MACHINE LEARNING ASSIGNMENT-4

V.Sonalika
700742916

Video link:
https://drive.google.com/file/d/16LHd6uEwrf1DfZA7Kdky79AxJRWvQMlJ/view?usp=share_link

Github link:
https://github.com/Sonalika2229/ML_Assignment4

1.Pandas:

```
[129]
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")
```

Importing all the required libraries.

```
[130] path ="/content/drive/MyDrive/Mldataset/data.csv"
df = pd.read_csv(path)
df.head()
```

	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

The Pandas library provides the pd.read_csv() function, which reads a CSV file and builds a DataFrame object from it.

```
df.describe()
```

	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

The df.describe() method in Pandas is a built-in function that creates descriptive statistics for the DataFrame df.

```
df.isnull().any()#Checking if the data has null values.
```

```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

```
[163] #Replace the null values with the mean

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

```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

The df.fillna() method uses the mean value of each column to fill in any missing values in the DataFrame df.
The df.isnull().any() is used to check if there are any missing values remaining in the DataFrame.

```
#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']})
```

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

The agg() method is used to do the aggregate computations of the Maxpulse and Calories on the dataframe df.

```
#Filtering the dataframe to select the rows with calories values between 500 and 1000.

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

The df.loc() function is used to select the rows with calories values between 500 and 1000

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

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

The df.loc() function is used to select the rows with calories values that are greater than 500 and the pulse value less than 100.

```
#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()
```

	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

New DataFrame df_modified is created that includes only Duration, Pulse, and Calories columns of the original DataFrame df.

```
#Delete the "Maxpulse" column from the main df dataframe

del df['Maxpulse']
```

The Maxpulse column is removed from the dataframe df using del df['MaxPulse']

```
[170] df.dtypes

Duration       int64
Pulse          int64
Calories     float64
dtype: object
```

The df.dtypes is used to display the data types of each column in the DataFrame.

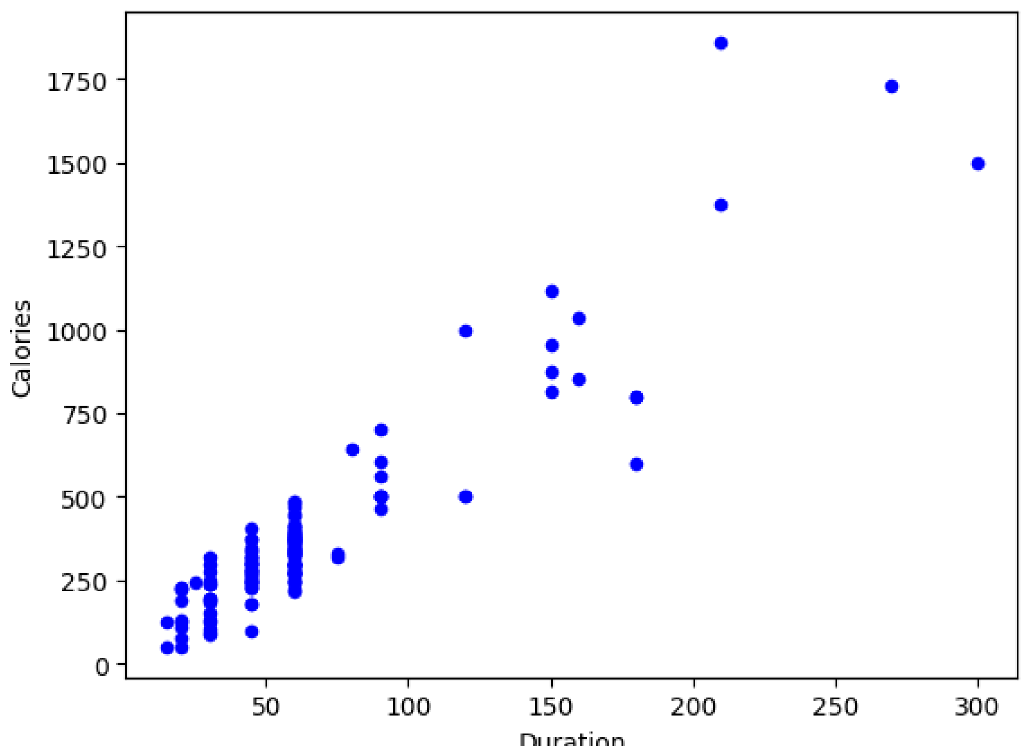
```
#Convert the datatype of Calories column to int datatype.

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

