Practical 1:-

import csv

num\_attributes = 6

a = []

print("\n The Given Training Data Set \n")

with open('/kaggle/input/enjoysport/ENJOYSPORT.csv', 'r') as csvfile:

reader = csv.reader(csvfile)

for row in reader:

a.append(row)

print(row)

print("\n The initial value of hypothesis:")

hypothesis = ['0'] \* num\_attributes

print(hypothesis)

for j in range(0, num\_attributes):

hypothesis[j] = a[0][j]

print("\n Find S: Finding a Maximally Specific Hypothesis\n")

for i in range(0, len(a)):

if a[i][num\_attributes] == 'yes':

for j in range(0, num\_attributes):

if a[i][j] != hypothesis[j]:

hypothesis[j] = '?'

else:

hypothesis[j] = a[i][j]

print(" For Training instance No:{0} the hypothesis is ".format(i), hypothesis)

print("\n The Maximally Specific Hypothesis for a given Training Examples :\n")

print(hypothesis)

Practical 4:-

import pandas as pd

from sklearn import tree

from sklearn.preprocessing import LabelEncoder

from sklearn.naive\_bayes import GaussianNB

# load data from CSV

data = pd.read\_csv('/kaggle/input/tennisdata-csv/tennisdata.csv')

print("THe first 5 values of data is :\n",data.head())

# obtain Train data and Train output

X = data.iloc[:,:-1]

print("\nThe First 5 values of train data is\n",X.head())

y = data.iloc[:,-1]

print("\nThe first 5 values of Train output is\n",y.head())

# Convert then in numbers

le\_outlook = LabelEncoder()

X.Outlook = le\_outlook.fit\_transform(X.Outlook)

le\_Temperature = LabelEncoder()

X.Temperature = le\_Temperature.fit\_transform(X.Temperature)

le\_Humidity = LabelEncoder()

X.Humidity = le\_Humidity.fit\_transform(X.Humidity)

le\_Windy = LabelEncoder()

X.Windy = le\_Windy.fit\_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())

le\_PlayTennis = LabelEncoder()

y = le\_PlayTennis.fit\_transform(y)

print("\nNow the Train output is\n",y)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.20)

classifier = GaussianNB()

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

from sklearn.metrics import accuracy\_score

print("Accuracy is:",accuracy\_score(classifier.predict(X\_test),y\_test))

Practical 5:-

# import necessary libraries

import pandas as pd

from sklearn import tree

from sklearn.preprocessing import LabelEncoder

from sklearn.naive\_bayes import GaussianNB

# load data from CSV

data = pd.read\_csv('/kaggle/input/documentt-csv/document.csv', names=['message', 'label'])

print("Total Instances of Dataset: ", data.shape[0])

data['labelnum'] = data.label.map({'pos': 1, 'neg': 0})

X = data.message

y = data.labelnum

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

Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y)

from sklearn.feature\_extraction.text import CountVectorizer

count\_v = CountVectorizer()

Xtrain\_dm = count\_v.fit\_transform(Xtrain)

Xtest\_dm = count\_v.transform(Xtest)

df = pd.DataFrame(Xtrain\_dm.toarray(), columns=count\_v.get\_feature\_names\_out())

print(df[0:5])

from sklearn.naive\_bayes import MultinomialNB

clf = MultinomialNB()

clf.fit(Xtrain\_dm, ytrain)

pred = clf.predict(Xtest\_dm)

for doc, p in zip(Xtrain, pred):

p = 'pos' if p == 1 else 'neg'

print("%s -> %s" % (doc, p))

from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score

print('Accuracy Metrics: \n')

print('Accuracy: ', accuracy\_score(ytest, pred))

Practical 7:-

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.mixture import GaussianMixture

import pandas as pd

import numpy as np

def load\_iris\_data():

iris = datasets.load\_iris()

X = pd.DataFrame(iris.data)

X.columns = ['Sepal\_Length', 'Sepal\_Width', 'Petal\_Length', 'Petal\_Width']

y = pd.DataFrame(iris.target)

y.columns = ['Targets']

return X, y

def visualize\_clusters(X, y, model\_labels, title, colormap):

plt.scatter(X.Petal\_Length, X.Petal\_Width, c=colormap[model\_labels], s=40)

plt.title(title)

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

def main():

X, y = load\_iris\_data()

kmeans\_model = KMeans(n\_clusters=3)

kmeans\_model.fit(X)

scaler = StandardScaler()

xs = scaler.fit\_transform(X)

gmm = GaussianMixture(n\_components=3)

gmm.fit(xs)

plt.figure(figsize=(14, 7))

colormap = np.array(['red', 'lime', 'black'])

plt.subplot(1, 3, 1)

plt.scatter(X.Petal\_Length, X.Petal\_Width, c=colormap[y.Targets], s=40)

plt.title('Real Clusters')

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

plt.subplot(1, 3, 2)

visualize\_clusters(X, y, kmeans\_model.labels\_, 'K-Means Clustering', colormap)

plt.subplot(1, 3, 3)

visualize\_clusters(X, y, gmm.predict(xs), 'GMM Clustering', colormap)

plt.show()

print('Observation: The GMM using EM algorithm-based clustering matched the true labels more closely than KMeans.')

if \_\_name\_\_ == "\_\_main\_\_":

main()

Practical 8:-

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

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt

import numpy as np

irisData = load\_iris()

X = irisData.data

y = irisData.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2, random\_state=42)

knn= KNeighborsClassifier(n\_neighbors=7)

knn.fit(X\_train,y\_train)

print(knn.predict(X\_test))

print(knn.score(X\_test,y\_test))

neighbours = np.arange(1,9)

train\_accuracy = np.empty(len(neighbours))

test\_accuracy = np.empty(len(neighbours))

for i,k in enumerate(neighbours):

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(X\_train,y\_train)

train\_accuracy[i] = knn.score(X\_train,y\_train)

test\_accuracy[i] = knn.score(X\_test,y\_test)

plt.plot(neighbours,test\_accuracy,label='Testing dataset Accuracy')

plt.plot(neighbours,train\_accuracy,label='Training dataset Accuracy')

plt.legend()

plt.xlabel('n\_neighbors')

plt.ylabel('Accuracy')

plt.show()