Machine Learning

Certified Journal

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

MASTER OF SCIENCE

(INFORMATION_TECHNOLOGY)

By

Anjali Rameshwar Nimje

DEPARTMENT OF INFORMATION TECHNOLOGY

KERALEEYA SAMAJAM (REGD.) DOMBIVLI'S

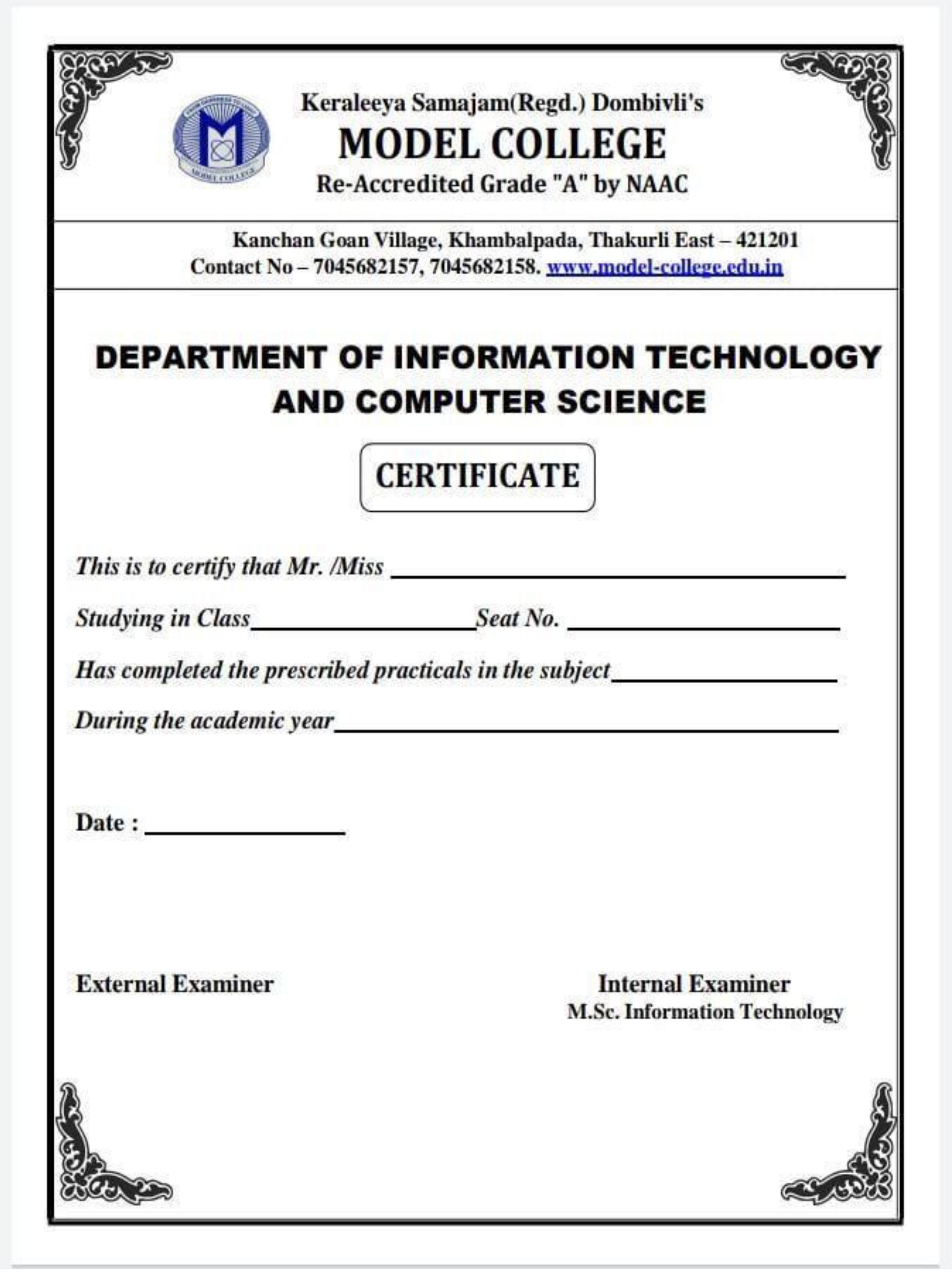
MODEL COLLEGE (AUTONOMOUS)

Re-Accredited 'A' Grade by NAAC

(Affiliated to University of Mumbai)

FOR THE YEAR

(2023-24)



Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss ______________________________

*Studying in Class*_______________ *Seat No.* ___________________

*Has completed the prescribed practicals in the subject*_______________

*During the academic year*___________________________________

Date : _____________

External Examiner

Internal Examiner
M.Sc. Information Technology

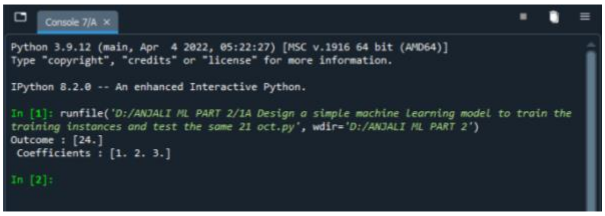
Sr No	Title	Date	Signature
1A	**Design A Simple Machine Learning Model To Train And Training Instances And Test The Same**	**21-10-2023**	
1B	**Implement And Demonstrate The Find-S Algorithm For Finding The Most Specific Hypothesis Based On A Given Set Of Training Data Samples. Read The Training Data From A Data.Csv File**	**21-10-23**	
2A	**Perform Data Loading, Feature Selection (Principal Component Analysis) And Feature Scoring And Ranking.**	**16-12-2023**	
2B	**For A Given Set Of Training Data Examples Stored In A .Csv File, Implement And Demonstrate The Candidate-Elimination Algorithm To Output A Description Of The Set Of All Hypotheses Consistent With The Training Examples**	**21-10-2023**	
3B	**Write A Program To Implement The Naïve Bayesian Classifier For A Sample Training Data Set Stored As A .Csv File. Compute The Accuracy Of The Classifier, Considering Few Test Data Sets.**	**4-11-2023**	
3B.1	**Write A Program To Implement A Decision Tree With Prediction, Test Score And Confusion Matrix.**	**4-11-2023**	
3B2	**Write A Program To Implement Random Forest With Prediction,Test Score And Confusion Matrix**.	**11-11-2023**	
4A	**For A Given Set Of Training Data Examples Stored In A .Csv File Implement Least Square Regression Algorithm.**	**11-11-2023**	
4B	**For A Given Set Of Training Data Examples Stored In A .Csv File Implement Logistic Regression Algorithm.**	**11-11-2023**	

5A	**Write A Program To Demonstrate The Working Of The Decision Tree Based Id3 Algorithm. Use An Appropriate Data Set For Building The Decision Tree And Apply This Knowledge To Classify A New Sample.**	**02-12-2023**	
5B	**Write A Program To Implement K-Nearest Neighbour Algorithm To Classify The Data Set.**	**11-11-2023**	
6A	**Implement The Different Distance Method Manhattan And Euclidean Distance**	**16-12-2023**	
6B	**Implement The Classification Model Using Clustering For The Following Techniques With K Means Clustering With Prediction, Test Score And Confusion Matrix.**	**02-12-2023**	
7A	**Implement The Classification Model Using Clustering For The Following Techniques With Hierarchical Clustering With Prediction, Test Score And Confusion Matrix**	**02-12-2023**	
9A	**Build An Artificial Neural Network By Implementing The Backpropagation Algorithm And Test The Same Using Appropriate Data Sets.**	**02-12-2023**	
9B	**Assuming the set of documents that need to be classified, using the naive bayesian classifier model to perform this task.**	**02-12-2023**	

