**01.a) Implement the FIND-S algorithm for finding the most specific hypothesis using the enjoy\_sport dataset.**

import numpy as np

import pandas as pd

data=pd.read\_csv("Enjoy\_sport1.csv")

print("Dataset : ",data)

concepts=data.iloc[:,:-1].values

target=data.iloc[:,-1].values

print("Concepts : ",concepts)

print("Target : ",target)

print("\n")

def train(conc,tar):

for i,val in enumerate(tar):

if val=="Yes":

specific\_h=conc[i].copy()

break

for i,val in enumerate(conc):

if tar[i]=="Yes":

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

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

specific\_h[x]="?"

else:

pass

return specific\_h

print("Final specific\_h :")

print(train(concepts,target))

**01.b)Construct a decision tree based on the ID3 algorithm. Use the Play\_Tennis dataset for building the decision tree and apply this knowledge to classify a new sample.**

import numpy as np

import pandas as pd

from sklearn.preprocessing import LabelEncoder

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

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

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

data=pd.read\_csv('Tennis.csv')

print("Dataset : ",data)

all\_cols=data.columns

features=all\_cols[1:5]

for i in data.columns:

data[i]=LabelEncoder().fit\_transform(data[i])

inputs=data.iloc[:,:-1].values

target=data.iloc[:,-1].values

print("Inputs : ",inputs)

print("Target : ",target)

print("\n")

x\_train,x\_test,y\_train,y\_test=train\_test\_split(inputs,target,test\_size=0.2,random\_state=0)

id3=DecisionTreeClassifier()

id3.fit(x\_train,y\_train)

y\_predict=id3.predict(x\_test)

print("Predicted value : ",y\_predict)

cm=confusion\_matrix(y\_test,y\_predict)

print("Confusion Matrix : ",cm)

print("Accuracy score of the model : ",accuracy\_score(y\_test,y\_predict))

tree.plot\_tree(id3,feature\_names=features)

**02.a)Demonstrate the working of the Candidate-Elimination algorithm to output a description of the set of all consistent hypotheses using the enjoy\_sport dataset.**

import numpy as np

import pandas as pd

data=pd.read\_csv('Enjoy\_sport1.csv')

print("Dataset : ",data)

concepts=data.iloc[:,:-1].values

target=data.iloc[:,-1].values

print("Concepts : ",concepts)

print("Target : ",target)

def initialize(concepts):

print("Initialization of specific\_h and general\_h")

specific\_h=['0']\*len(concepts[0])

print("Initial specific\_h : ",specific\_h)

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

print("Initial general\_h : ",general\_h)

return specific\_h,general\_h

def learn(concepts,target):

specific\_h,general\_h=initialize(concepts)

for i,h in enumerate(concepts):

print("Instance ",i+1," is ",h)

if target[i]=="Yes":

print("Instance is positive..")

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

if h[x]!=specific\_h[x] and i==0:

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

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

specific\_h[x]="?"

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

if target[i]=="No":

print("Instance is negative..")

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

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

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

else:

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

print("specific\_h after ",i+1," Instance : ",specific\_h)

print("general\_h after ",i+1," Instance : ",general\_h)

indices=[i for i,val in enumerate(general\_h) if val==["?","?","?","?","?","?"]]

for i in indices:

general\_h.remove(["?","?","?","?","?","?"])

indices=[i for i,val in enumerate(general\_h) if val==["?"]\*len(concepts[0])]

return specific\_h,general\_h

s\_final,g\_final=learn(concepts,target)

print("Final specific\_h : ",s\_final)

print("Final general\_h : ",g\_final)

import numpy as np

import pandas as pd

data=pd.read\_csv('Enjoy\_sport1.csv')

print("Dataset : ",data)

concepts=data.iloc[:,:-1].values

target=data.iloc[:,-1].values

print("Concepts : ",concepts)

print("Target : ",target)

def initialize(concepts):

print("Initialization of specific\_h and general\_h")

specific\_h=['0']\*len(concepts[0])

print("Initial specific\_h : ",specific\_h)

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

print("Initial general\_h : ",general\_h)

return specific\_h,general\_h

def learn(concepts,target):

specific\_h,general\_h=initialize(concepts)

for i,h in enumerate(concepts):

print("Instance ",i+1," is ",h)

if target[i]=="Yes":

print("Instance is positive..")

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

if h[x]!=specific\_h[x] and i==0:

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

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

specific\_h[x]="?"

