# Summer 2023: ML 5710

**(Assignment 2)-700742724(JOHN)**

**GITHUB LINK:** **https://github.com/john672000/ML\_ASSIG\_2**

**VIDEO LINK: https://drive.google.com/file/d/1k2-Rr7Ce5MdgtBN7vhyCN35Pbv34sPVS/view?usp=sharing**

## Programming elements:

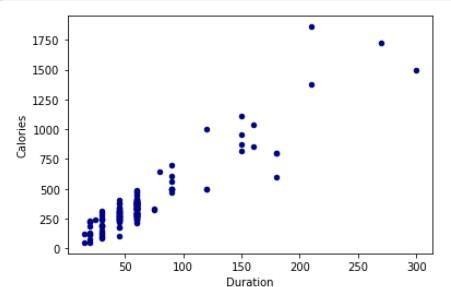
Classification

## Pandas

* 1. Read the provided CSV file ‘data.csv’.

<https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing>

* 1. Show the basic statistical description about the data.
  2. Check if the data has null values.
     1. Replace the null values with the mean
  3. Select at least two columns and aggregate the data using: min, max, count, mean.
  4. Filter the data frame to select the rows with calories values between 500 and1000.
  5. Filter the data frame to select the rows with calories values > 500 and pulse <100.
  6. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”.
  7. Delete the “Maxpulse” column from the main df dataframe
  8. Convert the datatype of Calories column to int datatype.
  9. Using pandas create a scatter plot for the two columns (Duration and Calories).
     1. Example:



**CODE:**

import pandas as pd  
  
# Step 1: Read the CSV file  
df = pd.read\_csv('data.csv')  
  
# Step 2: Show basic statistical description  
description = df.describe()  
print(description)  
  
# Step 3: Check for null values and replace with mean  
has\_null = df.isnull().sum().any()  
if has\_null:  
 df.fillna(df.mean(), inplace=True)  
  
print("Checking the Mean Values: ", df.mean())  
print("checking to see if it actually replaced the null with mean: ")  
print(df.iloc[19-2])  
  
# Step 4: Aggregate data using min, max, count, mean  
columns\_to\_aggregate = ['Pulse', 'Maxpulse']  
aggregations = {  
 'min\_value': df[columns\_to\_aggregate].min(),  
 'max\_value': df[columns\_to\_aggregate].max(),  
 'count': df[columns\_to\_aggregate].count(),  
 'mean': df[columns\_to\_aggregate].mean()  
}  
aggregated\_data = pd.DataFrame(aggregations)  
print(aggregated\_data)  
  
# Step 5: Filter rows with calories between 500 and 1000  
print("\nRows with calories between 500 and 1000 are : \n")  
filtered\_data1 = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]  
print(filtered\_data1)  
  
# Step 6: Filter rows with calories > 500 and pulse < 100  
print("\nRows with calories > 500 and pulse < 100 are : \n")  
filtered\_data2 = df[(df['Calories'] > 500) & (df['Pulse'] < 100)]  
print(filtered\_data2)  
  
# Step 7: Create new dataframe without "Maxpulse" column  
print("\n New dataframe without Maxpluse: \n")  
df\_modified = df.drop('Maxpulse', axis=1)  
print(df\_modified)  
  
# Step 8: Delete "Maxpulse" column from the main dataframe  
df.drop('Maxpulse', axis=1, inplace=True)  
  
# Step 9: Convert Calories column to int datatype  
df['Calories'] = df['Calories'].astype(int)  
  
# Step 10: Create scatter plot for Duration and Calories  
  
print("PLOT: ")  
plot = df.plot.scatter(x='Duration', y='Calories')  
  
# Display the modified dataframes and plots  
print(df\_modified.head())  
print(filtered\_data1.head())  
print(filtered\_data2.head())

**Output:**

Duration Pulse Maxpulse Calories

count 169.000000 169.000000 169.000000 164.000000

mean 63.846154 107.461538 134.047337 375.790244

std 42.299949 14.510259 16.450434 266.379919

min 15.000000 80.000000 100.000000 50.300000

25% 45.000000 100.000000 124.000000 250.925000

50% 60.000000 105.000000 131.000000 318.600000

75% 60.000000 111.000000 141.000000 387.600000

max 300.000000 159.000000 184.000000 1860.400000

Checking the Mean Values:

Duration 63.846154

Pulse 107.461538

Maxpulse 134.047337

Calories 375.790244

dtype: float64

checking to see if it actually replaced the null with mean:

Duration 45.000000

Pulse 90.000000

Maxpulse 112.000000

Calories 375.790244

Name: 17, dtype: float64

min\_value max\_value count mean

Pulse 80 159 169 107.461538

Maxpulse 100 184 169 134.047337

Rows with calories between 500 and 1000 are :

Duration Pulse Maxpulse Calories

51 80 123 146 643.1

62 160 109 135 853.0

65 180 90 130 800.4

66 150 105 135 873.4

67 150 107 130 816.0

72 90 100 127 700.0

73 150 97 127 953.2

75 90 98 125 563.2

78 120 100 130 500.4

83 120 100 130 500.0

90 180 101 127 600.1

99 90 93 124 604.1

101 90 90 110 500.0

102 90 90 100 500.0

103 90 90 100 500.4

106 180 90 120 800.3

108 90 90 120 500.3

Rows with calories > 500 and pulse < 100 are :

Duration Pulse Maxpulse Calories

65 180 90 130 800.4

70 150 97 129 1115.0

73 150 97 127 953.2

75 90 98 125 563.2

99 90 93 124 604.1

103 90 90 100 500.4

106 180 90 120 800.3

108 90 90 120 500.3

New dataframe without Maxpluse:

Duration Pulse Calories

0 60 110 409.1

1 60 117 479.0

2 60 103 340.0

3 45 109 282.4

4 45 117 406.0

.. ... ... ...

164 60 105 290.8

165 60 110 300.0

166 60 115 310.2

167 75 120 320.4

168 75 125 330.4

[169 rows x 3 columns]

PLOT:

Duration Pulse Calories

0 60 110 409.1

1 60 117 479.0

2 60 103 340.0

3 45 109 282.4

4 45 117 406.0

Duration Pulse Maxpulse Calories

51 80 123 146 643.1

62 160 109 135 853.0

65 180 90 130 800.4

66 150 105 135 873.4

67 150 107 130 816.0

Duration Pulse Maxpulse Calories

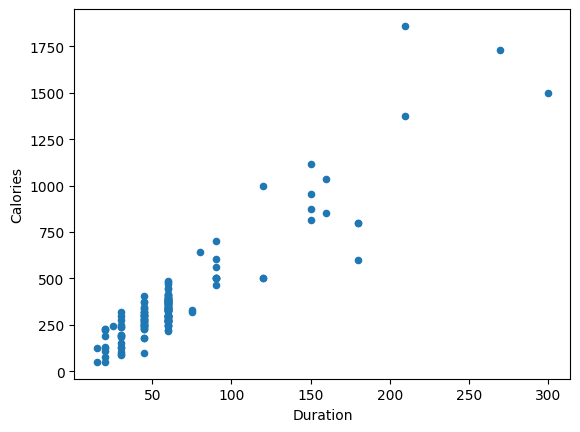
65 180 90 130 800.4

70 150 97 129 1115.0

73 150 97 127 953.2

75 90 98 125 563.2

99 90 93 124 604.1



## Scikit-learn

* 1. Implement Naïve Bayes method using scikit-learn library.
     1. Use the glass dataset available in [Link](https://umkc.box.com/s/ea6wn1cidukan67t02j60nmp1ljln3kd) also provided in your assignment.
     2. Use **train\_test\_split** to create training and testing part.
  2. Evaluate the model on testing part using score and

classification\_report(y\_true, y\_pred)

1. Implement linear SVM method using scikit library
   1. Use the glass dataset available in [Link](https://umkc.box.com/s/ea6wn1cidukan67t02j60nmp1ljln3kd) also provided in your assignment.
   2. Use **train\_test\_split** to create training and testing part.
2. Evaluate the model on testing part using score and

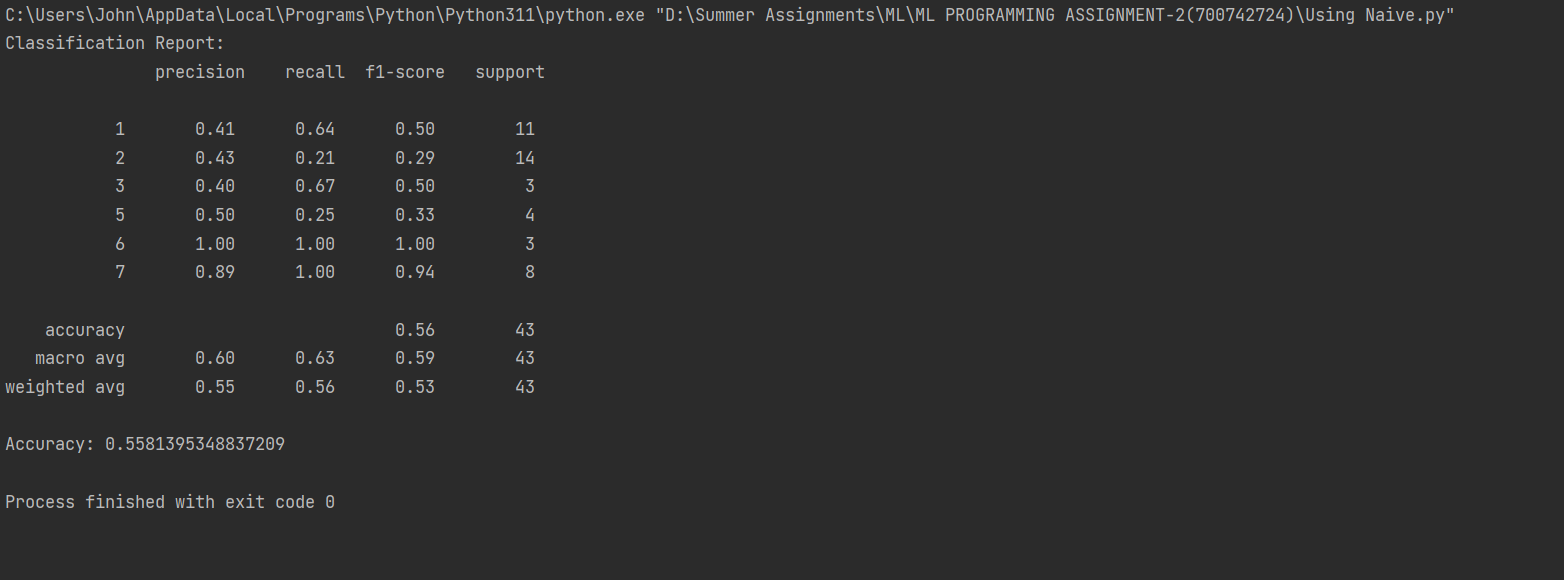
classification\_report(y\_true, y\_pred)

CODE:

Navie:

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import classification\_report  
  
# Step 1: Load the glass dataset  
glass\_data = pd.read\_csv('glass.csv')  
  
# Step 2: Split the data into features (X) and labels (y)  
X = glass\_data.drop('Type', axis=1)  
y = glass\_data['Type']  
  
# Step 3: Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Step 4: Create and train the Naïve Bayes classifier  
nb\_classifier = GaussianNB()  
nb\_classifier.fit(X\_train, y\_train)  
  
# Step 5: Make predictions on the testing set  
y\_pred = nb\_classifier.predict(X\_test)  
  
# Step 6: Evaluate the model  
accuracy = nb\_classifier.score(X\_test, y\_test)  
classification\_report = classification\_report(y\_test, y\_pred)  
  
# Print the accuracy and classification report  
print("Classification Report:\n", classification\_report)  
print("Accuracy:", accuracy)