PRACTICAL 1A : DESIGN A SIMPLE MACHINE LEARNING MODEL TO TRAIN AND TRAINING INSTANCES AND TEST THE SAME

```
from sklearn.linear_model import LinearRegression
from random import randint
TRAIN_SET_LIMIT = 1000
TRAIN_SET_COUNT = 100
TRAIN_INPUT = list()
TRAIN_OUTPUT = list()
for i in range(TRAIN_SET_COUNT):
a = randint(0, TRAIN_SET_LIMIT)
b = randint(0, TRAIN_SET_LIMIT)
c = randint(0, TRAIN_SET_LIMIT)
op = a+(2*b)+(3*c)
TRAIN_INPUT.append([a, b, c])
TRAIN_OUTPUT.append(op)
predictor = LinearRegression(n_jobs=-1)
predictor.fit(X=TRAIN_INPUT, y=TRAIN_OUTPUT)
X_TEST = [[3, 3, 5]]
outcome = predictor.predict(X=X_TEST)
coefficients = predictor.coef_
print('Outcome : {}\n Coefficients : {}'.format(outcome, coefficients))
```

Output



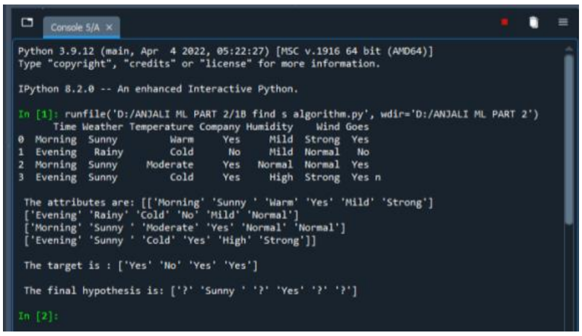
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/ANJALI ML PART 2/1A Design a simple machine learning model to train the
training instances and test the same 21 oct.py', wdir='D:/ANJALI ML PART 2')
Outcome : [24.]
 Coefficients : [1. 2. 3.]

In [2]:
```

PRACTICAL 1B. IMPLEMENT AND DEMONSTRATE THE FIND-S ALGORITHM FOR FINDING THE MOST SPECIFIC HYPOTHESIS BASED ON A GIVEN SET OF TRAINING DATA SAMPLES. READ THE TRAINING DATA FROM A DATA.CSV FILE

Code:

```
import numpy as np
data=pd.read_csv("D:\ANJALI ML PART 2\data.csv")
print(data,"n")
d=np.array(data)[:,:-1]
print("\n The attributes are:",d)
target=np.array(data)[:,-1]
print("\n The target is :",target)
def train(c,t):
for i,val in enumerate(t):
if val =="Yes":
specific_hypothesis=c[i].copy()
break
for i,val in enumerate(c):
if t[i]=="Yes":
for x in range (len(specific_hypothesis)):
if val[x]!= specific_hypothesis[x]:
specific_hypothesis[x]="?"
else:
pass
return specific_hypothesis
print("\n The final hypothesis is:",train(d,target))
```

Output



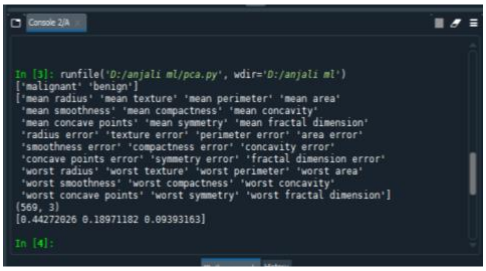
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/ANJALI ML PART 2/1B find s algorithm.py', wdir='D:/ANJALI ML PART 2')
      Time Weather Temperature Company Humidity    Wind Goes
0  Morning  Sunny         Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning  Sunny     Moderate     Yes   Normal  Normal  Yes
3  Evening  Sunny         Cold     Yes     High  Strong  Yes n

 The attributes are: [['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny ' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny ' 'Cold' 'Yes' 'High' 'Strong']]

 The target is : ['Yes' 'No' 'Yes' 'Yes']

 The final hypothesis is: ['?' 'Sunny ' '?' 'Yes' '?' '?']

In [2]:
```

PRACTICAL 2A: PERFORM DATA LOADING, FEATURE SELECTION (PRINCIPAL COMPONENT ANALYSIS) AND FEATURE SCORING AND RANKING.

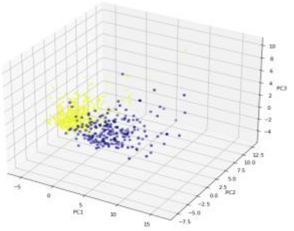
Code:

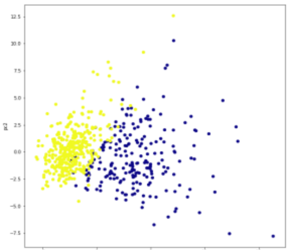
```
import pandas as pd
from sklearn.datasets import load_breast_cancer
data=load_breast_cancer()
data.keys()
print(data['target_names'])
print(data['feature_names'])
df1=pd.DataFrame(data['data'],columns=data['feature_names'])
scaling=StandardScaler()
scaling.fit(df1)
Scaled_data=scaling.transform(df1)
principal=PCA(n_components=3)
principal.fit(Scaled_data)
x=principal.transform(Scaled_data)
print(x.shape)
principal.components_
plt.figure(figsize=(10,10))
plt.scatter(x[:,0],x[:,1],c=data['target'],cmap='plasma')
plt.ylabel('pc1')
plt.ylabel('pc2')
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure(figsize=(10,10))
axis=fig.add_subplot(111,projection='3d')
axis.scatter(x[:,0],x[:,1],x[:,2],c=data['target'],cmap='plasma')
axis.set_xlabel("PC1",fontsize=10)
axis.set_ylabel("PC2",fontsize=10)
axis.set_zlabel("PC3",fontsize=10)
print(principal.explained_variance_ratio_)
```

Output :



```
In [3]: runfile('D:/anjali ml/pca.py', wdir='D:/anjali ml')
['malignant' 'benign']
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
(569, 3)
[0.44272026 0.18971182 0.09393163]

In [4]:
```