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

if target[i]=="No":

print("Instance is negative..")

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

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

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

else:

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

print("specific\_h after ",i+1," Instance : ",specific\_h)

print("general\_h after ",i+1," Instance : ",general\_h)

indices=[i for i,val in enumerate(general\_h) if val==["?","?","?","?","?","?"]]

for i in indices:

general\_h.remove(["?","?","?","?","?","?"])

indices=[i for i,val in enumerate(general\_h) if val==["?"]\*len(concepts[0])]

return specific\_h,general\_h

s\_final,g\_final=learn(concepts,target)

print("Final specific\_h : ",s\_final)

print("Final general\_h : ",g\_final)

**02.b)Perform Random Forest classification on the Pima Indians diabetes dataset.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

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

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

data=pd.read\_csv('diabetes.csv')

print("Dataset : ",data)

print("\n")

x=data.drop('Outcome',axis=1)

y=data['Outcome']

print("x : ",x)

print("y : ",y)

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

model=RandomForestClassifier(n\_estimators=5,max\_features=5)

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

y\_predict=model.predict(x\_test)

print("Predicted value : ",y\_predict)

print("\n")

print("---confusion matrix---")

cm=confusion\_matrix(y\_test,y\_predict)

print(cm)

print("\n")

from sklearn.metrics import classification\_report

print("---classification report---")

cr=classification\_report(y\_test,y\_predict)

print(cr)

asc=accuracy\_score(y\_test,y\_predict)

print("Accuracy Score of the model : ",asc)

print("\n")

index=np.arange(0,len(y\_test))

fig,ax=plt.subplots(1,1,figsize=(15,5))

plt.scatter(index,y\_test,c='red',label='Actual Value')

plt.scatter(index,y\_predict,c='blue',label='Predicted Value')

plt.legend()

plt.show()

**03.a)Write a program to implement the k-Nearest Neighbor classification algorithm on the Breast Cancer dataset and visualize the results**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn import datasets

dataset=datasets.load\_breast\_cancer()

data=dataset.data

target=dataset.target

print("Data : ",data)

print("Target : ",target)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(data,target,test\_size=0.2)

knn=KNeighborsClassifier(n\_neighbors=3)

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

y\_predict=knn.predict(x\_test)

print("Predicted value : ",y\_predict)

for i in range(3):

r1=np.where(y\_predict == i)

r2=np.where(y\_test == i)

if i==0:

m='\*'

c='red'

elif i==1:

m='o'

c='green'

elif i==2:

m='x'

c='blue'

plt.scatter(x\_test[r1,1],x\_test[r1,0],marker=m,color=c)

plt.show()

**03.b)Demonstrate the use of the Support Vector Machine algorithm for a regression problem on the Iris flower dataset and evaluate the performance of the model**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn import datasets

data=pd.read\_csv('Iris.csv')

print("Dataset : ",data)

x=data.iloc[:,[0]].values

y=data.iloc[:,1].values

print("x : ",x)

y=y.reshape(len(y),1)

print("y : ",y)

#Feature scaling

from sklearn.preprocessing import StandardScaler

sc\_x=StandardScaler()

sc\_y=StandardScaler()

x=sc\_x.fit\_transform(x)

y=sc\_y.fit\_transform(y)

# Training the SVM Model using training dataset

from sklearn.svm import SVR

regressor=SVR(kernel='rbf')

regressor.fit(x,y)

print("New Value")

y\_predict=sc\_y.inverse\_transform(regressor.predict(sc\_x.transform([[6.5]])).reshape(-1,1))

print("Predicted value : ",y\_predict)

plt.scatter(sc\_x.inverse\_transform(x),sc\_y.inverse\_transform(y).reshape(-1,1),color='red')

plt.plot(sc\_x.inverse\_transform(x),sc\_y.inverse\_transform(regressor.predict(x).reshape(-1,1)),color='blue')

plt.title('Iris - ID VS SepalLength (SVR)')

plt.xlabel('ID')

plt.ylabel('Sepal Length (cm)')

plt.show()

**04.a)Demonstrate the use of the K-Means clustering algorithm on the Mall\_Customers dataset. Use the elbow method to find the optimal number of clusters and visualize the clusters.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

print("Dataset : ",data)

print("\n")

x=data.iloc[:,[3,4]].values

# Using elbow method to find the optimal number of clusters..

from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++',random\_state=42)

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()

