Source Code:

```
import pandas as pd # For loading csv file data to numpy array
import numpy as np # For using data as array
import matplotlib.pyplot as plt # For plotting graph(x,y)
from datetime import datetime
print('----------------------Simple Linear Regression----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
#To import dataset
data=pd.read_csv('Advertising_Budget.csv')
data.head() #To show data

x=data['Impressions'] # Independent variable or predictor.
y=data['Budget'] # Dependent variable or outcome.

def linear_regression(x, y):
#simple linear regression equation is: Y = B0 + B1X
  N=len(x)
  x_mean=x.mean() # Mean of x value
  y_mean=y.mean() # Mean of y value
  #To find B0 and B1 value using formula
  B1_num = ((x - x_mean) * (y - y_mean)).sum()
  B1_den = ((x - x_mean)**2).sum()
  B1 = B1_num / B1_den # B1
  B0 = y_mean - (B1*x_mean) # B0
  print('B0: ',B0) ## B0 – is a constant (shows the value of Y when the value of X=0)
  print('B1: ',B1) ## B1 – the regression coefficient (shows how much Y changes for each unit change in X)
  reg_line = 'y = {} + {}β'.format(B0, round(B1, 3)) # To show Regression Line using y=B0+B1X format
  return (B0, B1, reg_line)

# To find correlation coefficient using formula
N = len(x)
num = (N * (x*y).sum()) - (x.sum() * y.sum())
```

```
den = np.sqrt((N * (x**2).sum() - x.sum()**2) * (N * (y**2).sum() - y.sum()**2))
R = num / den # Correlation coefficient

# Applying these functions to our data, we can print out the results:
B0, B1, reg_line=linear_regression(x,y)
print('"Regression Line:" ', reg_line)
print('"Correlation Coef.": ', R)
print('"Goodness of Fit": ', R**2)

# Plotting the Regression Line
plt.figure(figsize=(15,5)) # size of figure
plt.scatter(x, y, s=300, linewidths=1, edgecolor='black')
# To show description about SLR figure
text = '"X Mean: ${} (in millions)
Y Mean: ${} (in Million US$)
Correlation Coef.: {}
Goodness of Fit: {}
y = {} + {}X"'.format(round(x.mean(), 2),
                      round(y.mean(), 2),
                      round(R, 4),
                      round(R**2, 4),
                      round(B0, 3),
                      round(B1, 3))
plt.text(x=80, y=13, s=text, fontsize=12, bbox={'facecolor': 'grey', 'alpha': 0.6, 'pad': 15}) #To
give text description format
plt.title('Advertising_Budget')
plt.xlabel('Budget', fontsize=13)
plt.ylabel('Impressions(No.of viewers)', fontsize=13)
plt.plot(x, B0 + B1*x, c = 'g', linewidth=5, alpha=.5, solid_capstyle='round') # For plottng
linear line
plt.scatter(x=x.mean(), y=y.mean(), marker='*', s=10**2.5, c='r') # average point
```

Output:

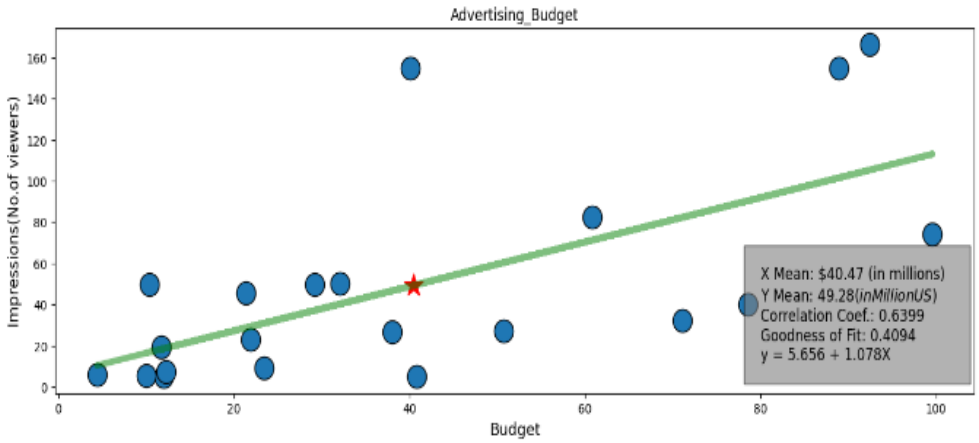
jupyter Farhana_Simple MachineLearning Model (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
----------------------Simple Linear Regression-----------------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 19/11/2022 14:42:05
B0:  5.656010401575564
B1:  1.0779286674122301
"Regression Line:"  y = 5.656010401575564 + 1.078β
"Correlation Coef.":  0.6398706128862057
"Goodness of Fit":  0.40943440123536856
```

Out[1]: <matplotlib.collections.PathCollection at 0x24979d7dae0>



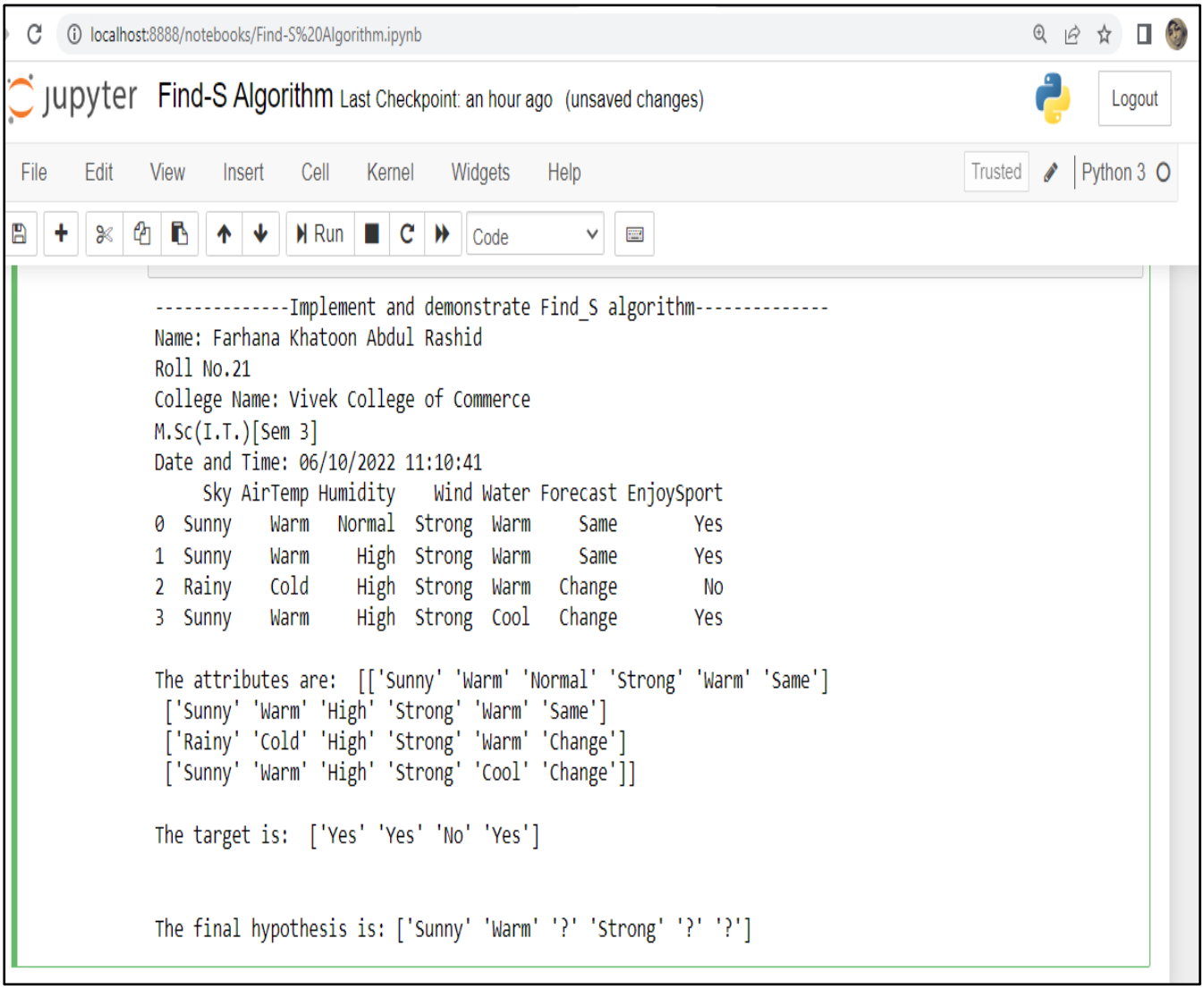
Type here to search 31°C ENG 14:42 19-11-2022

Source Code:

```
print("--------------Implement and demonstrate Find_S algorithm--------------")
print("Name: Farhana Khatoon Abdul Rashid")
print("Roll No.21")
print("College Name: Vivek College of Commerce")
print("M.Sc(I.T.)[Sem 3]")
from datetime import datetime
now=datetime.now()
#dd/mm/yy H:M:S
dt_string=now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time:", dt_string)
import pandas as pd
import numpy as np
#To read the data in the csv file
data = pd.read_csv("ENJOYSPORT.csv")
print(data)
print('')
#Making an array of all the attributes
attribute = np.array(data)[:,:-1]
print("The attributes are: ",attribute)
print('')
#Segregating the target that has positive and negative examples
target = np.array(data)[:,-1]
print("The target is: ",target)
print('')
#Training function to implement find-s algorithm
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
#Obtaining the final hypothesis
print('')
print("The final hypothesis is:",train(attribute,target))
```

Output:



Source Code:

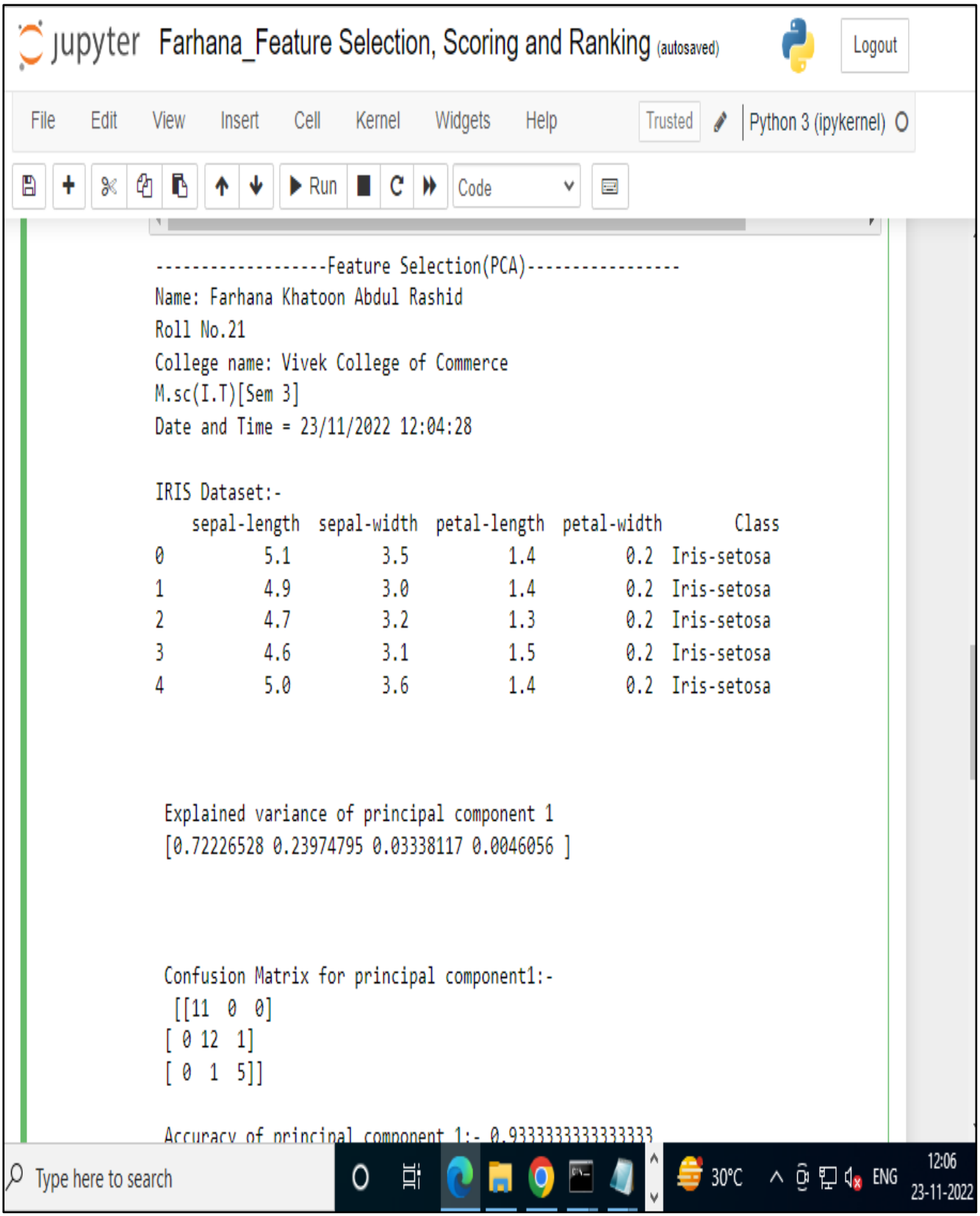
1. **Feature Selection using PCA:**

```
from datetime import datetime
print('------------------Feature Selection(PCA)----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
import numpy as np
import pandas as pd
import numpy as np
import pandas as pd
from sklearn import metrics
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Import the data set
# Read dataset to pandas dataframe
dataset = pd.read_csv("8-Irisdataset.csv", names=names)
print("\nIRIS Dataset:-\n", dataset.head())
print("\n")
X = dataset.drop('Class', 1)
y = dataset['Class']
# Splitting the dataset into the Training set 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=0.2, random_state=0)
#Standardize the Data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
#Applying PCA
from sklearn.decomposition import PCA
pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print("\n Explained variance of principal component 1\n", explained_variance)
print("\n")
```

```
#To use 1 principal component to train our algorithm
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
#Training and Making Predictions
from sklearn.ensemble import RandomForestClassifier
#I use random forest classification for making the predictions
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#Performance Evaluation
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("\n Confusion Matrix for principal component1:-\n ",cm)
print('\n Accuracy of principal component 1:-', accuracy_score(y_test, y_pred))
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents
        , columns = ['principal component 1', 'principal component 2'])
finalDf = pd.concat([principalDf, dataset[['Class']]], axis = 1)
print("\n ===Comparison between principal component1 and principal component 2===
\n\n",finalDf)
#Visualize 2D Projection
from matplotlib import 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):
    indicesToKeep = finalDf['Class'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
            , finalDf.loc[indicesToKeep, 'principal component 2']
            , c = color
            , s = 50)
```

```
ax.legend(targets)
ax.grid()
```

Output:



jupyter Farhana_Feature Selection, Scoring and Ranking (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

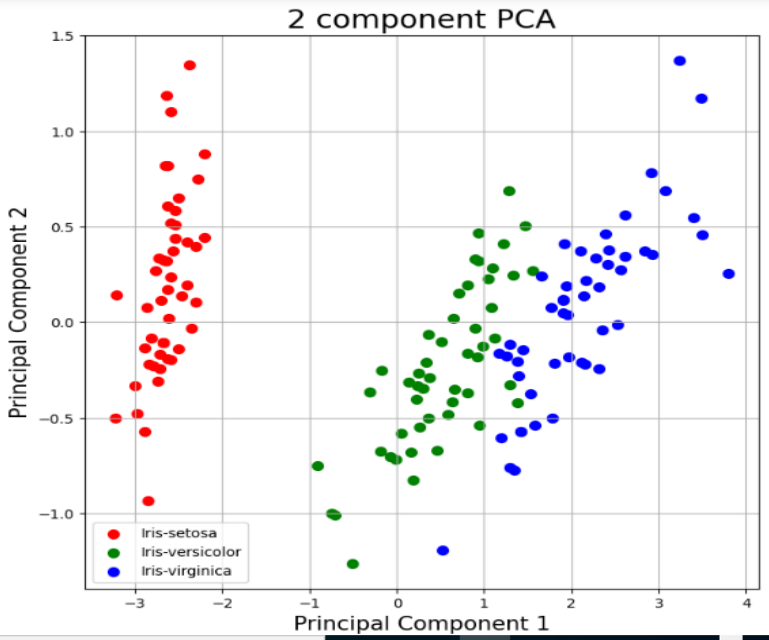
```
Explained variance of principal component 1
[0.72226528 0.23974795 0.03338117 0.0046056 ]



Confusion Matrix for principal component1:-
 [[11  0  0]
[ 0 12  1]
[ 0  1  5]]

Accuracy of principal component 1:- 0.9333333333333333

===Comparison between principal component1 and principal component 2===

     principal component 1  principal component 2          Class
0                -2.684207               0.326607     Iris-setosa
1                -2.715391              -0.169557     Iris-setosa
2                -2.889820              -0.137346     Iris-setosa
3                -2.746437              -0.311124     Iris-setosa
4                -2.728593               0.333925     Iris-setosa
..                     ...                    ...             ...
145               1.944017               0.187415  Iris-virginica
146               1.525664              -0.375021  Iris-virginica
147               1.764046               0.078519  Iris-virginica
148               1.901629               0.115877  Iris-virginica
149               1.389666              -0.282887  Iris-virginica

[150 rows x 3 columns]
```

Type here to search 30°C ENG 12:06 23-11-2022

jupyter Farhana_Feature Selection, Scoring and Ranking Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code



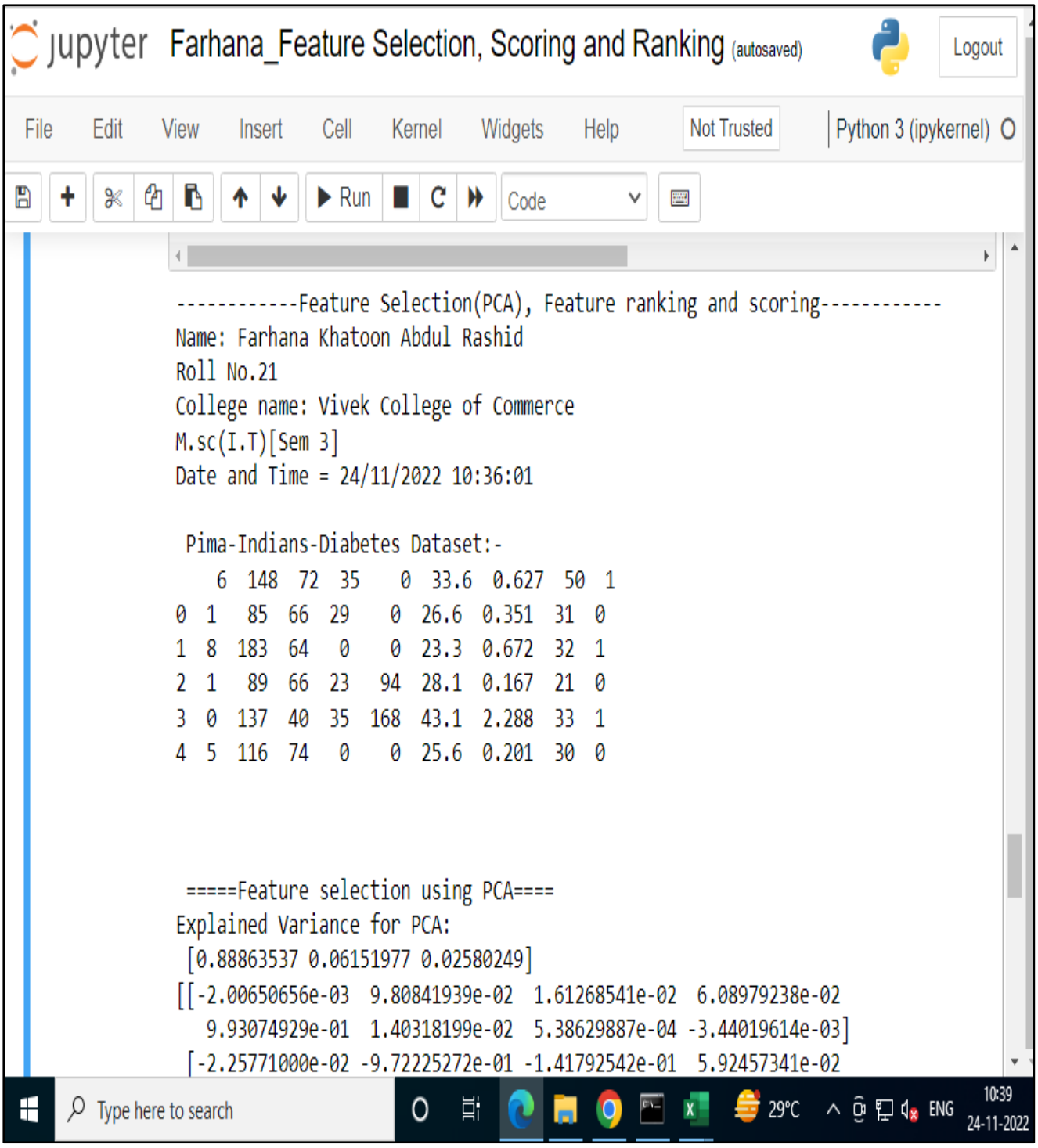
Type here to search 30°C ENG 12:07 23-11-2022

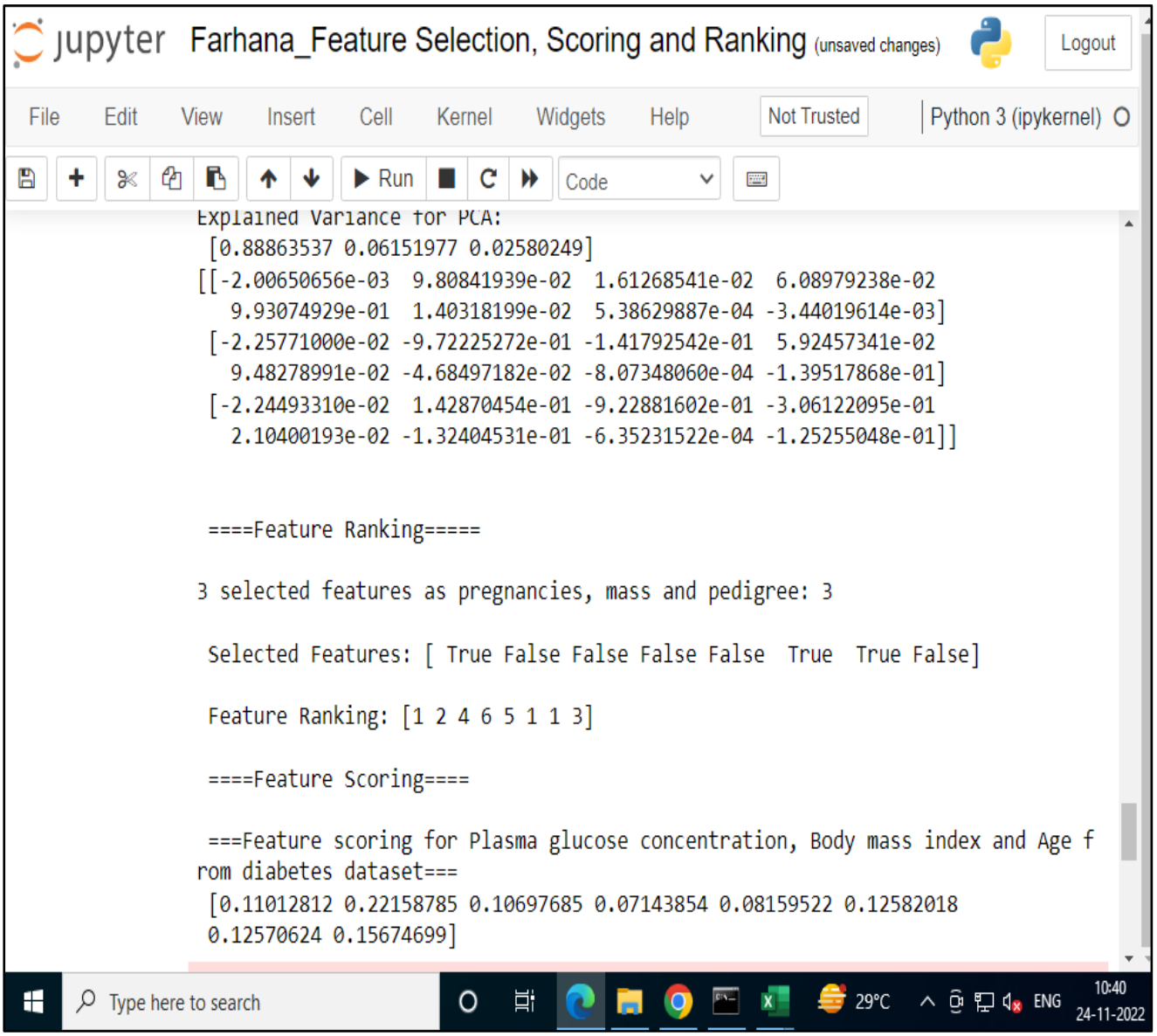
Source Code:

```
from datetime import datetime
print('------------Feature Selection(PCA), Feature ranking and scoring------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
#Importing libraries
import numpy
from pandas import read_csv
# Import the data set
# Read dataset to pandas dataframe
dataset = pd.read_csv("pima-indians-diabetes.csv")
print("\n Pima-Indians-Diabetes Dataset:-\n", dataset.head())
print("\n")
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = dataset.iloc[:,0:8].values
Y = dataset.iloc[:,8].values

#1.Feature selection using PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance for PCA:\n %s" % fit.explained_variance_ratio_)
print(fit.components_)
print("\n")
#2.Feature ranking using Recursive Feature Elimination (RFE)
# feature extraction
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
rfe = rfe = RFE(estimator=LogisticRegression(solver='lbfgs'), n_features_to_select=3)
fit = rfe.fit(X, Y)
print("3 selected features as pregnancies, mass and pedigree: %d" % fit.n_features_)
print("\n Selected Features: %s" % fit.support_)
print("\n Feature Ranking: %s" % fit.ranking_)
```

```
#3.Feature importance (Feature scoring)
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier(n_estimators=10)
model.fit(X, Y)
print("\n ===Feature scoring for Plasma glucose concentration, Body mass index and Age from diabetes dataset=== \n", model.feature_importances_)
```

Output:



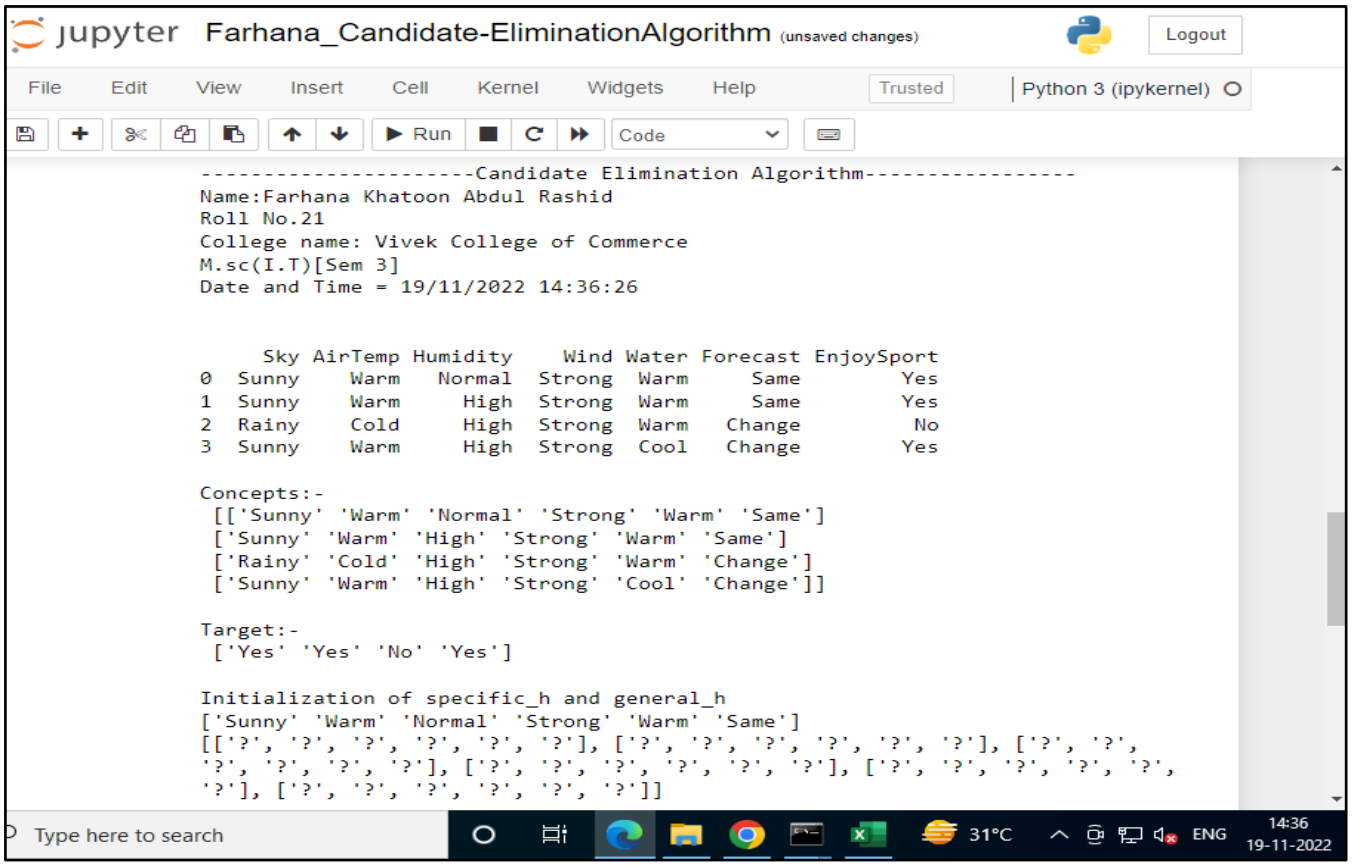

jupyter Farhana_Feature Selection, Scoring and Ranking (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Explained variance for PCA:
[0.88863537 0.06151977 0.02580249]
[[-2.00650656e-03 9.80841939e-02 1.61268541e-02 6.08979238e-02
9.93074929e-01 1.40318199e-02 5.38629887e-04 -3.44019614e-03]
[-2.25771000e-02 -9.72225272e-01 -1.41792542e-01 5.92457341e-02
9.48278991e-02 -4.68497182e-02 -8.07348060e-04 -1.39517868e-01]
[-2.24493310e-02 1.42870454e-01 -9.22881602e-01 -3.06122095e-01
2.10400193e-02 -1.32404531e-01 -6.35231522e-04 -1.25255048e-01]]
====Feature Ranking=====
3 selected features as pregnancies, mass and pedigree: 3
Selected Features: [True False False False False True True False]
Feature Ranking: [1 2 4 6 5 1 1 3]
====Feature Scoring====
===Feature scoring for Plasma glucose concentration, Body mass index and Age f
rom diabetes dataset===
[0.11012812 0.22158785 0.10697685 0.07143854 0.08159522 0.12582018
0.12570624 0.15674699]
Type here to search
29°C
ENG
10:40
24-11-2022

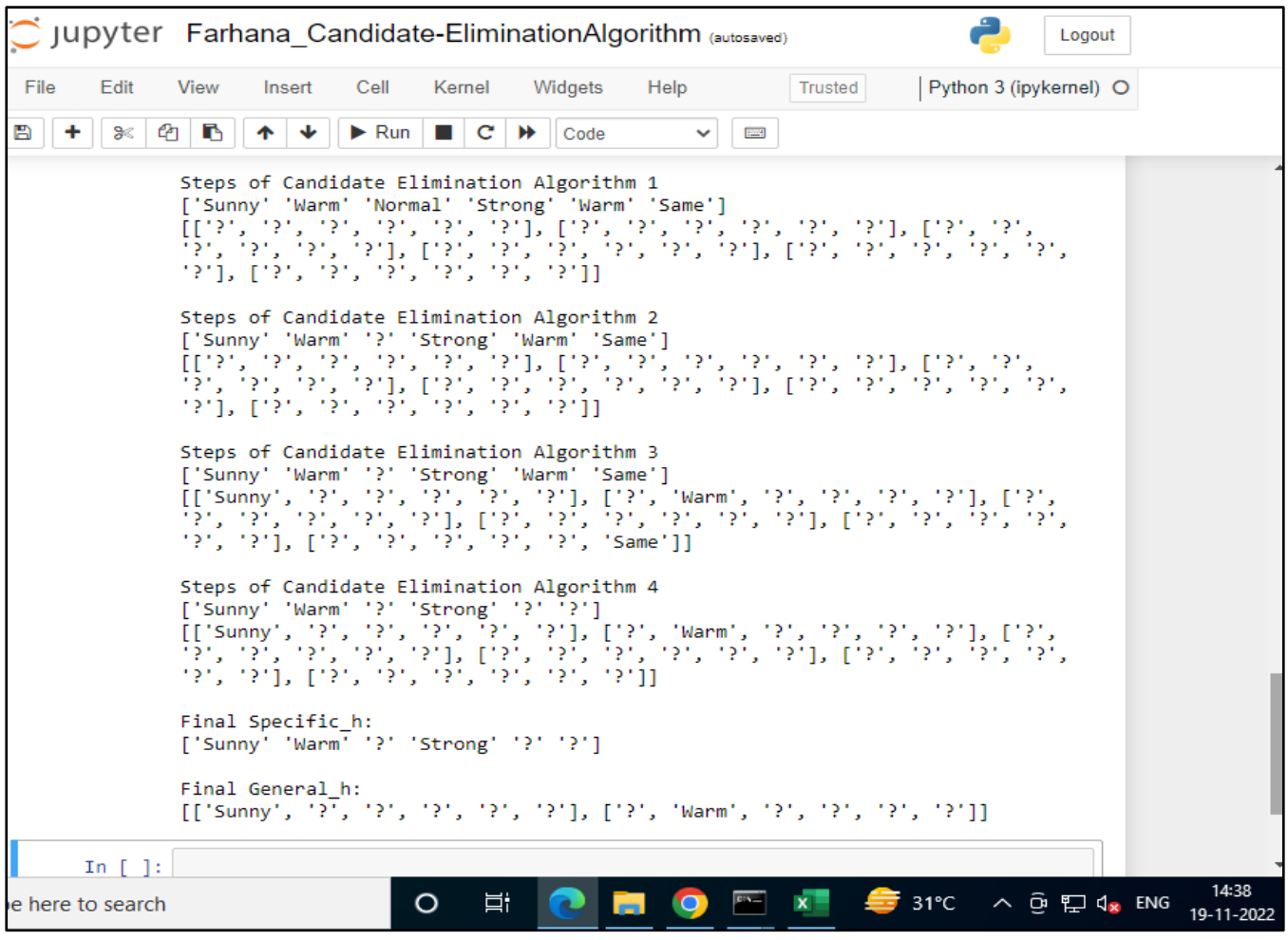
Source Code:

```
from datetime import datetime
print('----------------------Candidate Elimination Algorithm----------------')
print('Name: Teli Farhana Khatoon')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
print("\n")

import numpy as np
import pandas as pd
# Loading Data from a CSV File
data = pd.DataFrame(data=pd.read_csv('ENJOYSPORT.csv'))
print(data)
# Separating concept features from Target
concepts = np.array(data.iloc[:,0:-1])
print("\nConcepts:-\n", concepts)
# Isolating target into a separate DataFrame
# copying last column to target array
target = np.array(data.iloc[:,-1])
print("\nTarget:-\n", target)

def learn(concepts, target):

    '''
    learn() function implements the learning method of the Candidate elimination algorithm.
    Arguments:
        concepts - a data frame with all the features
        target - a data frame with corresponding output values
    '''

    # Initialise S0 with the first instance from concepts
    # .copy() makes sure a new list is created instead of just pointing to the same memory
location
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and general_h")
    print(specific_h)
```

```
    #h=["#" for i in range(0,5)]
    #print(h)

    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print(general_h)
    # The learning iterations
    for i, h in enumerate(concepts):

        # Checking if the hypothesis has a positive target
        if target[i] == "Yes":
            for x in range(len(specific_h)):

                # Change values in S & G only if values change
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'

        # Checking if the hypothesis has a positive target
        if target[i] == "No":
            for x in range(len(specific_h)):
                # For negative hypothesis change values only  in G
                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)
        print(general_h)

    # find indices where we have empty rows, meaning those that are unchanged
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        # remove those rows from general_h
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    # Return final values
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("\nFinal Specific_h:", s_final, sep="\n")
print("\nFinal General_h:", g_final, sep="\n")
```

Output:


jupyter Farhana_Candidate-EliminationAlgorithm (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
----------Candidate Elimination Algorithm----------
Name:Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 19/11/2022 14:36:26
Sky AirTemp Humidity Wind Water Forecast EnjoySport
0 Sunny Warm Normal Strong Warm Same Yes
1 Sunny Warm High Strong Warm Same Yes
2 Rainy Cold High Strong Warm Change No
3 Sunny Warm High Strong Cool Change Yes
Concepts:-
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
Target:-
['Yes' 'Yes' 'No' 'Yes']
Initialization of specific_h and general_h
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?']]
Type here to search
31°C
ENG
14:36
19-11-2022


jupyter Farhana_Candidate-EliminationAlgorithm (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Steps of Candidate Elimination Algorithm 1
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?']]
Steps of Candidate Elimination Algorithm 2
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?']]
Steps of Candidate Elimination Algorithm 3
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', 'Same']]
Steps of Candidate Elimination Algorithm 4
['Sunny' 'Warm' '?' 'Strong' '?' '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['Sunny' 'Warm' '?' 'Strong' '?' '?']
Final General_h:
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
In [ ]:
e here to search
31°C
ENG
14:38
19-11-2022

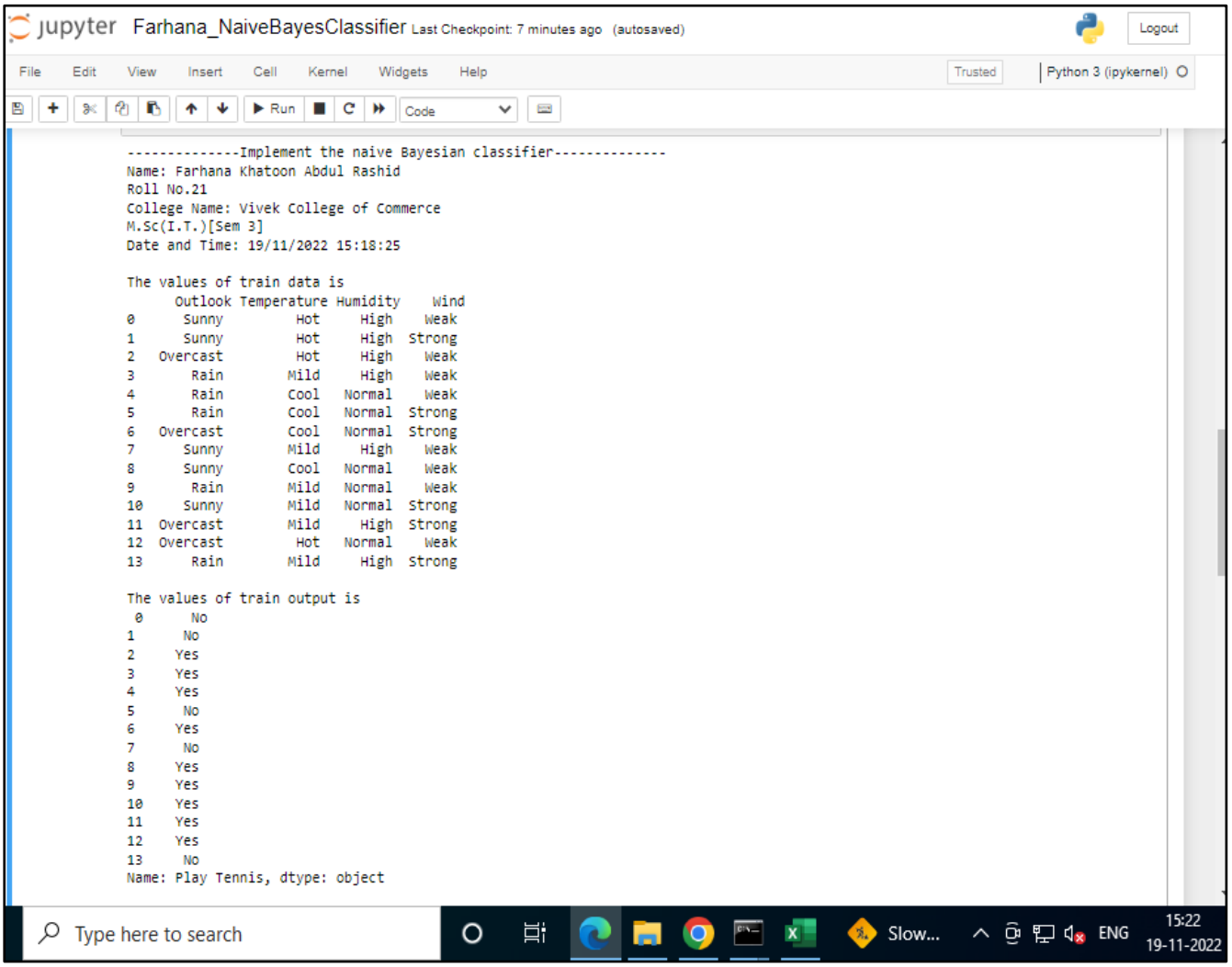
Source Code:

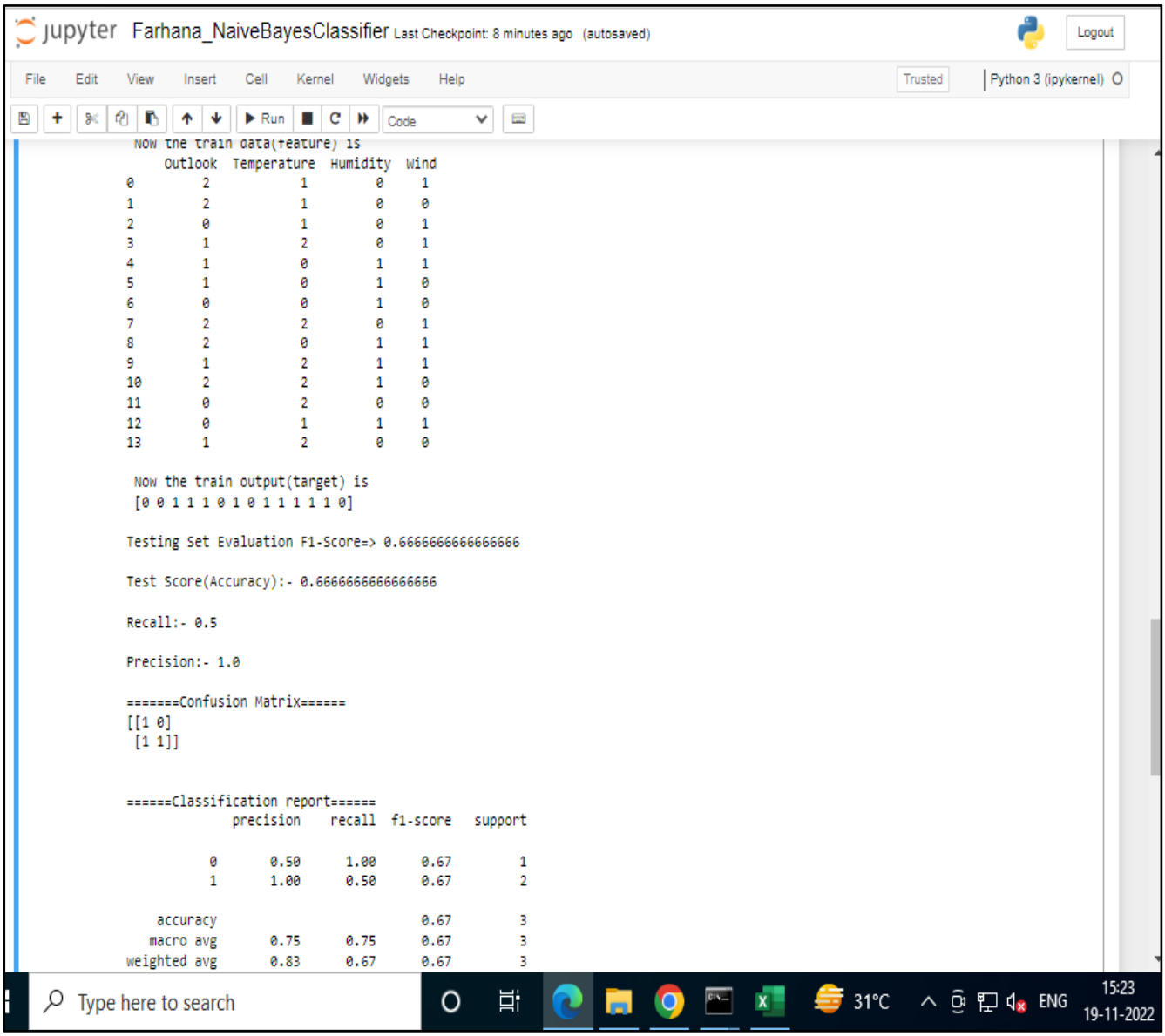
```
print("--------------Implement the naive Bayesian classifier--------------")
print("Name: Farhana Khatoon Abdul Rashid")
print("Roll No.21")
print("College Name: Vivek College of Commerce")
print("M.Sc(I.T.)[Sem 3]")
from datetime import datetime
now=datetime.now()
#dd/mm/yy H:M:S
dt_string=now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time:", dt_string)
# Import libraries
import pandas as pd
import numpy as np
# Import dataset
data=pd.read_csv('PlayTennis.csv')
data.head() # If you can to see the data you can print data

# Obtain train data and train output
x=data.iloc[:,:-1] #Feature
print("\nThe values of train data is\n", x)
y=data.iloc[:,-1] #Target
print("\nThe values of train output is\n", y)

# Convert then in number
from sklearn.preprocessing import LabelEncoder

le_Outlook=LabelEncoder()
x.Outlook=le_Outlook.fit_transform(x.Outlook)

le_Temperature=LabelEncoder()
```

```
x.Temperature=le_Temperature.fit_transform(x.Temperature)

le_Humidity=LabelEncoder()
x.Humidity=le_Humidity.fit_transform(x.Humidity)

le_Wind=LabelEncoder()
x.Wind=le_Wind.fit_transform(x.Wind)
print("\n Now the train data(feature) is\n", x)

le_PlayTennis=LabelEncoder()
y=le_PlayTennis.fit_transform(y)
print("\n Now the train output(target) is\n", y)

# Import train_test_split function
from sklearn.model_selection import train_test_split
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.20)

#Create a Gaussian Classifier
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()

#Train the model using the training sets
classifier.fit(X_train,y_train)

# generating predictions on the test set
y_pred = classifier.predict(X_test)
# Import cross_val_score, classification_report, and confusion_matrix
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score,recall_score, precision_score
from sklearn.metrics import f1_score
print('\nTesting Set Evaluation F1-Score=>',f1_score(y_test,y_pred))
```

```
print("\nTest Score(Accuracy):-", accuracy_score(y_test,y_pred))
print("\nRecall:-", recall_score(y_test,y_pred))
print("\nPrecision:-", precision_score(y_test,y_pred))
# Confusion matrix
cm=np.array(confusion_matrix(y_test,y_pred))
print("\n=======Confusion Matrix======")
print(cm)
print("\n")
# Classification report
cr=classification_report(y_test,y_pred)
print("======Classification report======")
print(cr)
print("\n")
```

Output:




jupyter Farhana_NaiveBayesClassifier Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Now the train data(feature) is
Outlook Temperature Humidity Wind
0 2 1 0 1
1 2 1 0 0
2 0 1 0 1
3 1 2 0 1
4 1 0 1 1
5 1 0 1 0
6 0 0 1 0
7 2 2 0 1
8 2 0 1 1
9 1 2 1 1
10 2 2 1 0
11 0 2 0 0
12 0 1 1 1
13 1 2 0 0
Now the train output(target) is
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Testing Set Evaluation F1-Score=> 0.6666666666666666
Test Score(Accuracy):- 0.6666666666666666
Recall:- 0.5
Precision:- 1.0
=======Confusion Matrix======
[[1 0]
[1 1]]
======Classification report======
precision recall f1-score support
0 0.50 1.00 0.67 1
1 1.00 0.50 0.67 2
accuracy 0.67 3
macro avg 0.75 0.75 0.67 3
weighted avg 0.83 0.67 0.67 3
Type here to search
31°C
ENG
15:23
19-11-2022

Source Code:

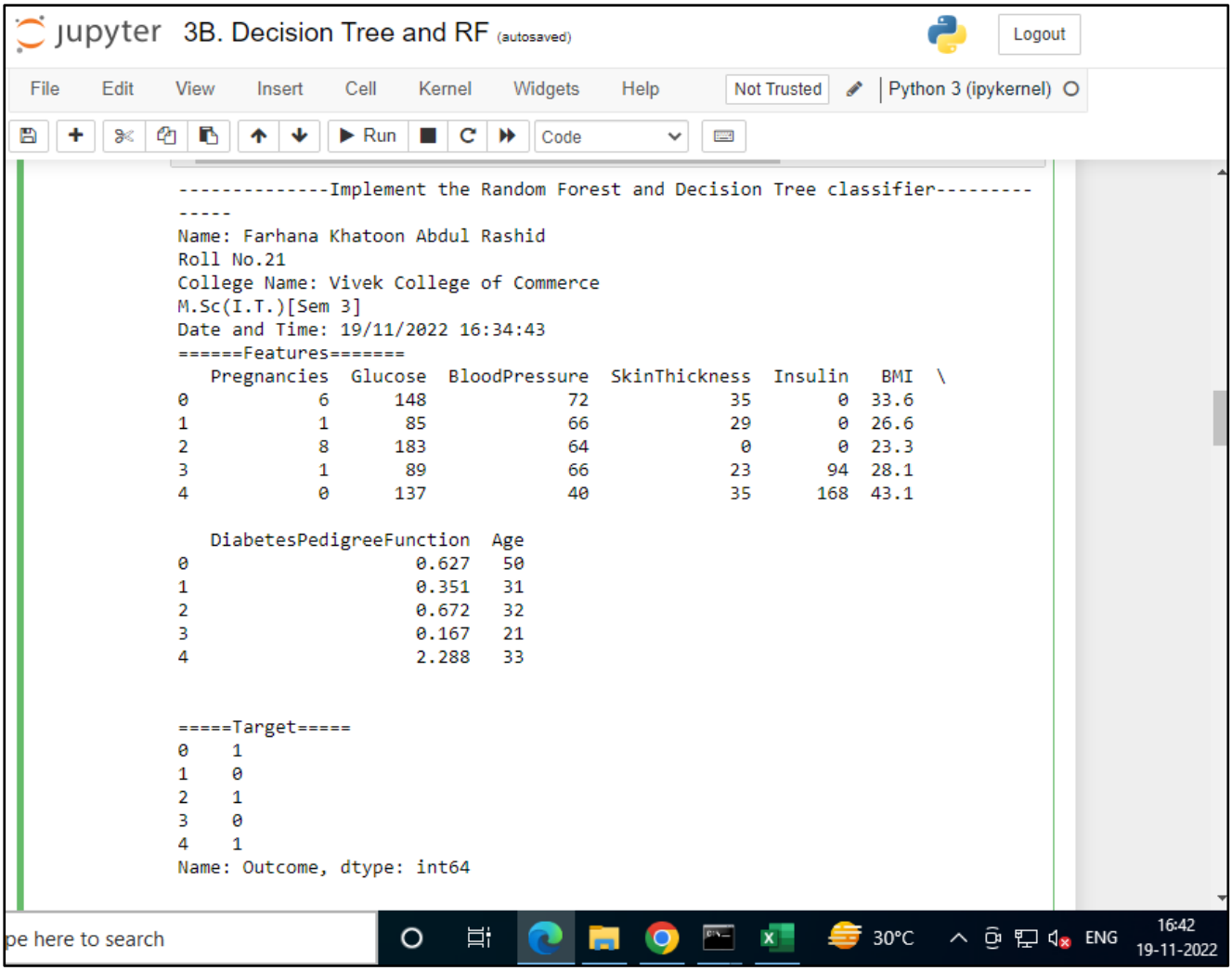
```
print("-------------Implement the Random Forest and Decision Tree classifier-------------")
print("Name: Farhana Khatoon Abdul Rashid")
print("Roll No.21")
print("College Name: Vivek College of Commerce")
print("M.Sc(I.T.)[Sem 3]")
from datetime import datetime
now=datetime.now()
#dd/mm/yy H:M:S
dt_string=now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time:", dt_string)
#Import libraries
import pandas as pd
import numpy as np
from sklearn.metrics import f1_score
#Import dataset
data=pd.read_csv('diabetes.csv')
data.head() #If you can to see the data you can print data
# Select feature data
x=data.drop('Outcome', axis=1)
print("======Features=======")
print(x.head())
print("\n")
# Select target data
y=data['Outcome']
print("=====Target=====")
print(y.head())
print("\n")


# Use train-test-split to split the data into training data and testing data
from sklearn.model_selection import train_test_split
# implementing train-test-split
```

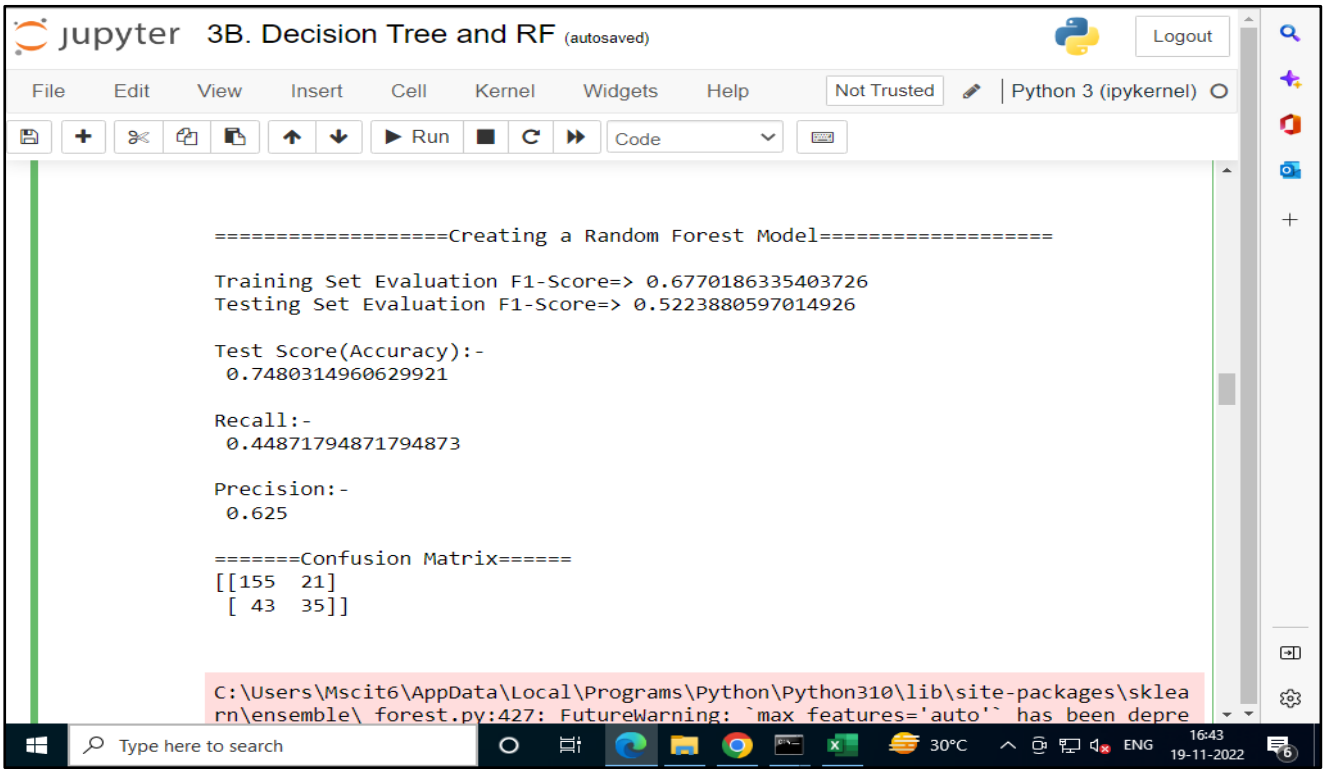
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=66)

# Import cross_val_score, classification_report, and confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score,recall_score, precision_score

print('\n====================Creating a Random Forest
Model====================\n')
# random forest model creation
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100,
                             max_depth=3,
                             max_features='auto',
                             min_samples_leaf=4,
                             bootstrap=True,
                             n_jobs=-1,
                             random_state=0)

rfc.fit(X_train,y_train)
# predictions
rfc_pred_train = rfc.predict(X_train) #for training prediction
print('Training Set Evaluation F1-Score=>',f1_score(y_train,rfc_pred_train))

# generating predictions on the test set
rfc_pred_test = rfc.predict(X_test)  #for testing prediction
print('Testing Set Evaluation F1-Score=>',f1_score(y_test,rfc_pred_test))
print("\nTest Score(Accuracy):-\n", accuracy_score(y_test,rfc_pred_test))
print("\nRecall:-\n", recall_score(y_test,rfc_pred_test))
print("\nPrecision:-\n", precision_score(y_test,rfc_pred_test))

# Confusion matrix
```

```
cm=np.array(confusion_matrix(y_test,rfc_pred_test))
print("\n=======Confusion Matrix======")
print(cm)
print("\n")
# Cross-validation
rfc_cv_score = cross_val_score(rfc, x, y, cv=10, scoring='roc_auc')
print("======Cross-validation for Random Forest======")
print( rfc_cv_score)
print("\n")
# Classification report
cr=classification_report(y_test,rfc_pred_test)
print("======Classification report======")
print(cr)
print("\n")
#Visualise Random Forest
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
rfc.estimators_[0]
fig = plt.figure(figsize=(15, 10))
plot_tree(rfc.estimators_[0],
        feature_names=x.columns,
        filled=True, impurity=True,
        rounded=True)

print('\n==================Creating a Decision Tree
Model==================\n')
# decision tree model creation
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion="gini", max_depth=3,#Max height of tree
                    min_samples_leaf=3, #Maximum leafsamples
                    random_state=100)
dtc.fit(X_train,y_train)
```

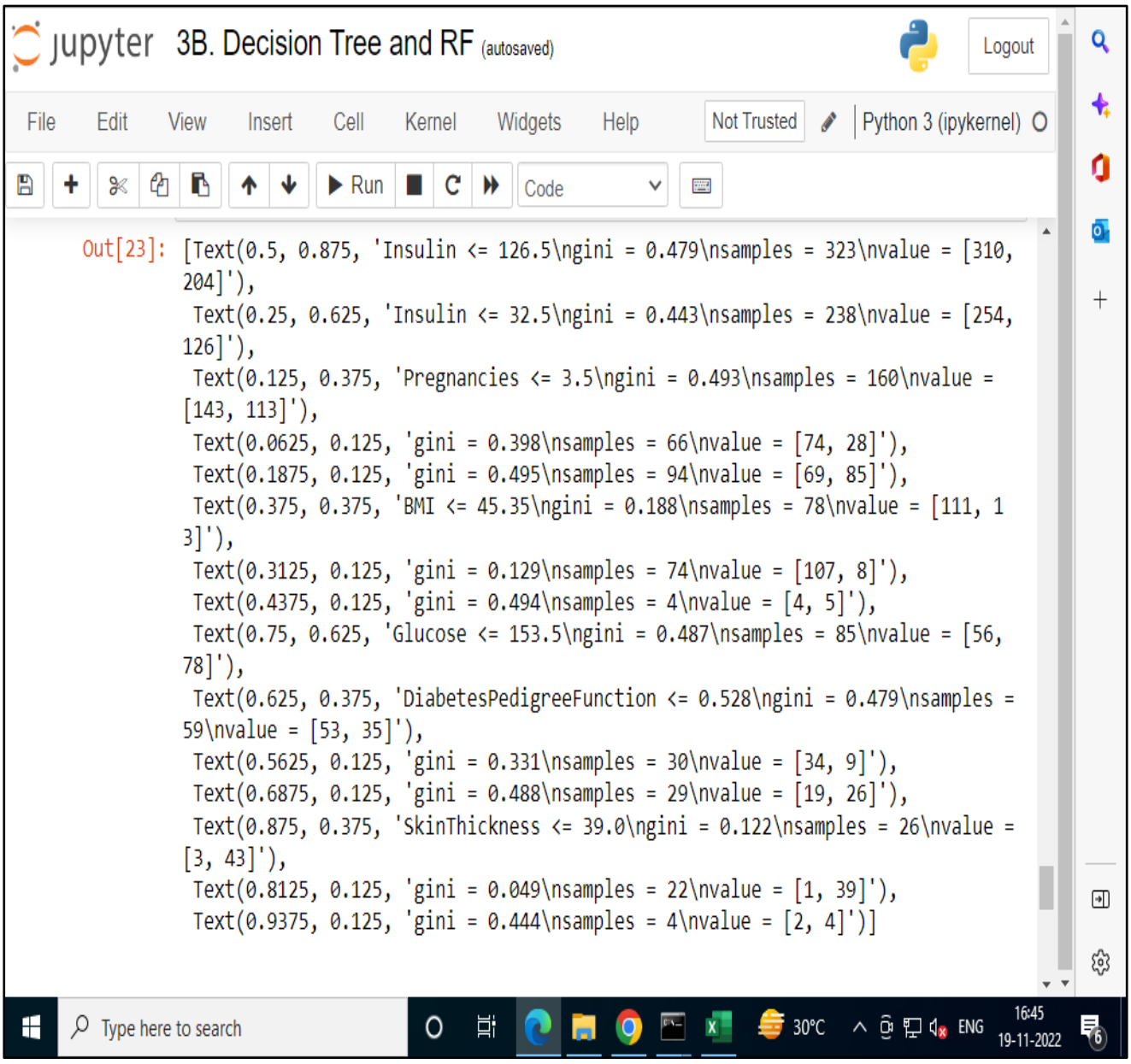
```
# predictions
dtc_pred_train = dtc.predict(X_train)
print('Training Set Evaluation F1-Score=>',f1_score(y_train,dtc_pred_train))
# generating predictions on the test set
dtc_pred_test = dtc.predict(X_test)
print('Testing Set Evaluation F1-Score=>',f1_score(y_test,dtc_pred_test))
print("\nTest Score(Accuracy):-\n", accuracy_score(y_test,dtc_pred_test))
print("\nRecall:-\n", recall_score(y_test,dtc_pred_test))
print("\nPrecision:-\n", precision_score(y_test,dtc_pred_test))
# Confusion matrix
cm=confusion_matrix(y_test,dtc_pred_test)
print("\n=======Confusion Matrix======")
print(cm)
print("\n")
# Cross-validation
dtc_cv_score = cross_val_score(dtc, x, y, cv=10, scoring='roc_auc')
print("======Cross-validation for Random Forest======")
print( dtc_cv_score)
print("\n")
# Classification report
cr=classification_report(y_test,dtc_pred_test)
print("======Classification report======")
print(cr)
print("\n")

# Visualizing decision tree
from sklearn import tree
tree.plot_tree(dtc)
```

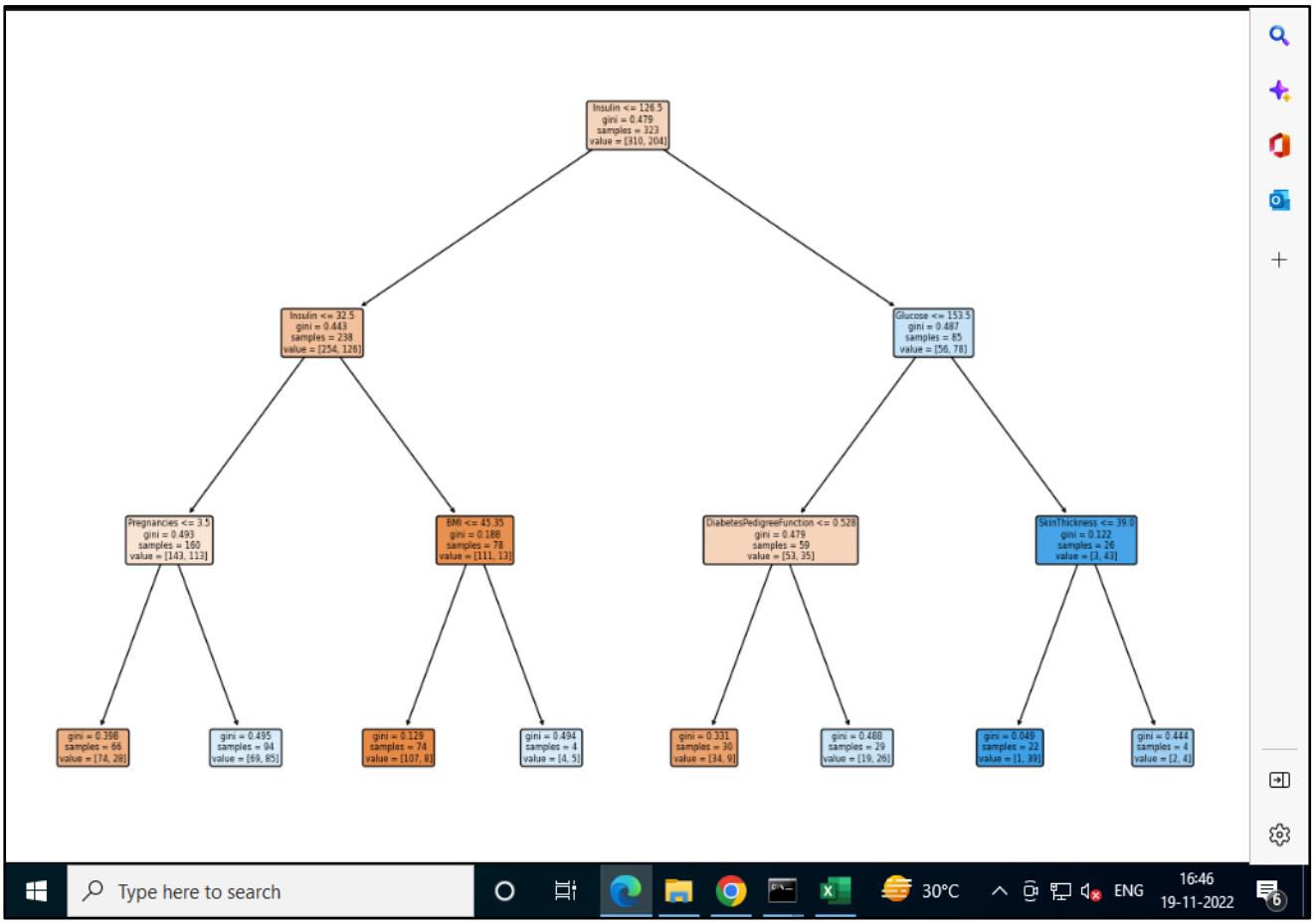
Output:

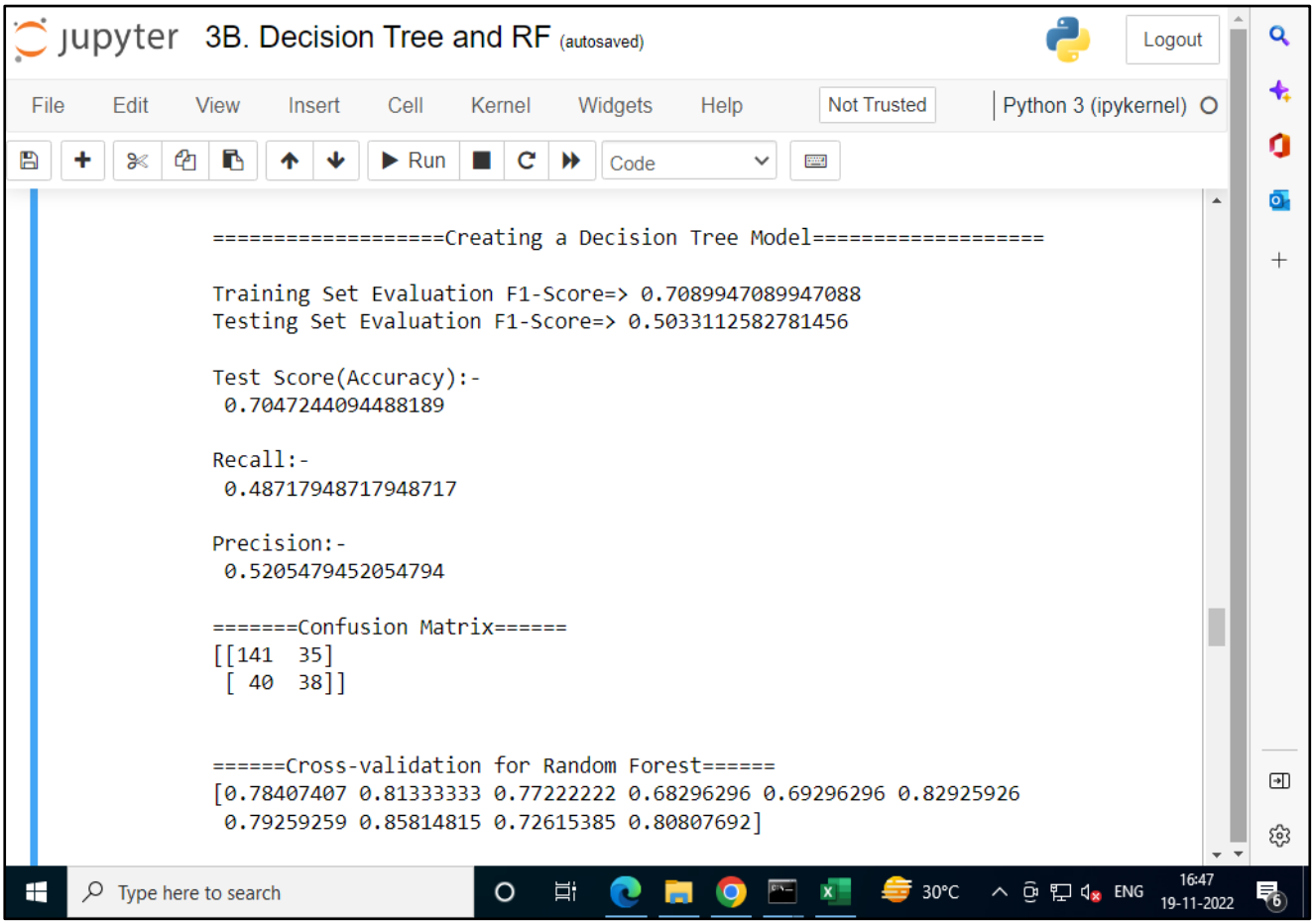

jupyter 3B. Decision Tree and RF (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
----------Implement the Random Forest and Decision Tree classifier---------
-----
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College Name: Vivek College of Commerce
M.Sc(I.T.)[Sem 3]
Date and Time: 19/11/2022 16:34:43
======Features=======
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \
0 6 148 72 35 0 33.6
1 1 85 66 29 0 26.6
2 8 183 64 0 0 23.3
3 1 89 66 23 94 28.1
4 0 137 40 35 168 43.1
DiabetesPedigreeFunction Age
0 0.627 50
1 0.351 31
2 0.672 32
3 0.167 21
4 2.288 33
=====Target=====
0 1
1 0
2 1
3 0
4 1
Name: Outcome, dtype: int64
pe here to search
30°C
ENG
16:42
19-11-2022


jupyter 3B. Decision Tree and RF (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
==========Creating a Random Forest Model==========
Training Set Evaluation F1-Score=> 0.6770186335403726
Testing Set Evaluation F1-Score=> 0.5223880597014926
Test Score(Accuracy):-
0.7480314960629921
Recall:-
0.44871794871794873
Precision:-
0.625
=======Confusion Matrix======
[[155 21]
[43 35]]
C:\Users\Mscit6\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\ensemble_forest.py:427: FutureWarning: `max_features='auto'` has been depre
Type here to search
30°C
ENG
16:43
19-11-2022

```
======Cross-validation for Random Forest======
[0.80518519 0.80740741 0.8437037  0.69037037 0.8        0.87407407
 0.84592593 0.88962963 0.83153846 0.85       ]


======Classification report======
              precision    recall  f1-score   support

           0       0.78      0.88      0.83       176
           1       0.62      0.45      0.52        78

    accuracy                           0.75       254
   macro avg       0.70      0.66      0.68       254
weighted avg       0.73      0.75      0.73       254
```




Insulin <= 126.5
gini = 0.479
samples = 323
value = [310, 204]
Insulin <= 32.5
gini = 0.443
samples = 238
value = [254, 126]
Glucose <= 153.5
gini = 0.487
samples = 85
value = [56, 78]
Pregnancies <= 3.5
gini = 0.493
samples = 160
value = [143, 113]
BMI <= 45.35
gini = 0.188
samples = 78
value = [111, 13]
DiabetesPedigreeFunction <= 0.528
gini = 0.479
samples = 59
value = [53, 35]
SkinThickness <= 39.0
gini = 0.122
samples = 26
value = [3, 43]
gini = 0.398
samples = 66
value = [74, 28]
gini = 0.495
samples = 94
value = [69, 85]
gini = 0.129
samples = 74
value = [107, 8]
gini = 0.494
samples = 4
value = [4, 5]
gini = 0.331
samples = 30
value = [34, 9]
gini = 0.488
samples = 29
value = [19, 26]
gini = 0.049
samples = 22
value = [1, 39]
gini = 0.444
samples = 4
value = [2, 4]
Type here to search
30°C
ENG
16:46
19-11-2022


jupyter 3B. Decision Tree and RF (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
==========Creating a Decision Tree Model==========
Training Set Evaluation F1-Score=> 0.7089947089947088
Testing Set Evaluation F1-Score=> 0.5033112582781456
Test Score(Accuracy):-
0.7047244094488189
Recall:-
0.48717948717948717
Precision:-
0.5205479452054794
=======Confusion Matrix======
[[141 35]
[40 38]]
======Cross-validation for Random Forest======
[0.78407407 0.81333333 0.77222222 0.68296296 0.69296296 0.82925926
0.79259259 0.85814815 0.72615385 0.80807692]
Type here to search
30°C
ENG
16:47
19-11-2022

jupyter 3B. Decision Tree and RF Last Checkpoint: 2 hours ago (autosaved)

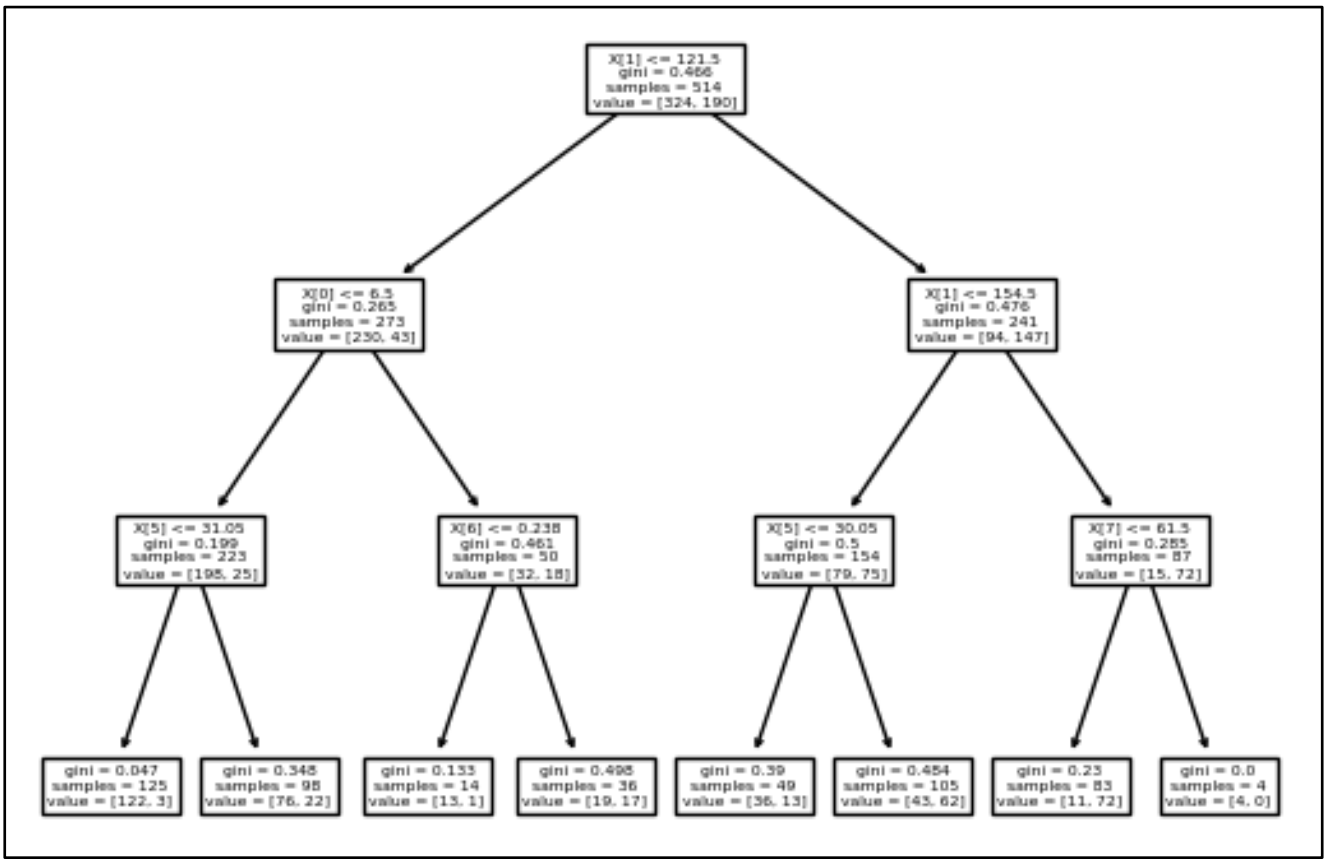
File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

```
======Cross-validation for Random Forest======
[0.78407407 0.81333333 0.77222222 0.68296296 0.69296296 0.82925926
 0.79259259 0.85814815 0.72615385 0.80807692]


======Classification report======
              precision    recall  f1-score   support

           0       0.78      0.80      0.79       176
           1       0.52      0.49      0.50        78

    accuracy                           0.70       254
   macro avg       0.65      0.64      0.65       254
weighted avg       0.70      0.70      0.70       254
```

Out[25]:

```
[Text(0.5, 0.875, 'X[1] <= 121.5\ngini = 0.466\nsamples = 514\nvalue = [324, 190]'),
 Text(0.25, 0.625, 'X[0] <= 6.5\ngini = 0.265\nsamples = 273\nvalue = [230, 43]'),
 Text(0.125, 0.375, 'X[5] <= 31.05\ngini = 0.199\nsamples = 223\nvalue = [198, 25]'),
 Text(0.0625, 0.125, 'gini = 0.047\nsamples = 125\nvalue = [122, 3]'),
 Text(0.1875, 0.125, 'gini = 0.348\nsamples = 98\nvalue = [76, 22]'),
 Text(0.375, 0.375, 'X[6] <= 0.238\ngini = 0.461\nsamples = 50\nvalue = [32, 18]'),
 Text(0.3125, 0.125, 'gini = 0.133\nsamples = 14\nvalue = [13, 1]'),
 Text(0.4375, 0.125, 'gini = 0.498\nsamples = 36\nvalue = [19, 17]'),
 Text(0.75, 0.625, 'X[1] <= 154.5\ngini = 0.476\nsamples = 241\nvalue = [94, 147]'),
 Text(0.625, 0.375, 'X[5] <= 30.05\ngini = 0.5\nsamples = 154\nvalue = [79, 75]'),
 Text(0.5625, 0.125, 'gini = 0.39\nsamples = 49\nvalue = [36, 13]'),
 Text(0.6875, 0.125, 'gini = 0.484\nsamples = 105\nvalue = [43, 62]'),
 Text(0.875, 0.375, 'X[7] <= 61.5\ngini = 0.285\nsamples = 87\nvalue = [15, 72]'),
 Text(0.8125, 0.125, 'gini = 0.23\nsamples = 83\nvalue = [11, 72]'),
 Text(0.9375, 0.125, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]')]
```



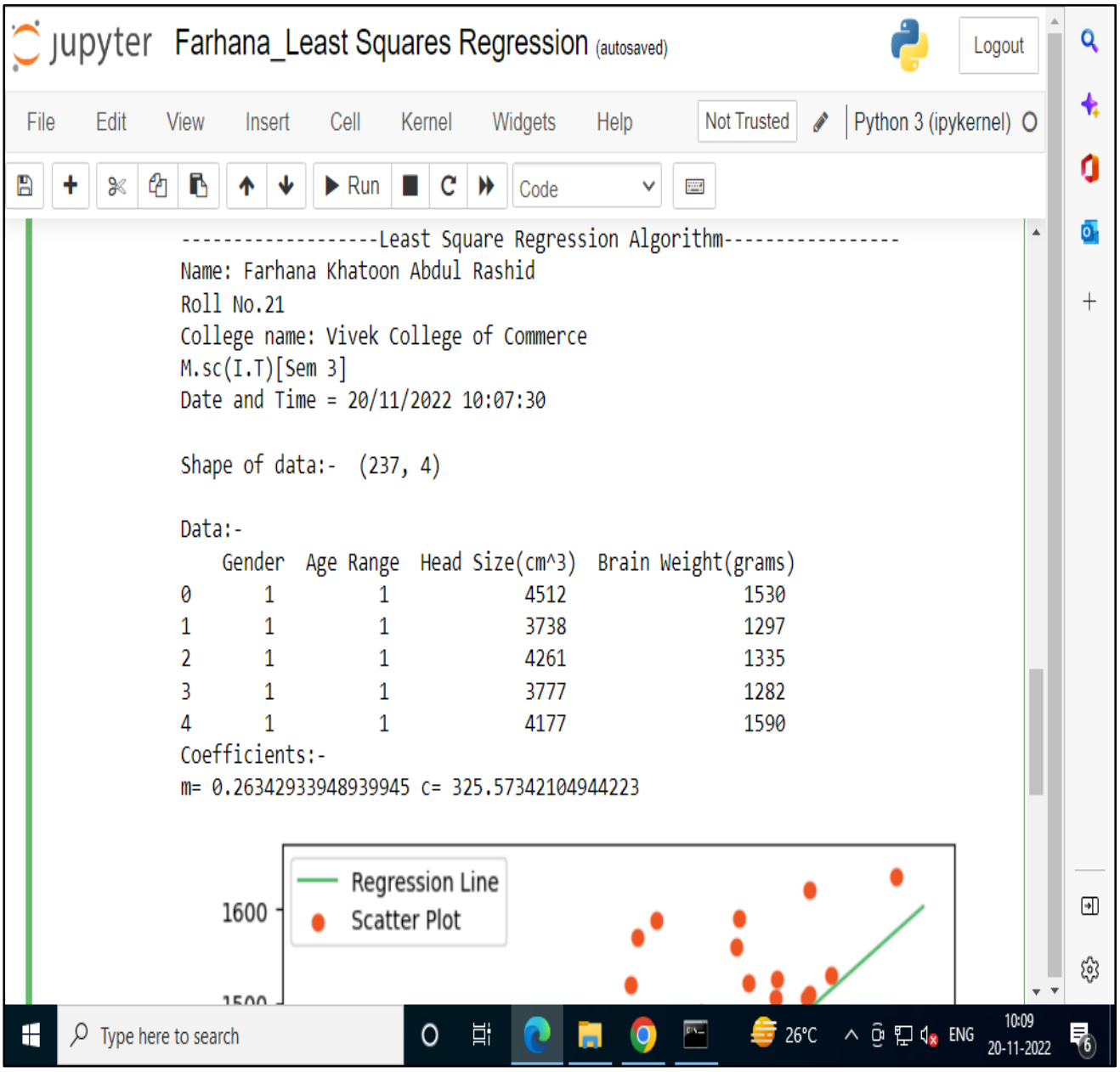
Source Code:

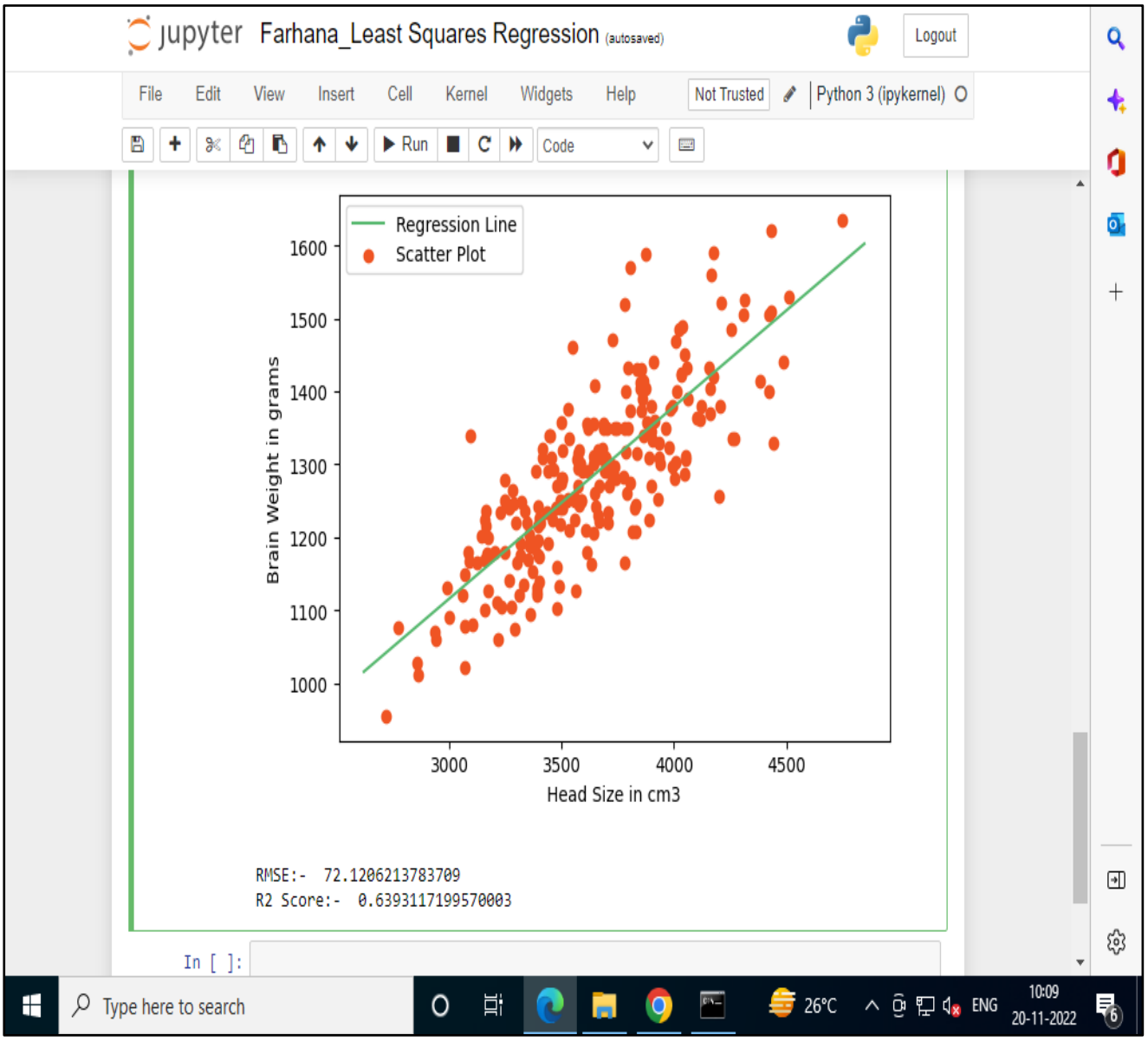
```
from datetime import datetime
print('------------------Least Square Regression Algorithm----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Import the data set
# Reading Data
data = pd.read_csv('headbrain.csv')
print("\nShape of data:- ", data.shape)
(237, 4)
print("\nData:- \n",data.head())
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = data['Head Size(cm^3)'].values
Y = data['Brain Weight(grams)'].values
# Mean X and Y
mean_x = np.mean(X)
mean_y = np.mean(Y)
# Total number of values
n = len(X)
#Calculate the values of the slope and y-intercept
#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=",m, "c=",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('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
#Model Evaluation
# Calculating Root Mean Squares Error
rmse = 0
for i in range(n):
    y_pred = c + m * X[i]
    rmse += (Y[i] - y_pred) ** 2
rmse = np.sqrt(rmse/n)
print("\nRMSE:- ", 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:- ", r2)
```

Output:-




jupyter Farhana_Least Squares Regression (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Regression Line
Scatter Plot
Brain Weight in grams
Head Size in cm3
1600
1500
1400
1300
1200
1100
1000
3000
3500
4000
4500
RMSE:- 72.1206213783709
R2 Score:- 0.6393117199570003
In [ ]:
Type here to search
26°C
ENG
10:09
20-11-2022

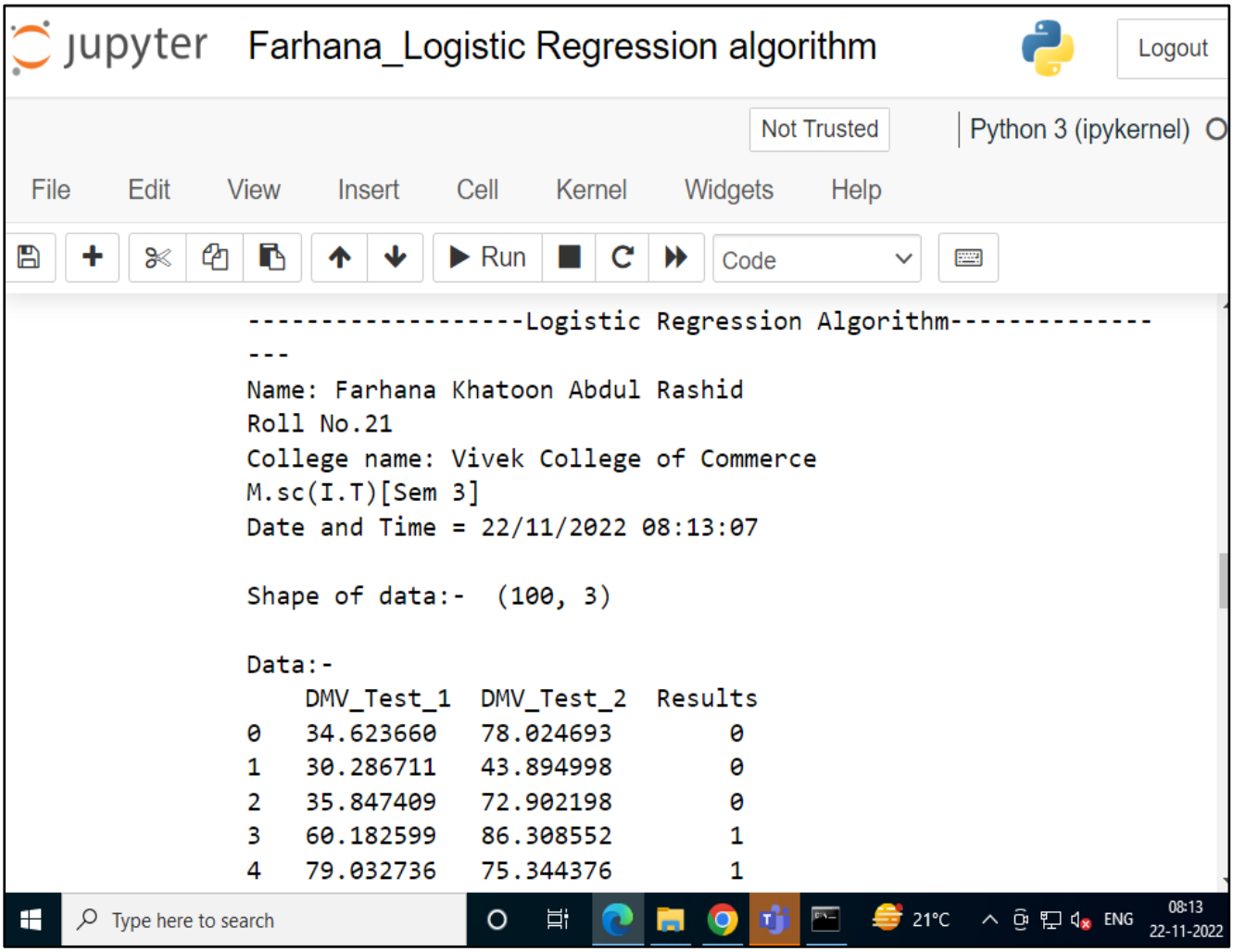
Source Code:

```
from datetime import datetime
print('------------------Least Square Regression Algorithm----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Import the data set
# Reading Data
dataset = pd.read_csv('DMVWrittenTests.csv')
print("\nShape of data:- ", dataset.shape)
(237, 4)
print("\nData:- \n",dataset.head())
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
# Computing X and Y
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
#Splitting the dataset into the Training set 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
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
#Training the Logistic Regression model on the Training Set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
#Predicting the Test set results
y_pred = classifier.predict(X_test)
print("\nPrediction on Test set:-", y_pred)
#Comparing the Real Values with Predicted Values
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(df.head())
#Confusion Matrix, Accuracy, Recall, Precision, and Classification report
from sklearn.metrics import confusion_matrix, recall_score, precision_score,
classification_report
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:-\n", cm)
from sklearn.metrics import accuracy_score
print ("\nAccuracy:- ", accuracy_score(y_test, y_pred))
print("\nRecall:-\n", recall_score(y_test,y_pred))
print("\nPrecision:-\n", precision_score(y_test,y_pred))
cr=classification_report(y_test,y_pred)
print("======Classification report======")
print(cr)
print("\n")
#Visualising the 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('Logistic Regression')
plt.xlabel('DMV_Test_1')
plt.ylabel('DMV_Test_2')
plt.legend()
plt.show()
```

Output:



jupyter Farhana_Logistic Regression algorithm (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
Prediction on Test set:- [1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 0]
   Real Values  Predicted Values
0            1                 1
1            0                 0
2            0                 0
3            0                 0
4            1                 1

Confusion Matrix:-
 [[11  0]
 [ 3 11]]

Accuracy:-  0.88

Recall:-
 0.7857142857142857

Precision:-
 1.0
======Classification report======
              precision    recall  f1-score   support

           0       0.79      1.00      0.88        11
           1       1.00      0.79      0.88        14

    accuracy                           0.88        25
   macro avg       0.89      0.89      0.88        25
weighted avg       0.91      0.88      0.88        25
```

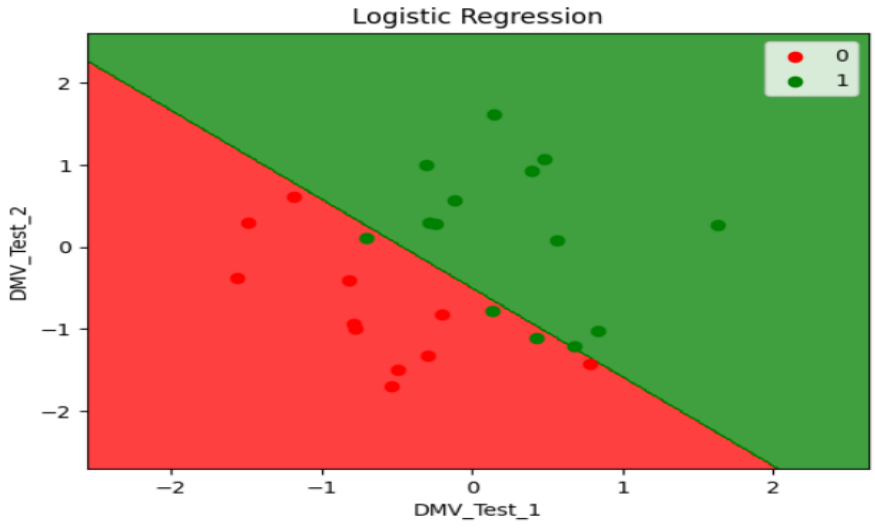
Type here to search 21°C ENG 08:14 22-11-2022

jupyter Farhana_Logistic Regression algorithm (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code



Type here to search 21°C ENG 08:15 22-11-2022

Source Code:

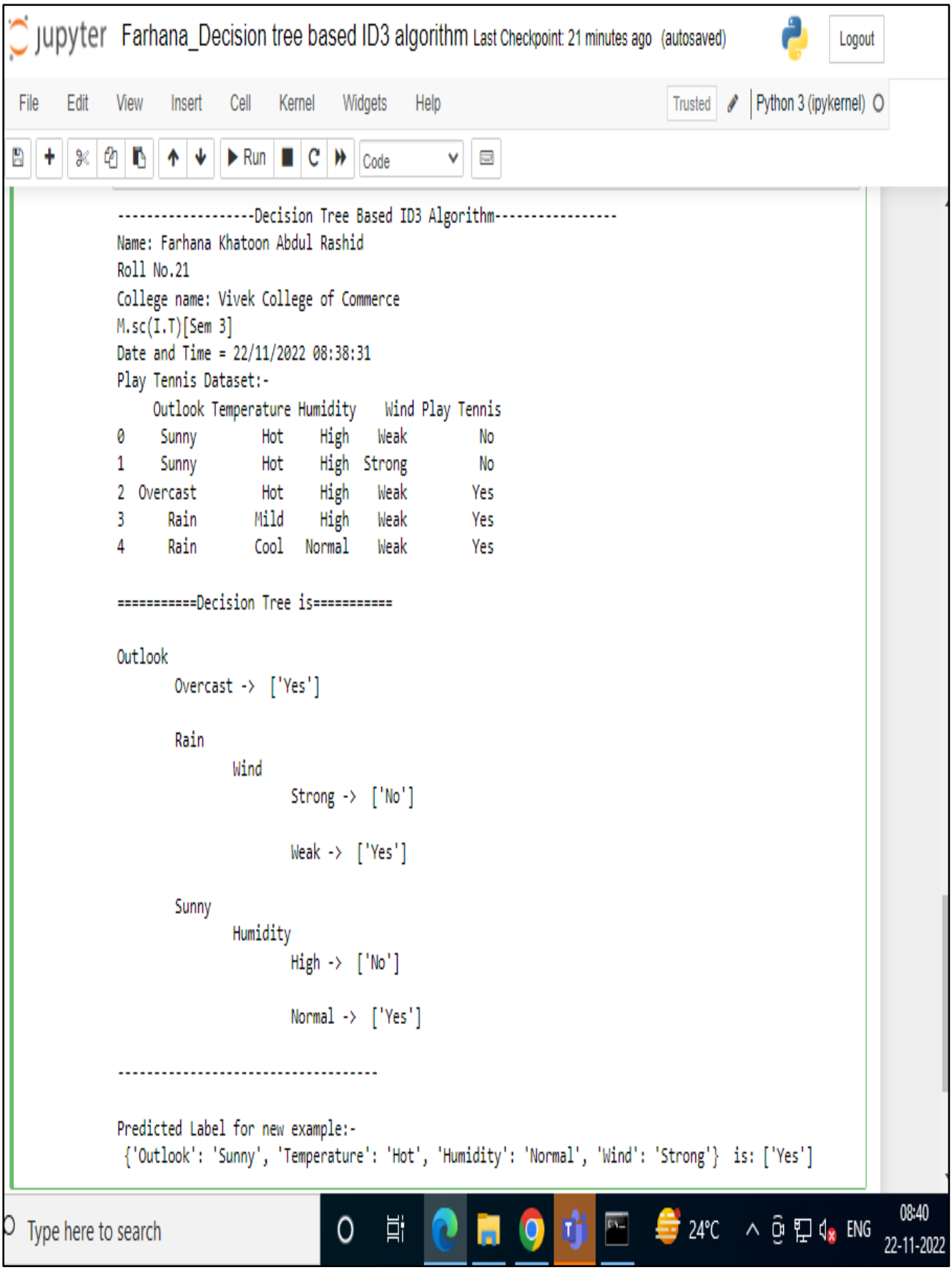
```
from datetime import datetime
print('------------------Decision Tree Based ID3 Algorithm----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import pandas as pd
import math
import numpy as np
# Import the data set
# Reading Data
data = pd.read_csv("PlayTennis.csv")
print("Play Tennis Dataset:-\n", data.head())
#Extract features from dataset
features = [feat for feat in data]
features.remove("Play Tennis")
#Create a class named Node with four members children, value, isLeaf and pred.
class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""


#Define a function called entropy to find the entropy of the dataset.
def entropy(examples):
```

```
    pos = 0.0
    neg = 0.0
    for _, row in examples.iterrows():
        if row["Play Tennis"] == "Yes":
            pos += 1
        else:
            neg += 1
    if pos == 0.0 or neg == 0.0:
        return 0.0
    else:
        p = pos / (pos + neg)
        n = neg / (pos + neg)
        return -(p * math.log(p, 2) + n * math.log(n, 2))

#Define a function named info_gain to find the gain of the attribute
def info_gain(examples, attr):
    uniq = np.unique(examples[attr])
    #print ("\n",uniq)
    gain = entropy(examples)
    #print ("\n",gain)
    for u in uniq:
        subdata = examples[examples[attr] == u]
        #print ("\n",subdata)
        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
        #print ("\n",gain)
    return gain

#Define a function named ID3 to get the decision tree for the given dataset
def ID3(examples, attrs):
    root = Node()
```

```
    max_gain = 0
    max_feat = ""
    for feature in attrs:
        #print ("\n",examples)
        gain = info_gain(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat
    #print ("\nMax feature attr",max_feat)
    uniq = np.unique(examples[max_feat])
    #print ("\n",uniq)
    for u in uniq:
        #print ("\n",u)
        subdata = examples[examples[max_feat] == u]
        #print ("\n",subdata)
        if entropy(subdata) == 0.0:
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["Play Tennis"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)
    return root
```

```
#Define a function named printTree to draw the decision tree
def printTree(root: Node, depth=0):
    for i in range(depth):
        print("\t", end="")
    print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    print()
    for child in root.children:
        printTree(child, depth + 1)

#Define a function named classify to classify the new example
def classify(root: Node, new):
    for child in root.children:
        if child.value == new[root.value]:
            if child.isLeaf:
                print ("Predicted Label for new example:-\n", new," is:", child.pred)
                exit
            else:
                classify (child.children[0], new)

#Finally, call the ID3, printTree and classify functions
root = ID3(data, features)
print("\n===========Decision Tree is===========\n")
printTree(root)
print ("----------------------------------\n")
new = {"Outlook":"Sunny", "Temperature":"Hot", "Humidity":"Normal", "Wind":"Strong"}
classify (root, new)
```

Output:


jupyter Farhana_Decision tree based ID3 algorithm Last Checkpoint: 21 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
----------Decision Tree Based ID3 Algorithm----------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 22/11/2022 08:38:31
Play Tennis Dataset:-
Outlook Temperature Humidity Wind Play Tennis
0 Sunny Hot High Weak No
1 Sunny Hot High Strong No
2 Overcast Hot High Weak Yes
3 Rain Mild High Weak Yes
4 Rain Cool Normal Weak Yes
==========Decision Tree is==========
Outlook
Overcast -> ['Yes']
Rain
Wind
Strong -> ['No']
Weak -> ['Yes']
Sunny
Humidity
High -> ['No']
Normal -> ['Yes']

Predicted Label for new example:-
{'Outlook': 'Sunny', 'Temperature': 'Hot', 'Humidity': 'Normal', 'Wind': 'Strong'} is: ['Yes']
Type here to search
24°C
ENG
08:40
22-11-2022

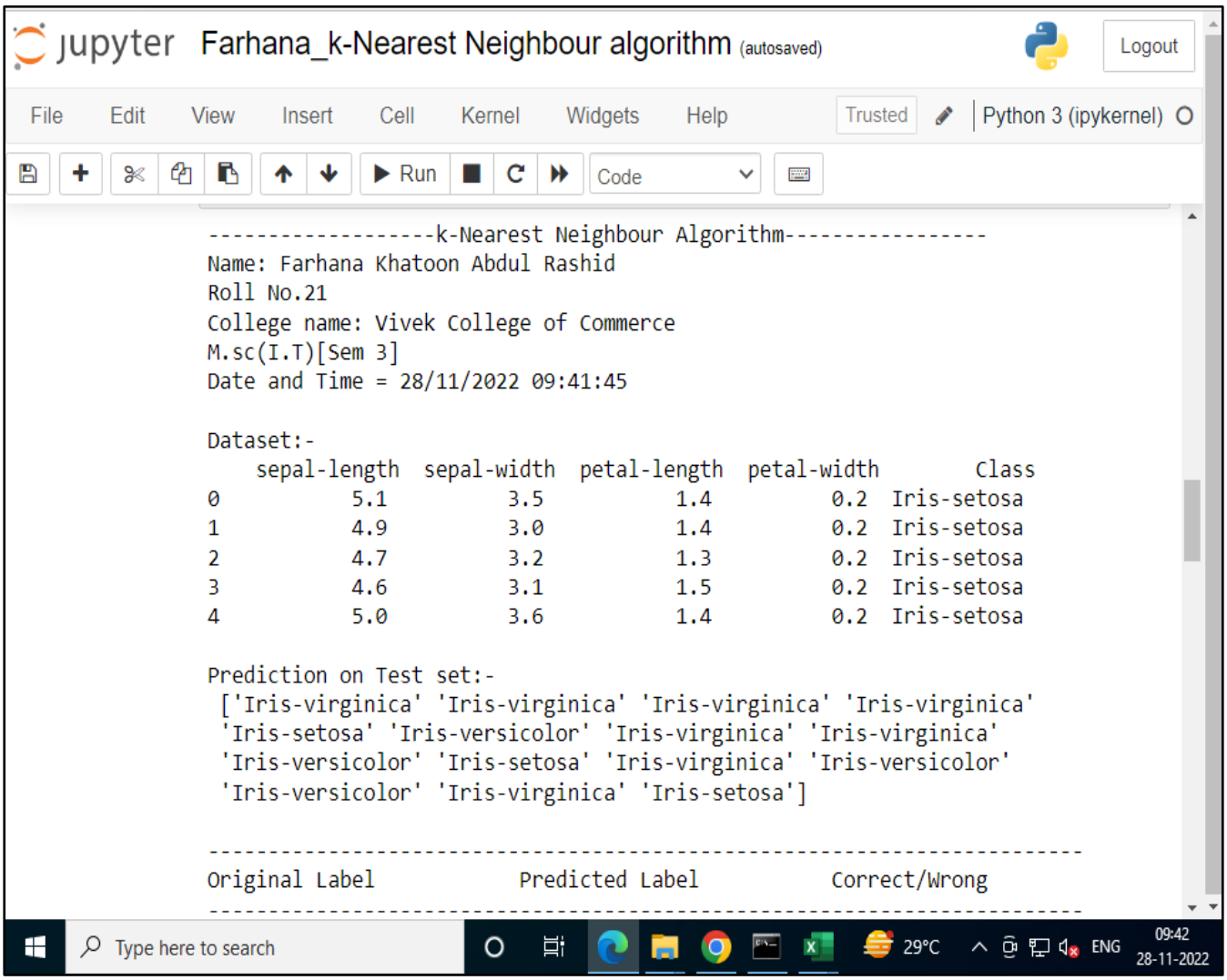
Source Code:

```
from datetime import datetime
print('------------------k-Nearest Neighbour Algorithm----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
import numpy as np
import pandas as pd
from sklearn import metrics
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Import the data set
# Read dataset to pandas dataframe
dataset = pd.read_csv("Iris.csv", names=names)
print("\nDataset:-\n", dataset.head())
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.10)
#Training the K-Nearest Neighbour model on the Training Set
from sklearn.neighbors import KNeighborsClassifier
# Instantiate learning model (k = 5)
classifier = KNeighborsClassifier(n_neighbors=5).fit(Xtrain, ytrain)
#Predicting the Test set results
ypred = classifier.predict(Xtest)
print("\nPrediction on Test set:-\n", ypred)
```

```
i = 0
print ("\n-------------------------------------------------------------------------")
print ('%-25s %-25s %-25s' % ('Original Label', 'Predicted Label', 'Correct/Wrong'))
print ("-------------------------------------------------------------------------")
for label in ytest:
    print ('%-25s %-25s' % (label, ypred[i]), end="")
    if (label == ypred[i]):
        print (' %-25s' % ('Correct'))
    else:
        print (' %-25s' % ('Wrong'))
    i = i + 1
print ("-------------------------------------------------------------------------")
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print("\nClassification Report:\n",metrics.classification_report(ytest, ypred))
print('Accuracy of the classifier is %0.2f' % metrics.accuracy_score(ytest,ypred))
# finding best k
# creating list of K for KNN
k_list = list(range(1,50,2))
# creating list of cv scores
cv_scores = []
# perform 10-fold cross validation
from sklearn.model_selection import cross_val_score
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, Xtrain, ytrain, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())
# changing to misclassification error
MSE = [1 - x for x in cv_scores]
best_k = k_list[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d." % best_k)
#Visualize optimal number of neighbors
import matplotlib.pyplot as plt
```

```
import seaborn as sns
plt.figure()
plt.figure(figsize=(15,10))
plt.title('The optimal number of neighbors', fontsize=20, fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=15)
plt.ylabel('Misclassification Error', fontsize=15)
sns.set_style("whitegrid")
plt.plot(k_list, MSE)
plt.show()
#Pairplot
plt.figure()
sns.pairplot(dataset,hue = "Class", size=3, markers=["o", "s", "D"])
plt.show()
```

Output:



jupyter Farhana_k-Nearest Neighbour algorithm (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
-----------------------------------------------------------------------
Original Label          Predicted Label          Correct/Wrong
-----------------------------------------------------------------------
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-setosa             Iris-setosa              Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-setosa             Iris-setosa              Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-versicolor          Wrong
Iris-versicolor         Iris-versicolor          Correct
Iris-virginica          Iris-virginica           Correct
Iris-setosa             Iris-setosa              Correct
-----------------------------------------------------------------------

Confusion Matrix:
 [[3 0 0]
 [0 3 0]
 [0 1 8]]
```

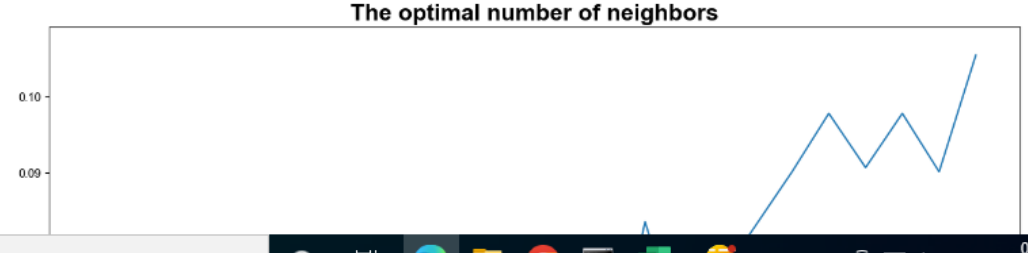
Type here to search 29°C ENG 09:43 28-11-2022

jupyter Farhana_k-Nearest Neighbour algorithm (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         3
Iris-versicolor       0.75      1.00      0.86         3
 Iris-virginica       1.00      0.89      0.94         9

       accuracy                           0.93        15
      macro avg       0.92      0.96      0.93        15
   weighted avg       0.95      0.93      0.94        15

Accuracy of the classifer is 0.93
The optimal number of neighbors is 5.

<Figure size 640x480 with 0 Axes>
```



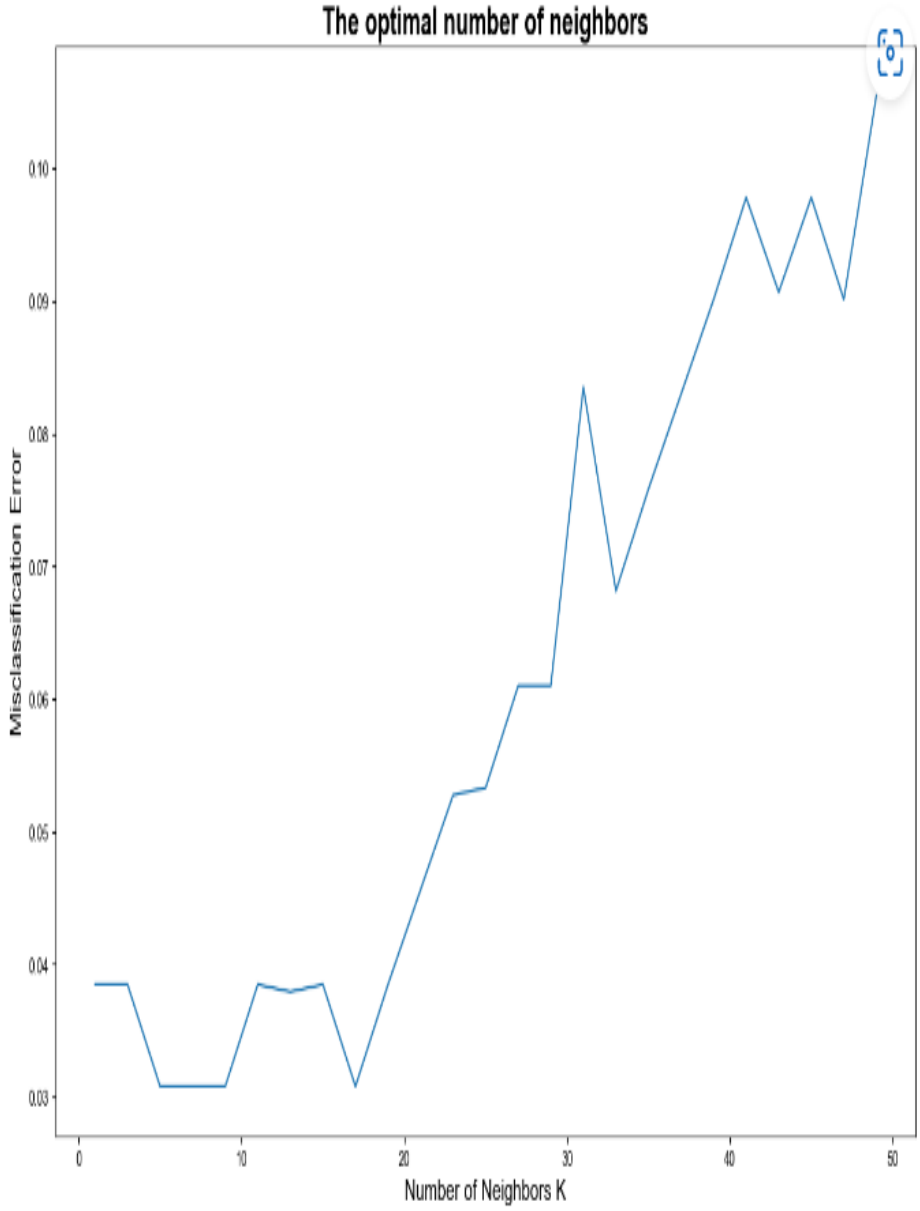
Type here to search 29°C ENG 09:43 28-11-2022

jupyter Farhana_k-Nearest Neighbour algorithm (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

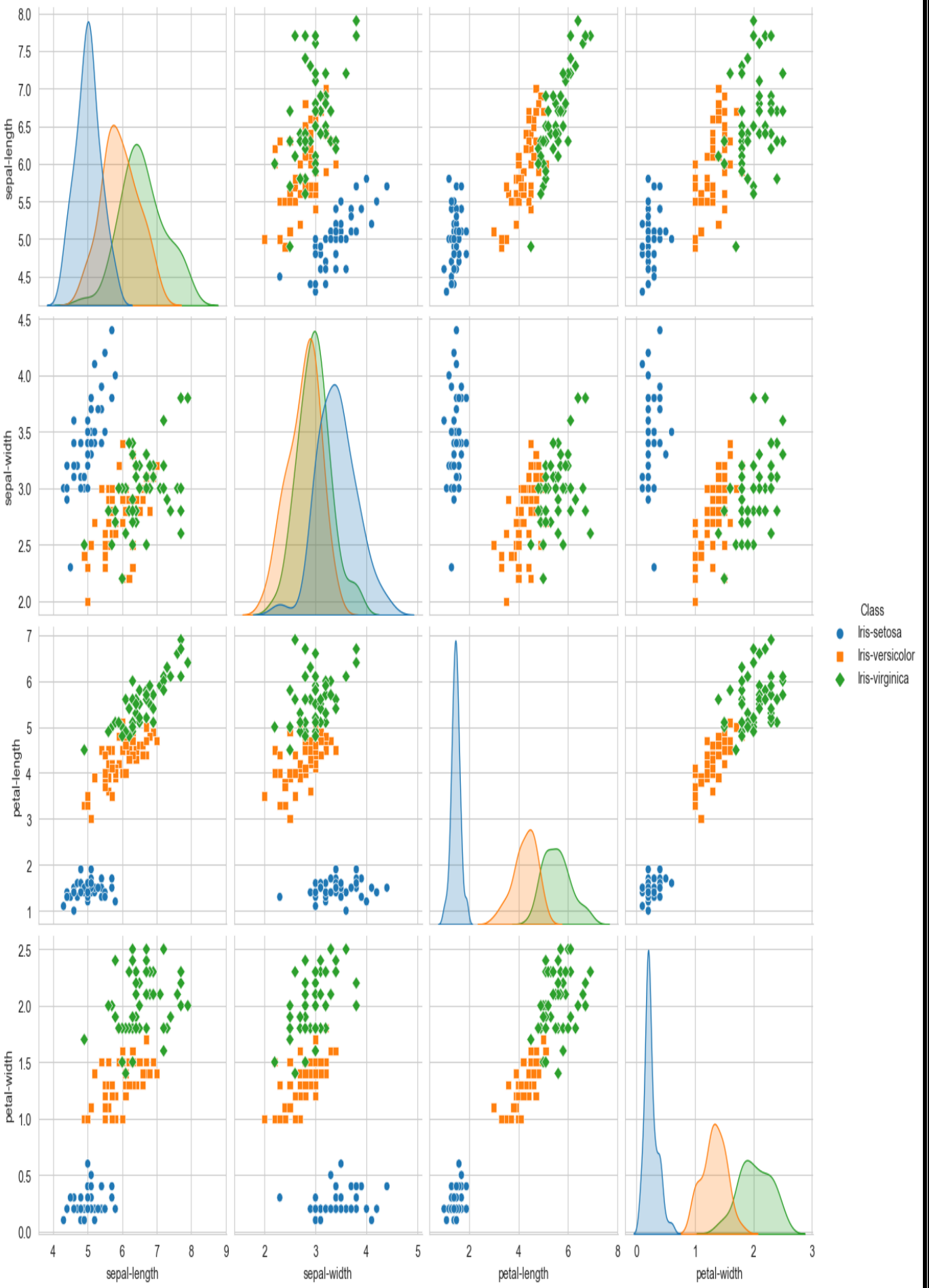
Run Code

```
<Figure size 640x480 with 0 Axes>
```



In [2]: #Pairplot

Type here to search 29°C ENG 09:45 28-11-2022


sepal-length
sepal-width
petal-length
petal-width
Class
Iris-setosa
Iris-versicolor
Iris-virginica

Source Code:

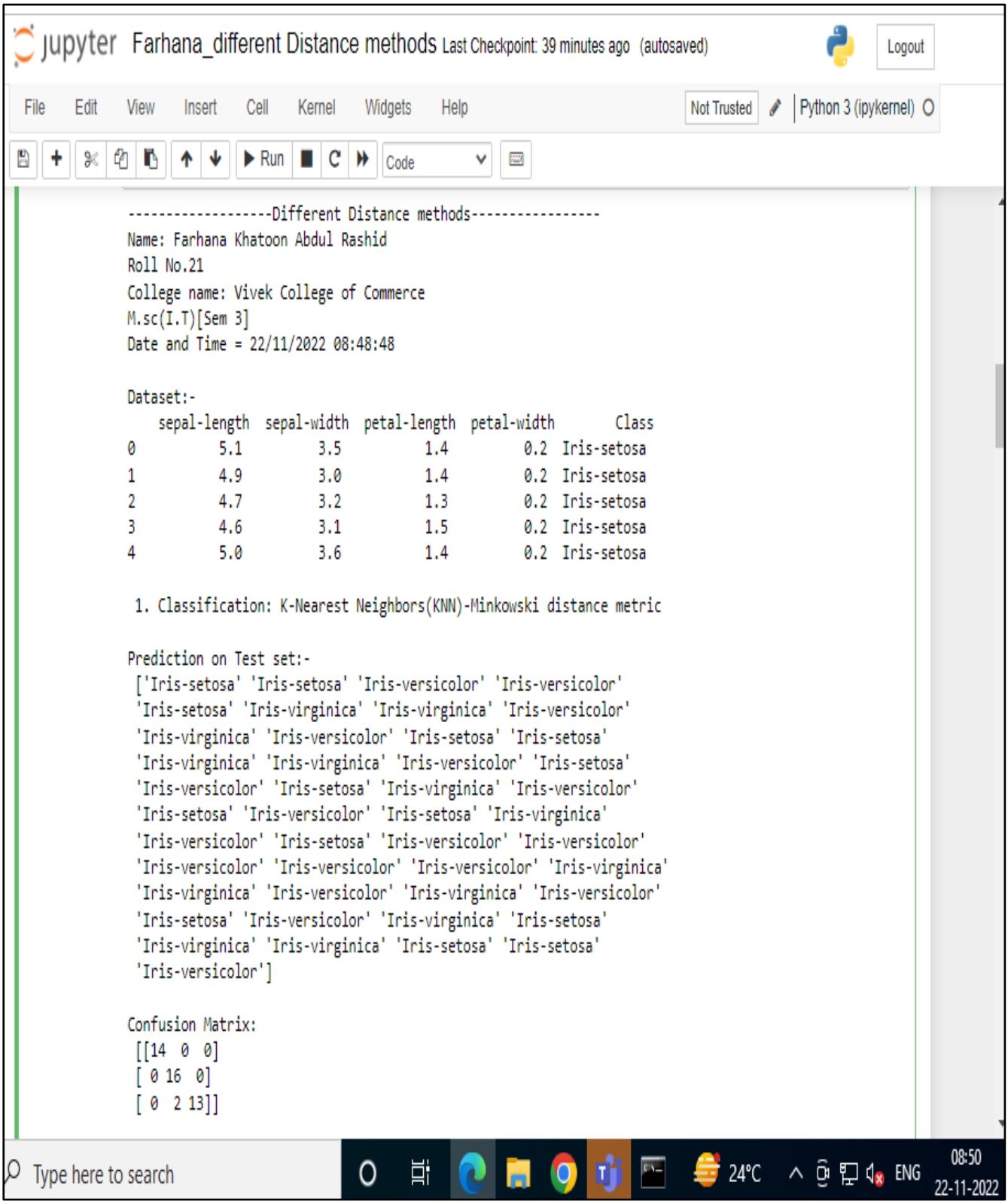
1. **KNN (Minkowski distance metric)**

```
from datetime import datetime
print('------------------Different Distance methods----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
import numpy as np
import pandas as pd
from sklearn import metrics
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Import the data set
# Read dataset to pandas dataframe
dataset = pd.read_csv("8-Irisdataset.csv", names=names)
print("\nDataset:-\n", dataset.head())
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3)
print("\n 1. Classification: K-Nearest Neighbors(KNN)-Minkowski distance metric")
#Training the K-Nearest Neighbour model on the Training Set
from sklearn.neighbors import KNeighborsClassifier
# Instantiate learning model (k = 5)
classifier = KNeighborsClassifier(n_neighbors = 6, p = 2, metric='minkowski').fit(Xtrain,
ytrain)
```

```
#Predicting the Test set results
ypred = classifier.predict(Xtest)
print("\nPrediction on Test set:-\n", ypred)
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print("\nClassification Report:\n",metrics.classification_report(ytest, ypred))
print('Accuracy of the classifer is %0.2f' % metrics.accuracy_score(ytest,ypred))
# finding best k
# creating list of K for KNN
k_list = list(range(1,50,2))
# creating list of cv scores
cv_scores = []
# perform 10-fold cross validation
from sklearn.model_selection import cross_val_score
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, Xtrain, ytrain, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())
# changing to misclassification error
MSE = [1 - x for x in cv_scores]
best_k = k_list[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d." % best_k)

from matplotlib import pyplot as plt
plt.figure()
dataset.boxplot(by="Class", figsize=(15, 10))
plt.show()
#Andrews Curves
from pandas.plotting import andrews_curves
plt.figure(figsize=(15,10))
andrews_curves(dataset, "Class")
plt.title('Andrews Curves Plot:(KNN)-Minkowski distance metric', fontsize=20,
fontweight='bold')
```