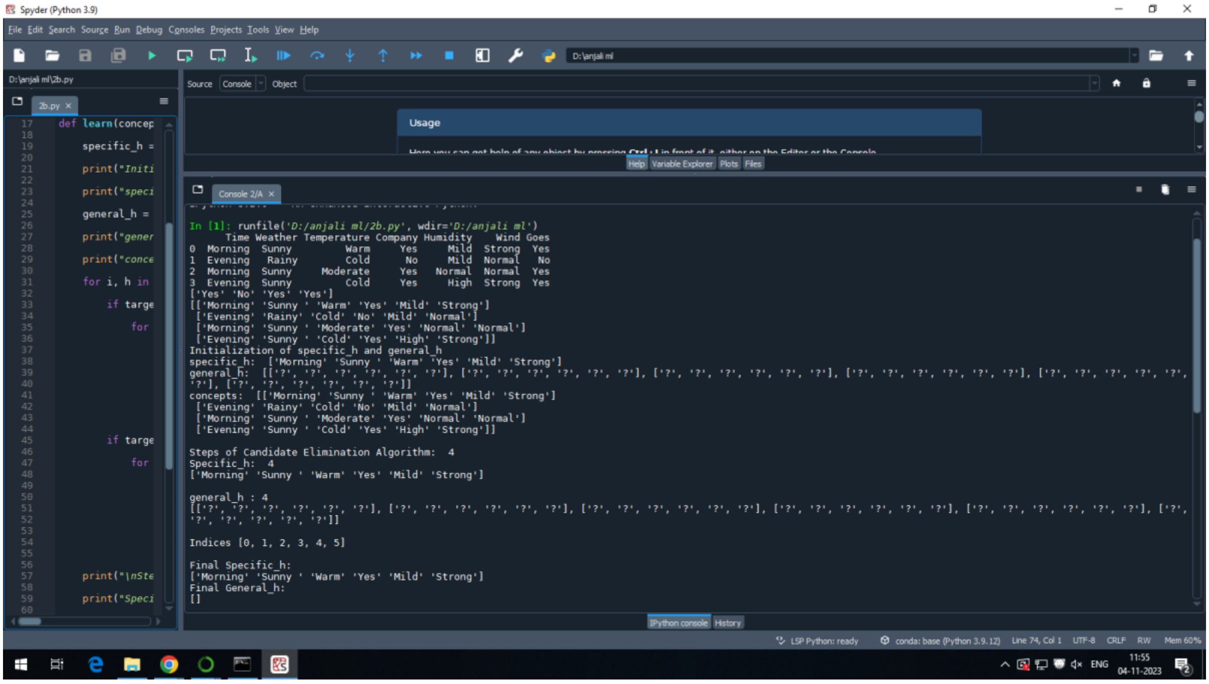


PRACTICAL 2B: FOR A GIVEN SET OF TRAINING DATA EXAMPLES STORED IN A .CSV FILE, IMPLEMENT AND DEMONSTRATE THE CANDIDATE-ELIMINATION ALGORITHM TO OUTPUT A DESCRIPTION OF THE SET OF ALL HYPOTHESES CONSISTENT WITH THE TRAINING EXAMPLES.

Code:

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('datac.csv'))
print(data)
concepts = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])
print(target)
print(concepts)
def learn(concepts, target):
specific_h = concepts[0].copy()
print("Initialization of specific_h and general_h")
print("specific_h: ",specific_h)
general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
print("general_h: ",general_h)
print("concepts: ",concepts)
for i, h in enumerate(concepts):
if target[i] == "yes":
for x in range(len(specific_h)):
#print("h[x]",h[x])
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("\nSteps of Candidate Elimination Algorithm: ",i+1)
print("Specific_h: ",i+1)
print(specific_h,"\n")
print("general_h :", i+1)
print(general_h)
indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
print("\nIndices",indices)
for i in indices:
general_h.remove(['?', '?', '?', '?', '?', '?'])
return specific_h, general_h
s_final,g_final = learn(concepts, target)
print("\nFinal Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

Output



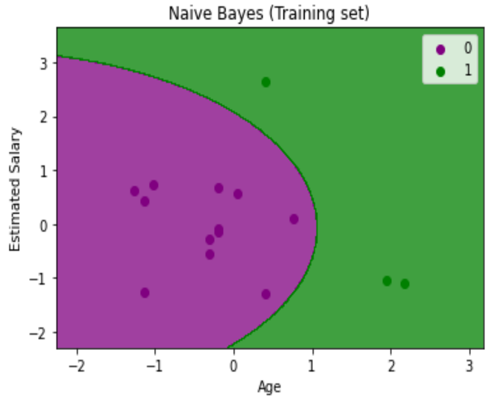
```
def learn(concep
    specific_h =
    print("Initi
    print("speci
    general_h =
    print("gener
    print("conce
    for i, h in
        if targe
            for


        if targe
            for



    print("\nSte
    print("Speci
```

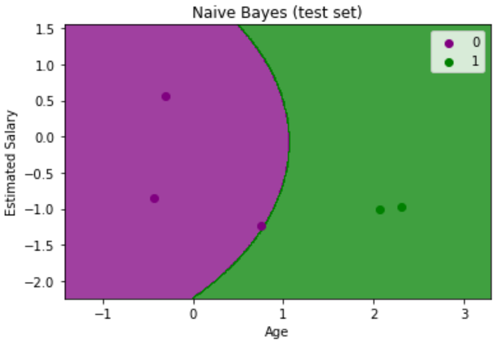
```
In [1]: runfile('D:/anjali ml/2b.py', wdir='D:/anjali ml')
     Time Weather Temperature Company Humidity    Wind Goes
0  Morning  Sunny        Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning  Sunny    Moderate     Yes   Normal  Normal  Yes
3  Evening  Sunny        Cold     Yes     High  Strong  Yes
['Yes' 'No' 'Yes' 'Yes']
[['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny ' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny ' 'Cold' 'Yes' 'High' 'Strong']]
Initialization of specific_h and general_h
specific_h:  ['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']
general_h:  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?']]
concepts:  [['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny ' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny ' 'Cold' 'Yes' 'High' 'Strong']]

Steps of Candidate Elimination Algorithm:  4
Specific_h:  4
['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']

general_h : 4
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?']]

Indices [0, 1, 2, 3, 4, 5]

Final Specific_h:
['Morning' 'Sunny ' 'Warm' 'Yes' 'Mild' 'Strong']
Final General_h:
[]
```

