PROGRAM-1

FIND-S

import pandas as pd

import itertools

df=pd.read\_csv('ENJOYSPORT.csv',header=None)

df.drop(0,axis=0,inplace=True)

df.head()

row\_len,col\_len=df.shape

a=[]

flag=0

for index,row in df.iterrows():

if(row[col\_len-1]=='1' or row[col\_len-1]==1):

if(flag==0):

flag=1

a.extend(row[0:col\_len-1])

for i in range(col\_len-1):

if(row[i]==a[i]):

a[i]==row[i]

else:

a[i]='?'

print(a)

new\_list=list()

for i in range(col\_len-1):

version\_space=set()

for index,row in df.iterrows():

version\_space.add(row[i])

version\_space.add('$')

version\_space.add('?')

new\_list.append(list(version\_space))

new\_list

version\_space=list(itertools.product(\*new\_list))

print('total no element in versionSpace',len(version\_space))

version\_space[:10]

def apply(key,df):

for index,row in df.iterrows():

flag=0

for i in range(col\_len-1):

if(key[i]=='?' or key[i]==row[i]):

flag+=1

else:

continue

if(flag==col\_len-1 and row[col\_len-1]=='1') or (flag!=col\_len-1 and row[col\_len-1]=='0'):

continue

else:

return 1

return 0

for i in version\_space:

p=apply(i,df)

if(p==0):

print(i)

PROGRAM-2

import numpy as np

import pandas as pd

data = pd.read\_csv(r'ENJOYSPORT.csv')

concepts = np.array(data.iloc[:,0:-1])

print("\nInstances are:\n",concepts)

target = np.array(data.iloc[:,-1],dtype=np.int64)

print("\nTarget Values are: ",target)

def learn(concepts, target):

specific\_h = concepts[0].copy()

print("\nInitialization of specific\_h and genearal\_h")

print("\nSpecific Boundary: ", specific\_h)

general\_h = [["?" for i in range(len(specific\_h))] for i in range(len(specific\_h))]

print("\nGeneric Boundary: ",general\_h)

for i, h in enumerate(concepts):

print("\nInstance", i+1 , "is ", h)

if target[i] == 1:

print("Instance is Positive ")

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

specific\_h[x] ='?'

general\_h[x][x] ='?'

if target[i] == 0:

print("Instance is Negative ")

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

general\_h[x][x] = specific\_h[x]

else:

general\_h[x][x] = '?'

print("Specific Bundary after ", i+1, "Instance is ", specific\_h)

print("Generic Boundary after ", i+1, "Instance is ", general\_h)

print("\n")

learn(concepts, target)

PROGRAM-3

Import pandas as pd

df=pd.read\_excel('dataset\_program\_3.xlsx')

df.head()

a=list(df.columns)

for i in a:

print(len(df[i].unique())<3)

df.drop('origin',axis=1,inplace=True)

df.head()

df.shape

df.isnull().sum()

a=df['displacement'].mean()

b=df['horsepower'].mean()

values = {"displacement": a, "horsepower":b }

df.fillna(value=values,inplace=True)

df.isnull().sum()

data=df.duplicated(keep=False)

data

data=df[df.duplicated(keep=False)]

data.sort\_values(by='mpg')

data = pd.DataFrame(df)

display(data.drop\_duplicates())

PROGRAM-4

import pandas as pd

import numpy as np

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

from matplotlib import pyplot as plt

import warnings

warnings.filterwarnings('ignore')

from sklearn import tree

df=pd.read\_csv("Iris.csv")

df.head()

df.drop('Id',axis=1,inplace=True)

df.head()

le = LabelEncoder()

df['Species']= le.fit\_transform(df['Species'])

df['Species'].unique()

X=df.iloc[:,:4]

y=df.iloc[:,4:]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,random\_state=123)

clf=DecisionTreeClassifier(criterion='entropy', max\_leaf\_nodes=3)

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

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

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

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

fn=['SepalLengthCm' ,'SepalWidthCm' ,'PetalLengthCm', 'PetalWidthCm']

cn=['Iris-setosa',' Iris-versicolor','Iris-virginica']

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (2,2), dpi=400)

tree.plot\_tree(clf,

feature\_names = fn,

class\_names=cn,

filled = True);

species\_check = clf.predict([[4.7, 3.2, 1.3, 0.2]])[0]

species\_check

PROGRAM-5

RANDOM FOREST

import pandas as pd

import numpy as np

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn import metrics

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from matplotlib import pyplot as plt

import warnings

warnings.filterwarnings('ignore')

from sklearn import tree

df=pd.read\_csv("Iris.csv")

df.head()

df.drop('Id',axis=1,inplace=True)

df.head()

le = LabelEncoder()

df['Species']= le.fit\_transform(df['Species'])

df['Species'].unique()

X=df.iloc[:,:4]

y=df.iloc[:,4:]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,random\_state=123)

clf=RandomForestClassifier(n\_estimators=3)

clf.fit(X\_train,y\_train.values.ravel())

