**1.**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**dataset = pd.read\_csv("suv\_data.csv")**

**dataset.head()**

**x = dataset.iloc[:, [2, 3]].values**

**y = dataset.iloc[:, 4].values**

**bool\_series=pd.isnull(dataset["Gender"])**

**dataset[bool\_series]**

**bool\_series=pd.notnull(dataset["Gender"])**

**dataset[bool\_series] dataset[10:25]**

**new\_data=dataset.dropna(axis=0,how='any')**

**new\_data**

**dataset.replace(to\_replace=np.nan,value=-99)**

**dataset["Gender"].fillna("No Gender",inplace=True)**

**dataset**

**print("Old data frame length:", len(dataset))**

**print("New data frame length:", len(dataset))**

**print("Number of rows with at least 1 NA value:",**

**len(dataset)-len(new\_data))**

**Old data frame length: 400**

**New data frame length: 400**

**Number of rows with at least 1 NA value: 0**

**new\_df1=dataset.fillna(method="ffill")**

**new\_df1**

**new\_df3=dataset.dropna(how='all')**

**new\_df3**

**2.**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**data = pd.read\_csv('Titanic-Dataset.csv') print(data)**

**x = data.drop('Survived', axis = 1)**

**y = data['Survived']**

**print(x)**

**print(y)**

**x.drop(['Name', 'Ticket', 'Cabin'],axis = 1, inplace =**

**True)**

**print(x)**

**x['Age'] = x['Age'].fillna(x['Age'].mean())**

**print(x)**

**x['Embarked'] =**

**x['Embarked'].fillna(x['Embarked'].mode()[0])**

**print(x)**

**x = pd.get\_dummies(x, columns = ['Sex',**

**'Embarked'],prefix = ['Sex', 'Embarked'],drop\_first =**

**True)**

**print(x)**

**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)**

**print(x\_train)**

**print(y\_train)**

**from sklearn.preprocessing import StandardScaler**

**std\_x = StandardScaler()**

**x\_train = std\_x.fit\_transform(x\_train)**

**x\_test = std\_x.transform(x\_test)**

**print(x\_train)**

**3.**

**from pandas import read\_csv**

**from numpy import set\_printoptions**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.feature\_selection import SelectKBest**

**from sklearn.feature\_selection import f\_classif**

**from matplotlib import pyplot**

**path=r'diabetes.csv'**

**names=['preg','plas','pres','skin','test','mass','peds','ag**

**e','class']**

**dataframe=read\_csv(path,names=names)**

**dataframe.head()**

**array=dataframe.values**

**x=array[:,0:8]**

**y=array[:,8]**

**print(x)**

**print(y)**

**x\_train,x\_test,y\_train,y\_test,=train\_test\_split(x,y,test\_**

**size=0.33,random\_state=1)**

**fs= SelectKBest(score\_func=f\_classif,k='all')**

**fs.fit(x\_train,y\_train)**

**x\_train\_fs=fs.transform(x\_train)**

**x\_test\_fs=fs.transform(x\_test)**

**for i in range(len(fs.scores\_)):**

**print('feature %d:%f'%(i,fs.scores\_[i]))**

**pyplot.bar([i for i in range(len(fs.scores\_))],fs.scores\_)**

**pyplot.show()**

**4.**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns from sklearn.feature\_selection**

**import chi2**

**df=pd.read\_csv('loandata.csv')**

**df.head()**

**from sklearn.preprocessing import LabelEncoder**

**for col in df.columns:**

**le=LabelEncoder()**

**df[col]=le.fit\_transform(df[col])**

**df.head()**

**x=df.iloc[:,0:6]**

**y=df.iloc[:,-1]**

**f\_score=chi2(x,y)**

**f\_score**

**p\_value=pd.Series(f\_score[1], index=x.columns)**

**p\_value.sort\_values(ascending=False,inplace=True)**

**p\_value**

**p\_value.plot(kind="bar")**

**plt.xlabel("Features", fontsize=20)**

**plt.ylabel("p\_values", fontsize=20)**

**plt.title("chi squared test base on p value")**

**plt.show()**

**5**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import seaborn as sns**

**from sklearn import tree**

**from sklearn import metrics**

**from sklearn.metrics import accuracy\_score, classification\_report**

**from sklearn.datasets import load\_iris**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.model\_selection import train\_test\_split**