# Training the kmeans model using training dataset

kmeans=KMeans(n\_clusters=5,init='k-means++',random\_state=42)

y\_kmeans=kmeans.fit\_predict(x)

print("Predicted values : ",y\_kmeans)

# Visualization of the model

plt.scatter(x[y\_kmeans == 0,0],x[y\_kmeans == 0,1],s=100,c='red',label='Cluster 1')

plt.scatter(x[y\_kmeans == 1,0],x[y\_kmeans == 1,1],s=100,c='blue',label='Cluster 2')

plt.scatter(x[y\_kmeans == 2,0],x[y\_kmeans == 2,1],s=100,c='green',label='Cluster 3')

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

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

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=300,c='yellow',label='Centroids')

plt.title('Clusters of Customers')

plt.xlabel('Anuual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**04.b)Demonstrate the application of Simple Linear regression on the Salary dataset.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

data=pd.read\_csv('Salary\_Data.csv')

print('Dataset : ',data)

x=data.iloc[:,:-1].values

y=data.iloc[:,-1].values

print("x : ",x)

print("y : ",y)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

model=LinearRegression()

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

y\_predict=model.predict(x\_test)

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

plt.plot(x\_test,y\_predict,color='blue')

plt.title('Experience Vs Salary')

plt.xlabel('Year of Experience')

plt.ylabel('Salary')

plt.show()

**05.a)Build an Artificial Neural Network by implementing the Backpropagation algorithm using the Churn\_Modelling dataset and evaluate the performance of the model**

import numpy as np

import pandas as pd

import tensorflow as tf

tf.\_\_version\_\_

data=pd.read\_csv("https://raw.githubusercontent.com/amppmann/ML-Lab-SEE/master/Folder\_05/Churn\_Modelling.csv")

print("Dataset : ",data.head())

# Preprocessing of the data

x=data.iloc[:,3:-1].values

y=data.iloc[:,-1].values

print("x : ",x)

print("y : ",y)

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

x[:,2]=le.fit\_transform(x[:,2])

print("Label Encoded x : ",x)

print("\n")

# Using OneHotEncoder to encode 'geography' column

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[1])],remainder='passthrough')

x=np.array(ct.fit\_transform(x))

print("One Hot Encoded x : ",x)

print("\n")

# Splitting the dataset into training and testing set

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)

# Feature scaling

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

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

x\_test=sc.transform(x\_test)

# Building the ANN Model

ann=tf.keras.models.Sequential()

# Adding input layer and first hidden layer

ann.add(tf.keras.layers.Dense(units=6,activation='relu'))

# Adding the second hidden layer

ann.add(tf.keras.layers.Dense(units=6,activation='relu'))

# Adding output layer

ann.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))

ann.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

ann.fit(x\_train,y\_train,batch\_size=32,epochs=10)

y\_predict=ann.predict(x\_test)

y\_predict=y\_predict>0.5

con=np.concatenate((y\_predict.reshape(len(y\_predict),1),y\_test.reshape(len(y\_test),1)),1)

print(con)

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

cm=confusion\_matrix(y\_test,y\_predict)

print("Confusion Matrix : ",cm)

print("Accuracy score of the model : ",accuracy\_score(y\_test,y\_predict))

**05.b)Demonstrate the application of Simple Linear regression on the housing dataset.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

data=pd.read\_csv('HousingDataset.csv')

print("Dataset : ",data.head())

x=data.iloc[:,[0]].values

y=data.iloc[:,[1]].values

print("x : ",x)

print("y : ",y)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

model=LinearRegression()

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

y\_predict=model.predict(x\_test)

print("Predicted value : ",y\_predict)

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

plt.plot(x\_test,y\_predict,color='blue')

plt.title('Price Vs Area')

plt.xlabel('Price')

plt.ylabel('Area')

plt.show()

**06.a)Write a program to implement the naïve Bayesian classifier for the Social\_Network\_Ads dataset. Compute the accuracy of the classifier and visualize the results.**

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

from matplotlib.colors import ListedColormap

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import StandardScaler

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('Social\_Network\_Ads.csv')

print("Dataset : ", data.head())

x = data.iloc[:, [0, 1]].values

y = data.iloc[:, 2].values

print("x : ", x)

print("y : ", y)

# Splitting the dataset into training and testing sets

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

x, y, test\_size=0.25, random\_state=0)

# Feature Scaling

sc = StandardScaler()

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

x\_test = sc.transform(x\_test)

# Fitting the Naive Bayes to the training set

classifier = GaussianNB()

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

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

print("Predicted value : ", y\_predict)

