**EXPERIMENT NO : 5**

**k-NN Algorithm**

**AIM : Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm**

from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

iris = load\_iris()

x = iris.data

y = iris.target

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)

c\_knn = KNeighborsClassifier(n\_neighbors=3)

c\_knn.fit(x\_train,y\_train)

y\_pred = c\_knn.predict(x\_test)

print("Accuracy : ",metrics.accuracy\_score(y\_test,y\_pred))

sample = [[2,2,2,2]]

pred = c\_knn.predict(sample)

pred\_v = [iris.target\_names[p] for p in pred]

print(pred\_v)

**EXPERIMENT NO : 6**

**Naive Bayes algorithm**

**AIM :Program to implement Naive Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm**

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

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

**from sklearn.naive\_bayes import GaussianNB**

**X,y=load\_iris(return\_X\_y=True)**

**X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.5,random\_state=0)**

**gnb=GaussianNB()**

**y\_pred=gnb.fit(X\_train,y\_train).predict(X\_test)**

**print(y\_pred)**

**x\_new=[[5,5,4,4]]**

**y\_new=gnb.fit(X\_train,y\_train).predict(x\_new)**

**print("predicted output for [[5,5,4,4]]:",y\_new)**

**print("Naive bayes score :",gnb.score(X\_test,y\_test))**

**EXPERIMENT NO :10**

**DECISION TREE**

**AIM : Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.**

from sklearn.datasets import load\_iris

from sklearn import metrics

from sklearn import tree

import matplotlib.pyplot as plt

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

iris = load\_iris()

x,y = iris.data,iris.target

x\_train,x\_test, y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.3,random\_state = 1)

clf = DecisionTreeClassifier(criterion = 'entropy')

clf = clf.fit(x\_train,y\_train)

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

print('Accuracy : ',metrics.accuracy\_score(y\_test,y\_pred))

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

tree.plot\_tree(clf,

fontsize = 10,

filled = True,

rounded = True,

class\_names = iris.target\_names,

feature\_names = iris.feature\_names)

plt.show()

**EXPERIMENT NO : 11**

**AIM :Program to implement k-means clustering technique using any standard dataset available in the public domain**

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.cluster import KMeans

iris = datasets.load\_iris()

X = iris.data

y = iris.target

plt.scatter(X[:,1], X[:,3],color='white', marker='o', edgecolor='red', s=50)

plt.grid()

plt.tight\_layout()

plt.show()

kmc = KMeans(n\_clusters=3, init='random', n\_init=10, max\_iter=300,tol=1e-04, random\_state=0)

y\_kmc = kmc.fit\_predict(X)

plt.scatter(X[y\_kmc == 0, 1], X[y\_kmc == 0, 3], s=50,c='lightgreen', marker='s', edgecolor='black', label='Cluster 1')

plt.scatter(X[y\_kmc == 1, 1], X[y\_kmc == 1, 3],s=50, c='orange', marker='o', edgecolor='black', label='Cluster 2')

plt.scatter(X[y\_kmc == 2, 1], X[y\_kmc == 2, 3], s=50, c='blue', marker='P', edgecolor='black', label='Cluster 3')

plt.scatter(kmc.cluster\_centers\_[:, 1], kmc.cluster\_centers\_[:, 3],

s=250, marker='\*', c='red', edgecolor='black', label='Centroids')

plt.legend(scatterpoints=1)

plt.grid()

plt.tight\_layout()

plt.show()

**EXPERIMENT NO :12**

**AIM : Implement a program to scrap the web page of any popular website**

**(scrapes the titles of articles from a website)**

import requests

from bs4 import BeautifulSoup

# Step 1: Send a request to the webpage

url = 'https://www.geeksforgeeks.org/mac

response = requests.get(url)

# Step 2: Parse the HTML content

soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all the article titles (assuming they are in <h2> tags)

titles = soup.find\_all('h2')

# Step 4: Print the titles

for title in titles:

    print(title.get\_text())