**Practical No 3**

import pandas as pd

import seaborn as sns

import os

df=pd.read\_csv("D:\Admission\_Predict.csv")

df.columns

df.shape

df.head()

from sklearn.preprocessing import Binarizer

bi=Binarizer(threshold=0.75)

df['Chance of Admit ']=bi.fit\_transform(df[['Chance of Admit ']])

df.head()

x=df.drop('Chance of Admit ',axis=1)

y=df['Chance of Admit ']

x

y

y=y.astype('int')

sns.countplot(x=y)

y.value\_counts()

#crossvalidation

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=0,test\_size=0.25)

x\_train.shape

x\_test.shape

x\_test

from sklearn.tree import DecisionTreeClassifier

classifier=DecisionTreeClassifier(random\_state=0)

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

**Practical no 2**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import os

df=pd.read\_csv("D:temperatures.csv")

df.describe()

df.head

#input Data

x=df['YEAR']

#output Data

y=df['ANNUAL']

plt.figure(figsize=(15,8))

plt.title('Temperature plot of india')

plt.xlabel('YEAR')

plt.ylabel('Annua laverage Temp')

plt.scatter(x,y)

x.shape

x=x.values

x=x.reshape(117,1)

x.shape

y.shape

from sklearn.linear\_model import LinearRegression

re=LinearRegression()

re.fit(x,y)

#slope or coeficient

re.coef\_

re.intercept\_

re.predict([[2024]])

predicted=re.predict(x)

predicted

y- predicted

import numpy as np

np.mean(abs(y-predicted)\*\*2)

from sklearn.metrics import mean\_squared\_error

mean\_squared\_error(y,predicted)

from sklearn.metrics import r2\_score

r2\_score(y,predicted)

re.score(x,y)

plt.scatter(x,y,label='actual',color='r',marker='.')

plt.plot(x,predicted,label='predicted',color='g')

plt.legend()

Practical no 4

import pandas as pd

import matplotlib.pyplot as plt

df=pd.read\_csv('D:\Mall\_customers.csv')

df

x=df.iloc[:,3:]

x

plt.title('unclustered data')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.scatter(x['Annual Income (k$)'],x['Spending Score (1-100)'])

from sklearn.cluster import KMeans,AgglomerativeClustering

Km=KMeans(n\_clusters=3)

x.shape

Km.fit\_predict(x)

#sam square error

Km.inertia\_

sse=[]

for K in range(1,16):

Km=KMeans(n\_clusters=k)

Km.fit\_predict(x)

sse.append(km.inertia\_)

sse

plt.title('Elbow Method')

plt.xlabel('Value of k')

plt.ylabel('SSE')

plt.grid()

plt.xticks(range(1,16))

plt.plot(range(1,16),sse,marker='.',color='red')

from sklearn.metrics import silhouette\_score

silh=[]

for k in range(2,16):

km=KMeans(n\_clusters=k)

labels= km.fit\_predict(x)

score= silhouette\_score(x,labels)

silh.append(score)

plt.title('Silhout Method')

plt.xlabel('value of k')

plt.ylabel('Silhoute Score')

plt.grid()

plt.xticks(range(2,16))

plt.bar(range(2,16),silh,color='red')

km=KMeanS(n\_clusters=5,random\_state=0)

labels=km.fit\_predict(x)

km=KMeanS(n\_clusters=5,random\_state=0)

labels=km.fit\_predict(x)

plt.figure(figsize=(16,9))

plt.subplot(1,2,1)

plt.title('unclustred data')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.scatter(x['Annual Income (k$)'],x['Spending Score (1-100)'])

plt.figure(figsize=(16,9))

cent=km.cluster\_centers\_

plt.subplot(1,2,1)

plt.title('clustered data')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.scatter(x[''])

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

result=pd.DataFrame({'actual': y\_test,'predicted': y\_pred})

result

from sklearn.metrics import ConfusionMatrixDisplay,accuracy\_score

from sklearn.metrics import classification\_report

ConfusionMatrixDisplay.from\_predictions(y\_test,y\_test)

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

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