```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

The astype() method is used to cast a column of a DataFrame to a specific data type. Here the Calories column is cast to the 64-bit integer data type using the np.int64.

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

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

```
<Axes: xlabel='Duration', ylabel='Calories'>
```



The scattered plot of the Duration and Calories columns in the DataFrame df is displayed.

Question: 2 Titanic Dataset

1. Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case inclass. a. Do you think we should keep this feature?
2. Do at least two visualizations to describe or show correlations.
3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

```
[173] path = "/content/drive/MyDrive/Mldataset/train.csv"
      df = pd.read_csv(path)
      df.head()
```


```
path = "/content/drive/MyDrive/Mldataset/train.csv"
df = pd.read_csv(path)
df.head()
```


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
#converting the  categorical data to numerical values for correlation calculation

le = preprocessing.LabelEncoder()
df['Sex'] = le.fit_transform(df.Sex.values)


#Calculation of correlation for 'Survived' and  'Sex' in data
correlation_Value= df['Survived'].corr(df['Sex'])

print(correlation_Value)
```

```
-0.5433513806577555
```

Yes, we should keep the 'Survived' and 'Sex' features helps classify the data accurately

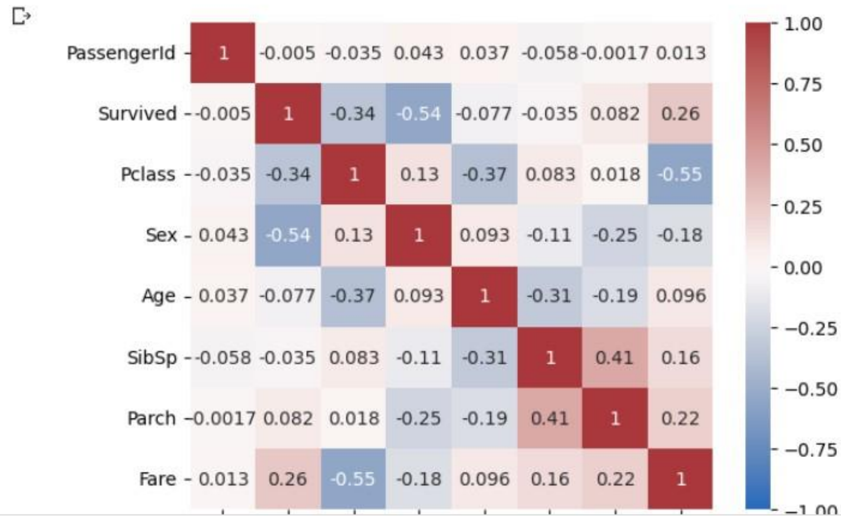
```
matrix = df.corr() #printing the correlation matrix
print(matrix)
```

```
             PassengerId  Survived    Pclass       Sex       Age     SibSp  \
PassengerId     1.000000 -0.005007 -0.035144  0.042939  0.036847 -0.057527
Survived       -0.005007  1.000000 -0.338481 -0.543351 -0.077221 -0.035322
Pclass         -0.035144 -0.338481  1.000000  0.131900 -0.369226  0.083081
Sex             0.042939 -0.543351  0.131900  1.000000  0.093254 -0.114631
Age             0.036847 -0.077221 -0.369226  0.093254  1.000000 -0.308247
SibSp          -0.057527 -0.035322  0.083081 -0.114631 -0.308247  1.000000
Parch          -0.001652  0.081629  0.018443 -0.245489 -0.189119  0.414838
Fare            0.012658  0.257307 -0.549500 -0.182333  0.096067  0.159651

                Parch      Fare
PassengerId -0.001652  0.012658
Survived     0.081629  0.257307
Pclass       0.018443 -0.549500
Sex         -0.245489 -0.182333
Age         -0.189119  0.096067
SibSp        0.414838  0.159651
Parch        1.000000  0.216225
Fare         0.216225  1.000000
```

```
# One way of visualizing the correlation matrix in form of the  spread chart

df.corr().style.background_gradient(cmap="Reds")
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.042939	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.543351	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	0.131900	-0.369226	0.083081	0.018443	-0.549500
Sex	0.042939	-0.543351	0.131900	1.000000	0.093254	-0.114631	-0.245489	-0.182333
Age	0.036847	-0.077221	-0.369226	0.093254	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.114631	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.245489	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	-0.182333	0.096067	0.159651	0.216225	1.000000

```python
sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag') #Second form of visuaizing correlation matriX
plt.show()
```




```python
#Loaded the data files test and train and merged files

trainraw = pd.read_csv('/content/drive/MyDrive/Mldataset/train.csv')
testraw = pd.read_csv('/content/drive/MyDrive/Mldataset/test.csv')
trainraw['train'] = 1
testraw['train'] = 0
df = trainraw.append(testraw, sort=False)
features = ['Age', 'Embarked', 'Fare', 'Parch', 'Pclass', 'Sex', 'SibSp']
target = 'Survived'
df = df[features + [target] + ['train']]
df['Sex'] = df['Sex'].replace(["female", "male"], [0, 1])
df['Embarked'] = df['Embarked'].replace(['S', 'C', 'Q'], [1, 2, 3])
train = df.query('train == 1')
test = df.query('train == 0')
```

[179]
```python
# Dropping the missing values from the train set.
train.dropna(axis=0, inplace=True)
labels = train[target].values
train.drop(['train', target, 'Pclass'], axis=1, inplace=True)
test.drop(['train', target, 'Pclass'], axis=1, inplace=True)
```

```
[180] #Test and train split

      A_train, A_val, B_train, B_val = train_test_split(train, labels, test_size=0.2, random_state=1)
```

```
[181] classifier = GaussianNB()

      classifier.fit(A_train, B_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
A_pred = classifier.predict(A_val)

# Summary of the predictions made by the classifier
print(classification_report(B_val, A_pred))
print(confusion_matrix(B_val, A_pred))
# Describing the Accuracy score
from sklearn.metrics import accuracy_score
print('Accuracy is',accuracy_score(B_val, A_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.80      0.80        85
         1.0       0.70      0.69      0.70        58

    accuracy                           0.76       143
   macro avg       0.75      0.74      0.75       143
weighted avg       0.75      0.76      0.75       143

[[68 17]
 [18 40]]
Accuracy is 0.7552447552447552
```

Question-3:

(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)
3. 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.
4. Evaluate the model on testing part using score and classification_report(y_true, y_pred)


```
[183] path = "/content/drive/MyDrive/Mldataset/glass.csv"
      df = pd.read_csv(path)
      df.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1

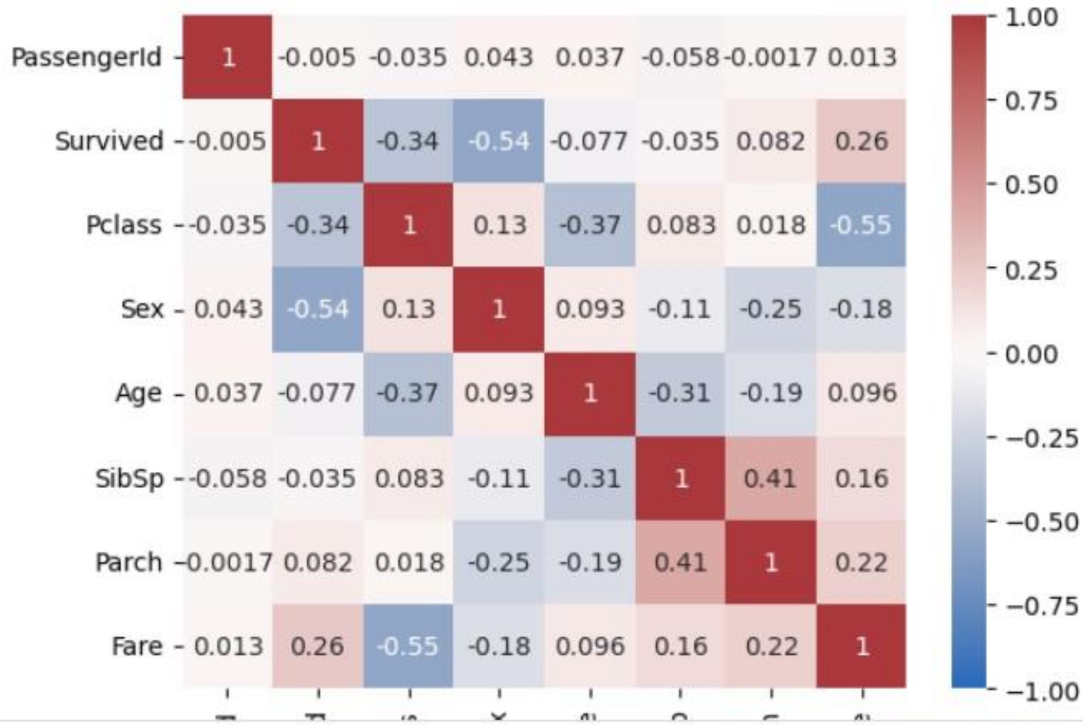

```
[183] path = "/content/drive/MyDrive/Mldataset/glass.csv"
      df = pd.read_csv(path)
      df.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

```
df.corr().style.background_gradient(cmap="Blues")
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000

```python
sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.show()
```



0s completed at 7:21 PM

[186]
```python
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'


A_train, A_val, B_train, B_val = train_test_split(df[::-1], df['Type'],test_size=0.2, random_state=1)

classifier = GaussianNB()

classifier.fit(A_train, B_train)


y_pred = classifier.predict(A_val)

# Summary of the predictions made by the classifier
print(classification_report(B_val, y_pred))
print(confusion_matrix(B_val, y_pred))
# Accuracy score

print('Accuracy is',accuracy_score(B_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.84        43
   macro avg       0.76      0.69      0.71        43
weighted avg       0.89      0.84      0.85        43

[[18  1  0  0  0  0]
 [ 1 11  0  0  0  0]
 [ 1  0  3  2  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
Accuracy is 0.8372093023255814
```

[187]

```
from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

classifier.fit(A_train, B_train)


y_pred = classifier.predict(A_val)

# Summary of the predictions made by the classifier
print(classification_report(B_val, y_pred))
print(confusion_matrix(B_val, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(B_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      0.95      0.97        19
           2       1.00      0.08      0.15        12
           3       0.26      1.00      0.41         6
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         4

    accuracy                           0.58        43
   macro avg       0.38      0.34      0.26        43
weighted avg       0.76      0.58      0.53        43

[[18  0  1  0  0  0]
 [ 0  1 10  1  0  0]
 [ 0  0  6  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  4  0  0  0]]
accuracy is 0.5813953488372093
```

The accuracy for Naïve Bayes method is 0.8372093023255814.Finally, because of the amount of data Naive Bayes algorithm gives better accuracy compared to Linear SVM.