titanic-dataset

February 1, 2024

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn import linear_model
```

Data collection and preproccessing

[ ]:
```
# load the dataset
titanic_data=pd.read_csv('/content/train.csv')
```

1 New Section

[ ]:
```
# display first 5 records in tah dataset
titanic_data.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					
4	5	0	3	Allen, Mr. William Henry	male	35.0	0					

```
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[ ]:
```
# no.of rows and column
titanic_data.shape
```

[ ]:
```
(891, 12)
```

[ ]:
```
# getting some information about the data
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

[ ]:
```
#check the number of missing value in each row
titanic_data.isnull().sum()
```

[ ]:
```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```

Handling missing data

[ ]:
```
# drop the cabin column from dataframe we need the add the axis where 1 represent column 0 represent rows
titanic_data=titanic_data.drop(columns='Cabin', axis=1)
```

[ ]:
```
# replacing the missing value in age by mean
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
```

[ ]:
```
# finding mode of the embarked column
print(titanic_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

[ ]:
```
print(titanic_data['Embarked'].mode()[0])
```

```
S
```

[ ]:
```
# replacing missing value in embarked column with mode value
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],inplace=True)
```

Data Analysis

[ ]:
```
# getting some staticalical measure about the data
titanic_data.describe()
```

[ ]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[ ]:
```
# finding the number of people survived and not survived
titanic_data['Survived'].value_counts()
```

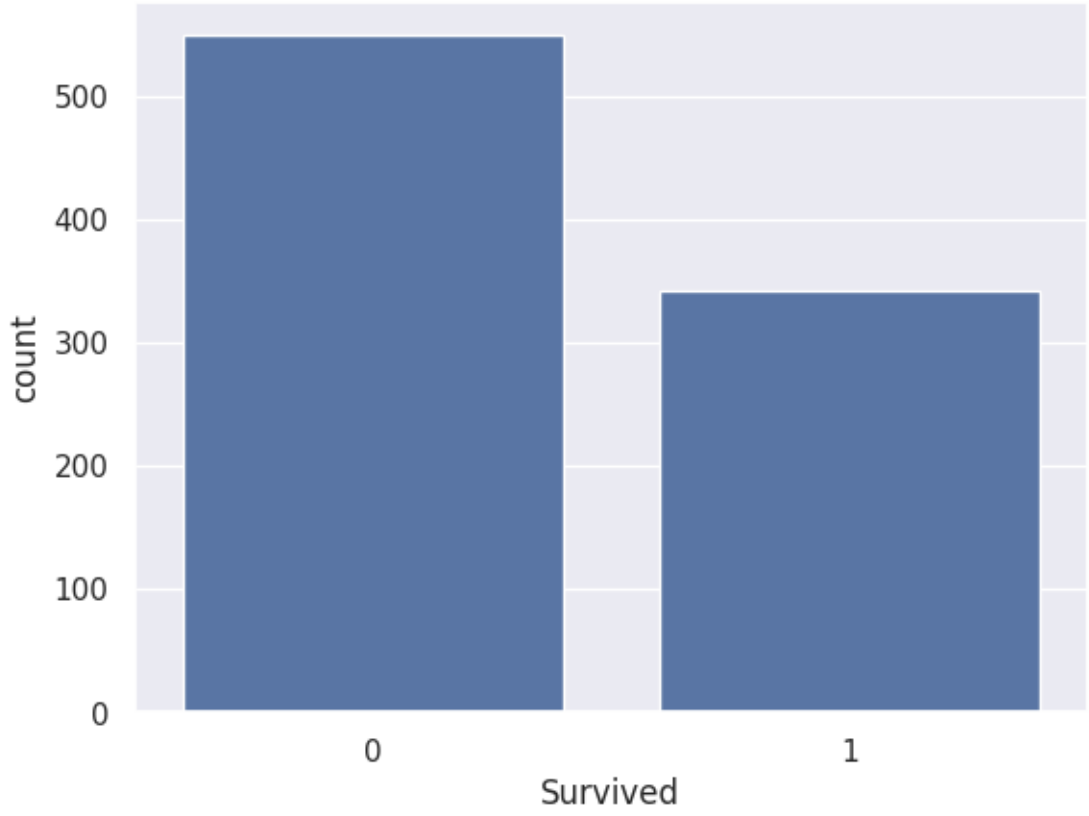
[ ]:
```
0    549
1    342
Name: Survived, dtype: int64
```

Data Visualization

[ ]:
```
sns.set()
```

[ ]:
```
# making a count plot for "Survived" column
sns.countplot(x='Survived',data=titanic_data)
```

