ML Assignment 2

Hima Bindu Desalanka

700739704

1. Pandas

1. Read the provided CSV file 'data.csv'. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing

2. Show the basic statistical description about the data.

3. Check if the data has null values

. a. Replace the null values with the mean

4. Select at least two columns and aggregate the data using: min, max, count, mean.

5. Filter the dataframe to select the rows with calories values between 500 and1000.

6. Filter the dataframe to select the rows with calories values > 500 and pulse

7. Create a new "df_modified" dataframe that contains all the columns from df except for "Maxpulse".

8. Delete the "Maxpulse" column from the main df dataframe

9. Convert the datatype of Calories column to int datatype.

10. Using pandas create a scatter plot for the two columns (Duration and Calories).

https://drive.google.com/drive/folders/1Fjle7i7Nagu-vEJO8SJ3tuB3On6pob7p?usp=sharing ---> Video link
https://github.com/hbdesalanka/assignment2 --> github link

In [1]:
```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_report, confusion_matrix
warnings.filterwarnings("ignore")
```

In [2]:
```
#Read the provided CSV file 'data.csv'
df = pd.read_csv("data.csv")
df.head()
```

Out[2]:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

In [3]:
```
#description about the data.
df.describe()
```

Out[3]:

	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

In [4]:
```
#if the data has null values.
df.isnull().any()
```

Out[4]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

In [5]:
```
#Replace the null values with the mean
df.fillna(df.mean(), inplace=True)
df.isnull().any()
```

Out[5]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

In [6]:
```
#Select at least two columns and aggregate the data using: min, max, count, mean.
df.agg({'Maxpulse':['min','max','count','mean'],'Calories':['min','max','count','mean']})
```

Out[6]:

	Maxpulse	Calories
min	100.000000	50.300000
max	184.000000	1860.400000
count	169.000000	169.000000
mean	134.047337	375.790244

In [7]:
```
#Filter the dataframe to select the rows with calories values between 500 and 1000.
df.loc[(df['Calories']>500)&(df['Calories']<1000)]
```

Out[7]:

	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
90	180	101	127	600.1
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

In [8]:
```
#Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
df.loc[(df['Calories']>500)&(df['Pulse']<100)]
```

Out[8]:

	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

In [9]:
```
#Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.
df_modified = df[['Duration','Pulse','Calories']]
df_modified.head()
```

Out[9]:

	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

In [10]:
```
#Delete the "Maxpulse" column from the main df dataframe
del df['Maxpulse']
```

In [11]:
```
#To display the first few rows of the table
df.head()
```

Out[11]:

	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

In [12]:
```
#To display the types of the rows
df.dtypes
```

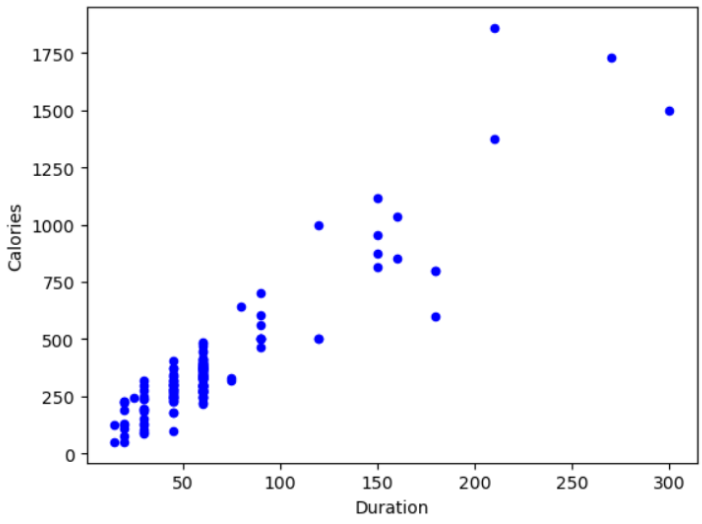
Out[12]:
```
Duration      int64
Pulse         int64
Calories    float64
dtype: object
```

In [13]:
```
#Convert the datatype of Calories column to int datatype.
df['Calories'] = df['Calories'].astype(np.int64)
df.dtypes
```