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

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

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

fn=['SepalLengthCm' ,'SepalWidthCm' ,'PetalLengthCm', 'PetalWidthCm']

cn=['Iris-setosa',' Iris-versicolor','Iris-virginica']

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=800)

tree.plot\_tree(clf.estimators\_[0],

feature\_names = fn,

class\_names=cn,

filled = True)

species\_check = clf.predict([[4.7,3.2,1.3,0.2]])[0]

species\_check

PROGRAM-6

import pandas as pd

dataset = pd.read\_csv('6 naivebasedataset.csv',header=None)

Y=dataset[8]

X=dataset.drop([8], axis=1)

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=1)

print('X\_trian = ',len(X\_train))

print('X\_test = ',len(X\_test))

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

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

y\_pred = gnb.predict(X\_test)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy\_score(y\_test, y\_pred)\*100)

PROGRAM-7

import numpy as np, pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.datasets import load\_files

import seaborn as sns

test=load\_files(r'C:\Users\Lishakka\OneDrive\Desktop\20news-bydate-test',encoding='latin1')

train=load\_files(r'C:\Users\Lishakka\OneDrive\Desktop\20news-bydate-train',encoding='latin1')

print("Number of unique classes {}".format(len(set(train.target))))

print("Number of training samples {} ".format(len(train.data)))

print("Number of test samples {}".format(len(test.data)))

train.data[1]

model = make\_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(train.data, train.target)

predicted\_categories = model.predict(test.data)

a=np.array(test.target\_names)[predicted\_categories]

print(a)

mat=confusion\_matrix(test.target, predicted\_categories)

sns.heatmap(mat,annot=True)

print(classification\_report(test.target, predicted\_categories,target\_names=set(a)))

Program-8

!pip install pgmpy

!pip install networkx

import pandas as pd

import numpy as np

from pgmpy.estimators import MaximumLikelihoodEstimator

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

import networkx as nx

import matplotlib.pyplot as plt

df=pd.read\_csv("8-dataset.csv")

df.head()

print(df.info())

print(pd.unique(df['age']))

model= BayesianModel([('age','heartdisease'),('gender','heartdisease'),('exang','heartdisease'),('cp','heartdisease'),('heartdisease','restecg')])

model.fit(df,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')

HeartDisease\_infer = VariableElimination(model)

print('\n 1. Probability of HeartDisease given evidence= cp')

q1=HeartDisease\_infer.query(variables=['heartdisease'],evidence={'gender':1})

print(q1)

print('\n 1. Probability of HeartDisease given evidence= restecg')

q2=HeartDisease\_infer.query(variables=['heartdisease'],evidence={'restecg':1})

print(q2)

graph = nx.DiGraph(model.edges())

nx.draw\_networkx(graph,with\_labels=True)

#plot

plt.show()

Program-9

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import LabelEncoder

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

from sklearn.metrics import classification\_report

from sklearn.mixture import GaussianMixture

from sklearn.inspection import DecisionBoundaryDisplay

df = pd.read\_csv("Iris.csv")

df.head()

df.drop('Id',axis=1,inplace=True)

df.head()

le = LabelEncoder()

df['Species']= le.fit\_transform(df['Species'])

df['Species'].unique()

X=df.iloc[:,:4]

y=df.iloc[:,4:]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,random\_state=123)

wcss = [] #within cluster sum square

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.title('The elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

x=df.iloc[:,:4]

gmm=GaussianMixture(n\_components=3)

gmm.fit(X\_train)

y\_prediction=gmm.predict(X\_test)

class\_names=["Iris-setosa","Iris-virginica","Iris-versicolor"]

print(classification\_report(y\_test, y\_prediction ,target\_names=class\_names))

x=df.iloc[:,:2]

gmm=GaussianMixture(n\_components=3)

gmm.fit(x)

disp=DecisionBoundaryDisplay.from\_estimator(gmm,

x,

response\_method="predict",

alpha=0.5)

disp.ax\_.scatter(df.iloc[:,:1], df.iloc[:,1:2], c=df['Species'].to\_numpy(), edgecolor="k")

plt.plot()

Program-10

import matplotlib.pyplot as plt

from sklearn.inspection import DecisionBoundaryDisplay

from sklearn.svm import SVC

from sklearn.preprocessing import LabelEncoder

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

from sklearn.metrics import classification\_report

import pandas as pd

df = pd.read\_csv(r"Iris.csv")

df.drop('Id',axis=1,inplace=True)

df.head()

df.info()

le = LabelEncoder()

df['Species']= le.fit\_transform(df['Species'])

df['Species'].unique()

X=df.iloc[:,:4]

y=df.iloc[:,4:]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,random\_state=123)

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)

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

y\_prediction=svm.predict(X\_test)

class\_names=["Iris-setosa","Iris-virginica","Iris-versicolor"]

print(classification\_report(y\_test, y\_prediction ,target\_names=class\_names))

x=df.iloc[:,:2]

svm2 = SVC(kernel="rbf", gamma=0.5, C=1.0)

svm2.fit(x, y)

DecisionBoundaryDisplay.from\_estimator(

svm2,

x,

response\_method="predict",

cmap=plt.cm.Spectral,

alpha=0.8,

)