[ ]:
```
<Axes: xlabel='Survived', ylabel='count'>
```

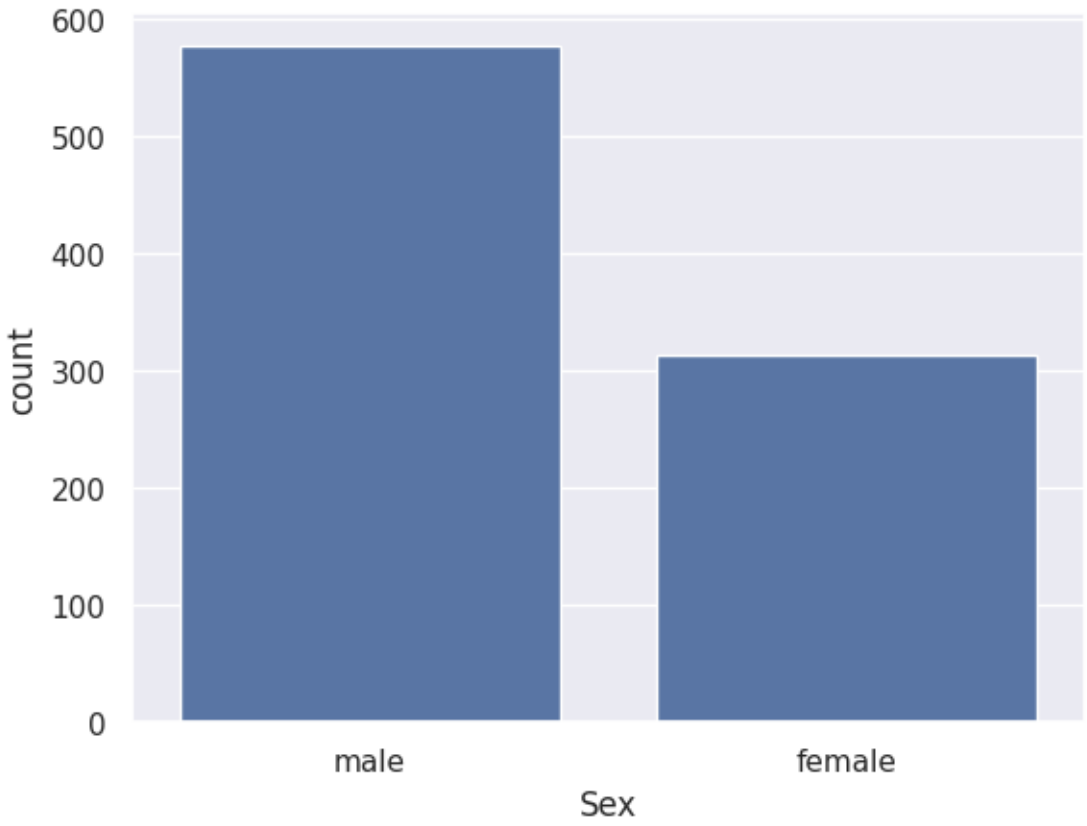


[ ]:
```
# finding the number of people survived and not survived
titanic_data['Sex'].value_counts()
```

[ ]:
```
male      577
female    314
Name: Sex, dtype: int64
```

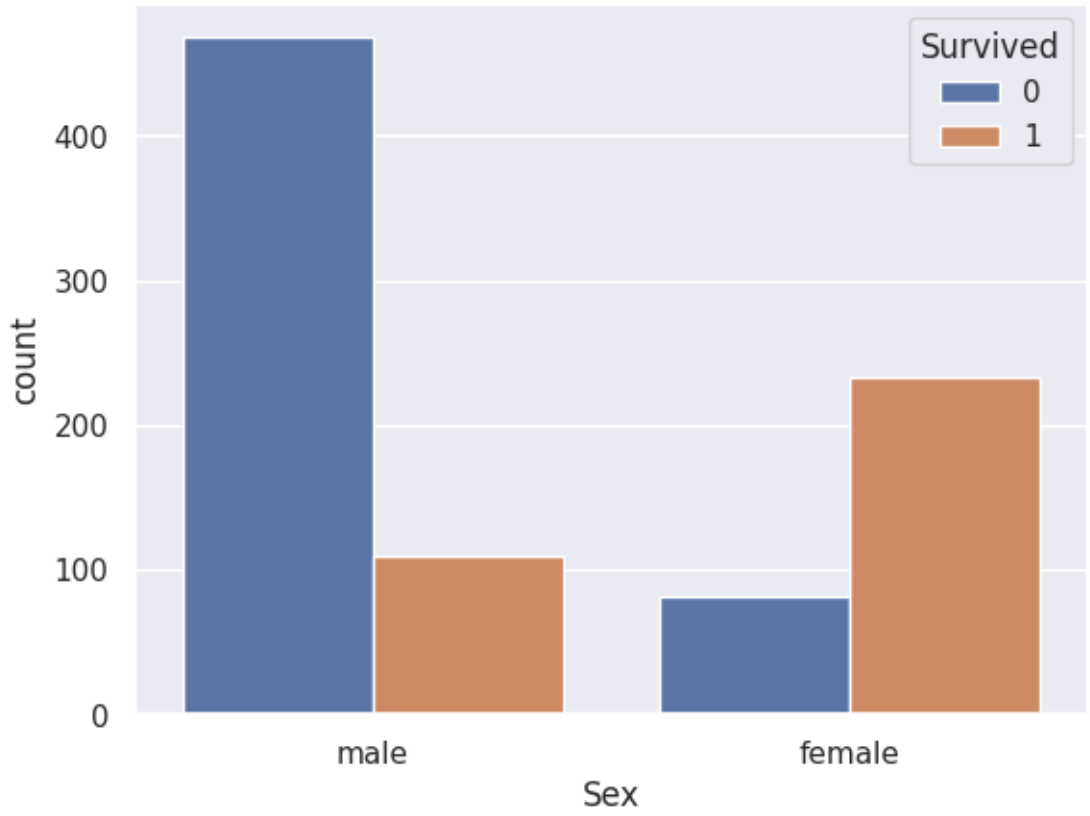
[ ]:
```
sns.countplot(x='Sex',data=titanic_data)
```

