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,rando
m_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).reshap
e(-1,1),color='red')

plt.plot(sc_x.inverse_transform(x),sc_y.inverse_transform(regressor.pr
edict(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,rando
m_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-SE
E/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])],re
mainder='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,rando
m_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=['accu
racy'])

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,rando
m_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,rando
m_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.pr
edict(x).reshape(-1,1)),color='blue')
```

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

plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```