**Ram Mohan Rao Yeduri**

**700740253**

**Machine Learning – Assignment2**

GitHub: https://github.com/rxy02530/ASSIGNMENT_2.git

Video link:
https://drive.google.com/file/d/1o3y3s5LxpOY8Ur823KgYDIJq7cNpIsaH/view?usp=drive_link

In [1]:
```
#importing the required libraries to work with Tabular data and also to implement algorithms

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.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")
```

```
C:\Users\ramye\AppData\Local\Temp\ipykernel_6200\1470931602.py:9: DeprecationWarning: Please use `pearsonr` from the `scipy.
stats` namespace, the `scipy.stats.stats` namespace is deprecated.
  from scipy.stats.stats import pearsonr
```

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).

In [1]:
```
#importing the required libraries to work with Tabular data and also to implement algorithms

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.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")
```

```
C:\Users\ramye\AppData\Local\Temp\ipykernel_6200\1470931602.py:9: DeprecationWarning: Please use `pearsonr` from the `scipy.
stats` namespace, the `scipy.stats.stats` namespace is deprecated.
  from scipy.stats.stats import pearsonr
```

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).

In [26]:
```
#3. Check if the data has null values.

data.isnull().any()
```

Out[26]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

In [27]:
```
#Replace the null values with the mean

data.fillna(data.mean(), inplace=True)
data.isnull().any()
```

Out[27]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

In [28]:
```
#4. Select at least two columns and aggregate the data using: min, max, count, mean.

data.agg({'Duration':['min','max','count','mean'],'Pulse':['min','max','count','mean']})
```

Out[28]:

| | Duration | Pulse |
|---|---|---|
| min | 15.000000 | 80.000000 |
| max | 300.000000 | 159.000000 |
| count | 169.000000 | 169.000000 |

In [29]:
```
#5. Filter the dataframe to select the rows with calories values between 500 and 1000.

data.loc[(data['Calories']>500)&(data['Calories']<1000)]
```

Out[29]:

| | 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 [30]:
```
#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
```

In [30]:
```
#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
```

```
data.loc[(data['Calories']>500)&(data['Pulse']<100)]
```

Out[30]:

| | 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 [31]:
```
#7. Create a new "data_modified" dataframe that contains all the columns from data except for "Maxpulse".

data_modified = data[['Duration','Pulse','Calories']]
data_modified.head()
```

Out[31]:

| | Duration | Pulse | Calories |
|---|---|---|---|
| **0** | 60 | 110 | 409.1 |
| **1** | 60 | 117 | 479.0 |
| **2** | 60 | 103 | 340.0 |
| **3** | 45 | 109 | 282.4 |

In [32]:
```
#8. Delete the "Maxpulse" column from the main df dataframe

del data['Maxpulse']
```

In [33]:
```
data.head()
```

Out[33]:

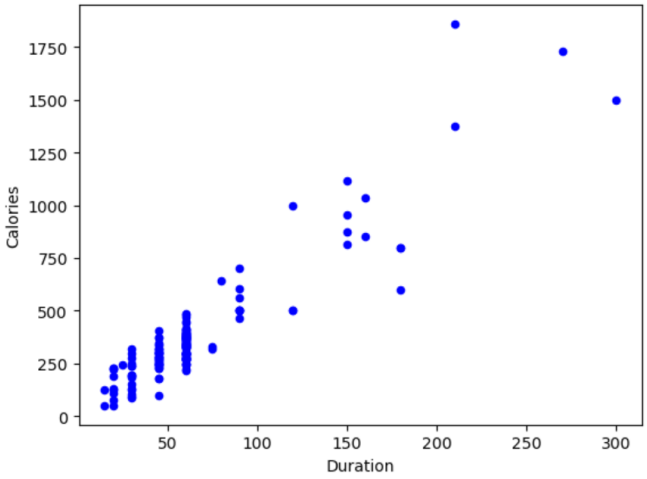
| | Duration | Pulse | Calories |
|---|---|---|---|
| **0** | 60 | 110 | 409.1 |
| **1** | 60 | 117 | 479.0 |
| **2** | 60 | 103 | 340.0 |
| **3** | 45 | 109 | 282.4 |
| **4** | 45 | 117 | 406.0 |

In [35]:
```
#9. Convert the datatype of Calories column to int datatype.

data['Calories'] = data['Calories'].astype(np.int64)
data.dtypes
```

Out[35]:
```
Duration     int64
Pulse        int64
Calories     int64
dtype: object
```

In [39]:
```
#10. Using pandas create a scatter plot for the two columns (Duration and Calories).

data.plot.scatter(x='Duration',y='Calories',c='blue')
```

Out[39]:
```
<Axes: xlabel='Duration', ylabel='Calories'>
```



Scikit-learn

(Glass Dataset)

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)

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)

In [4]:
```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats.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

get_ipython().run_line_magic('matplotlib', 'inline')
# Suppress warnings
warnings.filterwarnings("ignore")
glass_data=pd.read_csv("glass.csv")
X = glass_data.drop('Type', axis=1)
y = glass_data['Type']
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [7]:
```
classifier = GaussianNB()

classifier.fit(X_train, y_train)


y_pred = classifier.predict(X_test)
accuracy = classifier.score(X_test, y_test)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5581395348837209
              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 [8]:
```
glass=pd.read_csv("glass.csv")
```

In [9]:
```
from sklearn.svm import SVC, LinearSVC
classifier = LinearSVC()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [10]:
```
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
```

Out[10]:
```
SVC(kernel='linear')
```

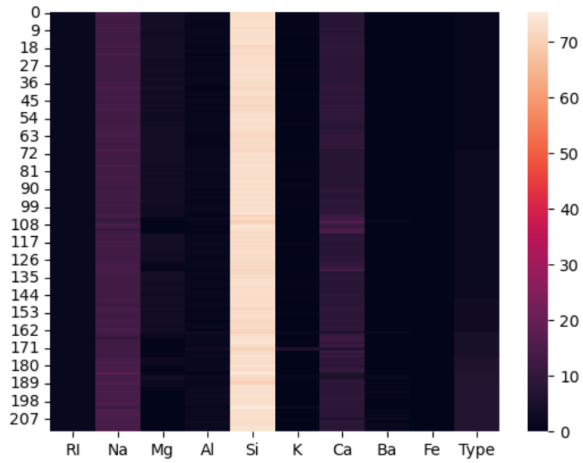
In [11]:
```
y_pred = svm_classifier.predict(X_test)
```

In [12]:
```
accuracy = svm_classifier.score(X_test, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7441860465116279
```

In [13]:
```
sns.heatmap(data=glass) #HeatMap Visualization for above dataset
```

Out[13]: <Axes: >




In [14]:
```
sns.scatterplot(data=glass)  #ScatterPlot Visualization for above dataset
```

Out[14]: <Axes: >