[ ]: <Axes: xlabel='Sex', ylabel='count'>



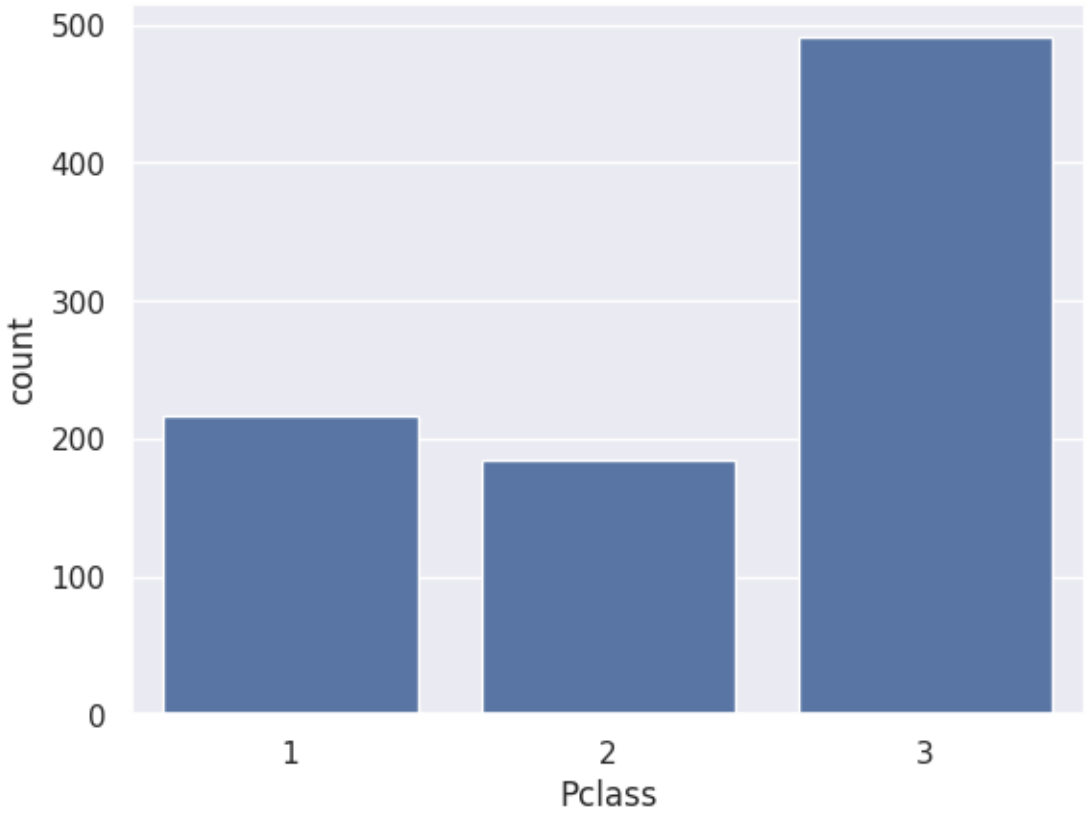
[ ]:
```
# nu.of survivors Gender wise
sns.countplot(data=titanic_data,x='Sex',hue='Survived' )
```

