```
1.def finds(data): hypo=['$']*(len(data[0])-1); [hypo:=row[:-1] if hypo[0]=='$' else [hypo[i] if hypo[i]==row[i] else '?' for i in range(len(hypo))] for row in data if row[-1]=='Yes']; return hypo
```

```
1.def candidate_elimination(data): s=["$"]*(len(data[0])-1); G=[["?"]*(len(data[0])-1)]; [(([s.__setitem__(i,attributes[i]) if s[i]=="$" else s.__setitem__(i,"?") if s[i]!=attributes[i] else None for i in range(len(s))], G.__setitem__(slice(None), [g for g in G if all(g[i]=="?" or g[i]==s[i] for i in range(len(s)))])) if label=="Yes" else G.__setitem__(slice(None), [g[:i]+[attributes[i]]+g[i+1:] for g in G for i in range(len(g)) if g[i]=="?"]) for attributes,label in [(row[:-1],row[-1]) for row in data]]; return s,G

S_final,G_final=candidate_elimination(data); print("Final specific hypothesis : ",S_final); print("Final General hypothesis : ",G_final)
```

```
2.import numpy as np; import pandas as pd; import matplotlib.pyplot as plt; import seaborn as sns; from sklearn.model_selection import train_test_split; from sklearn.linear_model import LinearRegression; from sklearn.metrics import mean_squared_error,r2_score

df=pd.read_csv('Salary_dataset.csv')  display(df.head()) df.info() print(df.describe())

plt.scatter(x=df['YearsExperience'],y=df['Salary']); plt.xlabel("Years of experience"); plt.ylabel("Salary"); plt.title("Salary vs Experience"); plt.show()

x=df[['YearsExperience']]; y=df['Salary']; model=LinearRegression(); x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

model.fit(x_train,y_train) print(model)

y_pred=model.predict(x_test)

mse=mean_squared_error(y_test,y_pred); r2=r2_score(y_test,y_pred); print(f"Mean squared error : {mse}"); print(f"R-squared error : {r2}")

plt.scatter(x_test,y_test,color="blue",label="Actual"); plt.plot(x_test,y_pred,color="red",linewidth=2,label="Predicted"); plt.xlabel("Year of experience"); plt.ylabel("Salary"); plt.title("Actual vs predicted value"); plt.legend(); plt.show()

print(f"Intercept : {model.intercept_}") print(f"Coeffecient : {model.coef_[0]}")
```

```
3.import pandas as pd; import matplotlib.pyplot as plt; from sklearn.model_selection import train_test_split; from sklearn.linear_model import LinearRegression; from sklearn.metrics import mean_squared_error,r2_score

df=pd.read_csv("ML_datasets/USA_Housing.csv") df.info()

x=df[["Avg. Area Income", "Avg. Area House Age", "Avg. Area Number of Rooms", "Avg. Area Numbery=df["Price"]
```

```
x_train, x_test, y_train, y_test=train_test_split(x,y, random_state=42, test_size=0.2)
model=LinearRegression() model.fit(x_train,y_train)

y_pred=model.predict(x_test)

print("Mean squared error ",mean_squared_error(y_pred, y_test)) print("R-squared value
",r2_score(y_test, y_pred))

print("Intercept : ",model.intercept_) print("Co-efficient : ",model.coef_)

plt.scatter(y_test,y_pred) plt.title("Actual vs Predicted") plt.show()
```

```
4.import pandas as pd; from sklearn.model_selection import train_test_split; from
sklearn.tree import DecisionTreeClassifier; from sklearn.metrics import
confusion_matrix,accuracy_score

file=pd.read_csv("play_tennis.csv") print(file.head())

outlook=file["outlook"].str.get_dummies(" "); temp=file["temp"].str.get_dummies(" ");
humid=file["humidity"].str.get_dummies(" "); wind=file["wind"].str.get_dummies(" ");
play=file["play"].str.get_dummies(" "); print(outlook)

df=pd.concat([outlook, temp, humid, wind, play], axis=1)

x=df.drop(["Yes","No"],axis=1)

y=df[“ Yes” ]

x_train, x_test, y_train, y_test=train_test_split(x,y, random_state=42,
test_size=0.3,stratify=y)

DT=DecisionTreeClassifier(criterion='entropy') DT.fit(x_train,y_train)
y_pred=DT.predict(x_test)

M=confusion_matrix(y_test,y_pred) print("Confusion Matrix : \n",M)
accuracy=accuracy_score(y_pred,y_test) print("Accuracy : ", accuracy)
```

```
5.from sklearn.datasets import load_breast_cancer import pandas as pd from
sklearn.model_selection import train_test_split from sklearn.linear_model import
Perceptron from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

dt=load_breast_cancer() y=dt.target x=dt.data

y=pd.DataFrame(y, columns=['class']) x=pd.DataFrame(x, columns=dt.feature_names)

x_train, x_test, y_train, y_test=train_test_split(x,y, random_state=42,test_size=0.2)

model=Perceptron(max_iter=1000, random_state=42,tol=1e-
3)model.fit(x_train,y_train.values.ravel()) print(model)
```

```
y_pred=model.predict(x_test) print("Accuracy : ",accuracy_score(y_pred,y_test))
print("Confusion matrix : \n",confusion_matrix(y_pred,y_test)) print("Classification report :
", classification_report(y_pred,y_test))

6.import pandas as pd

url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names=['sepal-length' ,'sepal-width' ,'petal-length','petal-width' ,'class'
irisdata=pd.read_csv(url,names=names) irisdata.head()

irisdata.info()

x=irisdata.iloc[:,0:4] print(x) y=irisdata.iloc[:,4:] y.head() print(y)

y["class"].unique()

from sklearn import preprocessing le=preprocessing. LabelEncoder()

y=y.apply(le.fit_transform)

from sklearn.model_selection import train_test_split print(x.shape,y.shape) x_train,
x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y) print("Dimension of x-train
:",x_train.shape,"x-test :",x_test.shape) print(y_test.head())

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
mlp=MLPClassifier(hidden_layer_sizes=(10,10,10),max_iter=2000)
mlp.fit(x_train,y_train.values.ravel())

predictions=mlp.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions)) print(classification_report(y_test, predictions))
print(y_train['class'].value_counts())

7.import pandas as pd from sklearn.neighbors import KNeighborsRegressor from
sklearn.metrics import mean_absolute_error, r2_score from sklearn.model_selection import
train_test_split

inc=pd.read_csv("salary_dataset.csv") print(inc.columns) x=inc[['YearsExperience'
]]y=inc['Salary']

xc, xv, yu, yi=train_test_split(x,y, test_size=0.3)

mod=KNeighborsRegressor() mod.fit(xc,yu.values.ravel()) y_pred=mod.predict(xv)
```

```
print("MSE:", mean_squared_error(yi, y_pred)) print("MAE:", mean_absolute_error(yi,
y_pred)) print("R2 Score:", r2_score(yi, y_pred))

8.import pandas as pd

url=https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data names=['sepal
length', 'sepal-width' ,'petal-length','petal-width' ,'class']
irisdata=pd.read_csv(url,names=names) irisdata.head()

irisdata.info()

x=irisdata.iloc[:,0:4]

y=irisdata.select_dtypes(include=[object]) y.head()

y["class"].unique()

from sklearn.model_selection import train_test_split print(x.shape, y. shape) x_train, x_test,
y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y) print("Dimension of x-train
:",x_train.shape,"x-test :",x_test.shape) print(y_test.head())

from sklearn.neighbors import KNeighborsClassifier knn=KNeighborsClassifier()
knn.fit(x_train,y_train.values.ravel())

predictions=knn.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,predictions)) print(classification_report(y_test,predictions))

9.from sklearn.cluster import KMeans import pandas as pd from sklearn.preprocessing
import MinMaxScaler from matplotlib import pyplot as plt %matplotlib inline import os
os.environ["OMP_NUM_THREADS"] = "1"

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,n_init=10) y_predicted=km.fit_predict(df[['Age','Income($)']])
print(y_predicted)

df['cluster']=y_predicted print(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($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='purple' , marker='*' ,