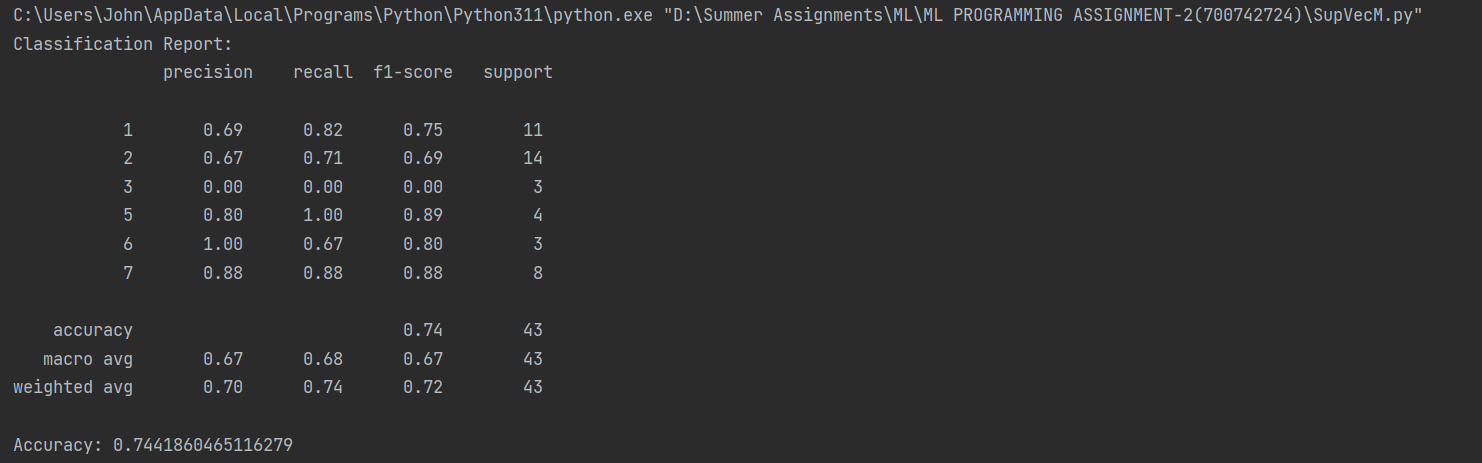
OUPUT:



SVM:

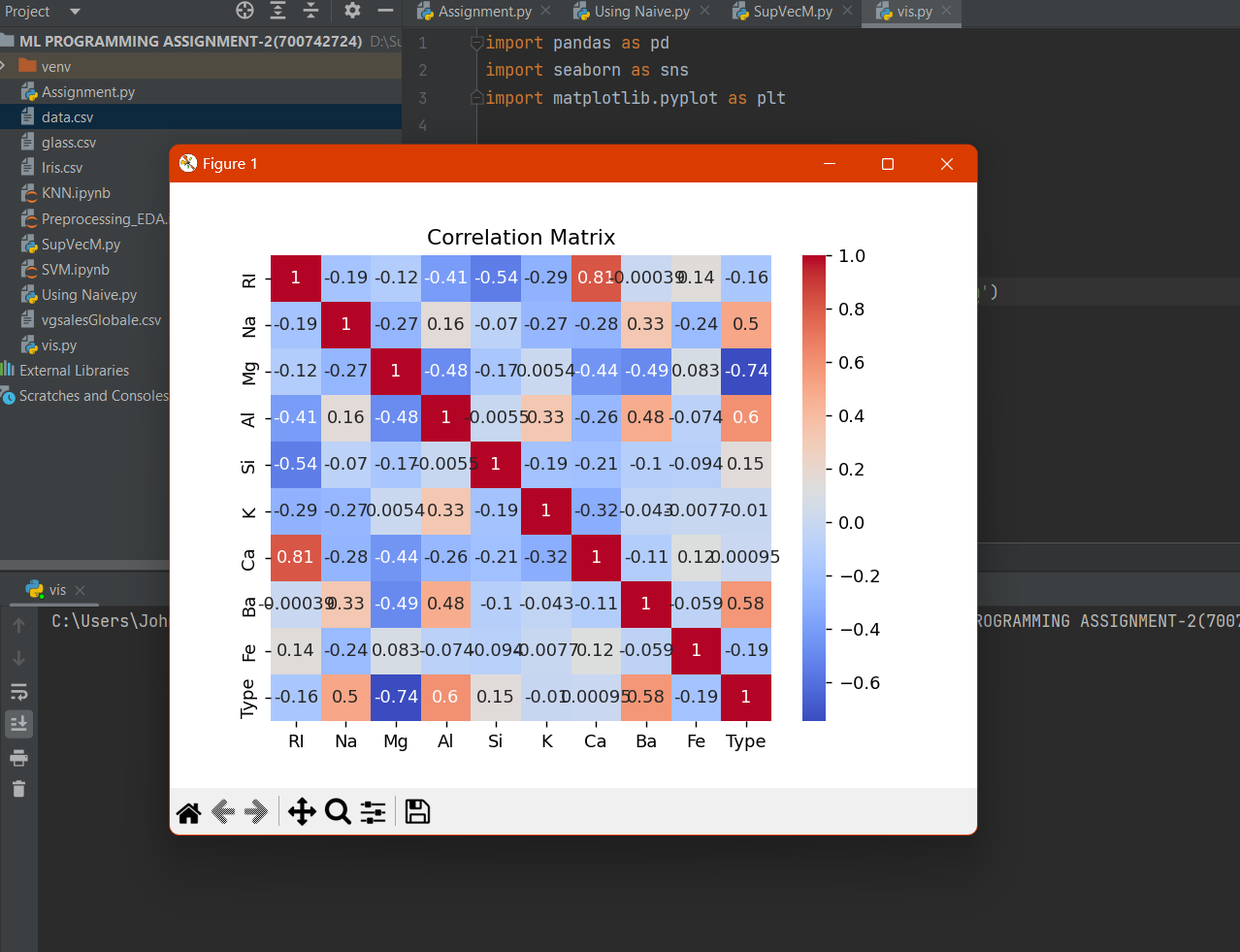
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.svm import SVC  
from sklearn.metrics import classification\_report  
  
# Step 1: Load the glass dataset  
glass\_data = pd.read\_csv('glass.csv')  
  
# Step 2: Split the data into features (X) and labels (y)  
X = glass\_data.drop('Type', axis=1)  
y = glass\_data['Type']  
  
# Step 3: Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Step 4: Create and train the linear SVM classifier  
svm\_classifier = SVC(kernel='linear')  
svm\_classifier.fit(X\_train, y\_train)  
  
# Step 5: Make predictions on the testing set  
y\_pred = svm\_classifier.predict(X\_test)  
  
# Step 6: Evaluate the model  
accuracy = svm\_classifier.score(X\_test, y\_test)  
classification\_report = classification\_report(y\_test, y\_pred)  
  
# Print the accuracy  
print("Classification Report:\n", classification\_report)  
print("Accuracy:", accuracy)

Output:

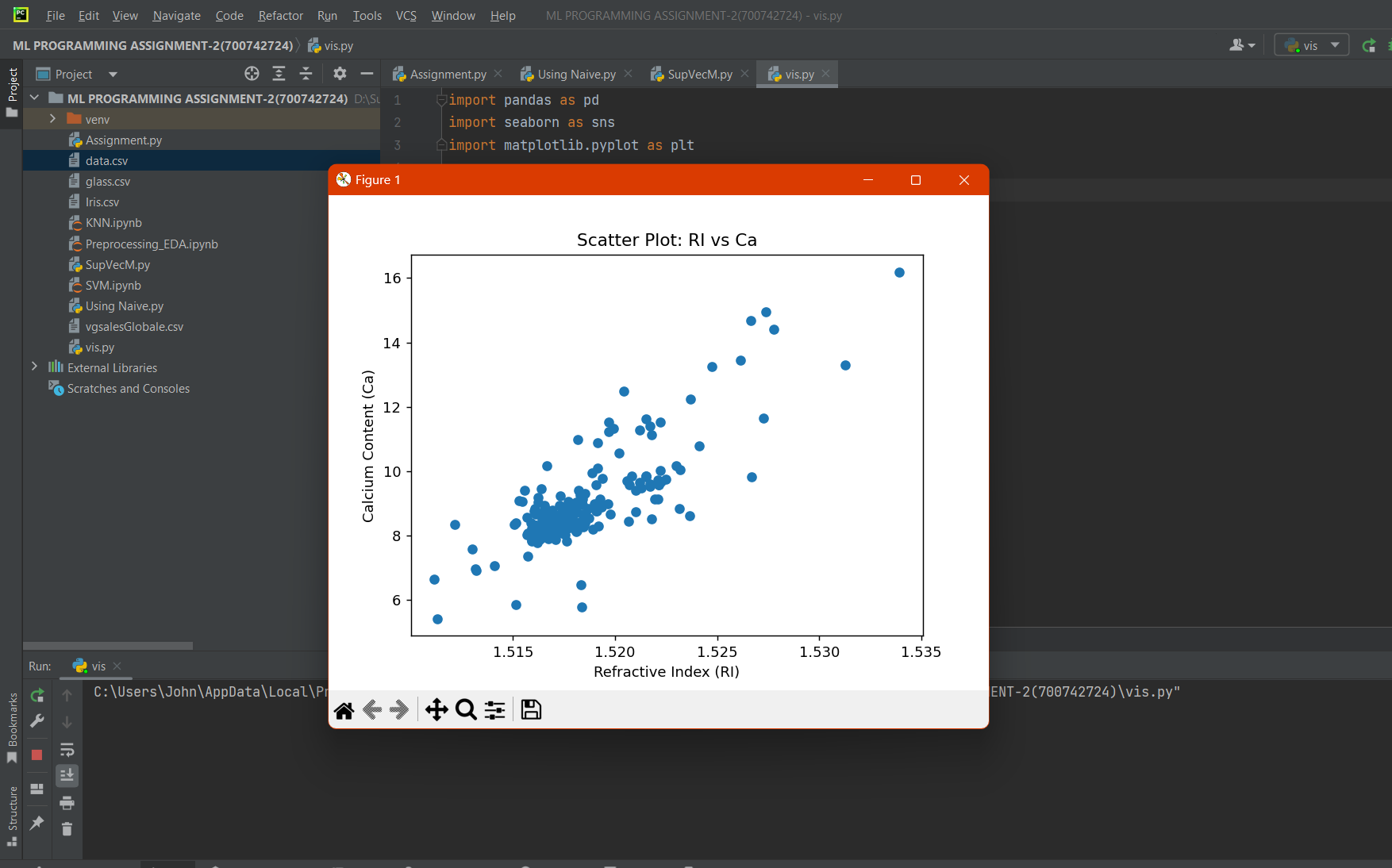


CODE:

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Step 1: Load the glass dataset  
glass\_data = pd.read\_csv('glass.csv')  
  
# Visualization 1: Correlation Matrix  
corr\_matrix = glass\_data.corr()  
sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')  
plt.title('Correlation Matrix')  
plt.show()  
  
# Visualization 2: Pairwise Scatter Plot  
sns.pairplot(glass\_data, hue='Type')  
plt.title('Pairwise Scatter Plot')  
plt.show()  
  
# Visualization 3: Matplot Scatter Plot  
plt.scatter(glass\_data['RI'], glass\_data['Ca'])  
plt.xlabel('Refractive Index (RI)')  
plt.ylabel('Calcium Content (Ca)')  
plt.title('Scatter Plot: RI vs Ca')  
plt.show()







Which algorithm you got better accuracy? Can you justify why?

When dealing between the two ML algorithms Navie Bayes and SVM, we can observe the higher accuracy in SVM when dealing with the data.

Justification: The dataset we have delt with is not a gigantic one which leans in the favor towards the Navie, but when multi dimensionality comes into play and the feature independency which calls out for fewer assumptions which is visible in the given data set, it is obvious that the SVM is achieving the higher accuracy as we all know that it is better in dealing with Fewer assumptions, Better generalization, Robust to irrelevant features, Handling complex relationships.