[ ]: <Axes: xlabel='Sex', ylabel='count'>



[ ]:
```
# making a count plot for "pclass" column
sns.countplot(x='Pclass',data=titanic_data)
```

[ ]: <Axes: xlabel='Pclass', ylabel='count'>



[ ]:
```
titanic_data = titanic_data.drop(columns=['Name', 'Ticket'], axis=1)
titanic_data.head()
```

[ ]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500	0
1	2	1	1	1	38.0	1	0	71.2833	1
2	3	1	3	1	26.0	0	0	7.9250	0
3	4	1	1	1	35.0	1	0	53.1000	0
4	5	0	3	0	35.0	0	0	8.0500	0

Encoding the Categorical columns

[ ]:
```
titanic_data['Sex'].value_counts()
```

[ ]:
```
0    577
1    314
Name: Sex, dtype: int64
```

[ ]:

[ ]:
```
titanic_data['Embarked'].value_counts()
```

[ ]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

[ ]:
```
#converting categorical columns
titanic_data.replace({'Sex': {'male': 0, 'female': 1}, 'Embarked': {'S': 0, 'C':
 ↪ 1, 'Q': 2}}, inplace=True)
```

[ ]:
```
titanic_data.head()
```

[ ]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500	0
1	2	1	1	1	38.0	1	0	71.2833	1
2	3	1	3	1	26.0	0	0	7.9250	0
3	4	1	1	1	35.0	1	0	53.1000	0
4	5	0	3	0	35.0	0	0	8.0500	0

Sepreating feature & target

[ ]:
```
X = titanic_data.drop(columns = ['PassengerId','Survived'],axis=1)
Y = titanic_data['Survived']
```

[ ]:
```
print(X)
```

```
     Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0         3    0  22.000000      1      0   7.2500         0
1         1    1  38.000000      1      0  71.2833         1
2         3    1  26.000000      0      0   7.9250         0
3         1    1  35.000000      1      0  53.1000         0
4         3    0  35.000000      0      0   8.0500         0
..      ...  ...        ...    ...    ...      ...       ...
886       2    0  27.000000      0      0  13.0000         0
887       1    1  19.000000      0      0  30.0000         0
888       3    1  29.699118      1      2  23.4500         0
889       1    0  26.000000      0      0  30.0000         1
890       3    0  32.000000      0      0   7.7500         2

[891 rows x 7 columns]
```

[ ]:
```
print(Y)
```

```
0      0
1      1
2      1
3      1
4      0
      ..
```

```
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

Splitting data into training data and testing data

[ ]:
```
X_train,X_test,y_train,y_test= train_test_split(X,Y,test_size=0.
↪2,random_state=2)
```

[ ]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(891, 7) (712, 7) (179, 7)
```

[ ]:
```
print(y_train.head())
print(y_train)
```

```
331    0
733    0
382    0
704    0
813    0
Name: Survived, dtype: int64
331    0
733    0
382    0
704    0
813    0
      ..
106    1
270    0
860    0
435    1
102    0
Name: Survived, Length: 712, dtype: int64
```

Model training

logistic regression model

[ ]:
```
model =linear_model.LogisticRegression()
```

[ ]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
```

```
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Sex          891 non-null    int64
 4   Age          891 non-null    float64
 5   SibSp        891 non-null    int64
 6   Parch        891 non-null    int64
 7   Fare         891 non-null    float64
 8   Embarked     891 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 62.8 KB
```

[ ]:
```
titanic_data.astype({'Age':'int','Fare':'int'}).dtypes
```

[ ]:
```
PassengerId    int64
Survived       int64
Pclass         int64
Sex            int64
Age            int64
SibSp          int64
Parch          int64
Fare           int64
Embarked       int64
dtype: object
```

[ ]:
```
# training the logistic regression model with training data
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
#accuracy on training data
X_train_prediction = model.predict(X_train)
```

[ ]:
```
print(X_train_prediction)
```

```
[0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1
```

```
0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0
1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1
0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0
0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0
0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1
0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0
0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 0
0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0
1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0
0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 0
0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1
0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0
1 0 0 1 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0
0 0 0 1 1 0 0 1 0]
```

[ ]:
```
training_data_accuracy = accuracy_score(y_train, X_train_prediction)
print('Accuracy_score_of_training_data : ', training_data_accuracy)
```

```
Accuracy_score_of_training_data :  0.8075842696629213
```

[ ]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
```

[ ]:
```
print(X_test_prediction)
```

```
[0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0
 1 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0
 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0
 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0]
```

[ ]:
```
test_data_accuracy = accuracy_score(y_test, X_test_prediction)
print('Accuracy_score_of_test data : ', test_data_accuracy)
```

```
Accuracy_score_of_test data :  0.7821229050279329
```