label plt.xlabel('Age') plt.ylabel('Income ($)') plt.legend() plt.show()
```

```
scaler=MinMaxScaler() scaler.fit(df[['Income($)']])
df['Income($)']=scaler.transform(df[['Income($)']]) scaler.fit(df[['Age']]) df['Age'
]=scaler.transform(df[['Age' ]]) print(df.head())

plt.scatter(df.Age,df['Income($)'])

km=KMeans(n_clusters=3) y_pred=km.fit_predict(df[['Age','Income($)']]) print(y_pred)

df['cluster' ]=y_pred 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($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[ : ,1], color='purple' , marker='*',
label plt.xlabel('Age') plt.ylabel('Income ($)') plt.legend()

see=[] k_rng=range(1,10) for k in k_rng: km=KMeans(n_clusters=k,n_init=10)
km.fit(df[['Age','Income($)']]) see.append(km.inertia_) print(see)

plt.xlabel('K') plt.ylabel('Sum of squared error') plt.plot(k_rng, see)

10.import numpy as np import matplotlib.pyplot as plt import pandas as pd

data=pd.read_csv("mall.csv") data.head()

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

import scipy.cluster.hierarchy as sch
dendogram=sch.dendrogram(sch.linkage(newd,method='ward')) plt.title('Dendrogram')
plt.xlabel('Customers') plt.ylabel('Euclidean distances') plt.show()

from sklearn.cluster import AgglomerativeClustering
agg_hc=AgglomerativeClustering(n_clusters=5,affinity='euclidean' , linkage='ward')
y_hc=agg_hc.fit_predict(newd)

plt.scatter(newd[y_hc == 0, 0], newd[y_hc == 0, 1], s=100, c='red', label='Cluster 1')
plt.scatter(newd[y_hc == 1, 0], newd[y_hc == 1, 1], s=100, c='blue', label='Cluster 2')
plt.scatter(newd[y_hc == 2, 0], newd[y_hc == 2, 1], s=100, c='green', label='Cluster 3')
plt.scatter(newd[y_hc == 3, 0], newd[y_hc == 3, 1], s=100, c='cyan', label='Cluster 4')
plt.scatter(newd[y_hc == 4, 0], newd[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()
```

```
11.import pandas as pd url="https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"names=['sepal_length' ,'sepal_width' , 'petal_length'
,'petal_width','species' ] df=pd.read_csv(url,names=names); print(df.head())

from sklearn.preprocessing import StandardScaler
features=['sepal_length','sepal_width','petal_length','petal_width'] x=df.loc[:,
features].values y=df.loc[:, ['species' ]].values x=StandardScaler().fit_transform(x) 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.decomposition import PCA pca=PCA(n_components=3)
principalComponents=pca.fit_transform(x)
principalDf=pd.DataFrame(data=principalComponents, columns=['principal component 1',
'principa.

finalDf=pd.concat([principalDf,df[['species' ]]],axis=1) finalDf.head()

import matplotlib.pyplot as plt fig=plt.figure(figsize=(8,8)) ax=fig.add_subplot(1,1,1)
ax.set_xlabel('principal component 1',fontsize=15) ax.set_ylabel('principal component
2',fontsize=15) ax.set_title('2 Component PCA', fontsize=20) targets=['Iris-setosa', 'Iris-
versicolor', 'Iris-virginica' ] colors=['r','g','b'] for target, color in zip(targets, colors):
ind=finalDf['species'] == target ax.scatter(finalDf.loc[ind, 'principal component 1']
,finalDf.loc[ind, 'principal component 2'] ,c=color ,S=50 ax.legend(targets) ax.grid()

pca. explained_variance_ratio_

from sklearn.linear_model import Perceptron
model=Perceptron(max_iter=1000,random_state=43,tol=1e-3)
model.fit(x_train,y_train.ravel()) pred=model.predict(x_test) print(pred)

from sklearn.metrics import accuracy_score print(accuracy_score(y_test,pred))

x1=finalDf.drop(['species'],axis=1) x1=StandardScaler().fit_transform(x1)
x1_train,x1_test,y_train,y_test=train_test_split(x1,y,test_size=0.3,random_state=42)

model=Perceptron(max_iter=1000, random_state=43, tol=1e-3)
model.fit(x1_train,y_train.ravel()) pred=model.predict(x1_test) print(pred)

accuracy_score(y_test,pred)
```