**iris = load\_iris()**

**iris = sns.load\_dataset('iris')**

**iris.head()**

**x=iris.iloc[:,:-1]**

**y=iris.iloc[:,-1]**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.33, random\_state=42)**

**treemodel = DecisionTreeClassifier()**

**treemodel.fit(x\_train, y\_train)**

**y\_pred = treemodel.predict(x\_test)**

**plt.figure(figsize=(20,30))**

**tree.plot\_tree(treemodel, filled=True)**

**print(classification\_report(y\_test, y\_pred))**

**from sklearn.metrics import confusion\_matrix**

**cm=confusion\_matrix(y\_test, y\_pred)**

**print("Confusion Matrix:")**

**print(cm)**

**from sklearn.metrics import accuracy\_score**

**accuracy\_score(y\_test, y\_pred)**

**6.**

**import numpy as nm**

**import matplotlib.pyplot as mtp**

**import pandas as pd**

**datasetpd.read\_csv('User\_data.csv")**

**x= dataset.iloc[:,[2,3]].values**

**print(x) from sklearn.preprocessing import**

**StandardScaler from sklearn.naive bayes import**

**GaussianNB**

**y=dataset.iloc[:,4].values**

**print(y)**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,**

**test\_size=0.25, random\_state=0)**

**sc StandardScaler()**

**x\_train sc.fit\_transform(x\_train)**

**x\_testsc.fit\_transform(x\_test)**

**classifier GaussianNB()**

**classifier.fit(x\_train, y\_train)**

**y\_pred classifier.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix**

**cm confusion\_matrix(y\_test, y\_pred)**

**print("Confusion Matrix:")**

**print(cm)**

**7.**

**import numpy as np**

**model's performance.**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**dataset pd.read\_csv('Social Network Ads.csv')**

**X dataset.iloc[:, :-1].values**

**ydataset.iloc[:,-1].values**

**dataset.head()**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X test, y train, y\_test train\_test\_split(X, y, test**

**size 0.20, random state-42)**

**X**

**y**

**from sklearn.preprocessing import StandardScaler**

**se StandardScaler()**

**X\_train sc.fit\_transform(X\_train)**

**X\_test sc.transform(X\_test)**

**print(X\_train)**

**from sklearn.neighbors import KNeighbors Classifier**

**classifier KNeighborsClassifier(n\_neighbors 5, metrie**

**'minkowski', p2)**

**classifier.fit(X\_train, y\_train)**

**print(classifier.predict(sc.transform([[46,28000]])))**

**y\_pred classifier.predict(X\_test)**

**print(np.concatenate((y\_pred.reshape(len(y\_pred), 1),**

**y\_test.reshape(len(y\_test),1)),1))**

**from sklearn.metrics import confusion\_matrix,**

**accuracy\_score**

**cm confusion\_matrix(y\_test, y\_pred)**

**print(cm)**

**accuracy\_score(y\_test, y\_pred)**

**8. import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**datasetpd.read\_csv('Salary\_Data.csv')**

**x=dataset.iloc[:, :-1],values**

**y=dataset.iloc[:,-1].values**

**dataset.head()**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_testtrain\_test\_split(x,y,test\_size=1/3, random\_state-0)**

