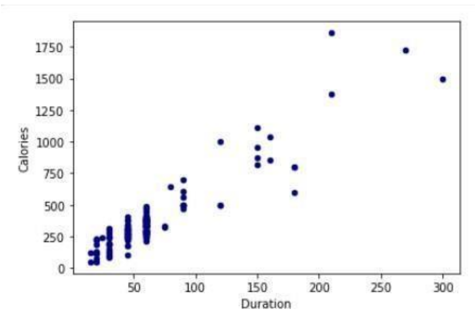
Summer 2023: ML 5710
(Assignment 2)

Anudeep Allamsetty - 700741171

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 and 1000.
6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
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).
    a. Example:



GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

In [4]:
```
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 [5]:
```
#Read the provided CSV file 'data.csv'
df = pd.read_csv("data.csv")
df.head()
```

Out[5]:

	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 [6]:
```
#description about the data.
df.describe()
```

Out[6]:

	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 [7]:
```
#if the data has null values.
df.isnull().any()
```

Out[7]:
```
Duration     False
Pulse        False
Maxpulse     False
Calories      True
dtype: bool
```

In [8]:
```
#Replace the null values with the mean
df.fillna(df.mean(), inplace=True)
df.isnull().any()
```

Out[8]:
```
Duration     False
Pulse        False
Maxpulse     False
Calories     False
dtype: bool
```

In [9]:
```
#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[9]:

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

In [10]:
```
#Filter the dataframe to select the rows with calories values between 500 and 1000.
df.loc[(df['Calories']>500)&(df['Calories']<1000)]
```

Out[10]:

	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

GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

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 [11]:
```
#Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
df.loc[(df['Calories']>500)&(df['Pulse']<100)]
```

Out[11]:

	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 [12]:
```
#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[12]:

	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 [13]:
```
#Delete the “Maxpulse” column from the main df dataframe
del df['Maxpulse']
```

In [14]:
```
#To display the first few rows of the table
df.head()
```

Out[14]:

	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 [15]:
```
#To display the types of the rows
df.dtypes
```

Out[15]:
```
Duration      int64
Pulse         int64
Calories    float64
dtype: object
```

In [16]:
```
#Convert the datatype of Calories column to int datatype.
df['Calories'] = df['Calories'].astype(np.int64)
df.dtypes
```

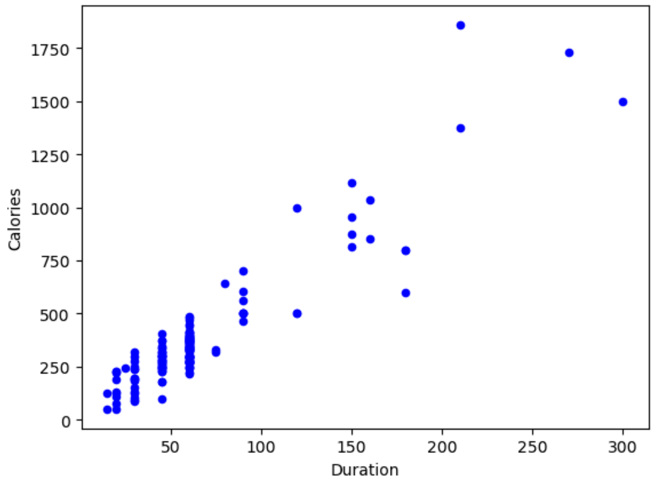
Out[16]:
```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

```
dtype: object
```

```
In [17]: #Using pandas create a scatter plot for the two columns (Duration and Calories).
         df.plot.scatter(x='Duration',y='Calories',c='blue')
```

```
Out[17]: <Axes: xlabel='Duration', ylabel='Calories'>
```



