1. WRITE A PROGRAM TO LOAD AND EXPLORE THE DATASET OF CSV AND EXCEL FILES USING PANDAS

import pandas as pd

CSV\_file\_path ="Sampledata1.csv"

excel\_file\_path ="data.xlsx"

data\_csv = pd.read\_csv(CSV\_file\_path)

print("csv file data:")

print(data\_csv)

data\_excel = pd.read\_excel(excel\_file\_path)

print("\nExcel file data:")

print(data\_excel)

print("\ndata descriptions:")

print("csv data description:")

print(data\_csv.describe())

print("\nExcel data description:")

print(data\_excel.describe())

print("\ndata types in csv file")

print(data\_csv.dtypes)

print("\ndata types in excel file:")

print(data\_excel.dtypes)

OUTPUT:

csv file data:

NAME AGE SCORE

0 Srikanth 28 85

1 Snigdha 22 78

2 Mary 31 92

Excel file data:

NAME COURSE SEM

0 Rajesh BCA 1

1 Ramesh BCA 2

2 Swati BCOM 1

3 Florina BCOM 3

4 Pooja BBA 2

5 Raghu BBA 4

data descriptions:

csv data description:

AGE SCORE

count 3.000000 3.0

mean 27.000000 85.0

std 4.582576 7.0

min 22.000000 78.0

25% 25.000000 81.5

50% 28.000000 85.0

75% 29.500000 88.5

max 31.000000 92.0

Excel data description:

SEM

count 6.000000

mean 2.166667

std 1.169045

min 1.000000

25% 1.250000

50% 2.000000

75% 2.750000

max 4.000000

data types in csv file

NAME object

AGE int64

SCORE int64

dtype: object

data types in excel file:

NAME object

COURSE object

SEM int64

dtype: object

2. WRITE A PROGRAM TP VISUALIZE THE DATASET TO GAIN INSIGHTS USING MATPLOTLIB BY PLOTTING SCATTER PLOTS, BAR CHARTS

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('data2ndprg.csv')

plt.figure(figsize=(14,7))

plt.subplot(1,2,1)

plt.scatter(data['Study hours'],data['Exam score'],color='dodgerblue',edgecolor='k',alpha=0.9)

plt.title('Study hours vs. Exam scores')

plt.xlabel('Study hours')

plt.ylabel('Exam scores')

plt.grid(True)

bins = [0,2,4,6,8,10,12]

labels = ['0-2','2-4','4-6','6-8','8-10','10-12']

data['Study hour Range'] = pd.cut(data['Study hours'],bins=bins,labels=labels,right=False)

grouped\_data = data.groupby('Study hour Range')['Exam score'].mean()

plt.subplot(1,2,2)

grouped\_data.plot(kind='bar', color='salmon')

plt.title('Average Exam score by Study hour Range')

plt.xlabel('Study hour Range')

plt.ylabel('Average Exam score')

plt.xticks(rotation=50)

plt.tight\_layout()

plt.show()

OUTPUT:

3. WRITE A PROGRAM TO HANDLE MISSING DATA, ENCODE CATEGORICAL VARIABLES, AND PERFORM FEATURES SCALING

import pandas as pd

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder, StandardScaler

data = {

'Age':[25,30,None,28,35],

'Gender':['Female','Male','Male','Female','Male'],

'Income':[50000,60000,45000,None,70000]

}

df = pd.DataFrame(data)

imputer = SimpleImputer(strategy='mean')

df[['Age','Income']] = imputer.fit\_transform(df[['Age','Income']])

print("Data after handling missing values:");

print(df)

encoder=OneHotEncoder()

encoded\_data=encoder.fit\_transform(df[['Gender']]).toarray()

encoded\_df = pd.DataFrame(encoded\_data, columns=encoder.get\_feature\_names\_out(['Gender']))

print("\nData after categorical encoding:")

print(encoded\_df)

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(df[['Age','Income']])

scaled\_df = pd.DataFrame(scaled\_data, columns=['Scaled Age','Scaled Income'])

print("\nData after feature scaling:")

print(scaled\_df)

OUTPUT:

Data after handling missing values:

Age Gender Income

25.0 Female 50000.0

1 30.0 Male 60000.0

2 29.5 Male 45000.0

3 28.0 Female 56250.0

4 35.0 Male 70000.0

Data after categorical encoding:

Gender\_Female Gender\_Male

0 1.0 0.0

1 0.0 1.0

2 0.0 1.0

3 1.0 0.0

4 0.0 1.0

Data after feature scaling:

Scaled Age Scaled Income

0 -1.382164 -0.727778

1 0.153574 0.436667

2 0.000000 -1.310001

3 -0.460721 0.000000

4 1.689312 1.601112

4. WRITE A PROGRAM TP IMPLEMENT A K-NEAREST NEIGHBOURS(KNN) CLASSIFER USING SCIKITLEARN AND TRAIN THE CLASSIFER ON THE DATASET AND EVALUATE ITS PERFORMANCE

import numpy as np

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

from sklearn. neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

x=np.array([[80,75],[95,90],[60,50],[45,30],[30,40],[85,95],[70,60],[50,55],[40,45],[60,70]])

y=np.array([1,1,0,0,0,1,1,0,0,1])

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

knn= KNeighborsClassifier(n\_neighbors=3)

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

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

accuracy=accuracy\_score(y\_test, y\_pred)

print("Accuracy on the test set: {:.2f}".format(accuracy))

exam\_score1 = float(input("Enter Exam Score 1:"))

exam\_score2 = float(input("Enter Exam Score 2:"))

user\_input = np.array([[exam\_score1, exam\_score2]])

predicted\_outcome = knn.predict(user\_input)

if predicted\_outcome[0] == 1:

print("Based on the exam scores provided, the student is preducted to pass.")

else:

print("Based on the exam scores provided, the student is preducted to fail.")

