**Assignment 2**

**Avinash Dakey**

**700740053**

**1.Pandas**

1.Answer

from google.colab import drive

drive.mount('/content/gdrive')

path\_to\_csv = '/content/data.csv'

import warnings

warnings.filterwarnings("ignore")



2. Answer

import pandas as pd

# Read the provided CSV file ‘data.csv’

df = pd.read\_csv(path\_to\_csv)

df

#A screenshot of a computer

Description automatically generated with medium confidence

3. Answer

# Show the basic statistical description about the data.

df.describe()

A screenshot of a computer

Description automatically generated with medium confidence

4. Answer

# Check if the data has null values.

print('Are there any null values: ',df.isnull().values.any())

# Replace the null values with the mean

df.fillna(df.mean(),inplace=True)

print('Are there any null values after using fillna: ',df.isnull().values.any())

A screenshot of a computer

Description automatically generated with medium confidence

5. Answer

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

aggre = df.groupby('Duration').agg({'Calories':['mean','min','max','count']})

aggre

A screenshot of a computer

Description automatically generated

6. Answer

# Filter the dataframe to select the rows with calories values between 500 and 1000

df[(df['Calories']>=500) & (df['Calories']<=1000)]

A screenshot of a computer

Description automatically generated with medium confidence

7. Answer

# Filter the dataframe to select the rows with calories values > 500 and pulse < 100

df[(df['Calories']>500) & (df['Pulse']<100)]

A screenshot of a computer

Description automatically generated with medium confidence

8. Answer

# Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”

df\_modified = df[['Duration', 'Pulse', 'Calories']]

df\_modified

**A screenshot of a computer

Description automatically generated with medium confidence**

9. Answer

# Delete the “Maxpulse” column from the main df dataframe

df = df.drop('Maxpulse', axis=1)

df

**A screenshot of a computer

Description automatically generated with medium confidence**

10. Answer

# Convert the datatype of Calories column to int datatype

df['Calories'] = df['Calories'].astype('int64')