PRACTICAL 3B: WRITE A PROGRAM TO IMPLEMENT THE NAÏVE BAYESIAN CLASSIFIER FOR A SAMPLE TRAINING DATA SET STORED AS A .CSV FILE. COMPUTE THE ACCURACY OF THE CLASSIFIER, CONSIDERING FEW TEST DATA SETS.

```
#IMPORTING DATASET
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset = pd.read_csv('D:\\anjali ml\\user_data.csv.')
x = dataset.iloc[:, [1, 2]].values
y = dataset.iloc[:, 3].values
#SPLITTING THE DATASET INTO TESTINGB AND TRAINING DATASET
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, 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)
#FITTING NAIVE BAYES TO TRAINING SET
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1=accuracy_score(y_test,y_pred)*100
print('The Accuracy is', sc1)
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step =0.01),nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],c = ListedColormap(('purple',
'green'))(i), label = j)
mtp.title('Naive Bayes (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
#VISUALIZING THE TEST SET RESULTS
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],c = ListedColormap(('purple',
'green'))(i), label = j)
mtp.title('Naive Bayes (test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output



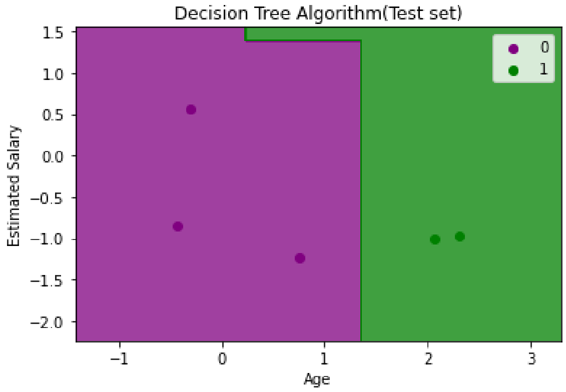


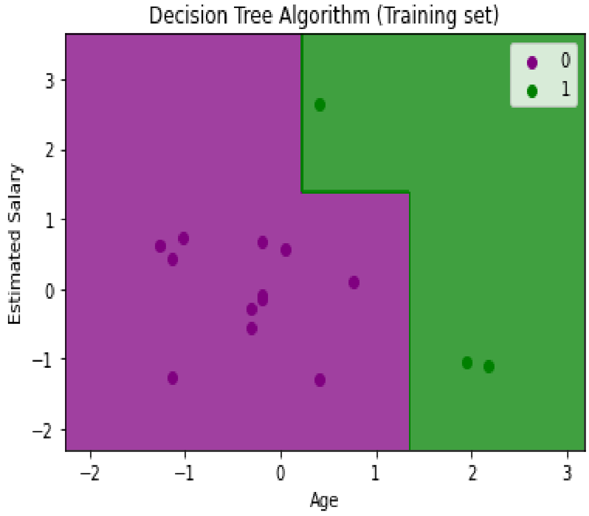
PRACTICAL 3.B.1 WRITE A PROGRAM TO IMPLEMENT A DECISION TREE WITH PREDICTION, TEST SCORE AND CONFUSION MATRIX.

Code:

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
#importing datasets
data_set= pd.read_csv('user_data.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, [1,2]].values
y= data_set.iloc[:, 3].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
#Fitting Decision Tree classifier to the training set
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)
DecisionTreeClassifier(class_weight=None,                              criterion='entropy',
max_depth=None,max_features=None,
max_leaf_nodes=None,min_impurity_decrease=0.0,min_samples_leaf=1,
min_samples_split=2,min_weight_fraction_leaf=0.0,random_state=0, splitter='best')
#Predicting the test set result
y_pred= classifier.predict(x_test)
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1 = accuracy_score(y_test, y_pred)*100
print("Accuracy is",sc1)
#Visulaizing the trianing set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i),label = j)
mtp.title('Decision Tree Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Decision Tree Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output



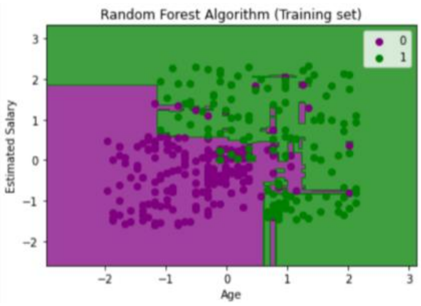


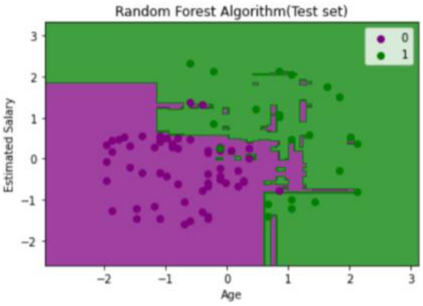
PRACTICAL NO.:3B. 2: WRITE A PROGRAM TO IMPLEMENT RANDOM FOREST WITH PREDICTION,TEST SCORE AND CONFUSION MATRIX.

Code:

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
#importing datasets
data_set= pd.read_csv('D:\Social_Network_Ads.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.20, random_state=0)
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
#Predicting the test set result
y_pred= classifier.predict(x_test)
print(y_pred)
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1 = accuracy_score(y_test, y_pred)*100
print('The Accuracy   is  ',sc1)
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
atplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:





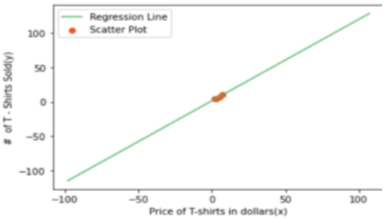
PRACTICAL NO.: 4. A: FOR A GIVEN SET OF TRAINING DATA EXAMPLES STORED IN A .CSV FILE IMPLEMENT LEAST SQUARE REGRESSION ALGORITHM.

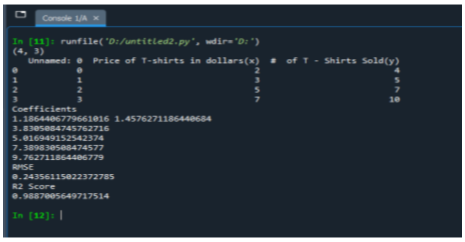
Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Reading Data
data = pd.read_csv('sale.csv')
print(data.shape)
(6, 2)
print(data.head())
# Coomputing X and Y
X = data['Price of T-shirts in dollars(x)'].values
Y = data['#  of T - Shirts Sold(y)'].values
# Mean X and Y
mean_x = np.mean(X)
mean_y = np.mean(Y)
# Total number of values
n = len(X)
# Using the formula to calculate 'm' and 'c'
numer = 0
denom = 0
for i in range(n):
numer += (X[i] - mean_x) * (Y[i] - mean_y)
denom += (X[i] - mean_x) ** 2
m = numer / denom
c = mean_y - (m * mean_x)
# Printing coefficients
print("Coefficients")
print(m, c)
# Plotting Values and Regression Line
max_x = np.max(X) + 100
min_x = np.min(X) - 100
# Calculating line values x and y
x = np.linspace(min_x, max_x, 1000)
y = c + m * x
# Ploting Line
plt.plot(x, y, color='#58b970', label='Regression Line')
# Ploting Scatter Points
plt.scatter(X, Y, c='#ef5423', label='Scatter Plot')
plt.xlabel('Price of T-shirts in dollars(x)')
plt.ylabel('#  of T - Shirts Sold(y)')
plt.legend()
plt.show()
# Calculating Root Mean Squares Error
rmse = 0
```

```
for i in range(n):
y_pred = c + m * X[i]
print(y_pred)
rmse += (Y[i] - y_pred) ** 2
rmse = np.sqrt(rmse/n)
print("RMSE")
print(rmse)
# Calculating R2 Score
ss_tot = 0
ss_res = 0
for i in range(n):
y_pred = c + m * X[i]
ss_tot += (Y[i] - mean_y) ** 2
ss_res += (Y[i] - y_pred) ** 2
r2 = 1 - (ss_res/ss_tot)
print("R2 Score")
print(r2)
```

Output





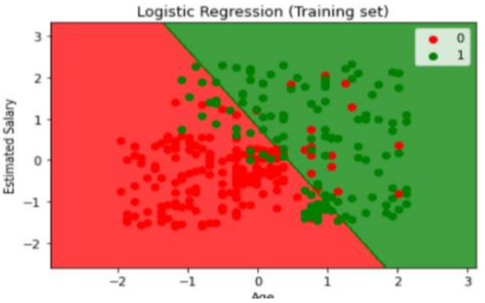
```
In [11]: runfile('D:/untitled2.py', wdir='D:')
(4, 3)
   Unnamed: 0  Price of T-shirts in dollars(x)  #  of T - Shirts Sold(y)
0           0                                2                        4
1           1                                3                        5
2           2                                5                        7
3           3                                7                       10
Coefficients
1.1864406779661016 1.4576271186440684
3.8305084745762716
5.016949152542374
7.389830508474577
9.762711864406779
RMSE
0.24356115022372785
R2 Score
0.9887005649717514

In [12]: |
```

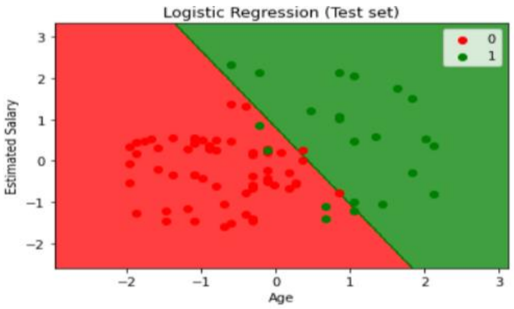
PRACTICAL NO.: 4. B: FOR A GIVEN SET OF TRAINING DATA EXAMPLES STORED IN A .CSV FILE IMPLEMENT LOGISTIC REGRESSION ALGORITHM.