```
plt.legend(loc=1, prop={'size': 15}, frameon=True,shadow=True, facecolor="white",
edgecolor="black")
plt.show()
```

Output:

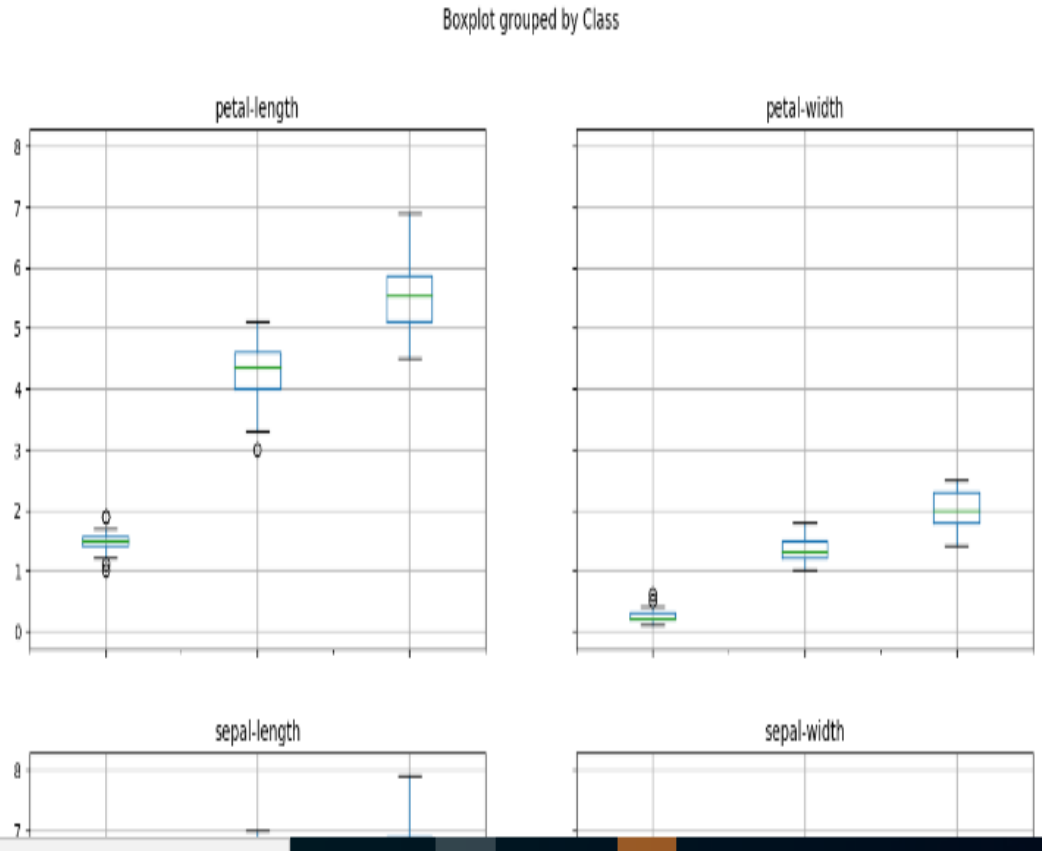


jupyter Farhana_different Distance methods Last Checkpoint: 39 minutes ago (autosaved) Logout

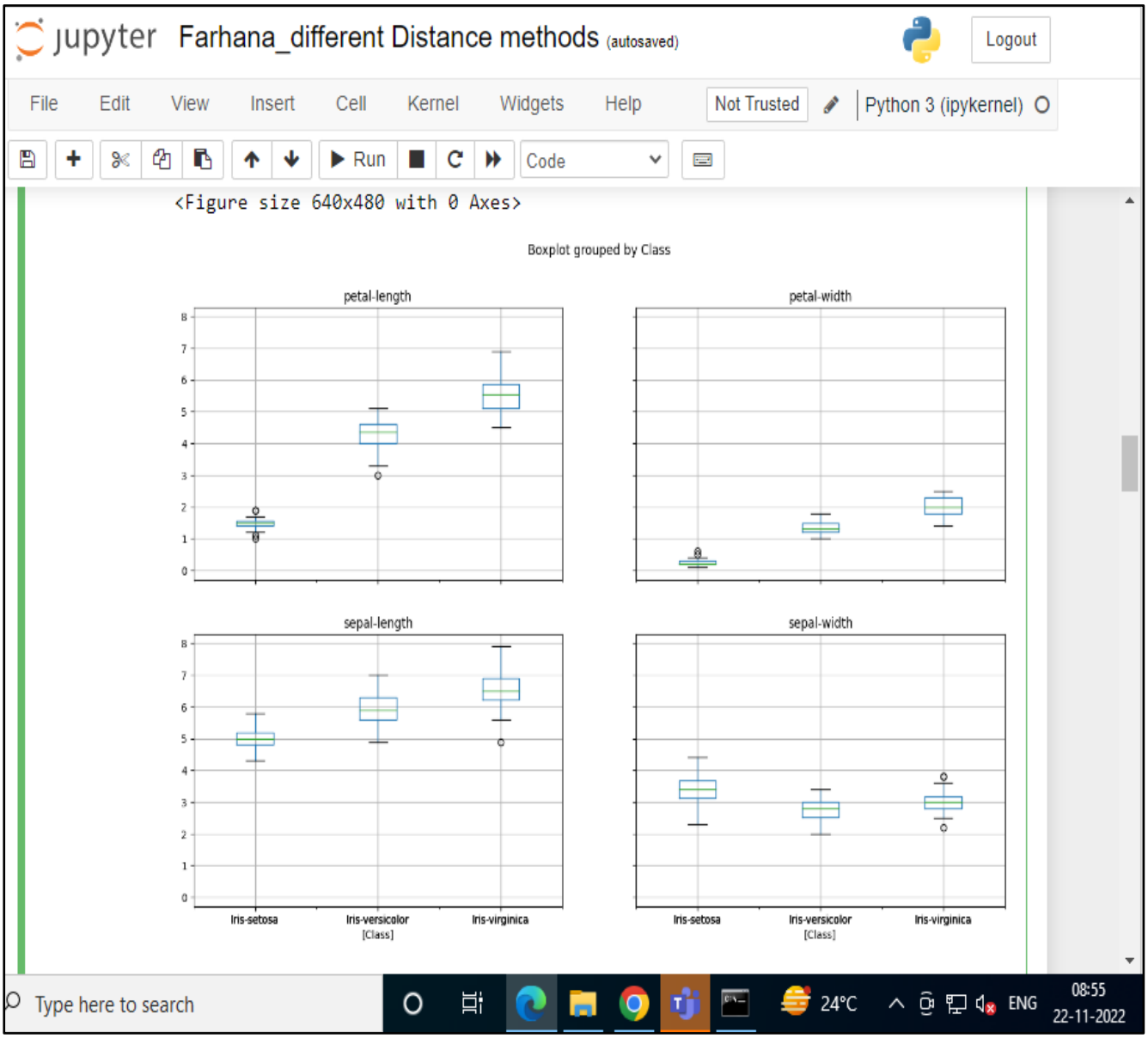
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

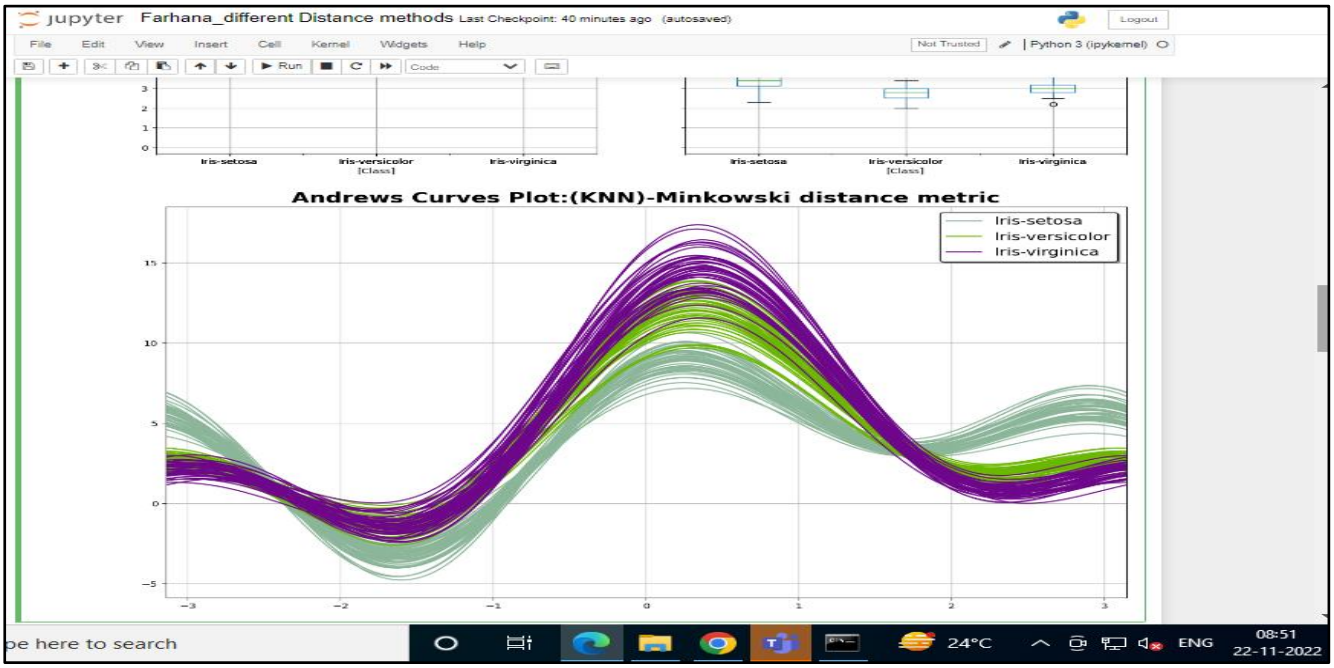
Run Code

```
Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.89      1.00      0.94        16
 Iris-virginica       1.00      0.87      0.93        15

       accuracy                           0.96        45
      macro avg       0.96      0.96      0.96        45
   weighted avg       0.96      0.96      0.96        45

Accuracy of the classifer is 0.96
The optimal number of neighbors is 9.

<Figure size 640x480 with 0 Axes>
```



Type here to search 24°C ENG 08:50 22-11-2022


jupyter Farhana_different Distance methods (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
<Figure size 640x480 with 0 Axes>
Boxplot grouped by Class
petal-length
petal-width
sepal-length
sepal-width
Iris-setosa
Iris-versicolor
Iris-virginica
[Class]
Type here to search
24°C
ENG
08:55
22-11-2022

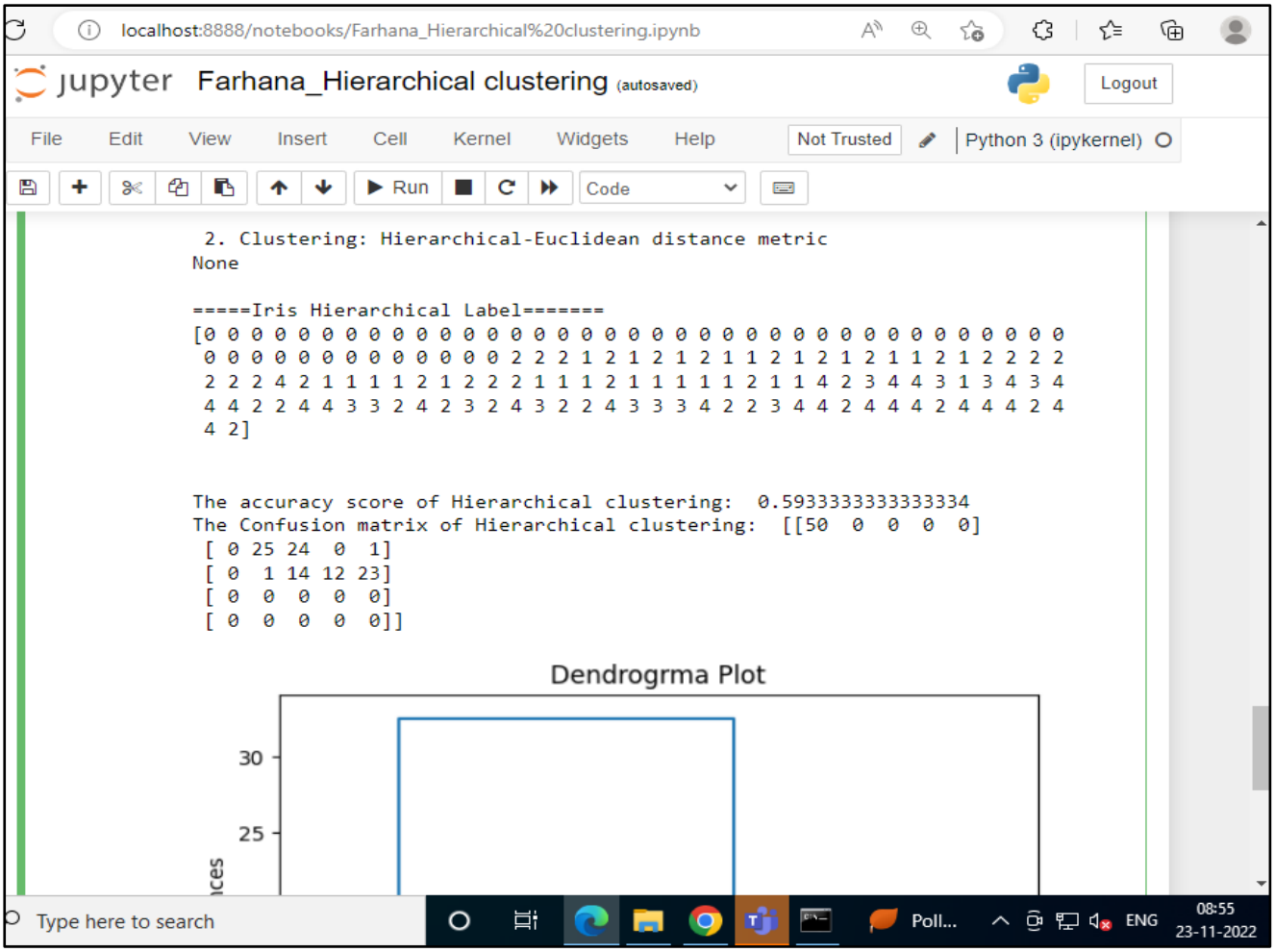

jupyter Farhana_different Distance methods Last Checkpoint: 40 minutes ago (autosaved)
Andrews Curves Plot:(KNN)-Minkowski distance metric
Iris-setosa
Iris-versicolor
Iris-virginica
24°C
ENG
08:51
22-11-2022

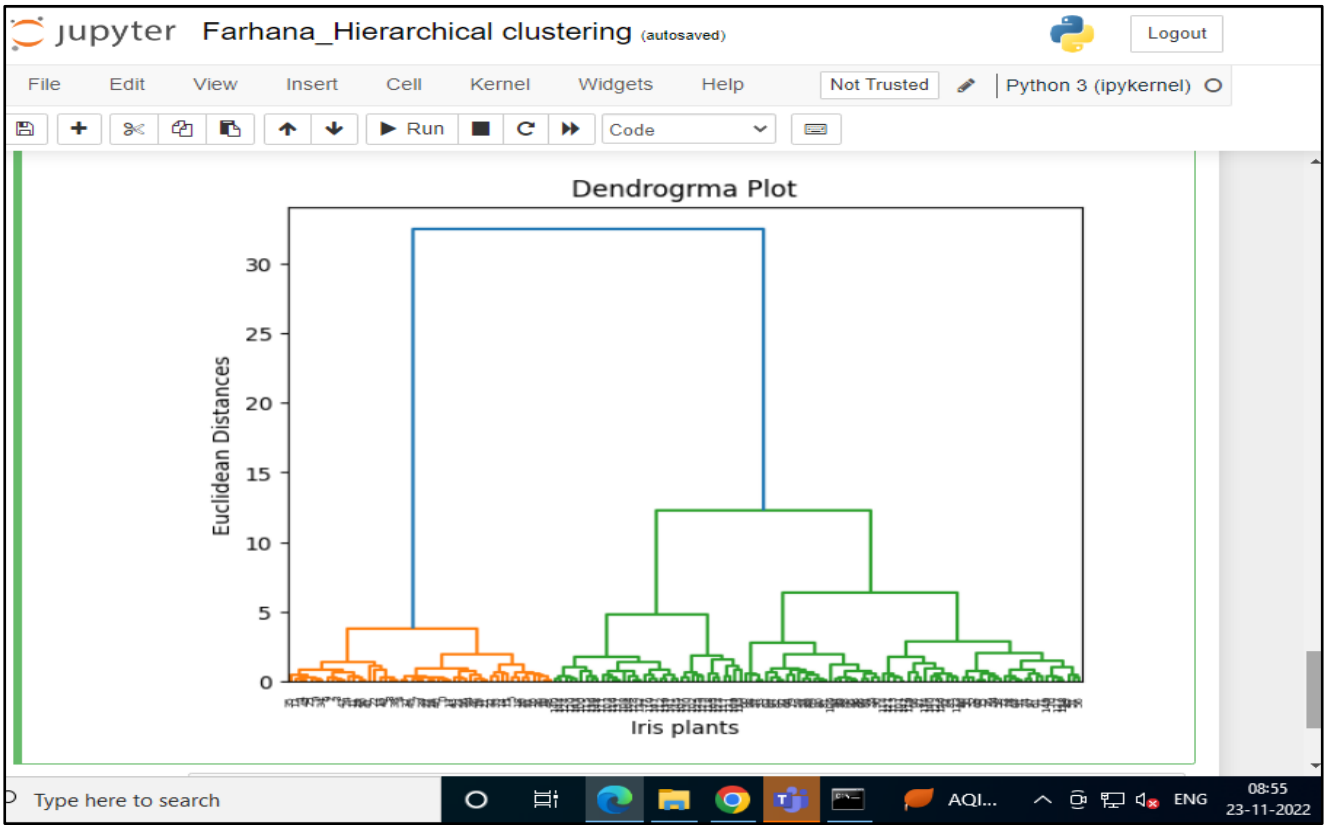
Source Code:

2. Hierarchical Clustering (Euclidean Distance)

```
print(print("\n 2. Clustering: Hierarchical-Euclidean distance metric"))
# Import the required libraries
from sklearn import datasets
import pandas as pd
from sklearn.utils import shuffle
import matplotlib.pyplot as mtp
import sklearn.metrics as sm
from matplotlib import pyplot as plt
import numpy as np
# import some data to play with
iris = datasets.load_iris()
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
#training the hierarchical model on dataset
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
y_pred= hc.fit_predict(X)

#labels that the algorithm has provided
print("\n=====Iris Hierarchical Label=======")
label=hc.labels_
print(label)
print("\n")
print('The accuracy score of Hierarchical clustering: ',sm.accuracy_score(y, hc.labels_))
print('The Confusion matrix of Hierarchical clustering: ',sm.confusion_matrix(y,
hc.labels_))

#Finding the optimal number of clusters using the Dendrogram
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(X, method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Iris plants")
mtp.show()
```

Output:


localhost:8888/notebooks/Farhana_Hierarchical%20clustering.ipynb
jupyter Farhana_Hierarchical clustering (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
2. Clustering: Hierarchical-Euclidean distance metric
None
=====Iris Hierarchical Label=======
[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 2 2 2 1 2 1 2 1 2 1 1 2 1 2 1 2 1 1 2 1 2 2 2 2
2 2 2 4 2 1 1 1 1 2 1 2 2 2 1 1 1 2 1 1 1 1 1 2 1 1 4 2 3 4 4 3 1 3 4 3 4
4 4 2 2 4 4 3 3 2 4 2 3 2 4 3 2 2 4 3 3 3 4 2 2 3 4 4 2 4 4 4 2 4 4 4 2 4
4 2]
The accuracy score of Hierarchical clustering: 0.5933333333333334
The Confusion matrix of Hierarchical clustering: [[50 0 0 0 0]
[0 25 24 0 1]
[0 1 14 12 23]
[0 0 0 0 0]
[0 0 0 0 0]]
Dendrogrma Plot
30
25
Type here to search
Poll...
ENG
08:55
23-11-2022


jupyter Farhana_Hierarchical clustering (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Dendrogrma Plot
Euclidean Distances
30
25
20
15
10
5
0
Iris plants
Type here to search
AQI...
ENG
08:55
23-11-2022

Source Code:

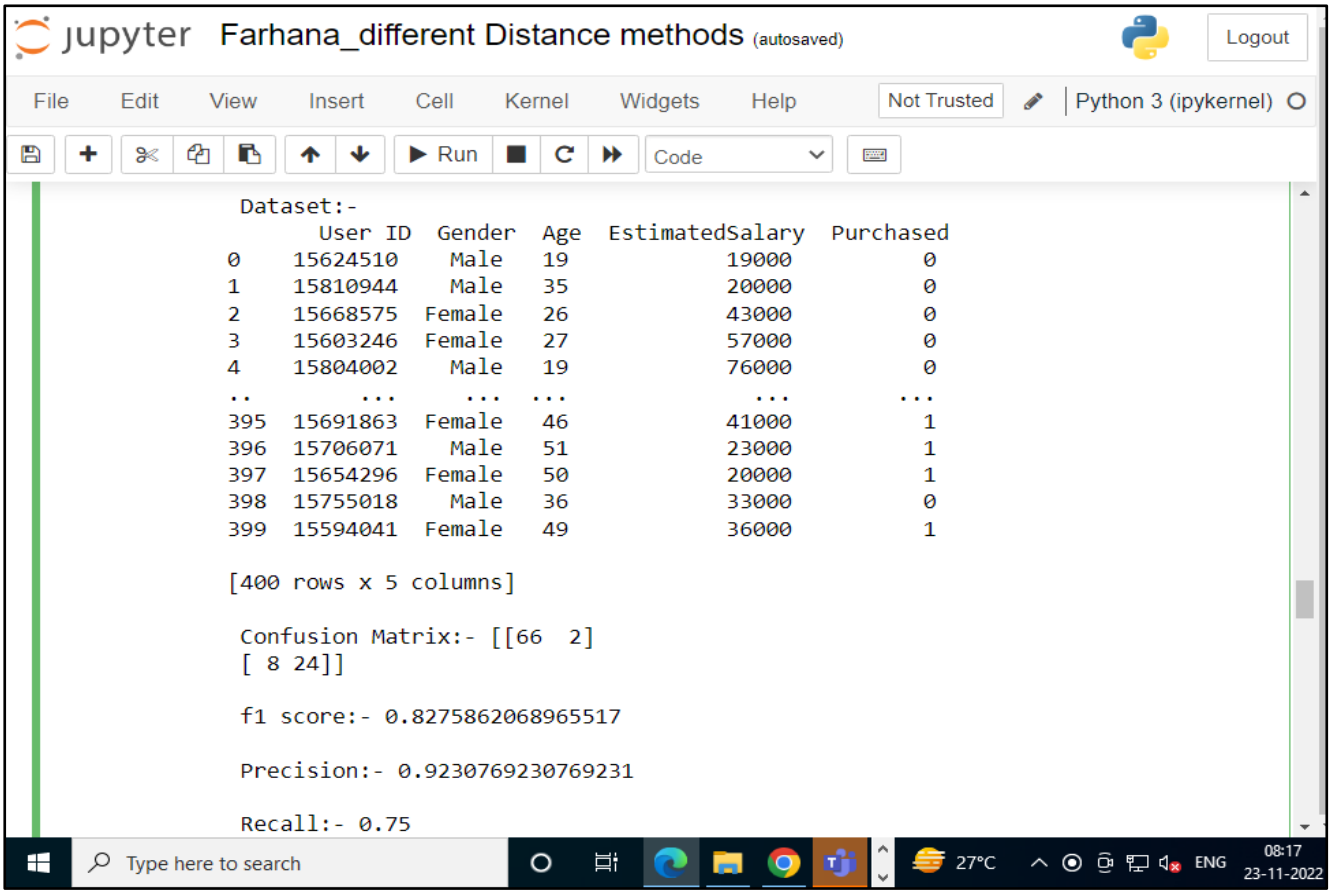
3. Manhattan Distance

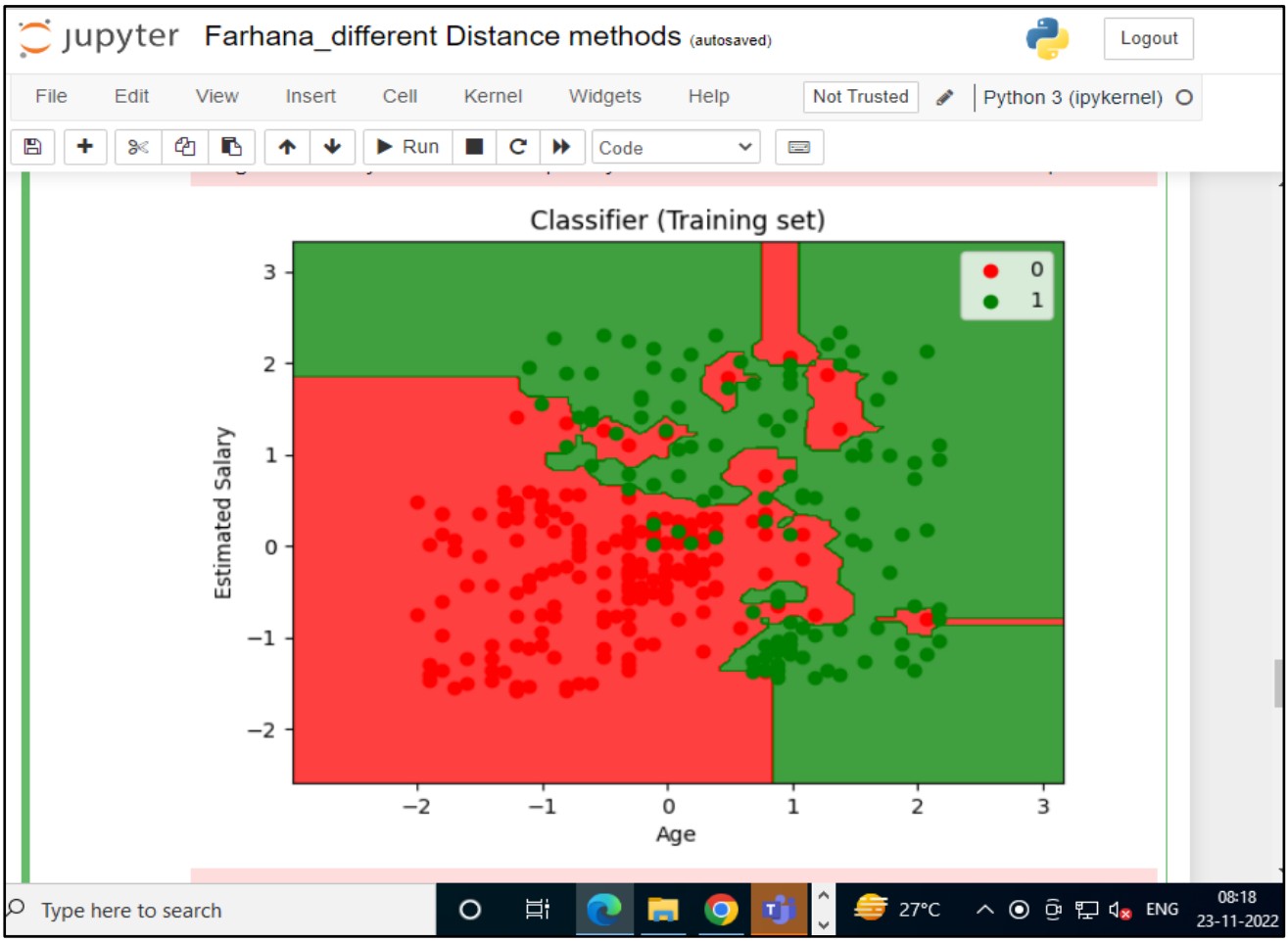
```
print("3. Classification: K-Nearest Neighbors(KNN)-Manhattan distance metric")
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
print("\n Dataset:-\n", dataset)
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
# Splitting the dataset into the Training set 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)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 2, metric='manhattan')
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, recall_score,
precision_score
cm = confusion_matrix(y_test, y_pred)
print("\n Confusion Matrix:-", cm)
f1 = f1_score(y_test, y_pred)
print("\n f1 score:-", f1)
```

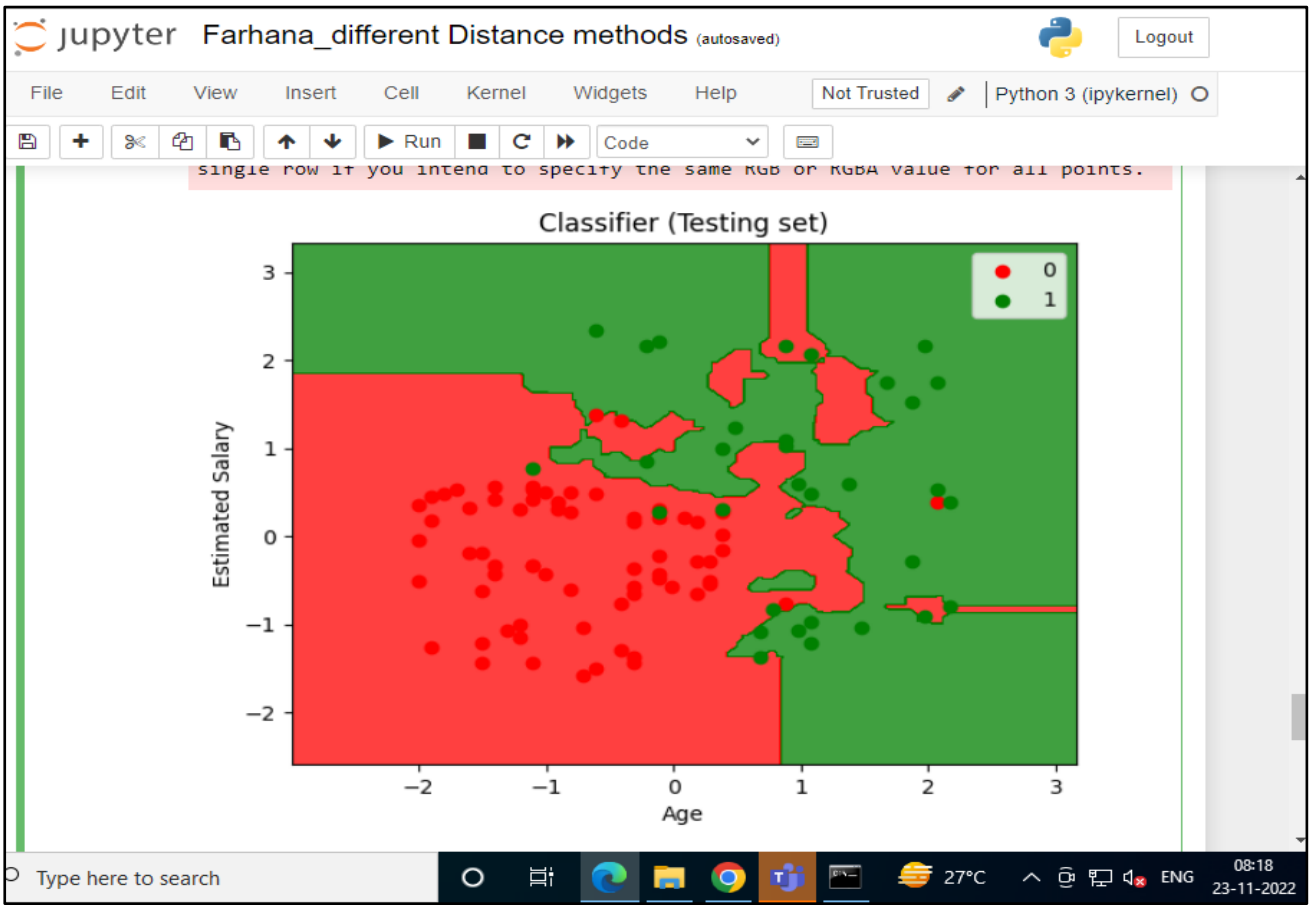
```
precision = precision_score(y_test, y_pred)
print("\n Precision:-", precision)
recall = recall_score(y_test, y_pred)
print("\n Recall:-", recall)

# 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('Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Testing set results
from matplotlib.colors import ListedColormap
X_test, y_test = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_test[:, 0].min() - 1, stop = X_test[:, 0].max() + 1,
step = 0.01),
```

```
                np.arange(start = X_test[:, 1].min() - 1, stop = X_test[:, 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_test)):
    plt.scatter(X_test[y_test == j, 0], X_test[y_test == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifier (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output:




jupyter Farhana_different Distance methods (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Classifier (Training set)
Estimated Salary
Age
0
1
Type here to search
27°C
ENG
08:18
23-11-2022

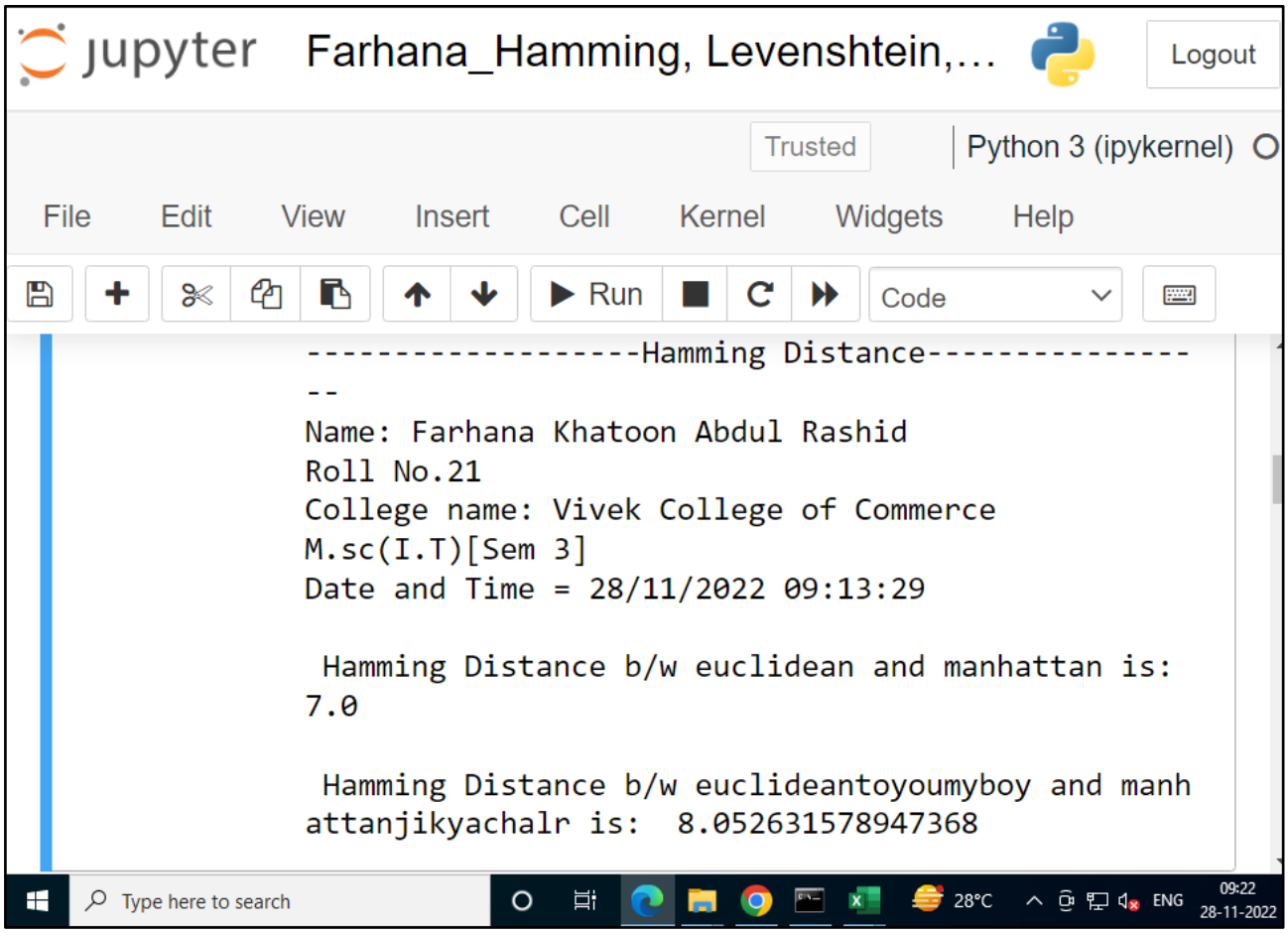

jupyter Farhana_different Distance methods (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
single row if you intend to specify the same RGB or RGBA value for all points.
Classifier (Testing set)
Estimated Salary
Age
0
1
Type here to search
27°C
ENG
08:18
23-11-2022

Source Code:

4. Hamming Distance

```
from datetime import datetime
print('------------------Hamming Distance----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)

# Hamming distance
from scipy.spatial import distance
import pandas as pd
# defining two strings
string_1 = 'euclidean'
string_2 = 'manhattan'

string_3 = 'euclideantoyoumyboy'
string_4 = 'manhattanjikyachalr'

# computing the hamming distance
hamming_distance = distance.hamming(list(string_1), list(string_2))*len(string_1)
print('\n Hamming Distance b/w', string_1, 'and', string_2, 'is: ', hamming_distance)

# computing the hamming distance
hamming_distance = distance.hamming(list(string_3), list(string_4))*len(string_1)
print('\n Hamming Distance b/w', string_3, 'and', string_4, 'is: ', hamming_distance)
```

Output:



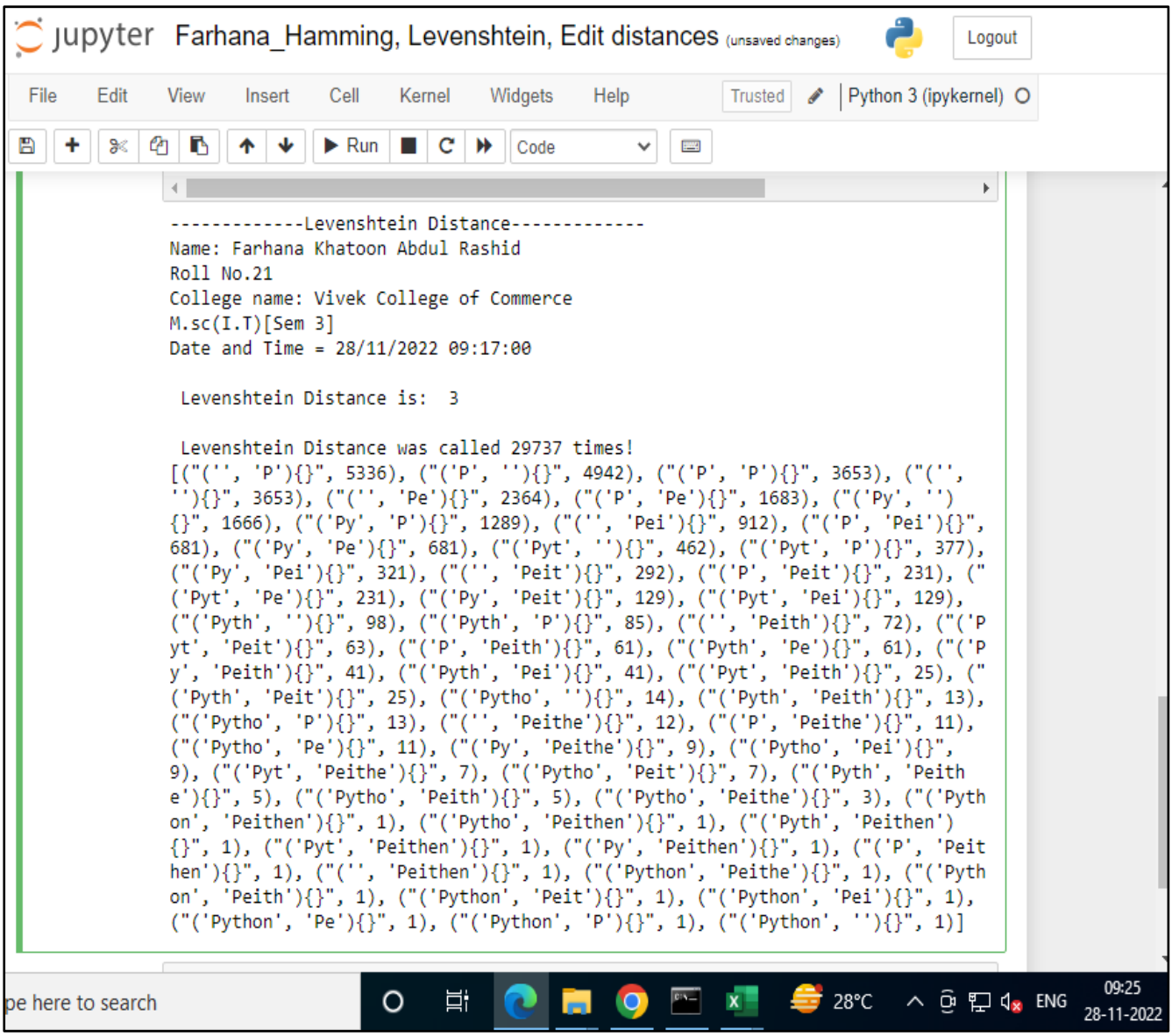
Source Code:

5. **Levenshtein Distance**

```
print('----------Levenshtein Distance----------')
from datetime import datetime
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
from collections import Counter
```

```
def call_counter(func):   #We count the number of calls by using a decorator function (recursive function)
    def helper(*args, **kwargs):
        helper.calls += 1
        key = str(args) + str(kwargs)
        helper.c[key] += 1
        return func(*args, **kwargs)
    helper.c = Counter()
    helper.calls = 0
    helper.__name__= func.__name__
    return helper
@call_counter

#Define Levenshtein Distance
def LevenshteinDistance(s, t):
    if s == "":
        return len(t)
    if t == "":
        return len(s)
    if s[-1] == t[-1]:
        cost = 0
    else:
        cost = 1

    res = min([LevenshteinDistance(s[:-1], t)+1,
               LevenshteinDistance(s, t[:-1])+1,
               LevenshteinDistance(s[:-1], t[:-1]) + cost])
    return res

print("\n Levenshtein Distance is: ", LevenshteinDistance("Python", "Peithen"))
print("\n Levenshtein Distance was called " + str(LevenshteinDistance.calls) + " times!")
print(LevenshteinDistance.c.most_common())
```

Output:



Source Code:

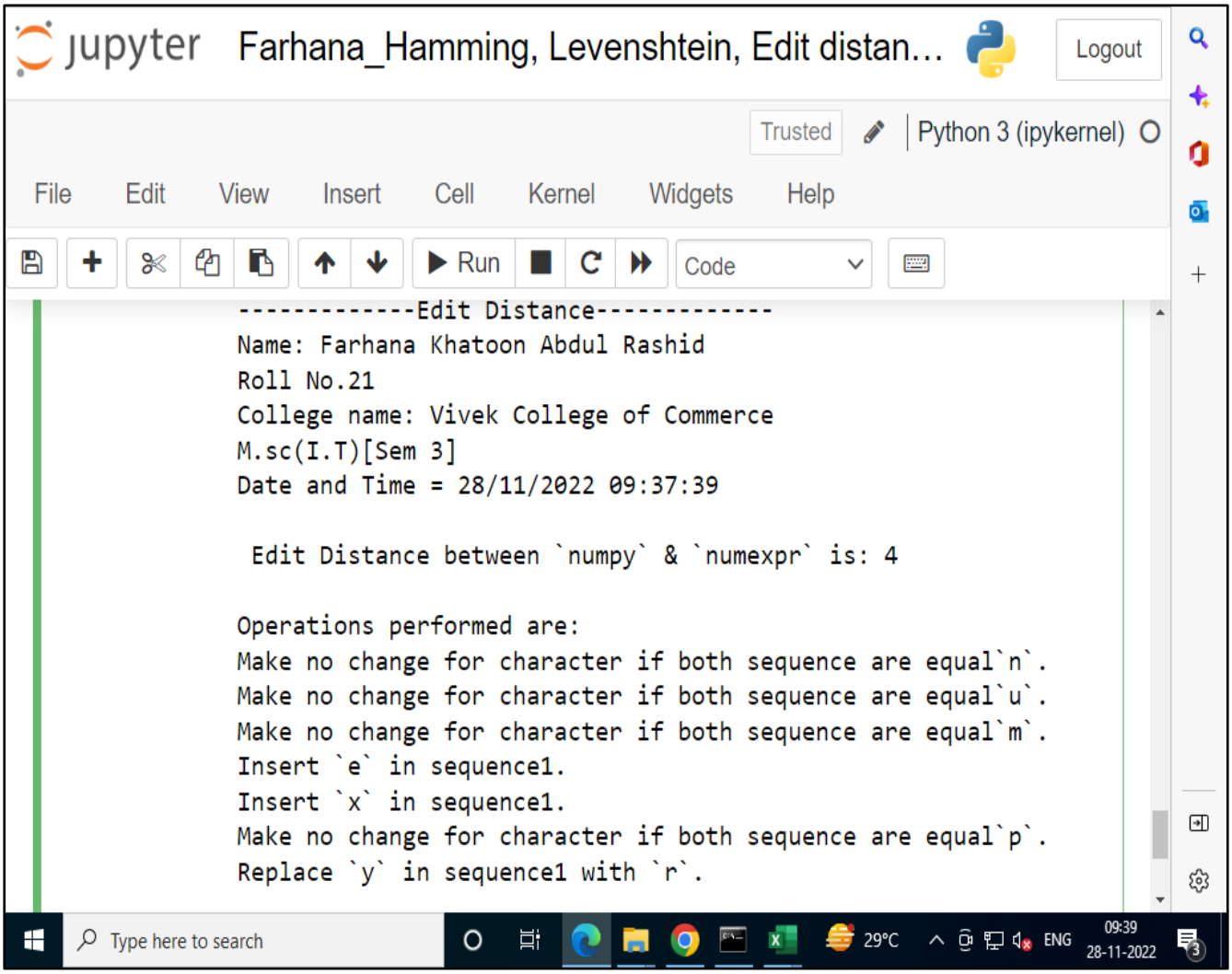
6. **Edit Distance**

```
print('----------Edit Distance----------')
from datetime import datetime
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
```

```
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
#Edit distance using Recursion
# define the costs
ins_cost = 1
del_cost = 1
sub_cost = 2
def edit_distance_recurse(seq1, seq2, operations=[]):
    """Returns the Edit Distance between the provided two sequences."""
    if len(seq2) == 0:
        operations = operations + ([f"Delete `{seq1}` from sequence1."] if len(seq1) else [])
        return len(seq1), operations
    if len(seq1) == 0:
        operations = operations + ([f"Insert `{seq2}` into sequence1."] if len(seq2) else [])
        return len(seq2), operations
    if seq1[0] == seq2[0]:
        operations = operations + [f"Make no change for character if both sequence are
equal`{seq1[0]}`."]
        return edit_distance_recurse(seq1[1: ], seq2[1: ], operations)
    # calculate cost if insertion was made
    ins_operations = operations + [f"Insert `{seq2[0]}` in sequence1."]
    insertion, ins_operations = edit_distance_recurse(seq1, seq2[1: ], ins_operations)
    # calculate cost if deletion was done
    del_operations = operations + [f"Delete `{seq1[0]}` from sequence1."]
    deletion, del_operations = edit_distance_recurse(seq1[1: ], seq2, del_operations)
    # calculate cost if substitution was done
    sub_operations = operations + [f"Replace `{seq1[0]}` in sequence1 with `{seq2[0]}`."]
    substitution, sub_operations = edit_distance_recurse(seq1[1: ], seq2[1: ], sub_operations)
        # calculate minimum cost
    min_cost = min(insertion + ins_cost, deletion + del_cost, substitution + sub_cost)
    if min_cost == (substitution + sub_cost):
        return min_cost, sub_operations
```

```
    elif min_cost == deletion + del_cost:
        return min_cost, del_operations
    else:
        return min_cost, ins_operations
#Let's test this function for some examples
seq1 = "numpy"
seq2 = "numexpr"
score, operations = edit_distance_recurse(seq1, seq2)
print(f"\n Edit Distance between `{seq1}` & `{seq2}` is: {score}")
print("\nOperations performed are:")
for operation in operations:
    print(operation)
```

Output:



Source Code:

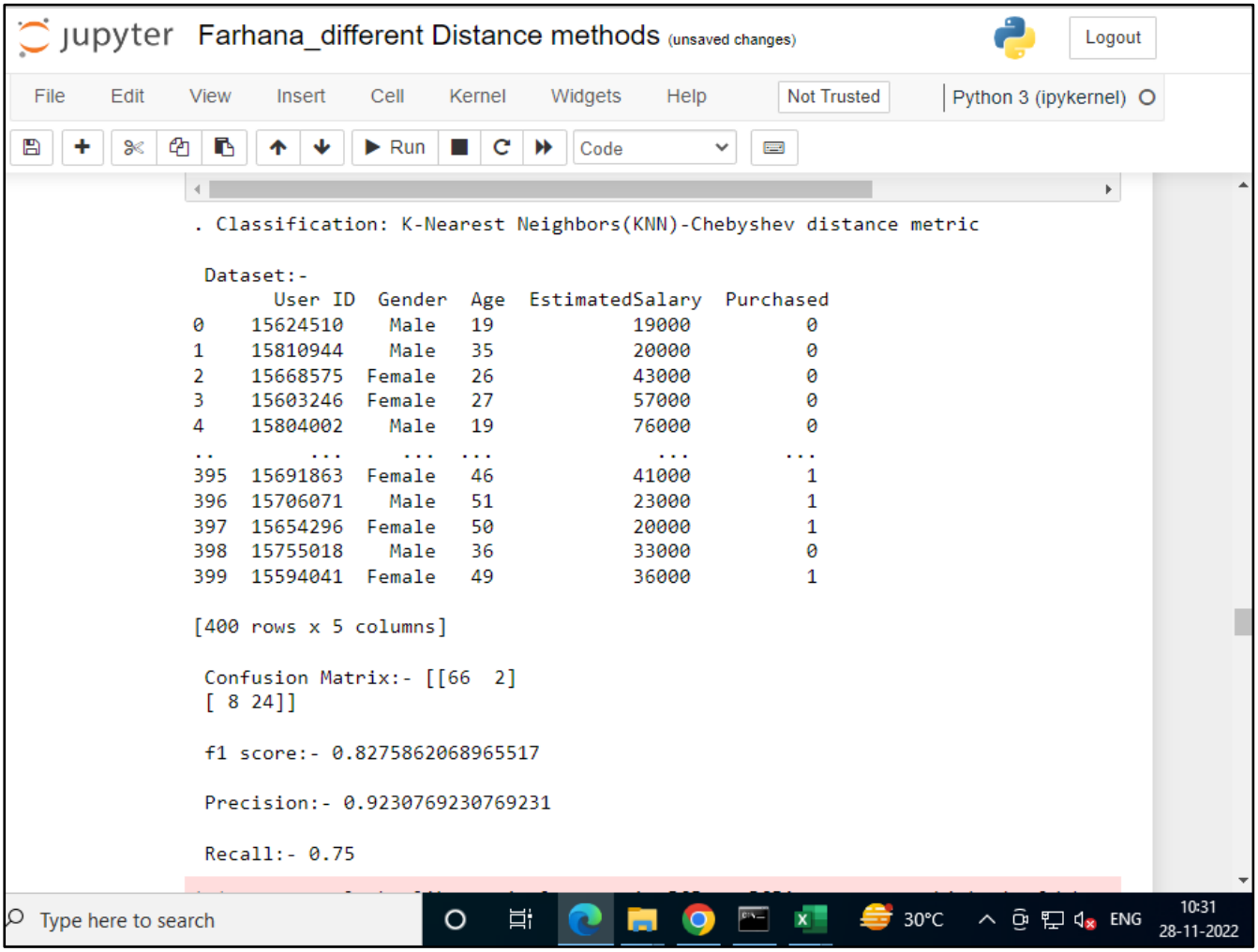
7. Chebyshev Distance

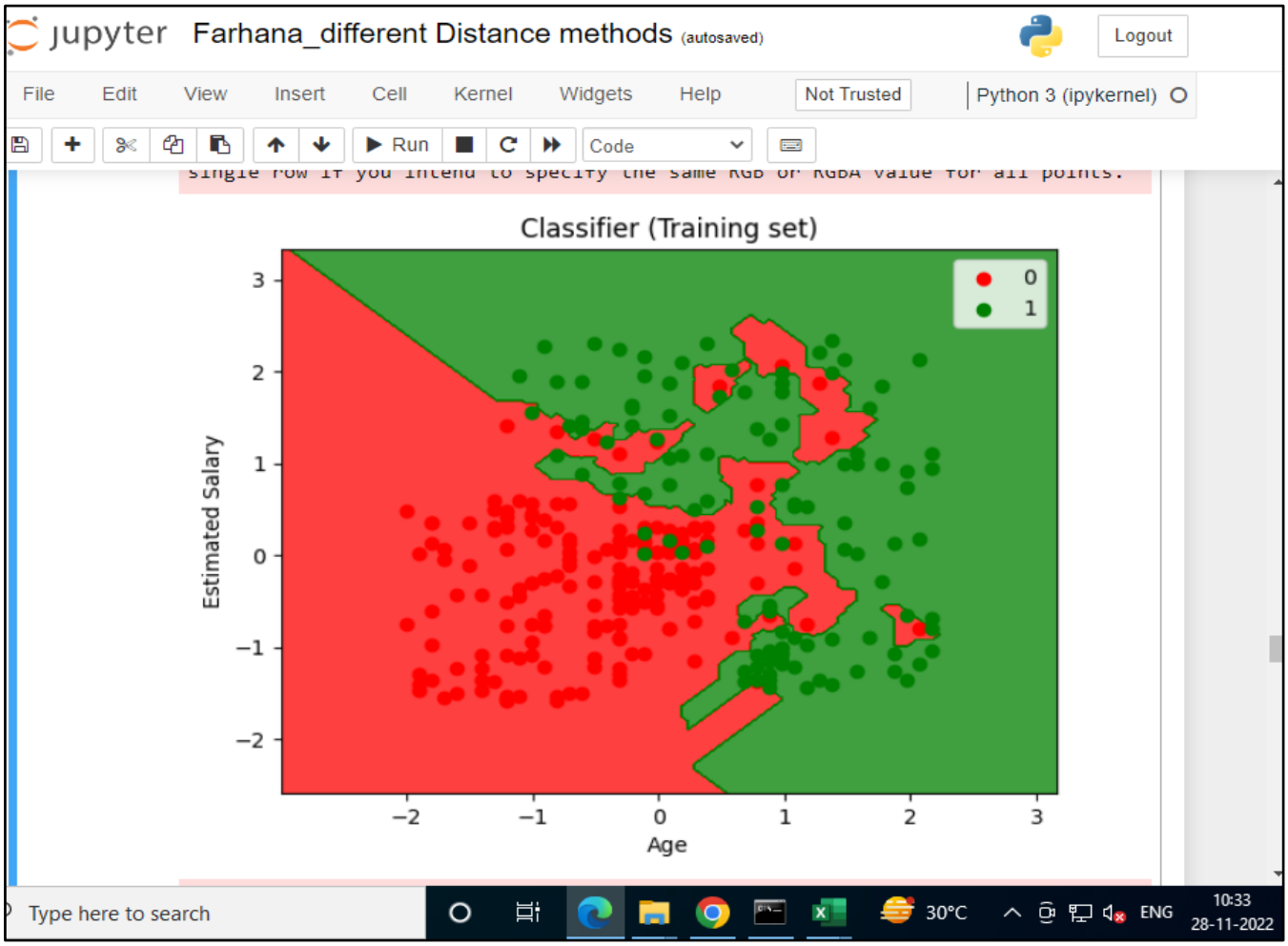
```
print(". Classification: K-Nearest Neighbors(KNN)-Chebyshev distance metric")
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
print("\n Dataset:-\n", dataset)
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
# Splitting the dataset into the Training set 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)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 2, metric='chebyshev')
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, recall_score,
precision_score
cm = confusion_matrix(y_test, y_pred)
print("\n Confusion Matrix:-", cm)
f1 = f1_score(y_test, y_pred)
print("\n f1 score:-", f1)
```

