**1.**import pandas as pd import numpy as np;; data=pd.read\_csv("tennis.csv") print(data)

d=np.array(data)[:,:-1] print("the attributes are: ",d) ;; target= np.array(data)[:,-1] print("the target is :",target) ;; def train(d, t):

specific\_hypothesis = None

for i, val in enumerate(t):

if val == "yes":

specific\_hypothesis = d[i].copy()

break

if specific\_hypothesis is None:

return "No positive example found in the target"

for i, val in enumerate(d):

if t[i] == "yes":

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

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

specific\_hypothesis[x] = '?'

return specific\_hypothesis ;;print("the final hypothesis is :",train(d,target))

--------------------------------------------------------------------------------------------------------------

1.import pandas as pd import numpy as np ;; data=pd.read\_csv("tennis.csv") print(data);;

d=np.array(data)[:,:-1] print("the attributes are: ",d) ;; target= np.array(data)[:,-1] print("the target is :",target) ;;

def learn(d, target):

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

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

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

print(general\_h)

for i, h in enumerate(d):

if target[i] == "yes":

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

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

specific\_h[x] ='?'

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

if target[i] == "no":

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(" steps of Candidate Elimination Algorithm",i+1)

print(specific\_h)

print(general\_h)

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

for i in indices:

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

return specific\_h, general\_h ;s\_final, g\_final = learn(d, target); print("Final Specific\_h:", s\_final, sep="\n"); print("Final General\_h:", g\_final, sep="\n")

**25.**

import pandas as pd ;; data=pd.read\_csv('salary\_data.csv') ;; data.head();;

import sklearn; from sklearn.model\_selection import train\_test\_split ; train , test=train\_test\_split(data,test\_size=0.3) ;; x\_train=train.drop('Salary',axis=1) y\_train=train['Salary'] ;; x\_test=test.drop('Salary',axis=1); y\_test=test['Salary'] ;;

x\_test.head() ;; y\_test.head() ;; from sklearn.linear\_model import LinearRegression ; from sklearn.metrics import mean\_squared\_error ; from math import sqrt; import matplotlib.pyplot as plt ;; model= LinearRegression() ;; model.fit(x\_train,y\_train) ;;

pred=model.predict(x\_test) pred ;; error=mean\_squared\_error(y\_test,pred); error ;;

print("Coefficient (slope):", model.coef\_[0]); print("Intercept:", model.intercept\_);;

plt.scatter(x\_test, y\_test, color='green', label='Testing data')

plt.plot(x\_train, model.predict(x\_train), color='red', label='Regression line')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.title('Salary vs. Years of Experience (Testing data)')

plt.legend()

plt.show();;

**36.**

import pandas as pd import numpy as np;;

data=pd.read\_csv("USA\_housing.csv");;

data.head() ;; x = data.drop(['Price', 'Address'],axis=1) ;; y=data['Price'] ;;

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.4) ;;

from sklearn.linear\_model import LinearRegression ; lm=LinearRegression() ;;

lm.fit(X\_train,Y\_train) ;; pred=lm.predict(X\_test) ;; pred ;; from sklearn import metrics;;

print("MSE : ",metrics.mean\_squared\_error(Y\_test,pred)) ;;

coefficients = lm.coef\_

print("Coefficients:")

for feature, coef in zip(x.columns, coefficients):

print(f"{feature}: {coef}")

**47**.import numpy as np ; import pandas as pd; import warnings warnings.filterwarnings("ignore", category=FutureWarning, module="sklearn") warnings.filterwarnings("ignore", category=UserWarning, module="sklearn") ;;

data=pd.read\_csv("tennis.csv") ; data.head() ;;

outlook=data["outlook"].str.get\_dummies(" ") ;temp = data["temp"].str.get\_dummies(" ") humidity =data["humidity"].str.get\_dummies(" ") play = data["play"].str.get\_dummies(" ") windy = pd.get\_dummies(data['windy'], drop\_first=True) ;;