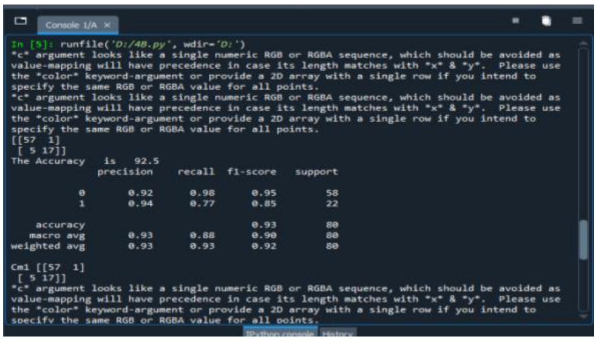
Code :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data_set= pd.read_csv('Social_Network_Ads.csv')
x= data_set.iloc[:,[2,3]].values
y= data_set.iloc[:, 4].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.20, random_state=0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1 = accuracy_score(y_test, y_pred)*100
print('The Accuracy   is  ',sc1)
from sklearn.metrics import classification_report
creport = classification_report(y_test, y_pred)
print(creport)
from sklearn.metrics import confusion_matrix
cm1= confusion_matrix(y_test, y_pred)
print('Cm1',cm1)
# Visualising the Training set results
from matplotlib.colors import ListedColormap
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('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
from matplotlib.colors import ListedColormap
X_set, y_set = x_test, y_testX1, 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('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output







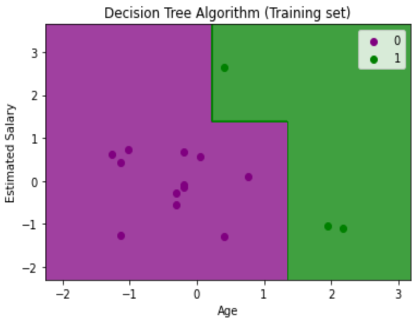
```
In [5]: runfile('D:/48.py', wdir='D:')
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
[[57  1]
 [ 5 17]]
The Accuracy   is   92.5
              precision    recall  f1-score   support

           0       0.92      0.98      0.95        58
           1       0.94      0.77      0.85        22

    accuracy                           0.93        80
   macro avg       0.93      0.88      0.90        80
weighted avg       0.93      0.93      0.92        80

Cm1 [[57  1]
 [ 5 17]]
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

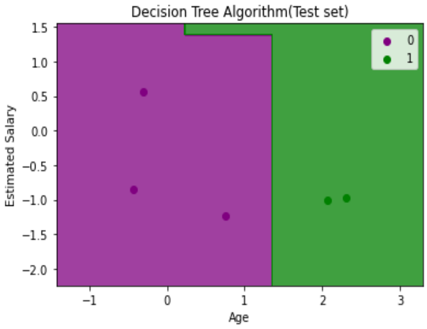
PRACTICAL 5A: WRITE A PROGRAM TO DEMONSTRATE THE WORKING OF THE DECISION TREE BASED ID3 ALGORITHM. USE AN APPROPRIATE DATA SET FOR BUILDING THE DECISION TREE AND APPLY THIS KNOWLEDGE TO CLASSIFY A NEW SAMPLE.

Code:

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
data_set = pd.read_csv('user_data.csv')
x = data_set.iloc[:, [1, 2]].values
y = data_set.iloc[:, 3].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
from sklearn.preprocessing import StandardScaler
st_x = StandardScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.transform(x_test)
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
random_state=0, splitter='best')
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1 = accuracy_score(y_test, y_pred) * 100
print("Accuracy is", sc1)
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start=x_set[:, 0].min() - 1, stop=x_set[:, 0].max() + 1,
step=0.01),
nm.arange(start=x_set[:, 1].min() - 1, stop=x_set[:, 1].max() + 1, step=0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha=0.75, cmap=ListedColormap(('purple', 'green')))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c=ListedColormap(('purple', 'green'))(i), label=j)
mtp.title('Decision Tree Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start=x_set[:, 0].min() - 1, stop=x_set[:, 0].max() + 1,
step=0.01),
nm.arange(start=x_set[:, 1].min() - 1, stop=x_set[:, 1].max() + 1, step=0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha=0.75, cmap=ListedColormap(('purple', 'green')))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c=ListedColormap(('purple', 'green'))(i), label=j)
mtp.title('Decision Tree Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:

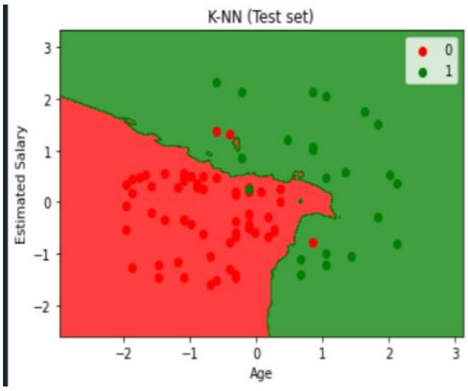


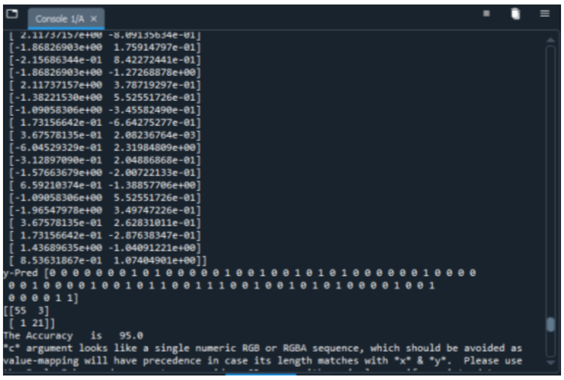


PRACTICAL NO.: 5. B: WRITE A PROGRAM TO IMPLEMENT K-NEAREST NEIGHBOUR ALGORITHM TO CLASSIFY THE DATA SET.

Code :
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#importing datasets
data_set= pd.read_csv('Social_Network_Ads.csv')
  # Age and EstimatedSalary as our independent variable matrix.# And take the Purchased
column in the dependent variable vector.
x= data_set.iloc[:,[2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.20, random_state=0)
print('len',len(y_test))
# Scaling the Datasets
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
print(x_train)
print(x_test)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print('y-Pred',y_pred)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
sc1 = accuracy_score(y_test, y_pred)*100
print('The Accuracy   is  ',sc1)
# Visualising the Training set results
from matplotlib.colors import ListedColormap
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('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
# Visualising the Test set results
from matplotlib.colors import ListedColormap
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('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output:





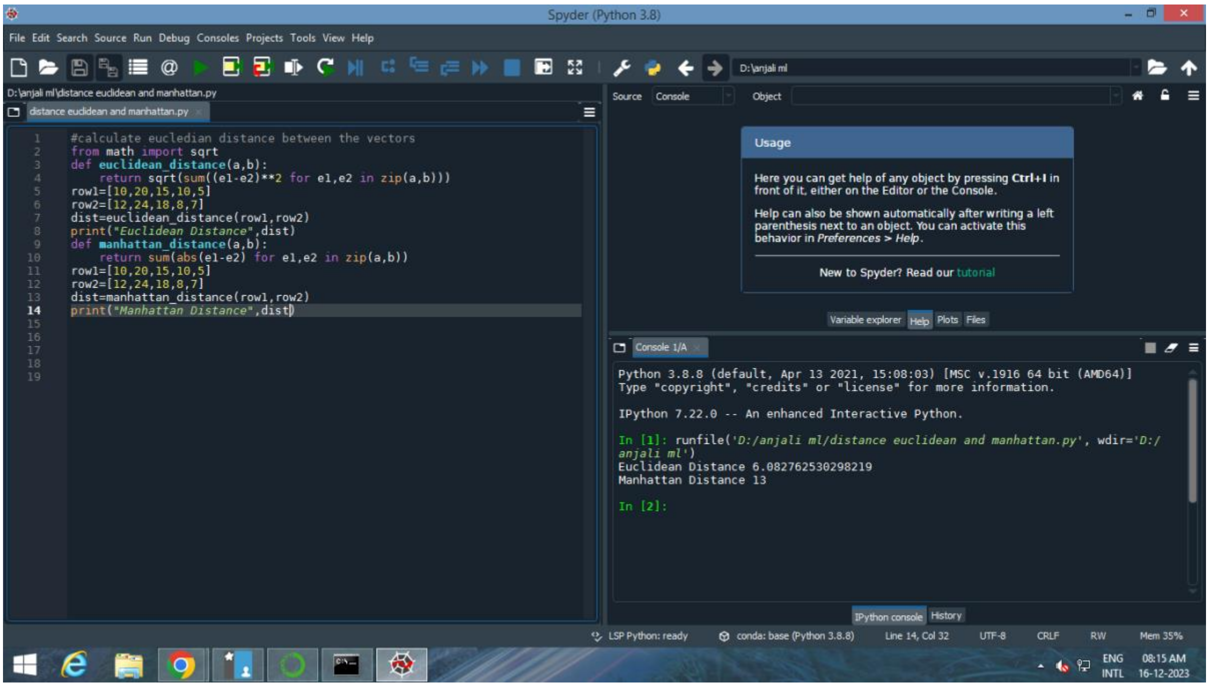
```
 [ 2.11737157e+00 -8.09135634e-01]
 [-1.86826903e+00  1.75914797e-01]
 [-2.15686344e-01  8.42272441e-01]
 [-1.86826903e+00 -1.27268878e+00]
 [ 2.11737157e+00  3.78719297e-01]
 [-1.38221530e+00  5.52551726e-01]
 [-1.09058306e+00 -3.45582490e-01]
 [ 1.73156642e-01 -6.64275277e-01]
 [ 3.67578135e-01  2.08236764e-03]
 [-6.04529329e-01  2.31984809e+00]
 [-3.12897090e-01  2.04886868e-01]
 [-1.57663679e+00 -2.00722133e-01]
 [ 6.59210374e-01 -1.38857706e+00]
 [-1.09058306e+00  5.52551726e-01]
 [-1.96547978e+00  3.49747226e-01]
 [ 3.67578135e-01  2.62831011e-01]
 [ 1.73156642e-01 -2.87638347e-01]
 [ 1.43689635e+00 -1.04091221e+00]
 [ 8.53631867e-01  1.07404901e+00]]
y-Pred [0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1]
[[55  3]
 [ 1 21]]
The Accuracy   is   95.0
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as
value-mapping will have precedence in case its length matches with *x* & *y*.  Please use
```

PRACTICAL 6A. IMPLEMENT THE DIFFERENT DISTANCE METHOD MANHATTAN AND EUCLIDEAN DISTANCE

Code:

```
#calculate eucledian distance between the vectors
from math import sqrt
def euclidean_distance(a,b):
    return sqrt(sum((e1-e2)**2 for e1,e2 in zip(a,b)))
row1=[10,20,15,10,5]
row2=[12,24,18,8,7]
dist=euclidean_distance(row1,row2)
print("Euclidean Distance",dist)
def manhattan_distance(a,b):
    return sum(abs(e1-e2) for e1,e2 in zip(a,b))
row1=[10,20,15,10,5]
row2=[12,24,18,8,7]
dist=manhattan_distance(row1,row2)
print("Manhattan Distance",dist)
```

Output



Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\anjali ml
D:\anjali ml\distance euclidean and manhattan.py
distance euclidean and manhattan.py
Source Console Object
Variable explorer Help Plots Files
Console 1/A
IPython console History
LSP Python: ready conda: base (Python 3.8.8) Line 14, Col 32 UTF-8 CRLF RW Mem 35%
ENG INTL 08:15 AM 16-12-2023

```
#calculate eucledian distance between the vectors
from math import sqrt
def euclidean_distance(a,b):
    return sqrt(sum((e1-e2)**2 for e1,e2 in zip(a,b)))
row1=[10,20,15,10,5]
row2=[12,24,18,8,7]
dist=euclidean_distance(row1,row2)
print("Euclidean Distance",dist)
def manhattan_distance(a,b):
    return sum(abs(e1-e2) for e1,e2 in zip(a,b))
row1=[10,20,15,10,5]
row2=[12,24,18,8,7]
dist=manhattan_distance(row1,row2)
print("Manhattan Distance",dist)
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

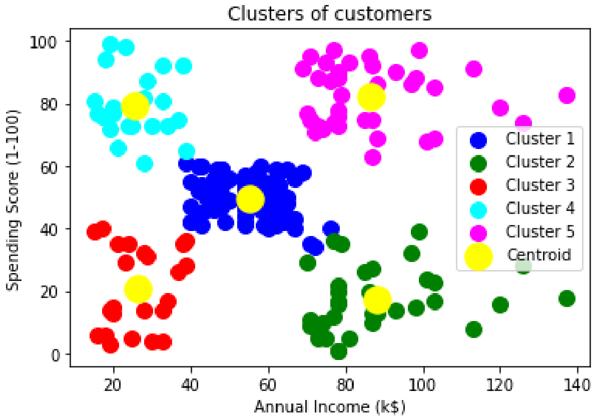
```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/anjali ml/distance euclidean and manhattan.py', wdir='D:/
anjali ml')
Euclidean Distance 6.082762530298219
Manhattan Distance 13

In [2]:
```

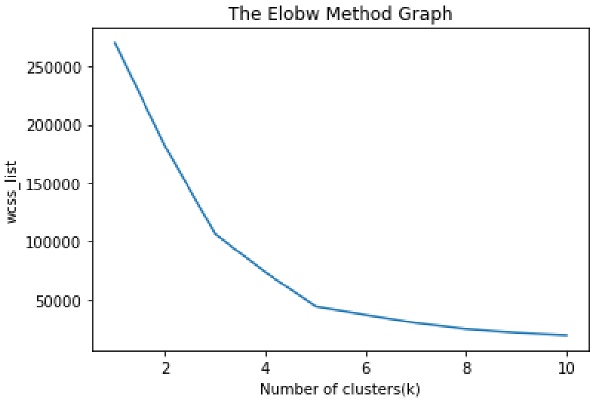
PRACTICAL 6B: IMPLEMENT THE CLASSIFICATION MODEL USING CLUSTERING FOR THE FOLLOWING TECHNIQUES WITH K MEANS CLUSTERING WITH PREDICTION, TEST SCORE AND CONFUSION MATRIX.

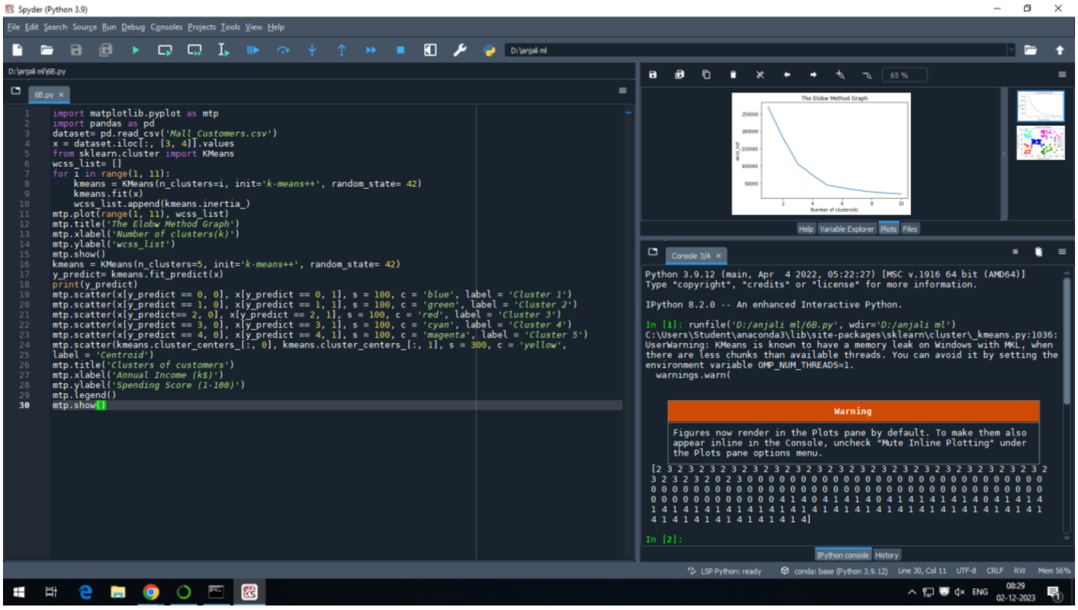
Code

```
import matplotlib.pyplot as mtp
import pandas as pd
dataset= pd.read_csv('Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
from sklearn.cluster import KMeans
wcss_list= []
for i in range(1, 11):
kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
kmeans.fit(x)
wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
print(y_predict)
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output