The program imports the necessary libraries for machine learning, data processing, visualization, and handling errors.A dataset is loaded from a CSV file and kept in a DataFrame with the name df.To provide a brief overview of the data, the code shows the top five rows of the DataFrame.For numerical columns in the DataFrame, it computes descriptive statistics like count, mean, standard deviation, minimum, quartiles, and maximum.The program scans the DataFrame for any missing values and returns a boolean value (True/False) for each column to indicate whether or not it has any.The code substitutes the mean value of each column for any missing data. After handling them, it checks once more to see if any missing values are still present.The code combines information such as minimum, maximum, count, and mean for the DataFrame's "Maxpulse" and "Calories" columns.The DataFrame is filtered according to predetermined criteria, such as choosing 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 duration, pulse, and calories columns from the original DataFrame are the sole columns in the newly formed DataFrame, df_modified. This altered DataFrame's initial few rows are shown.The DataFrame's 'Maxpulse' column gets removed.The code shows 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-learn 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

```
classification_report(y_true, y_pred)
```

GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

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

```
classification_report(y_true, y_pred)
```

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 [79]:
```
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 [80]:
```
# Load the glass dataset
glass_data = pd.read_csv('glass.csv')
```

In [81]:
```
# Split the dataset into features (X) and target variable (y)
X = glass_data.drop('Type', axis=1)
y = glass_data['Type']
```

In [82]:
```
# 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 [83]:
```
# Create a Naïve Bayes classifier
nb_classifier = GaussianNB()
```

In [84]:
```
# Train the classifier
nb_classifier.fit(X_train, y_train)
```

Out[84]:
```
▾ GaussianNB
GaussianNB()
```

In [85]:
```
# Predict the labels for the test set
y_pred = nb_classifier.predict(X_test)
```

In [86]:
```
# Evaluate the model
accuracy = nb_classifier.score(X_test, y_test)
report = classification_report(y_test, y_pred)
```

GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

In [87]:
```
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 [88]:
```
# Create a linear SVM classifier
svm_classifier = LinearSVC(max_iter=1000000)
```

In [89]:
```
# Train the classifier
svm_classifier.fit(X_train, y_train)
```

Out[89]:
```
LinearSVC
LinearSVC(max_iter=1000000)
```

In [90]:
```
# Predict the labels for the test set
y_pred_svm = svm_classifier.predict(X_test)
```

In [91]:
```
# Evaluate the model
accuracy_svm = svm_classifier.score(X_test, y_test)
report_svm = classification_report(y_test, y_pred_svm, zero_division=1)
```

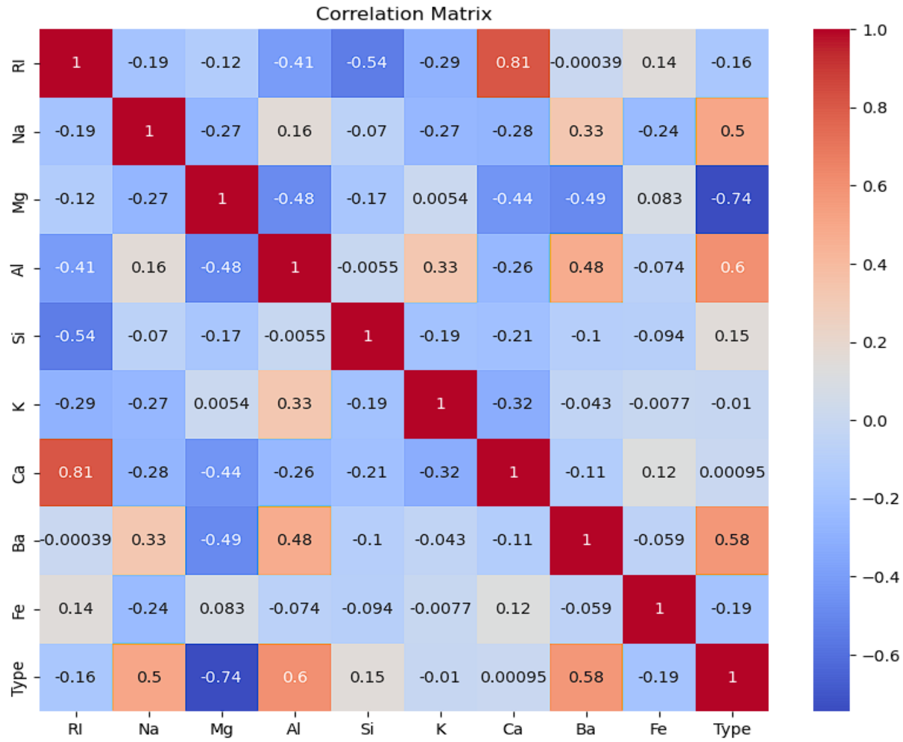
In [92]:
```
print("Accuracy (Linear SVM):", accuracy_svm)
print("Classification Report (Linear SVM):")
print(report_svm)
```