Out[13]:
```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

In [14]:
```
#Using pandas create a scatter plot for the two columns (Duration and Calories).
df.plot.scatter(x='Duration',y='Calories',c='blue')
```

Out[14]: <Axes: xlabel='Duration', ylabel='Calories'>



The program imports the required libraries for data processing, error handling, data visualization, and machine learning.A dataset is kept in a DataFrame with the name df after being loaded from a CSV file.The code displays the first five rows of the DataFrame to give a quick overview of the data.In the numerical columns,

It computes descriptive statistics like count, mean, standard deviation, minimum, quartiles, and maximum using the DataFrame.A boolean value (True/False) is returned for each column to indicate whether or not it has any missing values after the program searches the DataFrame for any.If any data is missing, the code fills in the gaps with the mean value of each column. It checks again to see if any missing values are still present after handling them. The code combines data for the "Maxpulse" and "Calories" columns of the DataFrame, including minimum, maximum, count, and mean.The DataFrame is filtered based on preset criteria, such as selecting rows where the 'Calories' column is greater than 500 and less than 1000, or where 'Calories' is larger than 500 and 'Pulse' is less than 100.The only columns in the brand-new DataFrame, df_modified, are the duration, pulse, and calories columns from the original DataFrame. The first few rows of this modified DataFrame are displayed.The 'Maxpulse' column in the DataFrame is deleted.The code displays data and changes the 'Calories' column's data type to a 64-bit integer type (int64).

2. Scikit-learn

1. Implement Naïve Bayes method using scikit-learnlibrary.

a. Use the glass dataset available in Link also provided in your assignment.

b. 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

a. Use the glass dataset available in Link also provided in your assignment.

b. Use train_test_split to create training and testing part.

2. Evaluate the model on testing part using score and Do at least two visualizations to describe or show correlations in the Glass Dataset.

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

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn.svm import LinearSVC
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the glass dataset
glass_data = pd.read_csv('glass.csv')
```

In [3]:
```
# Split the dataset into features (X) and target variable (y)
X = glass_data.drop('Type', axis=1)
y = glass_data['Type']
```

In [4]:
```
# Split the dataset 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)
```

In [5]:
```
# Create a Naïve Bayes classifier
nb_classifier = GaussianNB()
```

In [6]:
```
# Train the classifier
nb_classifier.fit(X_train, y_train)
```

Out[6]:
```
GaussianNB
GaussianNB()
```

In [7]:
```
# Predict the labels for the test set
y_pred = nb_classifier.predict(X_test)
```

In [8]:
```
# Evaluate the model
accuracy = nb_classifier.score(X_test, y_test)
report = classification_report(y_test, y_pred)
```