# Evaluation

cm = confusion\_matrix(y\_test, y\_predict)

asr = accuracy\_score(y\_test, y\_predict)

print("Confusion Matrix")

print(cm)

print("Accuracy Score of the model")

print(asr)

# FOR TRAINING SET

x\_set, y\_set = x\_train, y\_train

x1, x2 = np.meshgrid(np.arange(start=x\_set[:, 0].min()-1, stop=x\_set[:, 0].max(

)+1, step=0.01), np.arange(start=x\_set[:, 1].min()-1, stop=x\_set[:, 1].max()+1, step=0.01))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(

x1.shape), alpha=0.75, cmap=ListedColormap(('red', 'green')))

plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],

c=ListedColormap(('red', 'green'))(i), label=j)

plt.title('Naive Bayes (Training Set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# FOR TESTING SET

x\_set, y\_set = x\_test, y\_test

x1, x2 = np.meshgrid(np.arange(start=x\_set[:, 0].min()-1, stop=x\_set[:, 0].max(

)+1, step=0.01), np.arange(start=x\_set[:, 1].min()-1, stop=x\_set[:, 1].max()+1, step=0.01))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(

x1.shape), alpha=0.75, cmap=ListedColormap(('red', 'green')))

plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],

c=ListedColormap(('red', 'green'))(i), label=j)

plt.title('Naive Bayes (Testing Set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**06.b)Demonstrate the application of Simple Linear regression to predict the stock market prices of any organization.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

data=pd.read\_csv('Google\_stock.csv')

print("Dataset : ",data.head())

x=data.iloc[:,0].str.replace('/','').str.replace('-','').astype('int').values

y=data.iloc[:,-1].str.replace(',','').astype('int').values

x=x.reshape(len(x),1)

y=y.reshape(len(y),1)

print("x : ",x)

print("y : ",y)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

model=LinearRegression()

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

y\_predict=model.predict(x\_test)

print("Predicted value : ",y\_predict)

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

plt.plot(x\_test,y\_predict,color='blue')

plt.title('Date Vs Volume')

plt.xlabel('Date')

plt.ylabel('Volume (K)')

plt.show()

**07.a)Apply Hierarchical clustering on the Mall\_Customers dataset and visualize the clusters and plot the dendrograms.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

print("Dataset : ",data)

x=data.iloc[:,[3,4]].values

# Using the dendrogram to find the optimal number of clusters

import scipy.cluster.hierarchy as sch

dendrogram=sch.dendrogram(sch.linkage(x,method='ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean Distances')

plt.show()

# Training the Hierarchical clustering using training dataset

from sklearn.cluster import AgglomerativeClustering

hc=AgglomerativeClustering(n\_clusters=7,affinity='euclidean',linkage='ward')

y\_hc=hc.fit\_predict(x)

print("Predicted value : ",y\_hc)

plt.scatter(x[y\_hc == 0,0],x[y\_hc == 0,1],s=100,c='red',label='Cluster 1')

plt.scatter(x[y\_hc == 1,0],x[y\_hc == 1,1],s=100,c='blue',label='Cluster 2')

plt.scatter(x[y\_hc == 2,0],x[y\_hc == 2,1],s=100,c='green',label='Cluster 3')

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

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

plt.title('Clusters of Customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**07.b)Demonstrate the use of the Support Vector Machine algorithm for a regression problem on the Position\_Salaries dataset and evaluate the performance of the model.**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data=pd.read\_csv('Position\_Salaries.csv')

print("Dataset : ",data)

x=data.iloc[:,1:-1].values

y=data.iloc[:,-1].values

y=y.reshape(len(y),1)

print("x : ",x)

print("y : ",y)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc\_x=StandardScaler()

sc\_y=StandardScaler()

x=sc\_x.fit\_transform(x)

y=sc\_y.fit\_transform(y)

from sklearn.svm import SVR

regressor=SVR(kernel='rbf')

regressor.fit(x,y)

y\_predict=sc\_y.inverse\_transform(regressor.predict(sc\_x.transform([[6.5]])).reshape(-1,1) )

print("Predicted value : ",y\_predict)

plt.scatter(sc\_x.inverse\_transform(x),sc\_y.inverse\_transform(y),color='red')

plt.plot(sc\_x.inverse\_transform(x),sc\_y.inverse\_transform(regressor.predict(x).reshape(-1,1)),color='blue')

plt.title('Truth or Bluff (SVR)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()