data.drop(["outlook",'temp',"humidity","windy","play"],axis=1, inplace=True);;

data=pd.concat([outlook,temp,humidity,windy,play] , axis=1) ; data ;;

from sklearn.model\_selection import train\_test\_split ; x=data.drop(['yes','no'] , axis=1) ; y=data['no'] ; X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2,random\_state=42) ; X\_train.columns = X\_train.columns.astype(str) ;X\_test.columns = X\_test.columns.astype(str) ;; from sklearn.tree import DecisionTreeClassifier ; dtc = DecisionTreeClassifier(criterion='entropy'); dtc.fit(X\_train, y\_train) ;; pred=dtc.predict(X\_test) ;;

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

print("Accuracy :",accuracy\_score(y\_test,pred)) ;print("classification\_report : ") ;print(classification\_report(y\_test,pred)) ; print("confusion\_matrix : ") ;

print(confusion\_matrix(y\_test,pred));;

--------------------------------------------------------------------------------------------------------------

**69.**

import numpy as np; import pandas as pd ;; data=pd.read\_csv("Iris.csv") data.head() ;;

X=data.iloc[:,1:5] y=data.iloc[:,5:6] y.head() ;; from sklearn.preprocessing import LabelEncoder ; pr= LabelEncoder(); y=y.apply(pr.fit\_transform) ;; y.head() ;;

from sklearn.model\_selection import train\_test\_split ;; x\_train,x\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=42)

from sklearn.preprocessing import StandardScaler ; scaler=StandardScaler() ; scaler.fit(x\_train) ;; x\_train=scaler.transform(x\_train); x\_test=scaler.transform(x\_test) ;;

from sklearn.neural\_network import MLPClassifier ;;

model= MLPClassifier(hidden\_layer\_sizes=(10,10,10),max\_iter=1000) ;; model.fit(x\_train,y\_train.values.ravel()) ;; pred=model.predict(x\_test) ;; pred ;;

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

print("accuracy :",accuracy\_score(y\_test,pred))

print("classification\_report") ;print(classification\_report(y\_test,pred))

print("confusion\_matrix") ; print(confusion\_matrix(y\_test,pred))

**58**.import sklearn.datasets import numpy as np ;; cancer=sklearn.datasets.load\_breast\_cancer() ;; x=cancer.data y=cancer.target print(x.shape,y.shape) ;; import pandas as pd ; data=pd.DataFrame(cancer.data,columns=cancer.feature\_names) ;; data['class']=cancer.target ; data.head() ;; print(data['class'].value\_counts()) ;; print(cancer.target\_names) ;; data.groupby('class').mean() ;;

from sklearn.model\_selection import train\_test\_split ; x=data.drop('class',axis=1) ;y=data['class'] ;; type(x) ;; x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.1) ; x\_train=x\_train.values ;x\_test=x\_test.values ;;

from sklearn.metrics import accuracy\_score ; class Perceptron:

def \_\_init\_\_(self):

self.w=None; self.b=None

def model(self,X):

return 1 if(np.dot(self.w,X) >= self.b) else 0

def predict(self,X): Y=[]

for x in X:

result=self.model(x)

Y.append(result)

return np.array(Y)

def fit(self,X,Y,epochs=1,lr=1):

self.w=np.ones(X.shape[1]); self.b=0; accuracy={}; max\_accuracy=0

wt\_matrix=[]

for i in range(epochs): for x,y in zip(X,Y):; y\_pred=self.model(x); if y==1 and y\_pred==0: ;self.w=self.w+lr\*x; self.b=self.b-lr\*1; elif y==0 and y\_pred==1: self.w=self.w-lr\*x ; self.b=self.b+lr\*1; wt\_matrix.append(self.w) ; accuracy[i]=accuracy\_score(self.predict(X),Y);

if (accuracy[i] >=max\_accuracy):

max\_accuracy=accuracy[i]; chkptw=self.w; chkptb=self.b ;

self.w=chkptw self.b=chkptb print(max\_accuracy) ;