```
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 [93]:
```
# Create a correlation matrix
correlation_matrix = glass_data.corr()
```

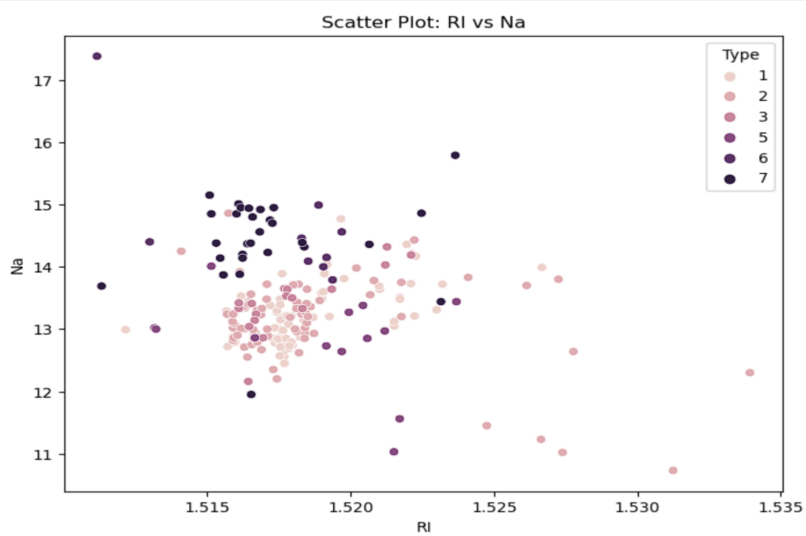
GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

In [94]:
```
# 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()
```



	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
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

In [95]:
```
# 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()
```



GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com

The code imports the required libraries for data manipulation, machine learning, and visualization, including pandas, scikit-learn, and seaborn.A DataFrame called glass_data is used to hold the glass dataset after it has been loaded from a CSV file.The dataset is divided into characteristics (X) and the desired outcome (y), where X contains all columns other than the "Type" column and y only the "Type" column.Using the train_test_split function of scikit-learn, the dataset is further divided into training and testing sets. A random seed is supplied for repeatability, and the testing set size is set to 20% of the data.The scikit-learn GaussianNB class is used to construct the Naive Bayes classifier (GaussianNB).The fit approach is used to train the Nave Bayes classifier on the training set. The trained Nave Bayes classifier makes predictions on the testing set.The score method, which compares the predicted labels with the actual labels, is used to determine the accuracy of the Naive Bayes classifier.The classification_report function is used to create the classification report, which contains metrics like precision, recall, and F1-score.The Nave Bayes classifier's accuracy and classification report are printed.The scikit-learn LinearSVC class is used to build a linear SVM classifier (LinearSVC).The fit approach is used to train the linear SVM classifier on the training set. On the testing set, predictions are made using the trained linear SVM classifier.Using the score approach, the linear SVM classifier's accuracy is determined.The classification_report function is used to create the classification report for the linear SVM classifier.The linear SVM classifier prints its accuracy and classification report.The data visualization process is then carried out by the code utilizing Seaborn and Matplotlib.The corr method, which determines the correlation between all pairs of variables, is used to construct a correlation matrix from the glass dataset.The heatmap function from seaborn is used to visualize the correlation matrix as a heatmap. The correlation between the RI (refractive index) and Na (sodium) variables is displayed in a scatter plot, with the color of each data point denoting the type of glass.Using matplotlib, the final scatter plot and correlation matrix heatmap are shown.From the Output we could see that the **Linear SVM has the high accuracy than the Naïve Bayes** for my model.

GIT HUB LINK: https://github.com/anudeep4c4/Programmin-Asst-2/
GIT HUB ID: allamsettyanudeep@gmail.com