OUTPUT:

Accuracy on the test set: 1.00

Enter Exam Score 1:90

Enter Exam Score 2:30

Based on the exam scores provided, the student is preducted to fail.

5. WRITE A PROGRAM TO IMPLEMENT A LINEAR REGRESSION MODEL FOR REGRESSION TASKS AND TRAIN THE MODEL ON A DATSET WITH CONTINUOUS TARGET VARIABLES

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

iris = load\_iris()

X = iris.data

y = iris.target

iris\_df = pd.DataFrame(data=X, columns=iris.feature\_names)

iris\_df ['target'] = y

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

linear\_regression = LinearRegression ()

linear\_regression.fit(X\_train, y\_train)

y\_pred = linear\_regression.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R-squared Score:", r2)

OUTPUT:

Mean Squared Error: 0.037113794407976866

R-squared Score: 0.9468960016420045

6. KNN WITHOUT USING BUILT-IN CLASSIFERS

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

X = data[['Age', 'EstimatedSalary']].values

y = data['Purchased'].values

def euclidean\_distance(x1, x2):

return np.sqrt(np.sum((x1 - x2) \*\* 2))

def knn\_predict(X, y, new\_data, k):

distances = []

for i in range(len(X)):

dist = euclidean\_distance(X[i], new\_data)

distances.append((dist, y[i]))

distances.sort(key=lambda x: x[0])

neighbors = distances[:k]

class\_votes = {0: 0, 1: 0}

for neighbor in neighbors:

class\_label = neighbor[1]

class\_votes[class\_label] += 1

return max(class\_votes, key=class\_votes.get)

new\_data = np.array([60, 90000])

k = 5

prediction = knn\_predict(X, y, new\_data, k)

if prediction == 0:

print("Predicted class: Not Purchased")

else:

print("Predicted class: Purchased")

OUTPUT:

Predicted class: Purchased

7. WRITE A PROGRAM TO IMPLEMENT A DECISION TREE CLASSIFIER USING SCIKIT-LEARN AND VISUALIZE THE DECISION TREE AND UNDERSTAND ITS SPLITS

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from matplotlib.pyplot import figure, show

import matplotlib.pyplot as plt

features=[[140,1],[130,0],[150,0],[170,1],[180,1],[100,0],[172,1]]

classifications=["play","don't play","don't play","play","play","don't play","play"]

import numpy as np

features=np.array(features)

classification=np.array(classifications)

clf=DecisionTreeClassifier()

clf=clf.fit(features, classifications)

predictions=clf.predict([[170,1]])

print("Decision Tree Classifier:")

print("predict Class Label for New Instance: [170,1]")

print("Class Label for New Instance is:", predictions[0])

plt.figure(figsize=(5,8))

plot\_tree(clf,feature\_names=["Temperature","Humidity"],class\_names=classifications, filled=True, rounded=True)

plt.show()

OUTPUT:

8. WRITE A PROGRAM TO IMPLEMENT K-MEANS CLUSTERING AND VISUALIZE CLUSTERS.

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import numpy as np

import pandas as pd

data = [[1,1],[1.5,1.8],[5,8],[8,8],[1,0.6],[9,11]]

print("Considered data for K-Means Clustering is:")

print(data)

print("Considered data as numpt- list is:")

data=np.array(data)

print(data)

print("Assumed K-Value is:")

k=3

print(k)

print("Kmeans object is given following values:")

kmeans=KMeans(n\_clusters=k, random\_state=42,n\_init=10)

print(kmeans)

kmeans.fit(data)

print("Integer Labels provided to each data point are:")

labels=kmeans.labels\_

print(labels)

print("Calculated centroid points are:")

centroids=kmeans.cluster\_centers\_

print(centroids)

plt.scatter(data[:,0],data[:,1], c=labels, cmap='viridis' )

plt.scatter(centroids[:,0], centroids[:,1], s=60, marker='x', c='red')

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("K-Means Clustering (k=" + ")")

plt.grid()

plt.show()

OUTPUT:

Considered data for K-Means Clustering is:

[[1, 1], [1.5, 1.8], [5, 8], [8, 8], [1, 0.6], [9, 11]]

Considered data as numpt- list is:

[[ 1. 1. ]

[ 1.5 1.8]

[ 5. 8. ]

[ 8. 8. ]

[ 1. 0.6]

[ 9. 11. ]]

Assumed K-Value is:

3

Kmeans object is given following values:

KMeans(n\_clusters=3, n\_init=10, random\_state=42)

Integer Labels provided to each data point are:

[1 1 2 2 1 0]

Calculated centroid points are:

[[ 9. 11. ]

[ 1.16666667 1.13333333]

[ 6.5 8. ]]

9. WRITE A PROGRAM TO IMPLIMENT K-MEANS CLUSTERING TO VISUALISE MORE CLUSTERING (N0 OF CLUSTERS AS 5)

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

X,y=make\_blobs(n\_samples=500,centers=4,cluster\_std=0.8,random\_state=42)

kmeans = KMeans(n\_clusters=5,random\_state=42)

kmeans.fit(X)

labels=kmeans.labels\_

plt.figure(figsize=(8,6))

plt.scatter(X[:,0],X[:,1],c=labels,cmap='viridis')

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],s=100,c='red',label='Centroids')

plt.title('K-Means Clustering')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

OUTPUT: