Assignment-4

localhost:8891/notebooks/Machine_learning_Assignment-4.ipynb#

jupyter Machine_learning_Assignment-4 Last Checkpoint: 9 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [34]:

```python
#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")
```

Question: 1

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).

jupyter Machine_learning_Assignment-4 Last Checkpoint: 10 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [36]:

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

df = pd.read_csv("data.csv")
df.head()
```

Out[36]:

	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 [37]:

```python
#2. Show the basic statistical description about the data.

df.describe()
```

Out[37]:

	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 [38]: #3. Check if the data has null values.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [38]:
```
#3. Check if the data has null values.

df.isnull().any()
```

Out[38]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

In [39]:
```
#Replace the null values with the mean

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

Out[39]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

In [40]:
```
#4. 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[40]:

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

In [41]:
```
#5. Filter the dataframe to select the rows with calories values between 500 and 1000.
```

localhost:8891/notebooks/Machine_learning_Assignment-4.ipynb#

jupyter Machine_learning_Assignment-4 Last Checkpoint: 10 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [41]:
```
#5. Filter the dataframe to select the rows with calories values between 500 and 1000.

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

Out[41]:

	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 [42]:
```
#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

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

Out[42]:

	Duration	Pulse	Maxpulse	Calories
65	180	90	130	800.4

Greenshot

Greenshot
Exported to: Document1

Jupyter Machine_learning_Assignment-4 Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [42]:
```
#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

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

Out[42]:

	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 [43]:
```
#7. 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[43]:

	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 [44]:
```
#8. Delete the "Maxpulse" column from the main df dataframe
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [44]:
```
#8. Delete the "Maxpulse" column from the main df dataframe

del df['Maxpulse']
```

In [45]:
```
df.head()
```

Out[45]:

	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 [46]:
```
df.dtypes
```

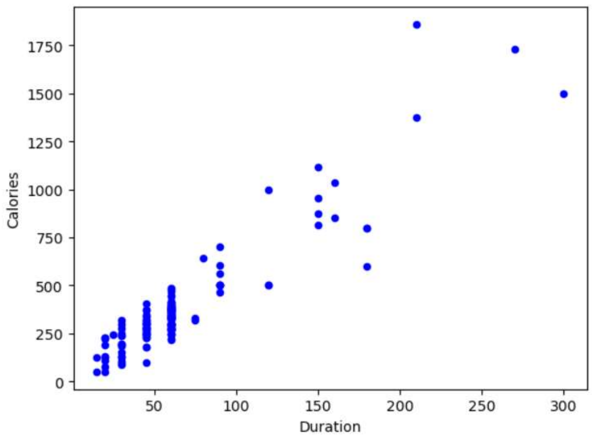
Out[46]:
```
Duration      int64
Pulse         int64
Calories    float64
dtype: object
```

In [47]:
```
#9. Convert the datatype of Calories column to int datatype.

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

Out[47]:
```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

In [48]:
```
#10. Using pandas create a scatter plot for the two columns (Duration and Calories).
```

In [48]:
```
#10. Using pandas create a scatter plot for the two columns (Duration and Calories).

df.plot.scatter(x='Duration',y='Calories',c='blue')
```
Out[48]: <Axes: xlabel='Duration', ylabel='Calories'>



Question: 2

Titanic Dataset

Jupyter Machine_learning_Assignment-4 Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

Titanic Dataset

1. Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case in class. 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

In [49]:
```
#Loading the data file into te program
df=pd.read_csv("train.csv")

df.head()
```
Out[49]:

	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

In [50]:
```
#converted categorical data to numerical values for correlation calculation

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


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

print(correlation_Value)
```
```
-0.543351380657755
```

the accuracy is just 54%.so it is not necessary

jupyter Machine_learning_Assignment-4 Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
Ans: the accuracy is just 54%.so it is not necessary
```

In [51]:
```
#print correlation matrix
matrix = df.corr()
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
```

In [52]:
```
# One way of visualizing correlation matrix in form of spread chart

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

Out[52]:

	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

jupyter Machine_learning_Assignment-4 Last Checkpoint: 12 minutes ago (autosaved)

Logout

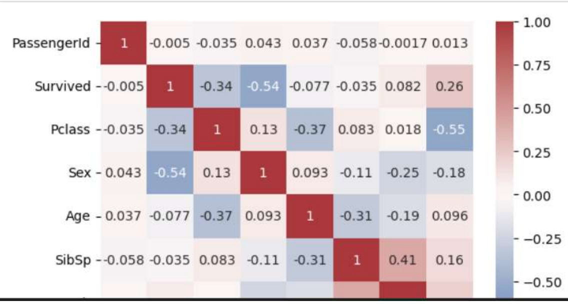
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

	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

In [53]:
```
#Second form of visuaizing correlation matriX using heatmap() from seaborn

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



jupyter Machine_learning_Assignment-4 Last Checkpoint: 13 minutes ago (autosaved)

Logout

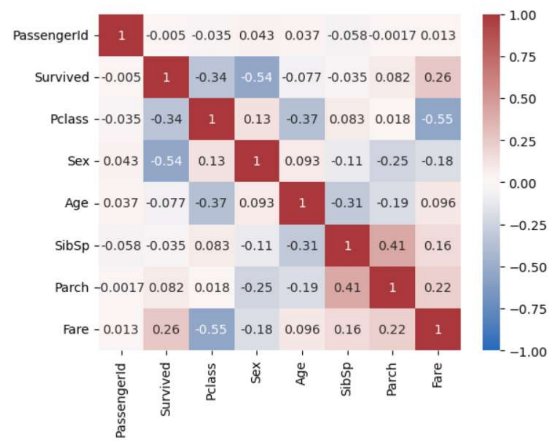
File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

Run Code

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



In [54]:

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

train_raw = pd.read_csv('train.csv')
test_raw = pd.read_csv('test.csv')
```

jupyter Machine_learning_Assignment-4 Last Checkpoint: 13 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [54]:
```python
#Loaded data files test and train and merged files

train_raw = pd.read_csv('train.csv')
test_raw = pd.read_csv('test.csv')
train_raw['train'] = 1
test_raw['train'] = 0
df = train_raw.append(test_raw, 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')
```

In [55]:
```python
# Drop 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)
```

In [56]:
```python
#Test and train split

X_train, X_val, Y_train, Y_val = train_test_split(train, labels, test_size=0.2, random_state=1)
```

In [57]:
```python
classifier = GaussianNB()

classifier.fit(X_train, Y_train)
```

Out[57]:
```
▾ GaussianNB
GaussianNB()
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [58]:
```python
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('accuracy is',accuracy_score(Y_val, y_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)

1. Implement linear SVM method using scikit library
    a. Use the glass dataset available in Link also provided in your assignment.

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

2. Evaluate the model on testing part using score and

In [59]:
```
glass=pd.read_csv("glass.csv")
glass.head()
```

Out[59]:

	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

In [60]:
```
glass.corr().style.background_gradient(cmap="Reds")
```

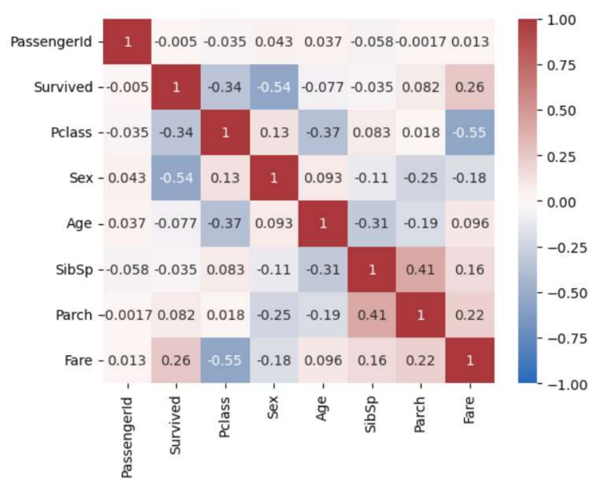
Out[60]:

	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

jupyter Machine_learning_Assignment-4 Last Checkpoint: 15 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [61]:
```
sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.show()
```



In [62]:
```
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [62]:
```
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'


X_train, X_val, Y_train, Y_val = train_test_split(glass[::-1], glass['Type'],test_size=0.2, random_state=1)

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

print('accuracy is',accuracy_score(Y_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]
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
           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
```

In [63]:
```
from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

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('accuracy is',accuracy_score(Y_val, y_pred))
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

▶ Run Code

```python
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      0.95      0.97        19
           2       0.57      1.00      0.73        12
           3       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.77        43
   macro avg       0.39      0.45      0.41        43
weighted avg       0.67      0.77      0.70        43

[[18  1  0  0  0  0]
 [ 0 12  0  0  0  0]
 [ 0  5  0  0  0  1]
 [ 0  1  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  1  0  0  0  3]]
accuracy is 0.7674418604651163
```

Justification:

The Naive Bayes technique gave us a superior accuracy of 0.8372093023255814. While Linear SVM works better with linear regression logics, Naive Bayes analysis works better with probabilistic ideas. However in order to function more accurately, SVM needs a lot of data to train and test. Hence, compared to Linear SVM, Naive Bayes algorithm provides superior accuracy because of the volume of data.

The Naive Bayes technique gave us a superior accuracy of 0.8372093023255814. While Linear SVM works better with linear regression logics, Naive Bayes analysis works better with probabilistic ideas. However, in order to function more accurately, SVM needs a lot of data to train and test. Hence, compared to Linear SVM, Naive Bayes algorithm provides superior accuracy because of the volume of data.

Video link: https://www.loom.com/share/081b23b321a54147a0fa5ae683848bb4

Github link: https://github.com/kadiresanjayreddy/machinelearningAssignment-4