Practical No 6 : Data Analysis [III]

```
IN[1]: 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.naive_bayes import GaussianNB
         from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_re
         from sklearn.preprocessing import LabelEncoder
IN[2]: data = pd.read_csv("Iris.csv")
IN[3]: data.head(5)
IN[4]: data.describe(include='all')
IN[5]: data.info()
IN[6]: print(data.shape) data['Species'].unique()
IN[7]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
IN[8]: data.isnull().sum()
IN[9]: x= data.iloc[:,1:5]
         y= data.iloc[:,5:]
IN[10]: encode= LabelEncoder()
          y= encode.fit_transform(y)
IN[11]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0
IN[12]:  naive_bayes=GaussianNB()
           naive_bayes.fit(x_train,y_train)
           pred=naive_bayes.predict(x_test)
IN[13]: pred
IN[14]: y_test
IN[15]: matrix = confusion_matrix(y_test, pred, labels=naive_bayes.classes_) print(matrix)
          tp, fn, fp, tn = confusion_matrix(y_test, pred, labels=[1, 0]).reshape(-1)
IN[16]:  From sklearn.metrics import ConfusionMatrixDisplay
          conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix, display_labels=naive_baye
          conf_matrix.plot(cmap=plt.cm.YlGn)
          plt.show()
```

```
IN[17]: print(classification_report(y_test,pred))

IN[18]: print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test, pred)))

         print('Error Rate:', (fp + fn) / (tp + tn + fn + fp))

         print('Sensitivity (Recall or True positive rate):', tp / (tp + fn))

         print('Specificity (True negative rate):', tn / (fp + tn))

         print('Precision (Positive predictive value):', tp / (tp + fp))

        print('False Positive Rate:', fp / (tn + fp))
```

Practical No: 5 Data Analysis [ii]

```
IN[1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn.model_selection import train_test_split
       from sklearn.linear_model import LogisticRegression
       from sklearn.preprocessing import StandardScaler
       from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,classification_rep
IN[2]: df=pd.read_csv("Social_Network_Ads.csv")
IN[3]: df.head(5)
IN[4]: df.info()
IN[5]: df.describe()
IN[6]: df.isnull().sum()
IN[7]: df.shape
IN[8]: x= df.iloc[:,2:4]
IN[9]: y= df.iloc[:,4]
IN[10]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_stat
IN[11]: scale = StandardScaler()
         X_train = scale.fit_transform(x_train)
         X_test = scale.transform(x_test)
IN[12]: log_reg = LogisticRegression(random_state = 0)
         log_reg.fit(x_train, y_train)
         pred = log_reg.predict(x_test)
         print(x_test[:10])
         print('-'*15)
         print(pred[:10])

IN[13]: print('Expected Output:',pred[:10])
         print('-'*15)
         print('Predicted Output:\n' ,y_test[:10])

IN[14]: matrix= confusion_matrix(y_test,pred,labels=log_reg.classes_)
         print(matrix)
        tp,fn,fp,tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
IN[15]: conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix, display_labels=log_reg.cl
        conf_matrix.plot(cmap=plt.cm.Blues)
IN[16]: print(classification_report(y_test,pred))
IN[17]: from sklearn.metrics import accuracy_score
        print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test, pred)))
        print('Error Rate:', (fp + fn) / (tp + tn + fn + fp))
        print('Sensitivity (Recall or True positive rate):', tp / (tp + fn))
        print('Specificity (True negative rate):', tn / (fp + tn))
        print('Precision (Positive predictive value):', tp / (tp + fp))
        print('False Positive Rate:', fp / (tn + fp))
```

Practical No: 4 Data Analysis [i]

```
IN[1]: import pandas as pd
       import numpy as np
       import matplotlib.pyplot as plt
       import seaborn as sns

IN[2]: df = pd.read_csv('boston.csv')
IN[3]: df.head()
IN[4]: df.tail()
IN[5]: df.describe()
IN[6]: df.shape
IN[7]: df.dtypes
IN[8]: df .info()
IN[9]: df .isna() .sum()
IN[10]: mean_value = df ['CRIM' ] .mean()
IN[11]: means = df .mean()
        df .fillna (value =means , inplace =True )
       print(df.isnull().sum())
IN[12]: target_feature = 'MEDV'
IN[13]: x = df.drop(target_feature, axis=1)
        y = df[target_feature]
IN[14]: x.head()
IN[15]: y.head()
IN[16]: from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)

IN[17]: from sklearn.linear_model import LinearRegression
        regression = LinearRegression()
IN[18]: train_score=round(regression.score(x_train,y_train)*100,2)
        print('Train score of Linear Regression:',train_score)

IN[19]: print('Coefficients" ', regression.coef_)
```

```
IN[20]: predictions = regression.predict(x_test)

IN[21]: predictions

IN[22]: plt.scatter(y_test, predictions)
        plt.xlabel('Y Test')
        plt.ylabel('Predicted Y')

IN[23]: from sklearn.metrics import r2_score
        score = round(r2_score(y_test,predictions)*100,2)
        print("r_2 score:", score)

IN[24]: round(regression.score(x_test, y_test)*100,2)
```

Practical No: 9 Data Visulazation[ii]

```
IN[1]: import seaborn as sns
       titanic = sns.load_dataset("titanic")
IN[2]: titanic
IN[3]: titanic.head(10)
IN[4]: titanic.info
IN[5]: titanic.describe()
IN[6]: titanic.loc[:,["survived","alive"]]
IN[7]: sns.boxplot(x="sex",y="age",data=titanic)
IN[8]: sns.boxplot(x="sex",y="age",data=titanic,hue="survived")
```

Practical No: 10 Data Visulization [iii]

```
IN[1]: import matplotlib.pyplot as plt
       import seaborn as sns
       import pandas as pd
IN[2]: df = pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/
      ↪Iris-Dataset/master/Iris.csv")
      df.head()

IN[3]: df.describe()
IN[4]: df.shape
IN[5]: df["Species"].unique()
IN[6]: df.groupby("Species").size()
IN[7]: df.info()
IN[8]: corr = df.corr()
       plt.subplots(figsize=(10,6))
       sns.heatmap(corr, annot=True)
IN[9]: def graph(y):
       sns.boxplot(x="Species", y=y, data=df)
       plt.figure(figsize=(10,10))
      plt.subplot(221)
      graph('SepalLengthCm')
      plt.subplot(222)
      graph('SepalWidthCm')
      plt.subplot(223)
     graph('PetalLengthCm')
     plt.subplot(224)
     graph('PetalWidthCm')
     plt.show()
IN[10]: sns.boxplot(x='SepalWidthCm', data=df)
        plt.show()
       sns.boxplot(x='SepalLengthCm', data=df)
```

```
plt.show()

sns.boxplot(x='PetalWidthCm', data=df)

plt.show()

sns.boxplot(x='PetalLengthCm', data=df)

plt.show()
```