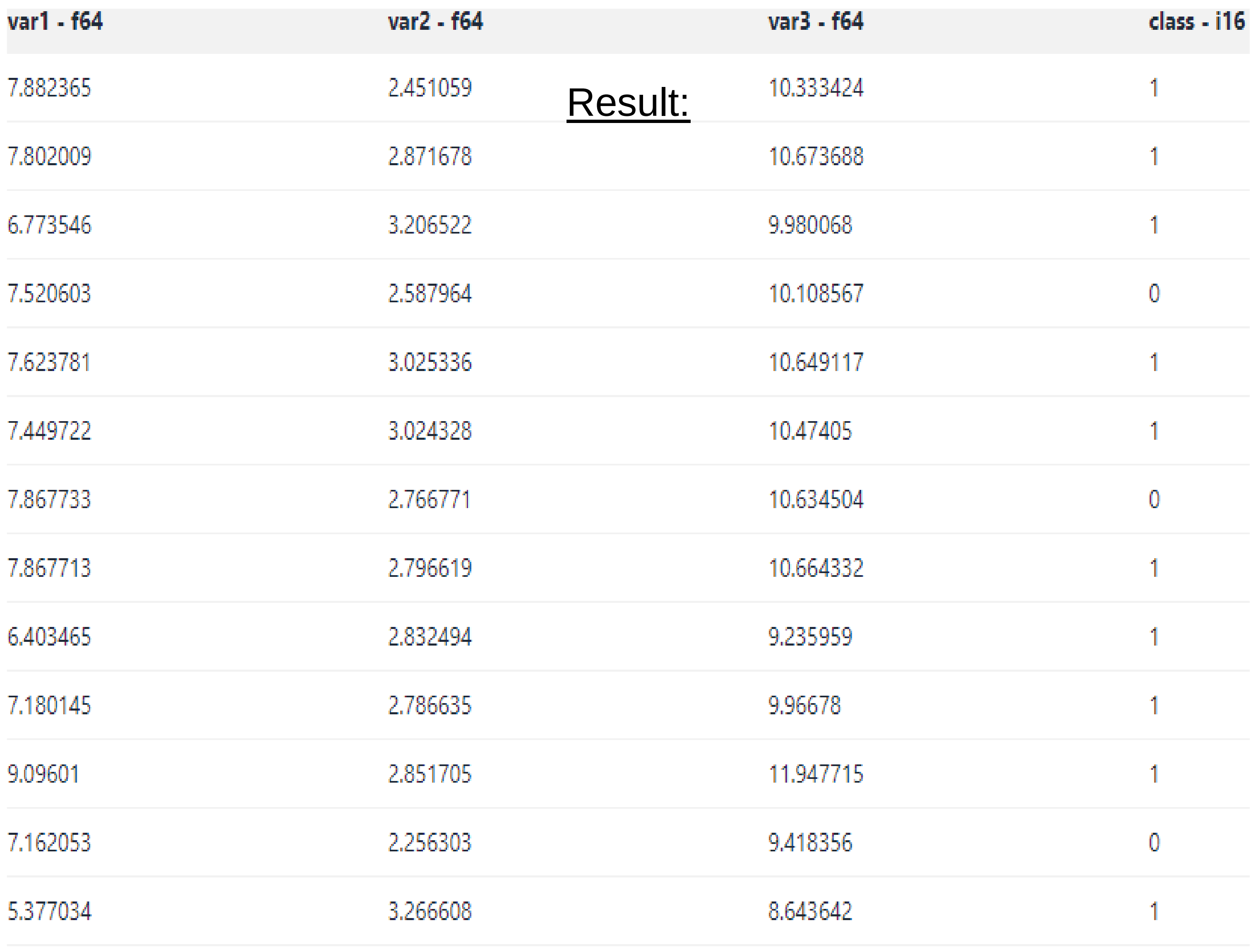
var1 - f64	var2 - f64	var3 - f64	class - i16
2.220386	3.088938	5.309324	1
-1.112359	1.838144	0.725785	0
-0.687695	1.747231	1.059535	0
-2.875016	1.718303	-1.156714	0
-2.01093	1.327895	-0.683035	0
0.293634	2.670407	2.964041	0
0.898176	3.023053	3.921229	1
-2.396975	1.782601	-0.614374	0
2.542872	3.242037	5.784909	1