In [9]:
```
print("Accuracy:", accuracy)
print("Classification Report:")
print(report)
```

```
Accuracy: 0.5581395348837209
Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

In [10]:
```
# Create a linear SVM classifier
svm_classifier = LinearSVC(max_iter=1000000)
```

In [11]:
```
# Train the classifier
svm_classifier.fit(X_train, y_train)
```

Out[11]:
```
LinearSVC(max_iter=1000000)
```

In [12]:
```
# Predict the labels for the test set
y_pred_svm = svm_classifier.predict(X_test)
```

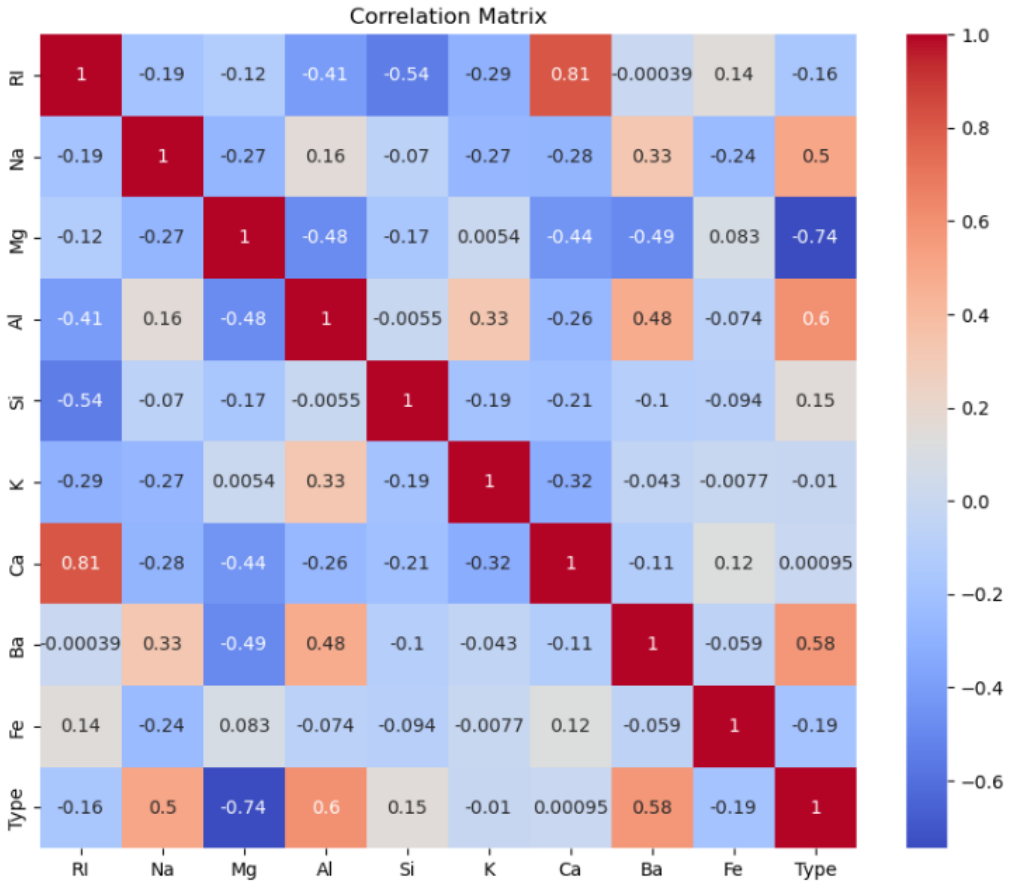
In [13]:
```
# Evaluate the model
accuracy_svm = svm_classifier.score(X_test, y_test)
report_svm = classification_report(y_test, y_pred_svm, zero_division=1)
```

In [14]:
```
print("Accuracy (Linear SVM):", accuracy_svm)
print("Classification Report (Linear SVM):")
print(report_svm)
```

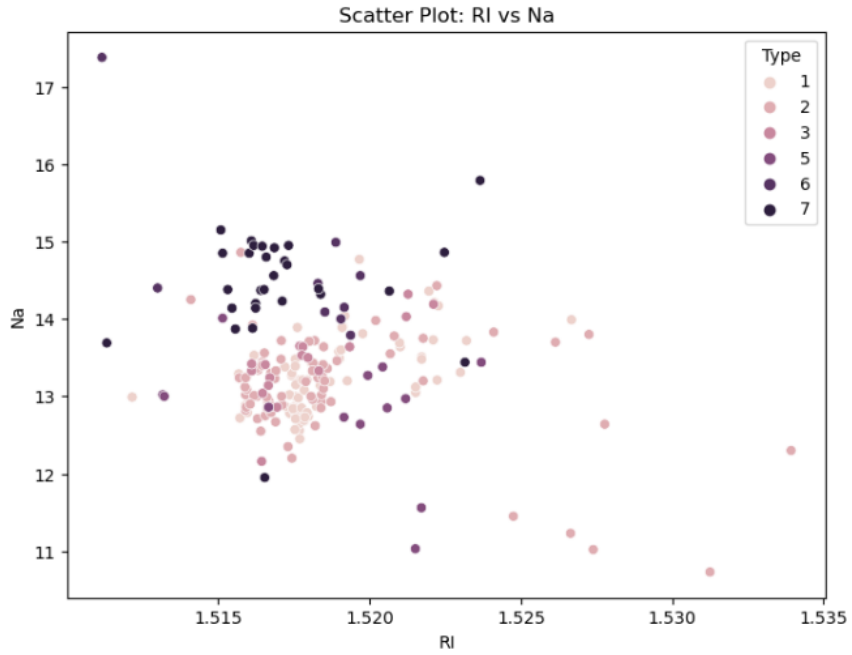
```
Accuracy (Linear SVM): 0.6511627906976745
Classification Report (Linear SVM):
              precision    recall  f1-score   support