```python
import matplotlib.pyplot as mtp
import pandas as pd
dataset= pd.read_csv('Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
from sklearn.cluster import KMeans
wcss_list= []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
print(y_predict)
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

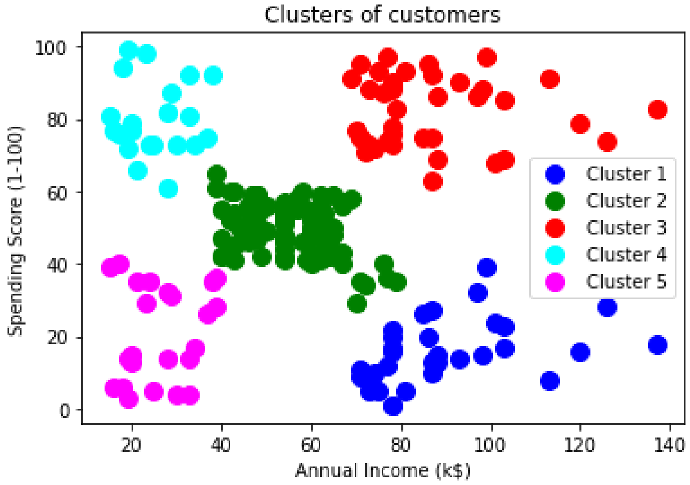
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/anjali ml/6B.py', wdir='D:/anjali ml')
C:\Users\Student\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(

                               Warning
   Figures now render in the Plots pane by default. To make them also
   appear inline in the Console, uncheck "Mute Inline Plotting" under
   the Plots pane options menu.
 [2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
 3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4
 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]

In [2]:
```

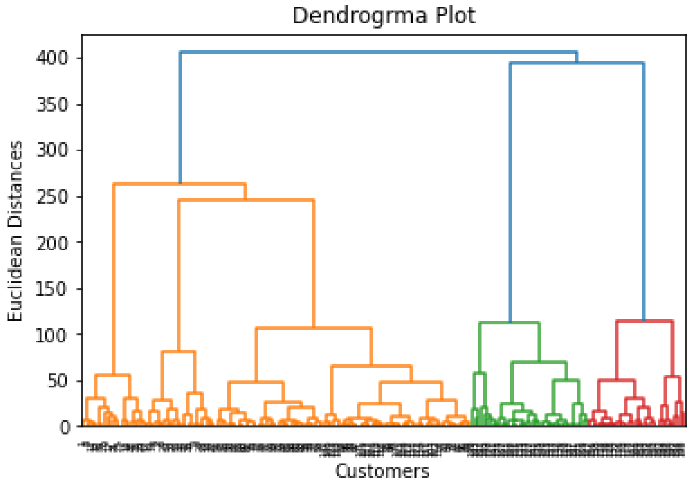
PRACTICAL 7A : IMPLEMENT THE CLASSIFICATION MODEL USING CLUSTERING FOR THE FOLLOWING TECHNIQUES WITH HIERARCHICAL CLUSTERING WITH PREDICTION, TEST SCORE AND CONFUSION MATRIX

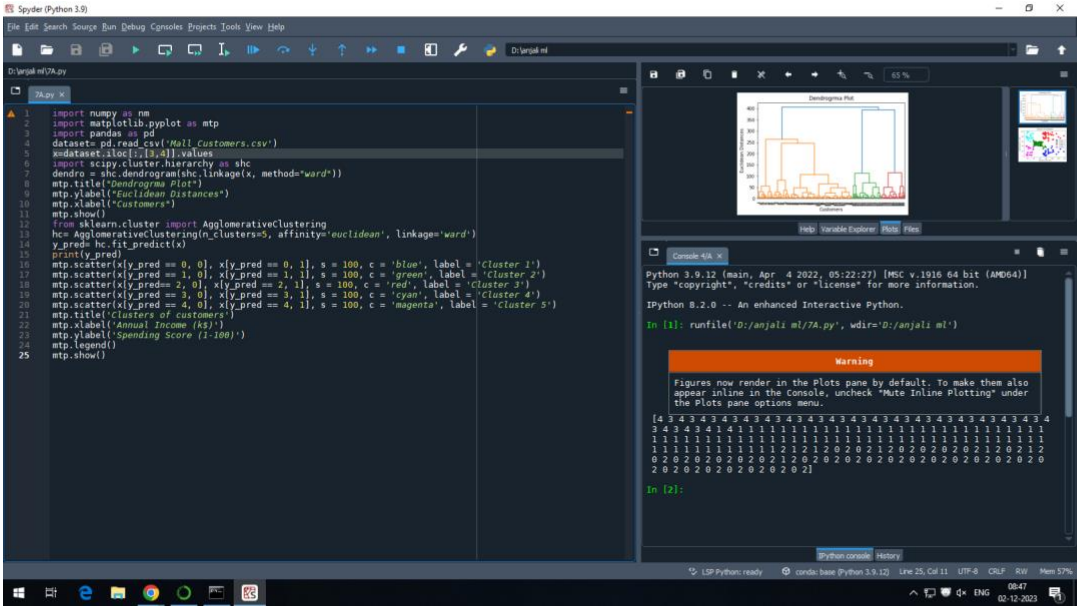
Code

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset= pd.read_csv('Mall_Customers.csv')
x=dataset.iloc[:,[3,4]].values
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(x, method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Customers")
mtp.show()
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
y_pred= hc.fit_predict(x)
print(y_pred)
mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output







```python
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset= pd.read_csv('Mall_Customers.csv')
x=dataset.iloc[:,[3,4]].values
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(x, method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Customers")
mtp.show()
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
y_pred= hc.fit_predict(x)
print(y_pred)
mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

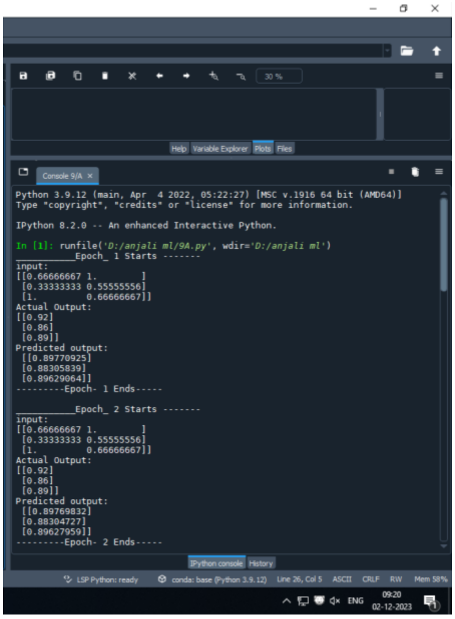
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/anjali ml/7A.py', wdir='D:/anjali ml')

                              Warning
  Figures now render in the Plots pane by default. To make them also
  appear inline in the Console, uncheck "Mute Inline Plotting" under
  the Plots pane options menu.