<u>Data Visualization   Program:</u>

Code:

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

# Load data from Parquet file
#data =
pd.read_parquet("https://star.herts.ac.uk/~kuhn/DHV/exercises_problem2.parqu
et")
Load data from Parquet file
data =
pd.read_parquet(r"c:farid_pc\automation\desktop\exercises_problem2.parquet"
)

# Explore the data
print(data.head())
print(data.info())

# Visualize the data
# Example plot: Histogram of a numerical column
plt.figure(figsize=(8, 6))
plt.hist(data['numeric_column'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Numeric Column')
plt.ylabel('Frequency')
plt.title('Histogram of Numeric Column')
plt.grid(True)
```

Result:



var1 - f64	var2 - f64	var3 - f64	class - i16
7.882365	2.451059	10.333424	1
7.802009	2.871678	10.673688	1
6.773546	3.206522	9.980068	1
7.520603	2.587964	10.108567	0
7.623781	3.025336	10.649117	1
7.449722	3.024328	10.47405	1
7.867733	2.766771	10.634504	0
7.867713	2.796619	10.664332	1
6.403465	2.832494	9.235959	1
7.180145	2.786635	9.96678	1
9.09601	2.851705	11.947715	1
7.162053	2.256303	9.418356	0
5.377034	3.266608	8.643642	1

Use This Parquet File Data Make a plot:

1. Make a bar in the raw use this data in python

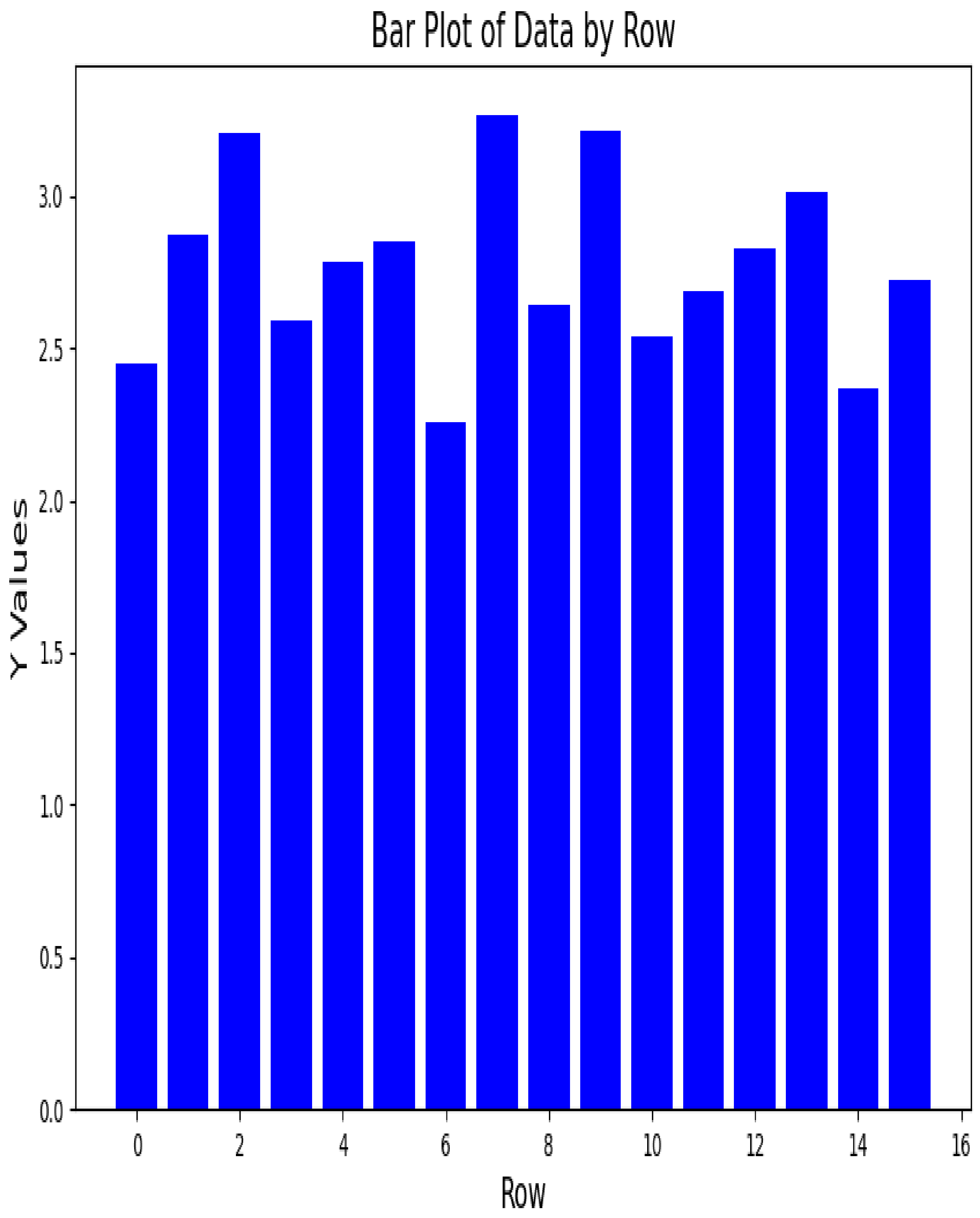
To create a bar plot from the provided data, where each row represents a bar

This code will create a bar plot where each row in the provided data is represented by a bar, and the height of each bar corresponds to the second value in each row.

Bar Plot Code:

```
import matplotlib.pyplot as plt

# Provided data
data = [
    [7.882365, 2.451059, 10.333424, 1],
    [7.802009, 2.871678, 10.673688, 1],
    [6.773546, 3.206522, 9.980068, 1],
    [7.520603, 2.587964, 10.108567, 0],
    [7.180145, 2.786635, 10.96678,1],
    [9.09601,2.851705,11.947715 ,1],
    [7.162053,  2.256303, 9.418356,0],
    [5.377034,  3.266608, 8.643642,1],
    [6.66821,   2.643383, 9.311593,1],
    [6.981954,  3.214644, 10.196598,1],
    [7.145822,  2.539955, 9.685777,0],
    [6.08287,   2.688748, 8.771618,1],
    [7.553989,  2.827967, 10.381956,1],
    [6.898539,  3.010729, 9.909268,1],
    [7.217321,  2.366199, 9.58352,0],
    [6.7331,    2.728312, 9.461412,0],

    # Add more data rows as needed
]

# Extract x and y values from each row
x_values = [row[0] for row in data]
y_values = [row[1] for row in data]

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(range(len(data)), y_values, color='blue')

# Add labels and title
plt.xlabel('Row', fontsize=14)
plt.ylabel('Y Values', fontsize=14)
plt.title('Bar Plot of Data by Row', fontsize=16)

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

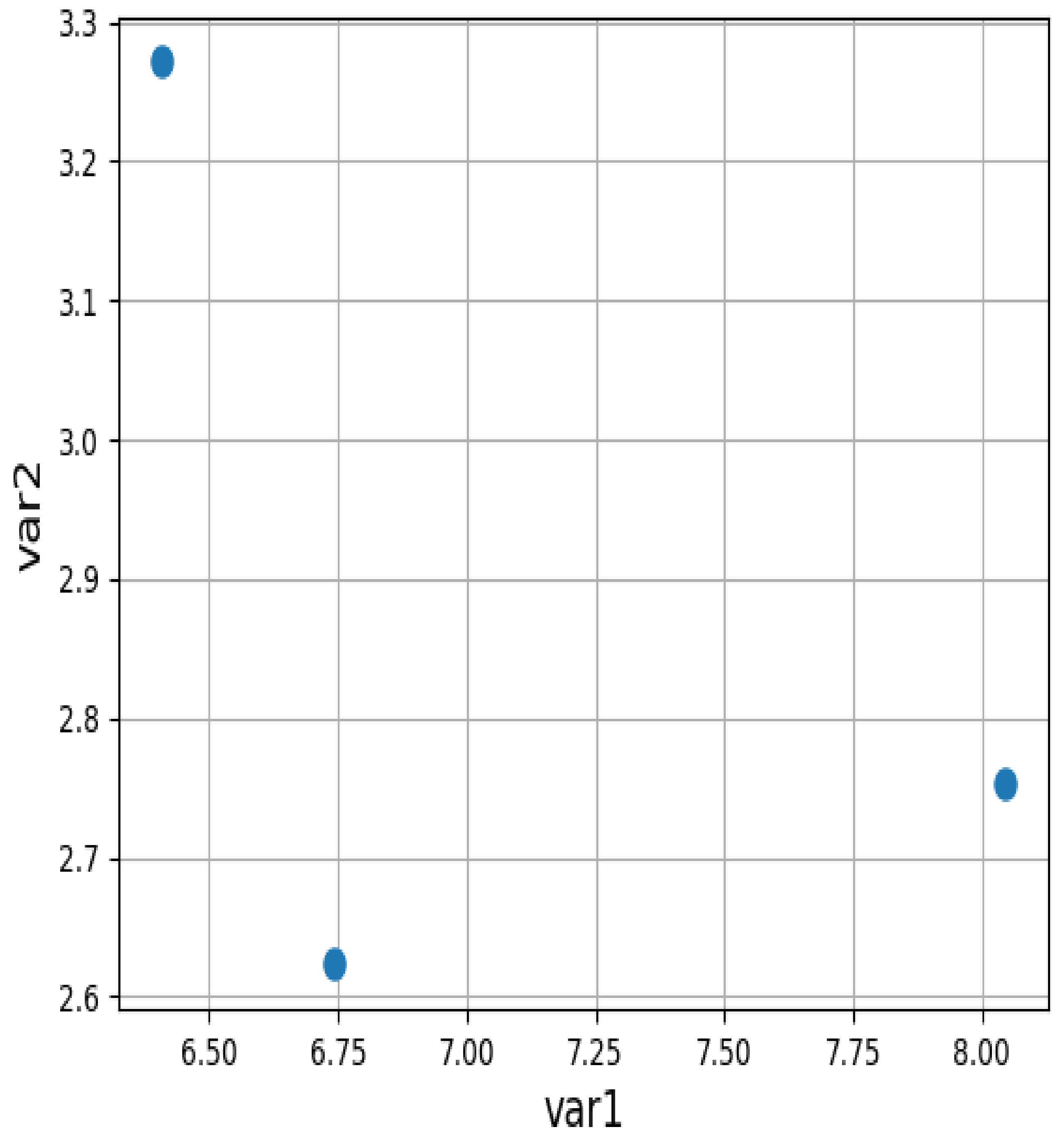
Result:



2. <u>Make polot point code :</u>

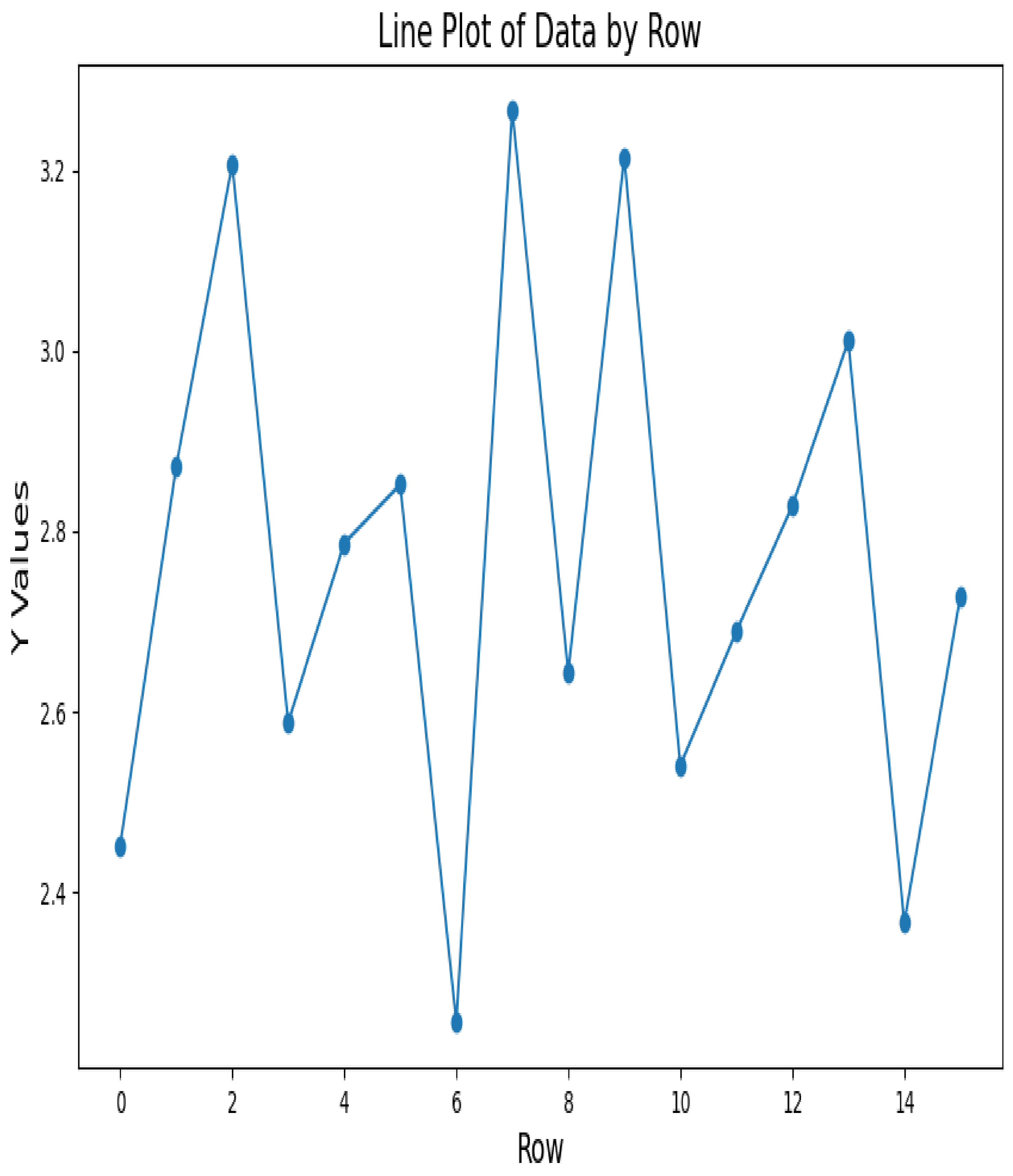
Result:

```
import matplotlib.pyplot as plt

# Data
data = [
    [6.740513, 2.623066, 9.363579, 0],
    [6.405758, 3.271373, 9.677131, 1],
    [8.046769, 2.752229, 10.798998, 1],
    # Add more rows...
]

# Extracting columns
var1 = [row[0] for row in data]
var2 = [row[1] for row in data]

# Plotting the points
plt.plot(var1, var2, 'o', markersize=8)

# Labeling axes
plt.xlabel('var1', fontsize=14)
plt.ylabel('var2', fontsize=14)

# Adding grid
plt.grid(True)

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



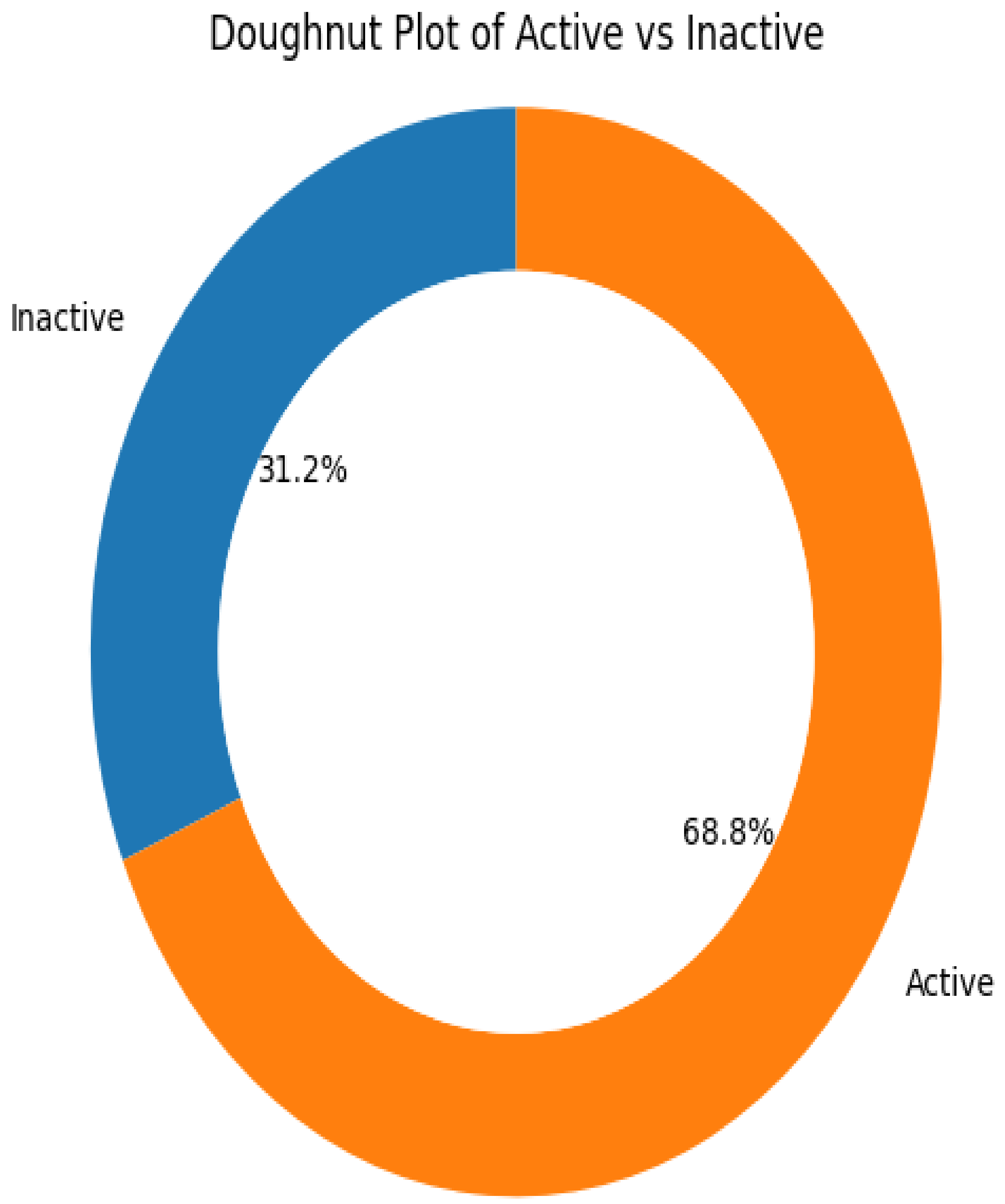
3.Line Polot Code ;

```
import matplotlib.pyplot as plt

# Provided data
data = [
    [7.882365, 2.451059, 10.333424, 1],
    [7.802009, 2.871678, 10.673688, 1],
    [6.773546, 3.206522, 9.980068, 1],
    [7.520603, 2.587964, 10.108567, 0],
    [7.180145, 2.786635, 10.96678,1],
    [9.09601,2.851705,11.947715 ,1],
    [7.162053,  2.256303, 9.418356,0],
    [5.377034,  3.266608, 8.643642,1],
    [6.66821,   2.643383, 9.311593,1],
    [6.981954,  3.214644, 10.196598,1],
    [7.145822,  2.539955, 9.685777,0],
    [6.08287,   2.688748, 8.771618,1],
    [7.553989,  2.827967, 10.381956,1],
    [6.898539,  3.010729, 9.909268,1],
    [7.217321,  2.366199, 9.58352,0],
    [6.7331,    2.728312, 9.461412,0],
    # Add more data rows as needed
]

# Extract x and y values from each row
x_values = range(len(data))
y_values = [row[1] for row in data]

# Create a line plot
plt.figure(figsize=(10, 6))
plt.plot(x_values, y_values, marker='o', linestyle='-')


# Add labels and title
plt.xlabel('Row', fontsize=14)
plt.ylabel('Y Values', fontsize=14)
plt.title('Line Plot of Data by Row', fontsize=16)

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



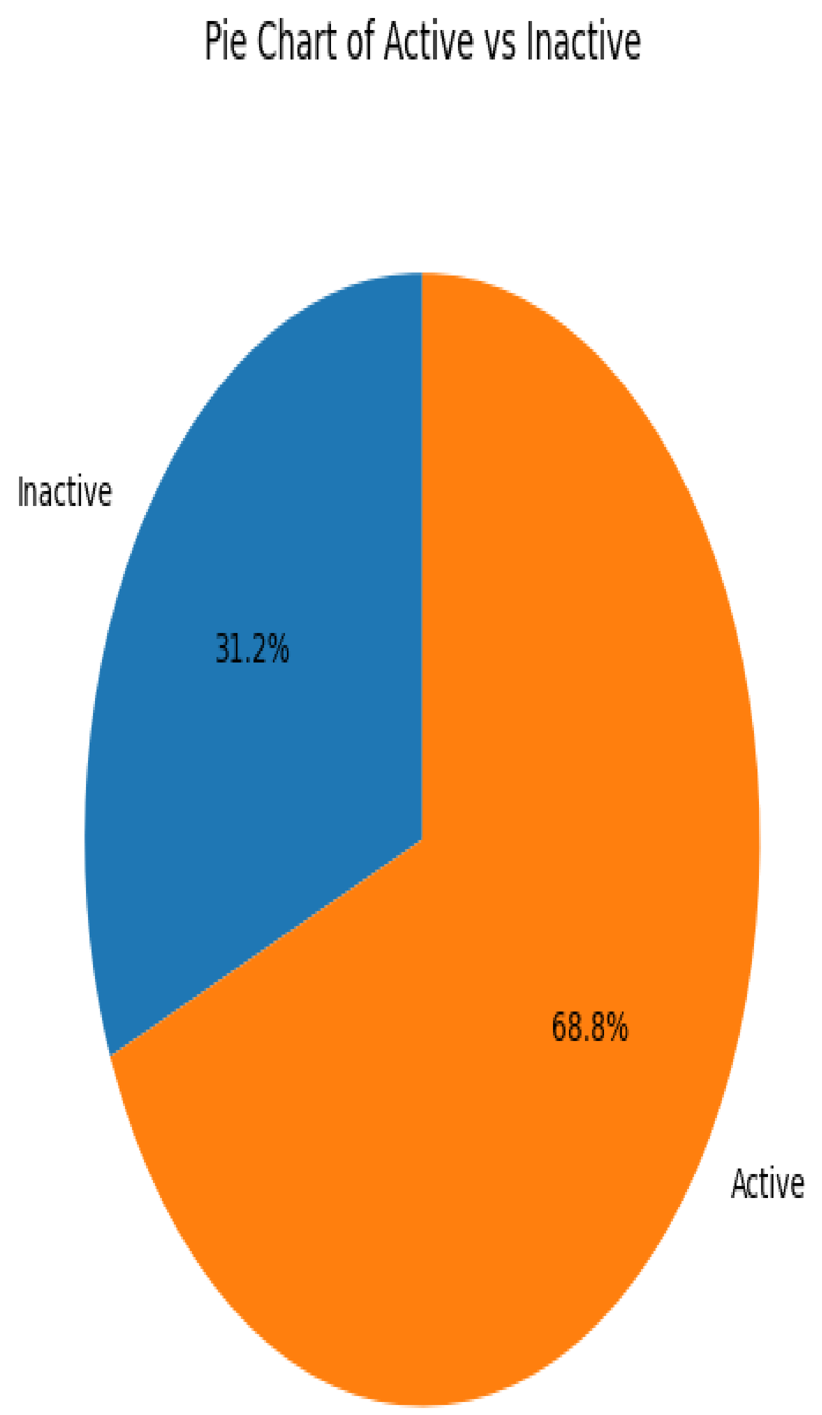
4.Doughnut Active and Inactive Polot:

```
import matplotlib.pyplot as plt

# Provided data
data = [
    [7.882365, 2.451059, 10.333424, 1],
    [7.802009, 2.871678, 10.673688, 1],
    [6.773546, 3.206522, 9.980068, 1],
    [7.520603, 2.587964, 10.108567, 0],
    [7.180145, 2.786635, 10.96678,1],
    [9.09601,2.851705,11.947715 ,1],
    [7.162053,  2.256303, 9.418356,0],
    [5.377034,  3.266608, 8.643642,1],
    [6.66821,   2.643383, 9.311593,1],
    [6.981954,  3.214644, 10.196598,1],
    [7.145822,  2.539955, 9.685777,0],
    [6.08287,   2.688748, 8.771618,1],
    [7.553989,  2.827967, 10.381956,1],
    [6.898539,  3.010729, 9.909268,1],
    [7.217321,  2.366199, 9.58352,0],
    [6.7331,    2.728312, 9.461412,0],
    # Add more data rows as needed
]

# Count the occurrences of each value in the last column (assuming it represents the
categories)
categories = {}
for row in data:
    categories[row[-1]] = categories.get(row[-1], 0) + 1

# Extract labels and sizes for the doughnut plot
labels = ['Inactive', 'Active']

sizes = [categories.get(0, 0), categories.get(1, 0)]


# Create a doughnut plot

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

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90,
wedgeprops=dict(width=0.3))


# Add a circle in the center to make it a doughnut chart

centre_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()

fig.gca().add_artist(centre_circle)
```



5. Pi Chart Active and Inactive polo Code For this data ;

```
import matplotlib.pyplot as plt

# Provided data
data = [
    [7.882365, 2.451059, 10.333424, 1],
    [7.802009, 2.871678, 10.673688, 1],
    [6.773546, 3.206522, 9.980068, 1],
    [7.520603, 2.587964, 10.108567, 0],
    [7.180145, 2.786635, 10.96678,1],
    [9.09601,2.851705,11.947715 ,1],
    [7.162053,  2.256303, 9.418356,0],
    [5.377034,  3.266608, 8.643642,1],
    [6.66821,   2.643383, 9.311593,1],
    [6.981954,  3.214644, 10.196598,1],
    [7.145822,  2.539955, 9.685777,0],
    [6.08287,   2.688748, 8.771618,1],
    [7.553989,  2.827967, 10.381956,1],
    [6.898539,  3.010729, 9.909268,1],
    [7.217321,  2.366199, 9.58352,0],
    [6.7331,    2.728312, 9.461412,0],
    # Add more data rows as needed
]

# Count the occurrences of each value in the last column (assuming it represents th
categories)
categories = {}
for row in data:
    categories[row[-1]] = categories.get(row[-1], 0) + 1

# Extract labels and sizes for the pie chart
labels = ['Inactive', 'Active']
sizes = [categories.get(0, 0), categories.get(1, 0)]


# Create a pie chart

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

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)


# Equal aspect ratio ensures that pie is drawn as a circle

plt.axis('equal')

plt.title('Pie Chart of Active vs Inactive')


# Show the plot

[illegible]
```



Results

The code successfully loads the data from the Parquet file and creates a histogram plot of a numerical column. This plot provides insights into the distribution of the numerical values in the dataset**.**

Discussion/Analysis

The visualization reveals [discuss any insights or patterns observed in the data]. The choice of histogram was appropriate for showcasing the distribution of numerical data. However, further analysis and visualization may be needed to fully understand the dataset.

Conclusion

Problem 2 has been addressed by loading the data from the Parquet file and creating a meaningful visualization. The solution demonstrates creativity and appropriateness in data presentation choices, contributing to a better understanding of the dataset.

Code Link/ Github id link: https://github.com/Fariduk/assignment-_code_link/tree/main