**print(x\_train)**

**print(x\_test)**

**print(y train)**

**print(y\_test)**

**from sklearn.linear model import LinearRegression**

**regressor-LinearRegression()**

**regressor.fit(x\_train.y\_train)**

**y\_pred-regressor.predict(x\_test)**

**print(y\_test)**

**print(y\_pred)**

**print(np.concatenate((y\_test.reshape(len(y\_test),1),y\_**

**pred.reshape(len(y\_pred),131,13)**

**from sklearn.metrics import mean\_squared\_error**

**mean mean\_squared\_error(y\_test,y\_pred)**

**mean**

**plt.scatter(x\_train,y\_train,color='red')**

**plt.plot(x train regressor.predict(x\_train),color="blue")**

**plt.title('salary vs Experience(Training set)')**

**plt.xlabel('years of Experience')**

**plt.ylabel('salary)**

**plt.show()**

**plt.scatterfx test, y test, color-'red')**

**plt.plot(x train, regressor.predict(x\_train),**

**color="blue")**

**plt.titlet Salary vs Experience (Test set)')**

**plt.xlabel("Years of Experience')**

**plt.ylabel("Salary)**

**plt.show()**

**9.**

**import numpy as np**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_squared\_error,**

**r2\_score**

**from sklearn.preprocessing import OneHotEncoder**

**from sklearn.compose import ColumnTransformer**

**data = pd.read\_csv('50\_Startups.csv')**

**df = pd.DataFrame(data)**

**print("Dataset:")**

**print(df.head())**

**X = df.drop(columns=['Profit'])**

**y = df['Profit']**

**column\_transformer = ColumnTransformer(**

**transformers=[('encoder', OneHotEncoder(),**

**['State'])], remainder='passthrough')**

**X = column\_transformer.fit\_transform(X)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,**

**test\_size=0.2, random\_state=42)**

**model = LinearRegression()**

**model.fit(X\_train, y\_train)**

**y\_pred = model.predict(X\_test)**

**mse = mean\_squared\_error(y\_test, y\_pred)**

**r2 = r2\_score(y\_test, y\_pred)**

**print(f"Mean Squared Error: {mse}")**

**print(f"R-squared Score: {r2}")**

**print("Coefficients:", model.coef\_)**

**print("Intercept:", model.intercept\_)**

**10.**

**import numpy as nm**

**import matplotlib.pyplot as mtp**

**import pandas as pd**

**dataset = pd.read\_csv('Mall\_Customers.csv')**

**x = dataset.iloc[:, [3, 4]].values**

**dataset.head()**

**import scipy.cluster.hierarchy as shc**

**dendro = shc.dendrogram(shc.linkage(x,**

**method="ward"))**

**mtp.title("Dendrogrma Plot")**

**mtp.ylabel("Euclidean Distances")**

**mtp.xlabel("Customers")**

**mtp.show()**

**from sklearn.cluster import AgglomerativeClustering**

**hc = AgglomerativeClustering(n\_clusters=5,**

**metric='euclidean', linkage='ward')**

**y\_pred = hc.fit\_predict(x)**

**#visulaizing the clusters**

**mtp.scatter(x[y\_pred == 0, 0], x[y\_pred == 0, 1], s =100, c = 'blue', label = 'Cluster 1')**

**mtp.scatter(x[y\_pred == 1, 0], x[y\_pred == 1, 1], s =100, c = 'green', label = 'Cluster 2')**

**mtp.scatter(x[y\_pred== 2, 0], x[y\_pred == 2, 1], s =100, c = 'red', label = 'Cluster 3')**

**mtp.scatter(x[y\_pred == 3, 0], x[y\_pred == 3, 1], s =100, c = 'cyan', label = 'Cluster 4')**

**mtp.scatter(x[y\_pred == 4, 0], x[y\_pred == 4, 1], s =100, c = 'magenta', label = 'Cluster 5')**

**mtp.title('Clusters of customers')**

**mtp.xlabel('Annual Income (k$)')**

**mtp.ylabel('Spending Score (1-100)')**

**mtp.legend()**

**mtp.show()**