[4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 1 2
 0 2 0 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]

In [2]:
```

PRACTICAL 9A : BUILD AN ARTIFICIAL NEURAL NETWORK BY IMPLEMENTING THE BACKPROPAGATION ALGORITHM AND TEST THE SAME USING APPROPRIATE DATA SETS.

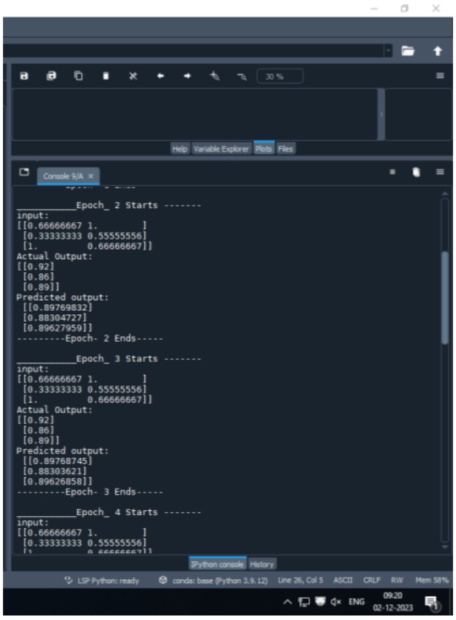
Code

```
import numpy as np
x=np.array(([2,9],[1,5],[3,6]), dtype=float)
y=np.array(([92],[86],[89]), dtype=float)
x=x/np.amax(x,axis=0)
y=y/100
def sigmoid (x):
return 1/(1 + np.exp(-x))
def der_sigmoid (x):
return x*(1-x)
epoch=5
lr=0.1
inputlayer_neurons =2
hiddenlayer_neurons = 3
output_neurons = 1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
hinp1=np.dot(x,wh)
hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp= outinp1+bout
output = sigmoid(outinp)

EO = y-output
outgrad = der_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)
hiddengrad = der_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_output) * lr
wh += x.T.dot(d_hiddenlayer) * lr
print("_____________Epoch_", i+1, "Starts -------")
print("input:\n" + str(x))
print("Actual Output: \n" + str(y))
print("Predicted output: \n", output)
print("---------Epoch-", i+1, "Ends-----\n")
print("Input: \n" + str(x))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" , output)
```

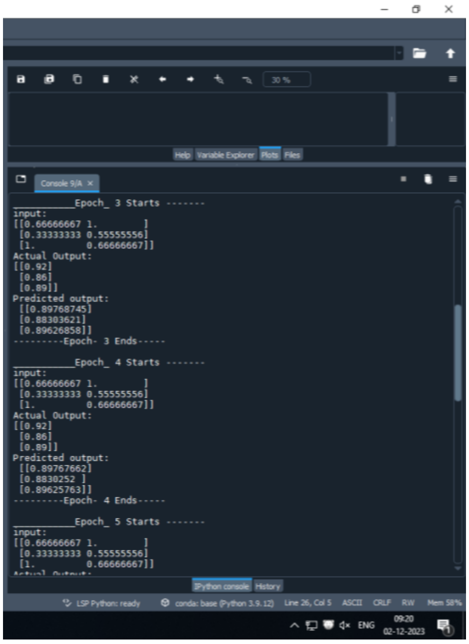
Output



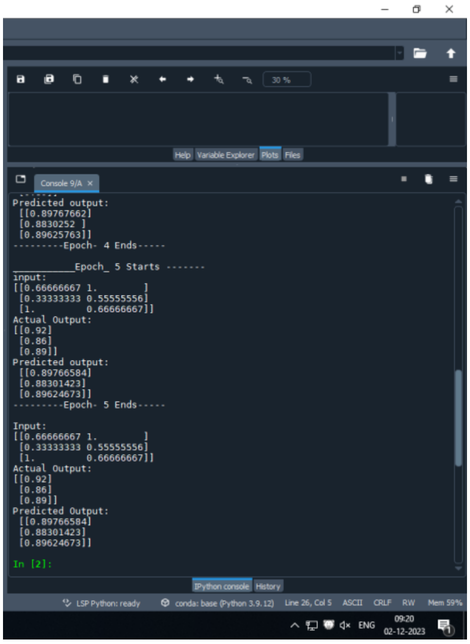
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/anjali ml/9A.py', wdir='D:/anjali ml')
___________Epoch_ 1 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89770925]
 [0.88305839]
 [0.89629064]]
---------Epoch- 1 Ends-----

___________Epoch_ 2 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89769832]
 [0.88304727]
 [0.89627959]]
---------Epoch- 2 Ends-----
```



```
___________Epoch_ 2 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89769832]
 [0.88304727]
 [0.89627959]]
---------Epoch- 2 Ends-----

___________Epoch_ 3 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89768745]
 [0.88303621]
 [0.89626858]]
---------Epoch- 3 Ends-----

___________Epoch_ 4 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
```



```
___________Epoch_ 3 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89768745]
 [0.88303621]
 [0.89626858]]
---------Epoch- 3 Ends-----

___________Epoch_ 4 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89767662]
 [0.8830252 ]
 [0.89625763]]
---------Epoch- 4 Ends-----

___________Epoch_ 5 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
```

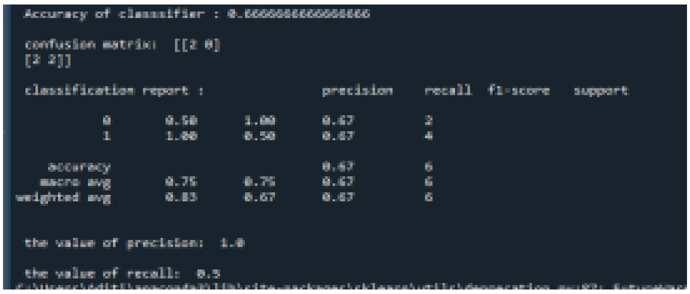


```
Predicted output:
 [[0.89767662]
 [0.8830252 ]
 [0.89625763]]
---------Epoch- 4 Ends-----

___________Epoch_ 5 Starts -------
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89766584]
 [0.88301423]
 [0.89624673]]
---------Epoch- 5 Ends-----

Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.89766584]
 [0.88301423]
 [0.89624673]]

In [2]:
```

PRACTICAL 9B : ASSUMING A SET OF DOCUMENTS THAT NEED TO BE CLASSIFIED, USE THE NAÏVE BAYESIAN CLASSIFIER MODEL TO PERFORM THIS TASK.

Code:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
data = pd.read_csv('navebyase.csv', names=['text','label'])
print("\n The Dataset is :\n", data)
print("\n The Dimensions of the dataset", data.shape)
data['labelnum'] = data.label.map({'positive':1, 'negitive':0})
x= data.text
y= data.labelnum
print(x)
print(y)
vectorizer = TfidfVectorizer()
data = vectorizer.fit_transform(x)
print("\n the TF-IDF features of Dataset:\n")
df = pd.DataFrame(data.toarray(), columns= vectorizer.get_feature_names())
df.head()
x_train, x_test, y_train, y_test = train_test_split(data, y, test_size=0.3, random_state=2)
print("\n the total number of taraning data :" ,y_train.shape)
print("\n the total number of test data :" ,y_test.shape)
clf = MultinomialNB().fit(x_train, y_train)
predicted = clf.predict(x_test)
print('\n Accuracy of classsifier :', metrics.accuracy_score(y_test, predicted))
print("\n confusion matrix: ",metrics.confusion_matrix(y_test, predicted))
print("\n classification report :",metrics.classification_report(y_test, predicted))
print("\n the value of precision: ", metrics.precision_score(y_test, predicted))
print("\n the value of recall: ",metrics.recall_score(y_test, predicted))
```



```
 Accuracy of classsifier : 0.6666666666666666

 confusion matrix:  [[2 0]
 [2 2]]

 classification report :              precision    recall  f1-score   support

           0       0.50      1.00      0.67         2
           1       1.00      0.50      0.67         4

    accuracy                           0.67         6
   macro avg       0.75      0.75      0.67         6
weighted avg       0.83      0.67      0.67         6


 the value of precision:  1.0

 the value of recall:  0.5
C:\Users\Aditi\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarn
```