df.dtypes

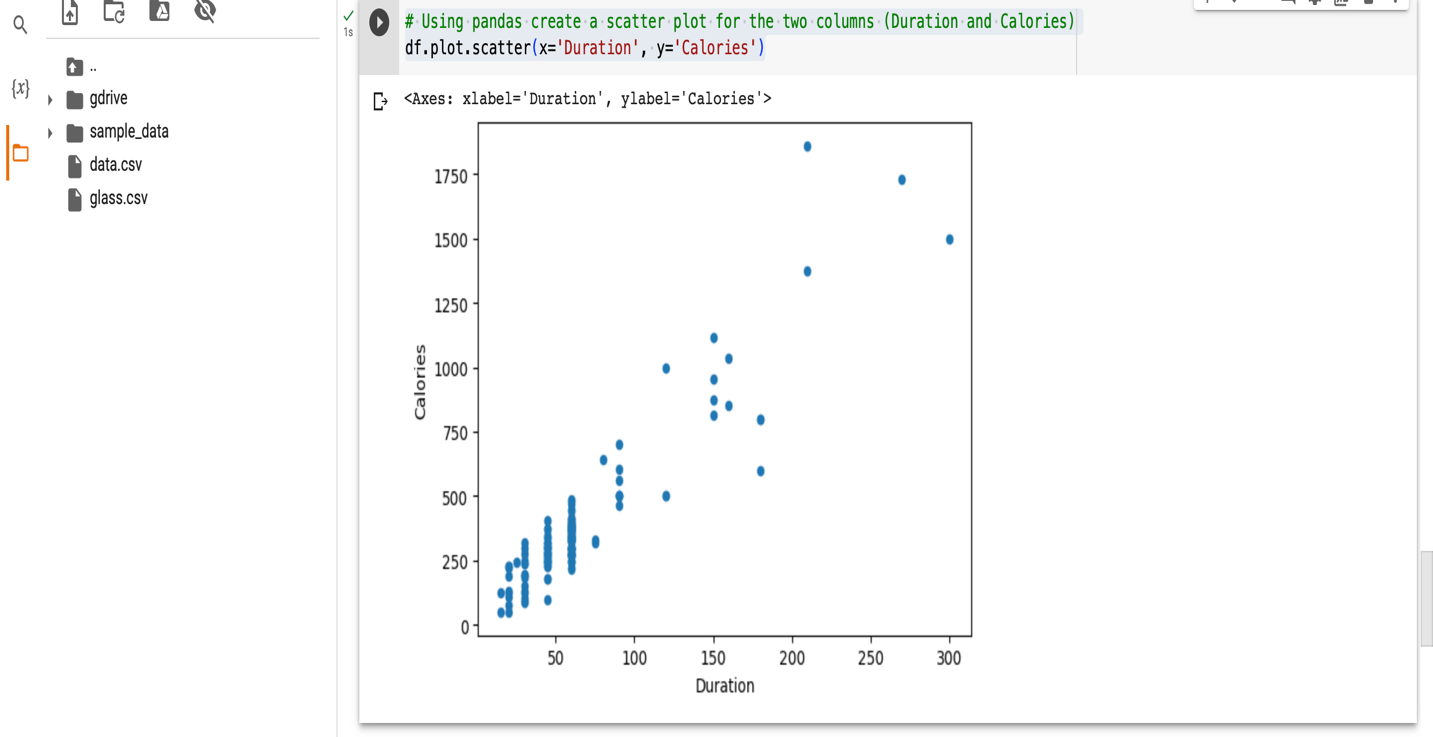
**A screenshot of a computer

Description automatically generated with low confidence**

a.Example:

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

df.plot.scatter(x='Duration', y='Calories')

****

**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.

1. glass=pd.read\_csv("/content/glass.csv")
2. glass.head()
3. des=glass.corr()
4. glass.corr().style.background\_gradient(cmap="Greens")

**A screenshot of a computer

Description automatically generated with medium confidence**

b.Use **train\_test\_split** to create training and testing part.

1. features **=** ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
2. target **=** 'Type'
3. X\_train, X\_val, Y\_train, Y\_val **=** train\_test\_split(glass[::**-**1], glass[target],test\_size**=**0.2, random\_state**=**1)

**2. Evaluate the model on testing part using score and**

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

import warnings

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, classification\_report, confusion\_matrix

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('\naccuracy is',accuracy\_score(Y\_val, y\_pred))

**A screenshot of a computer

Description automatically generated**

1. Implement linear SVM method using scikit library
   * 1. Use the glass dataset available in Link also provided in your assignment.
     2. Use **train\_test\_split** to create training and testing part.

We have already imported the glass dataset and also used train\_test\_split() method to split the dataset into train dataset and test dataset while implementing Naïve Bayes method.

from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

classifier.fit(X\_train, Y\_train)

**A picture containing text, screenshot, font

Description automatically generated**

**2. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)**

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('\naccuracy is',accuracy\_score(Y\_val, y\_pred))

**A screenshot of a computer

Description automatically generated**

**1.Do at least two visualizations to describe or show correlations in the Glass Dataset.**

**1)** import seaborn as sns

sns.heatmap(data=glass) #HeatMap Visualization for above dataset

**A screenshot of a graph

Description automatically generated with medium confidence**

**2)** sns.scatterplot(data=glass) #ScatterPlot Visualization for above dataset

**A screenshot of a computer

Description automatically generated with medium confidence**

**2.Which algorithm you got better accuracy? Can you justify why?**

According to the above accuracy scores Naive Bayes method is best for data visualization than that of Support Vector Machine method. The performance of the each algorithm depends on several factors. So, few algorithms works well for only few of the problems and does not work well for other problems. By evaluating the model using various algorithms we can compare and then state which one is best.