           1       0.60      0.82      0.69        11
           2       0.53      0.57      0.55        14
           3       1.00      0.00      0.00         3
           5       1.00      0.25      0.40         4
           6       1.00      0.67      0.80         3
           7       0.80      1.00      0.89         8

    accuracy                           0.65        43
   macro avg       0.82      0.55      0.56        43
weighted avg       0.71      0.65      0.62        43
```

In [15]:
```
# Create a correlation matrix
correlation_matrix = glass_data.corr()
```

In [16]:
```
# Plot the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```


Correlation Matrix
RI 1 -0.19 -0.12 -0.41 -0.54 -0.29 0.81 -0.00039 0.14 -0.16
Na -0.19 1 -0.27 0.16 -0.07 -0.27 -0.28 0.33 -0.24 0.5
Mg -0.12 -0.27 1 -0.48 -0.17 0.0054 -0.44 -0.49 0.083 -0.74
Al -0.41 0.16 -0.48 1 -0.0055 0.33 -0.26 0.48 -0.074 0.6
Si -0.54 -0.07 -0.17 -0.0055 1 -0.19 -0.21 -0.1 -0.094 0.15
K -0.29 -0.27 0.0054 0.33 -0.19 1 -0.32 -0.043 -0.0077 -0.01
Ca 0.81 -0.28 -0.44 -0.26 -0.21 -0.32 1 -0.11 0.12 0.00095
Ba -0.00039 0.33 -0.49 0.48 -0.1 -0.043 -0.11 1 -0.059 0.58
Fe 0.14 -0.24 0.083 -0.074 -0.094 -0.0077 0.12 -0.059 1 -0.19
Type -0.16 0.5 -0.74 0.6 0.15 -0.01 0.00095 0.58 -0.19 1
RI Na Mg Al Si K Ca Ba Fe Type
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

In [17]:
```
# Create a scatter plot to show the correlation between two variables
plt.figure(figsize=(8, 6))
sns.scatterplot(x="RI", y="Na", hue="Type", data=glass_data)
plt.title("Scatter Plot: RI vs Na")
plt.show()
```



The code imports the necessary libraries, such as pandas, scikit-learn, and seaborn, for data manipulation, machine learning, and visualization.The glass dataset is loaded from a CSV file and stored in a DataFrame called glass_data.The dataset is separated into characteristics (X) and the desired result (y), where X includes all columns except for the "Type" column and y only includes the "Type" column.The dataset is further split into training and testing sets using scikit-learn's train_test_split function. For consistency, a random seed is provided, and 20% of the data is designated for testing.The Naive Bayes classifier (GaussianNB) is built using the scikit-learn GaussianNB class. .The Naive Bayes

classifier is trained on the training set using the fit approach. On the testing set, the trained Nave Bayes classifier makes predictions.The Naive Bayes classifier's accuracy is assessed using the score method, which contrasts the predicted labels with the actual labels.The classification report, which includes metrics like precision, recall, and F1- score, is produced using the classification_report function.The classification report and accuracy of the Nave Bayes classifier are printed.A linear SVM classifier (LinearSVC) is created using the scikit-learn LinearSVC class.The linear SVM classifier is trained on the training set using the fit approach. The trained linear SVM classifier makes predictions on the testing set. The accuracy and classification report from the linear SVM classifier are printed.The code then uses Seaborn and Matplotlib to carry out the data visualization process.The glass dataset is used to create a correlation matrix using the corr method, which establishes the correlation between each pair of variables.The correlation matrix is shown as a heatmap using the seaborn heatmap function. A scatter plot is used to show the relationship between the RI (refractive index) and Na (sodium) variables, with the color of each data point designating the kind of glass.The final scatter plot and correlation matrix heatmap are displayed using matplotlib.The output showed that the Linear SVM for my model has higher accuracy than the Naive Bayes.

https://drive.google.com/drive/folders/1Fjle7i7Nagu-vEJO8SJ3tuB3On6pob7p?usp=sharing ---> Video link

https://github.com/hbdesalanka/assignment2 --> github link