import matplotlib.pyplot as plt plt.plot(accuracy.values()) plt.ylim([0,1]) plt.show() return np.array(wt\_matrix);; percept=Perceptron() ;; wt\_matrix=percept.fit(x\_train,y\_train,10000,0.5) ;; y\_predict=percept.predict(x\_test);;

from sklearn.metrics import classification\_report; print(classification\_report(y\_test,y\_predict)) ;; from sklearn.metrics import accuracy\_score;; accuracy = accuracy\_score(y\_test, y\_predict);

print("Accuracy:", accuracy) ;; from sklearn.metrics import confusion\_matrix; conf=confusion\_matrix(y\_test, y\_predict); print("Confusion\_Matrix : "); print(conf)

**710.**import pandas as pd; import numpy as np;; data=pd.read\_csv("salary\_data.csv");;

data.head();; x =data.drop('Salary', axis=1); y= data['Salary'];; from sklearn.model\_selection import train\_test\_split; x\_train , x\_test,y\_train,y\_test =train\_test\_split(x,y, test\_size = 0.3) ;; from sklearn.neighbors import KNeighborsRegressor ; from sklearn.metrics import mean\_squared\_error ;;

model = KNeighborsRegressor(n\_neighbors=3); model.fit(x\_train, y\_train);

pred=model.predict(x\_test); error = mean\_squared\_error(y\_test,pred) ;; pred;;

error ;; ----------------------------------------------------------------------------------------------

**811**. import pandas as pd;import numpy as np;from sklearn.datasets import load\_iris;;

data=load\_iris();; X=data.data ; y=data.target ;y ;;

from sklearn.model\_selection import train\_test\_split; X\_train,X\_test ,y\_train,y\_test = train\_test\_split(X,y, test\_size = 0.3,random\_state=42);; from sklearn.neighbors import KNeighborsClassifier ; model=KNeighborsClassifier(n\_neighbors=7) ;; model.fit(X\_train,y\_train);; pred=model.predict(X\_test);; pred ;; from sklearn.metrics import accuracy\_score ; print("Accuracy :",accuracy\_score(y\_test,pred)) ;;

from sklearn.metrics import confusion\_matrix; print("Confusion matrix : "); print(confusion\_matrix(y\_test,pred));; from sklearn.metrics import classification\_report; print("Classifiaction report : "); print(classification\_report(y\_test,pred));;

**----------------------------------------------------------------------------------------------**

**912**.import numpy as np ;import matplotlib.pyplot as plt ;import seaborn as sns;sns.set();;

from sklearn.datasets import fetch\_20newsgroups; data = fetch\_20newsgroups() ;data.target\_names;; categories = ['talk.religion.misc', 'soc.religion.christian', 'sci.space', 'comp.graphics'] ;train = fetch\_20newsgroups(subset='train', categories=categories) ;test = fetch\_20newsgroups(subset ='test', categories=categories);; print(train.data[5]);;

from sklearn.feature\_extraction.text import TfidfVectorizer; from sklearn.naive\_bayes import MultinomialNB; from sklearn.pipeline import make\_pipeline; model=make\_pipeline(TfidfVectorizer(), MultinomialNB());;

model.fit(train.data, train.target); labels = model.predict(test.data);;

from sklearn.metrics import confusion\_matrix;mat = confusion\_matrix(test.target, labels);sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,

xticklabels=train.target\_names, yticklabels=train.target\_names);plt.xlabel('true label')

;plt.ylabel('predicted label');; mat ;; def predict\_category(s, train=train, model=model): ;pred = model.predict([s]) ;return train.target\_names[pred[0]];; predict\_category('sending a payload to the ISS') ;; predict\_category('discussing islam vs atheism');; predict\_category('determining the screen resolution');;

from sklearn.metrics import classification\_report , confusion\_matrix , accuracy\_score;

print("classification\_report : ");print(classification\_report(test.target,labels));

print("confusion\_matrix : ");print(confusion\_matrix(test.target,labels));

print("Accuracy : ",accuracy\_score(test.target,labels));;

**1013**. import pandas as pd;from sklearn.cluster import KMeans;from sklearn.preprocessing import MinMaxScaler;from matplotlib import pyplot as plt;import warnings;

warnings.filterwarnings("ignore", category=FutureWarning, module="sklearn");

warnings.filterwarnings("ignore", category=UserWarning, module="sklearn");;

df=pd.read\_csv("income.csv"); df.head();; plt.scatter(df.Age,df['Income($)']) ;plt.xlabel('Age'); plt.ylabel('Income($)');; km=KMeans(n\_clusters=3) ;y\_predicted=km.fit\_predict(df[['Age','Income($)']]) ;y\_predicted;; df['cluster']=y\_predicted; df.head();;

df1 = df[df.cluster==0]; df2 = df[df.cluster==1]; df3 = df[df.cluster==2]; plt.scatter(df1.Age,df1['Income($)'],color='green');plt.scatter(df2.Age,df2['Income($)'],color='red');plt.scatter(df3.Age,df3['Income($)'],color='black'); plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],color='purple',marker='\*',

plt.xlabel('Age');plt.ylabel('Income ($)');plt.legend();; scaler=MinMaxScaler() ; scaler.fit(df[['Income($)']]); df['Income($)']=scaler.transform(df[['Income($)']]); scaler.fit(df[["Age"]]); df['Age']=scaler.transform(df[['Age']]);; df.head() ;; plt.scatter(df.Age,df['Income($)']) ;; km=KMeans(n\_clusters=3) ;y\_predicted=km.fit\_predict(df[['Age','Income($)']]); y\_predicted;; df['cluster']=y\_predicted;; df.head() ;; km.cluster\_centers\_;; df1 = df[df.cluster==0]

df2 = df[df.cluster==1];

df3 =df[df.cluster==2];plt.scatter(df1.Age,df1['Income($)'],color='green');

plt.scatter(df2.Age,df2['Income($)'],color='red'); plt.scatter(df3.Age,df3['Income($)'],color='black'); plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],color='purple',marker='\*',

plt.legend() ;; sse = [] k\_rng = range(1,10) ;for k in k\_rng: ; km = KMeans(n\_clusters=k) ;km.fit(df[['Age','Income($)']]); sse.append(km.inertia\_) ;; sse ;; plt.xlabel('K') ; plt.ylabel('sum of squared error') ; plt.plot(k\_rng,sse) ;;

**1114**. import numpy as np; import matplotlib.pyplot as plt; import pandas as pd;;

data=pd.read\_csv('Mall\_Customers.csv') ; data.head() ;; newdata=data.iloc[:,[3,4]].values;; import scipy.cluster.hierarchy as sch ;

dendrogram = sch.dendrogram(sch.linkage(newdata, method = 'ward')) ; plt.title('

Dendrogram') ;plt.xlabel('Customers'); plt.ylabel('Euclidean distances') ; plt.show();; import warnings; warnings.filterwarnings("ignore", category=FutureWarning, module="sklearn") ; warnings.filterwarnings("ignore", category=UserWarning, module="sklearn") ;;

from sklearn.cluster import AgglomerativeClustering; Agg\_hc=AgglomerativeClustering(n\_clusters=5,affinity='euclidean',linkage='ward');

y\_hc=Agg\_hc.fit\_predict(newdata);;

plt.scatter(newdata[y\_hc == 0, 0], newdata[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster1');plt.scatter(newdata[y\_hc == 1, 0], newdata[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster2'); plt.scatter(newdata[y\_hc == 2, 0], newdata[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster3'); plt.scatter(newdata[y\_hc == 3, 0], newdata[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster4'); plt.scatter(newdata[y\_hc == 4, 0], newdata[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster5'); plt.title('Clusters of customers');

plt.xlabel('Annual Income (k$)'); plt.ylabel('Spending Score (1-100)') ; plt.legend() ; plt.show()