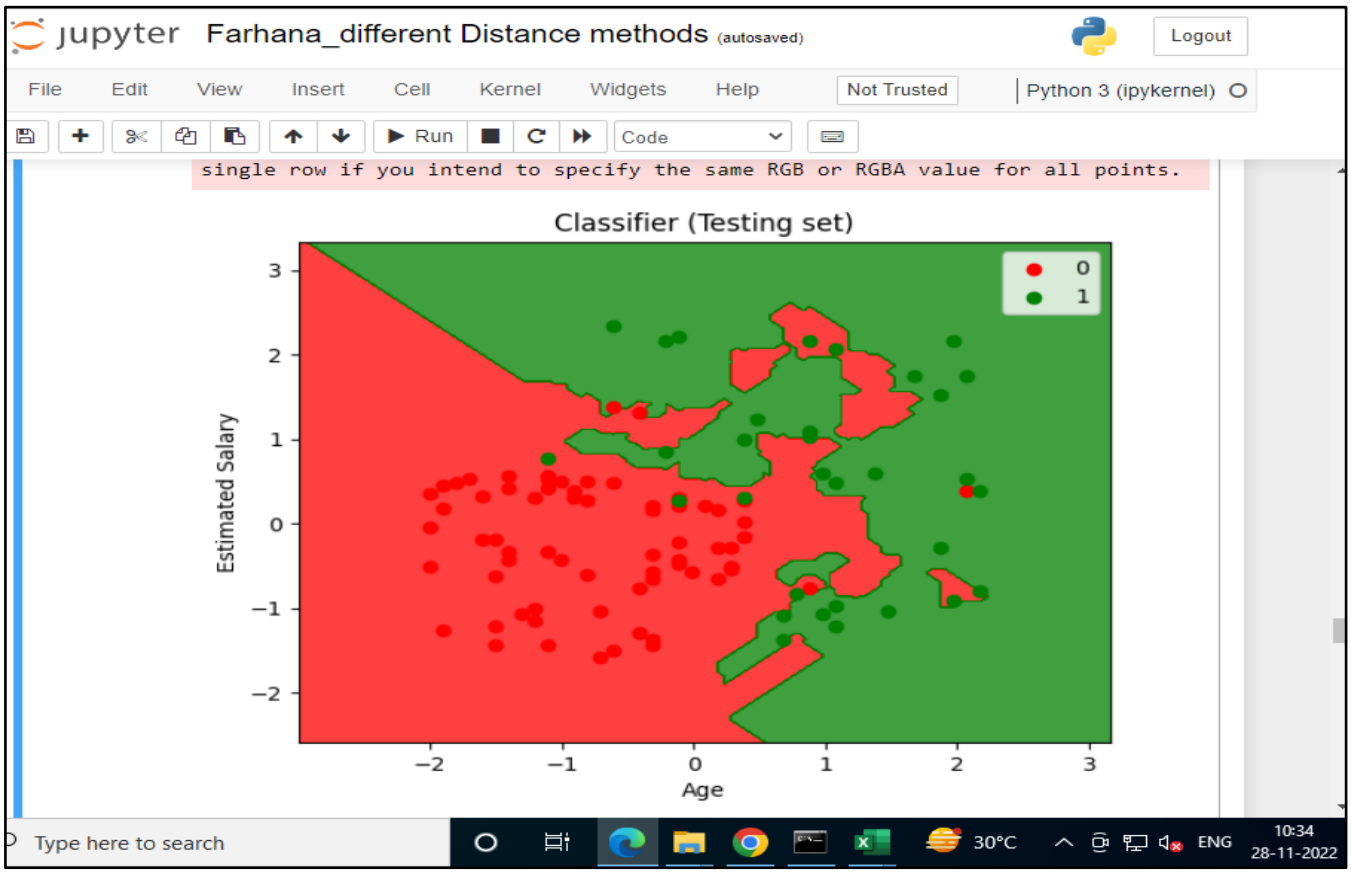
```
precision = precision_score(y_test, y_pred)
print("\n Precision:-", precision)
recall = recall_score(y_test, y_pred)
print("\n Recall:-", recall)
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = 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('Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
# Visualising the Testing set results
from matplotlib.colors import ListedColormap
X_test, y_test = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_test[:, 0].min() - 1, stop = X_test[:, 0].max() + 1,
step = 0.01),
                     np.arange(start = X_test[:, 1].min() - 1, stop = X_test[:, 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_test)):
    plt.scatter(X_test[y_test == j, 0], X_test[y_test == j, 1],
            c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifier (Testing set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output:




jupyter Farhana_different Distance methods (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Classifier (Training set)
Estimated Salary
Age
0
1
Type here to search
30°C
ENG
10:33
28-11-2022


jupyter Farhana_different Distance methods (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
single row if you intend to specify the same RGB or RGBA value for all points.
Classifier (Testing set)
Estimated Salary
Age
0
1
Type here to search
30°C
ENG
10:34
28-11-2022

Source Code:

```
from datetime import datetime
print('-------------------KMeans Clustering-----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np
# import some data to play with
iris = datasets.load_iris()
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
#Call the KMeans implementation to instantiate a model and fit it.The parameter n_clusters
is the number of clusters
model = KMeans(n_clusters=3)
model.fit(X)
#labels that the algorithm has provided
print("\n=====Iris KmMeans Label=======")
print(model.labels_)
```

```
print("\n")
#Let us look at the location of the final clusters:
print("=======Location of the final clusters=======")
print(model.cluster_centers_)
print("\n")

plt.figure(figsize=(14,7))
colormap = np.array(['red', 'lime', 'black'])
# Plot the Original Classifications
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
#plotting the centroids and clustered data points of the clusters
plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1], s=100, c='blue',
label='Centroids')
plt.title('K Mean clustering Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of K-Mean clustering: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrix of K-Mean clustering: ',sm.confusion_matrix(y,
model.labels_))
```

Output:

jupyter Farhana_KMeans Clustering (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
--------------------KMeans Clustering-----------------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 22/11/2022 09:57:53

=====Iris KmMeans Label=======
[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 1 1 2 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2
 2 1]


=======Location of the final clusters=======
[[5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]
 [6.85       3.07368421 5.74210526 2.07105263]]


The accuracy score of K-Mean clustering:  0.8933333333333333
The Confusion matrix of K-Mean clustering:  [[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
```

Real Classification K Mean clustering Classification

Type here to search 28°C ENG 09:58 22-11-2022

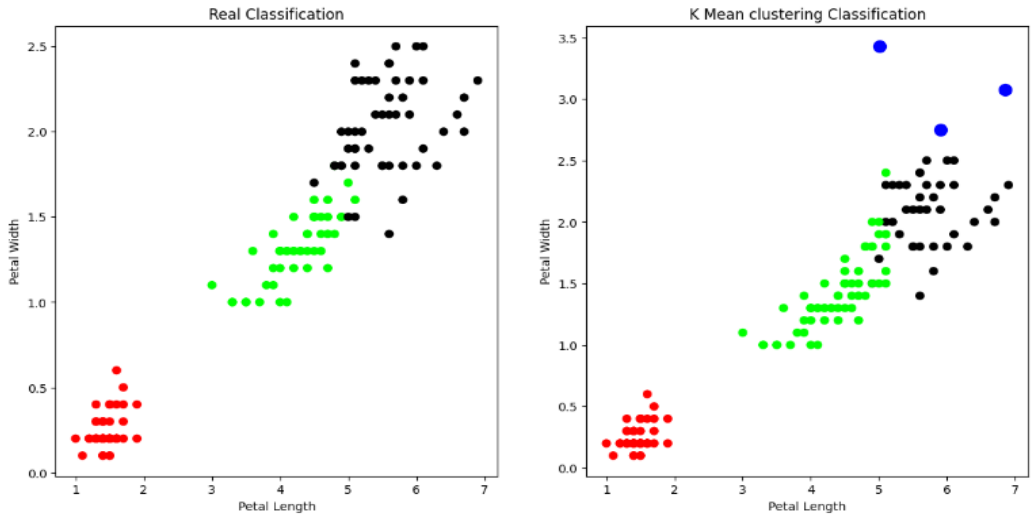
jupyter Farhana_KMeans Clustering (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
The accuracy score of K-Mean clustering:  0.8933333333333333
The Confusion matrix of K-Mean clustering:  [[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
```



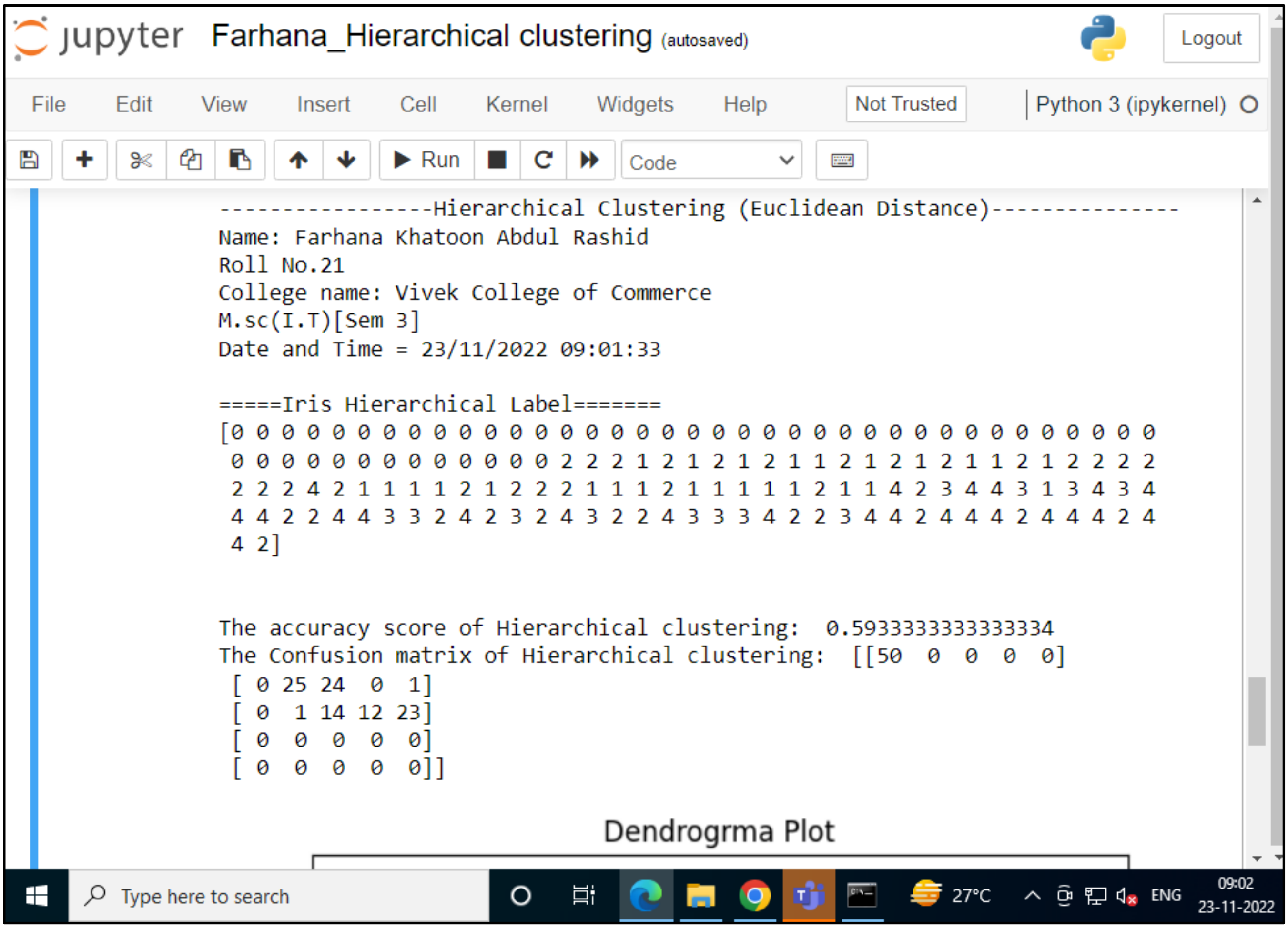
Type here to search 28°C ENG 10:01 22-11-2022

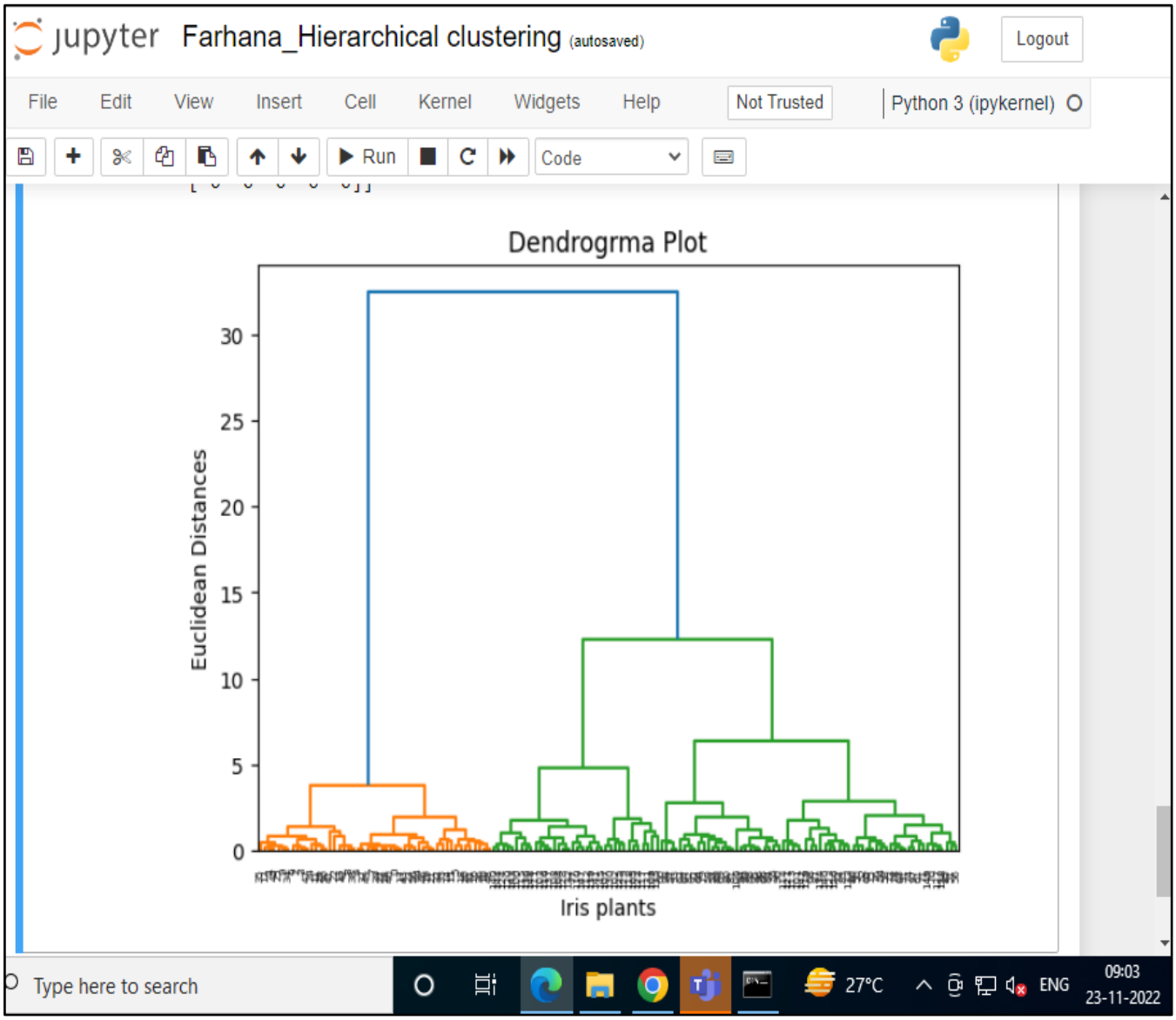
Source Code:

```
from datetime import datetime
print('----------------Hierarchical Clustering (Euclidean Distance)---------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
from sklearn import datasets
import pandas as pd
from sklearn.utils import shuffle
import matplotlib.pyplot as mtp
import sklearn.metrics as sm
from matplotlib import pyplot as plt
import numpy as np
# import some data to play with
iris = datasets.load_iris()
#Assigning ‘X’ as independent variable and ‘Y’ as dependent variable
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
#training the hierarchical model on dataset
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
y_pred= hc.fit_predict(X)
#labels that the algorithm has provided
print("\n=====Iris Hierarchical Label=======")
```

```
label=hc.labels_
print(label)
print("\n")
print('The accuracy score of Hierarchical clustering: ',sm.accuracy_score(y, hc.labels_))
print('The Confusion matrix of Hierarchical clustering: ',sm.confusion_matrix(y,
hc.labels_))
#Finding the optimal number of clusters using the Dendrogram
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(X, method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Iris plants")
mtp.show()
```

Output:



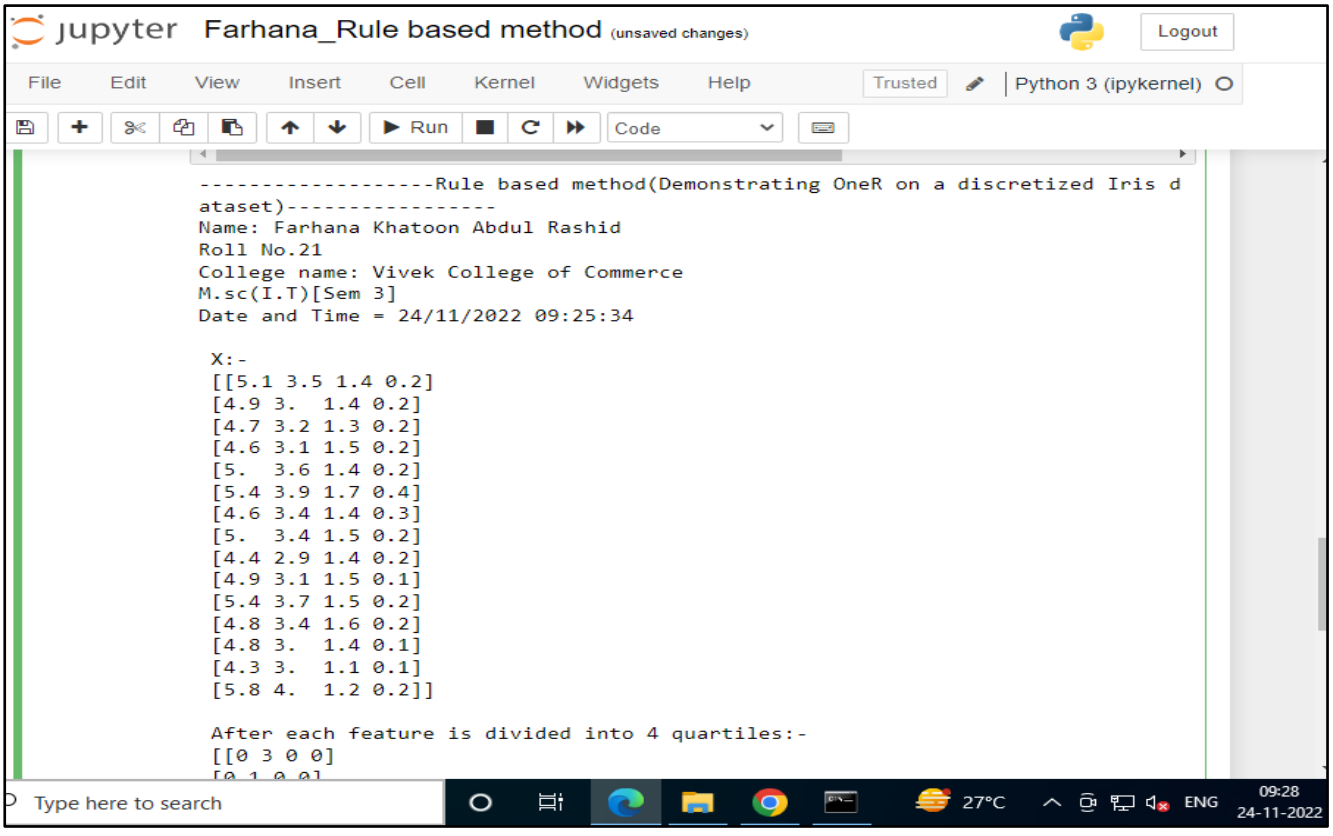

jupyter Farhana_Hierarchical clustering (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Dendrogrma Plot
Euclidean Distances
0
5
10
15
20
25
30
Iris plants
Type here to search
27°C
ENG
09:03
23-11-2022

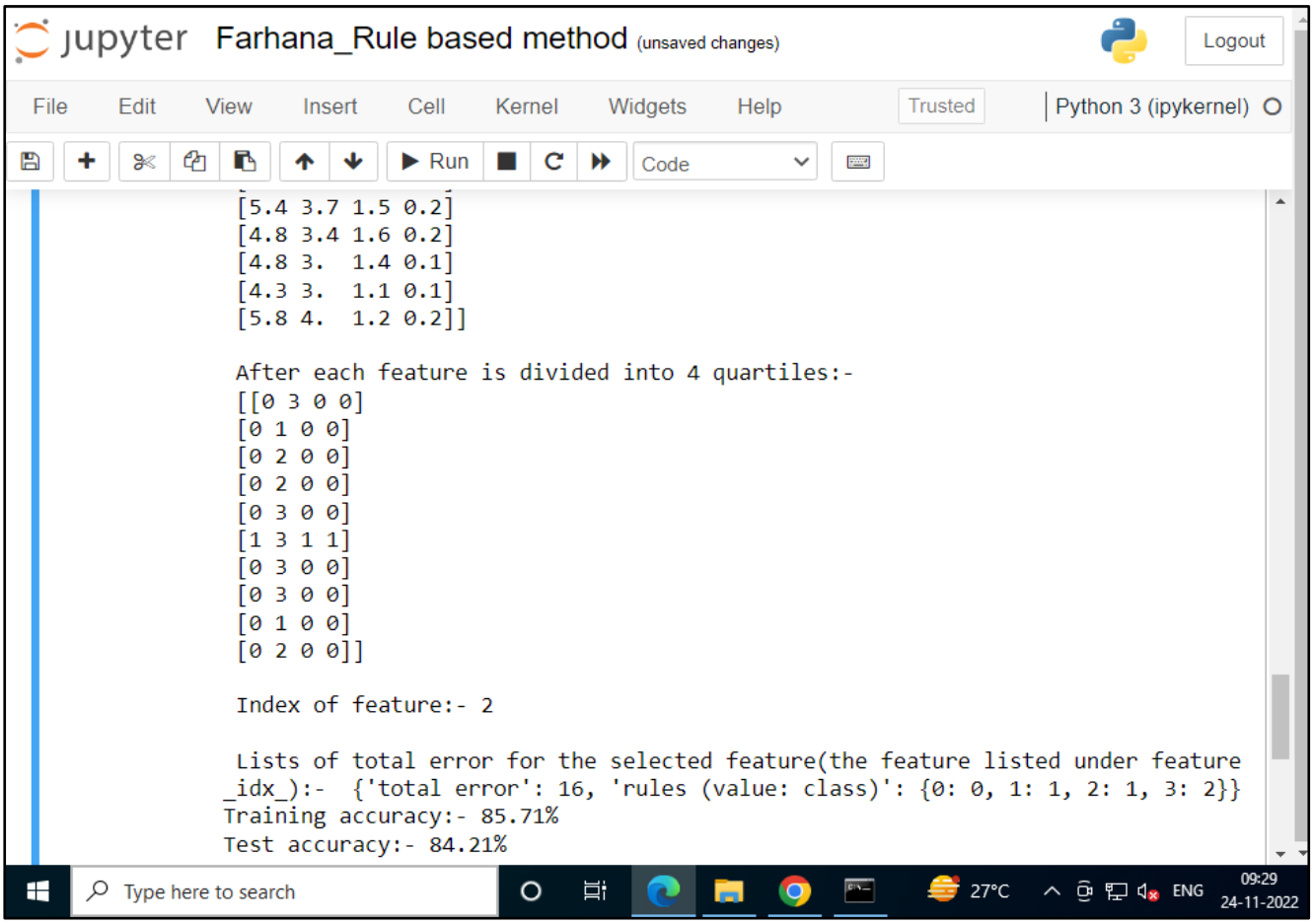
Source Code:

```
from datetime import datetime
print('------------------Rule based method(Demonstrating OneR on a discretized Iris dataset)-
---------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
import numpy as np
import pandas as pd
from sklearn import metrics
from mlxtend.data import iris_data
X, y = iris_data()
print("\n X:- \n", X[:15])
import numpy as np
def get_feature_quartiles(X):
    X_discretized = X.copy()
    for col in range(X.shape[1]):
        for q, class_label in zip([1.0, 0.75, 0.5, 0.25], [3, 2, 1, 0]):
            threshold = np.quantile(X[:, col], q=q)
            X_discretized[X[:, col] <= threshold, col] = class_label
    return X_discretized.astype(np.int)
Xd=get_feature_quartiles(X)
print(" \n After each feature is divided into 4 quartiles:- \n", xd[:10])
#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
Xd_train, Xd_test, y_train, y_test = train_test_split(Xd, y, random_state=0, stratify=y)
#Now train a OneRClassifier model on the training set using the fit method
from mlxtend.classifier import OneRClassifier
```

```
oner = OneRClassifier()
oner.fit(Xd_train, y_train)
#The column index of the selected feature is accessible via the feature_idx_ attribute after model fitting
print("\n Index of feature:-", oner.feature_idx_)
print("\n Lists of total error for the selected feature(the feature listed under feature_idx_):- ", oner.prediction_dict_)
#Making prediction
#For training
y_pred = oner.predict(Xd_train)
train_acc = np.mean(y_pred == y_train)
print(f'Training accuracy:- {train_acc*100:.2f}%')
#For testing
from sklearn import metrics
y_pred = oner.predict(Xd_test)
test_acc = np.mean(y_pred == y_test)
print(f'Test accuracy:- {test_acc*100:.2f}%')
```

Output:



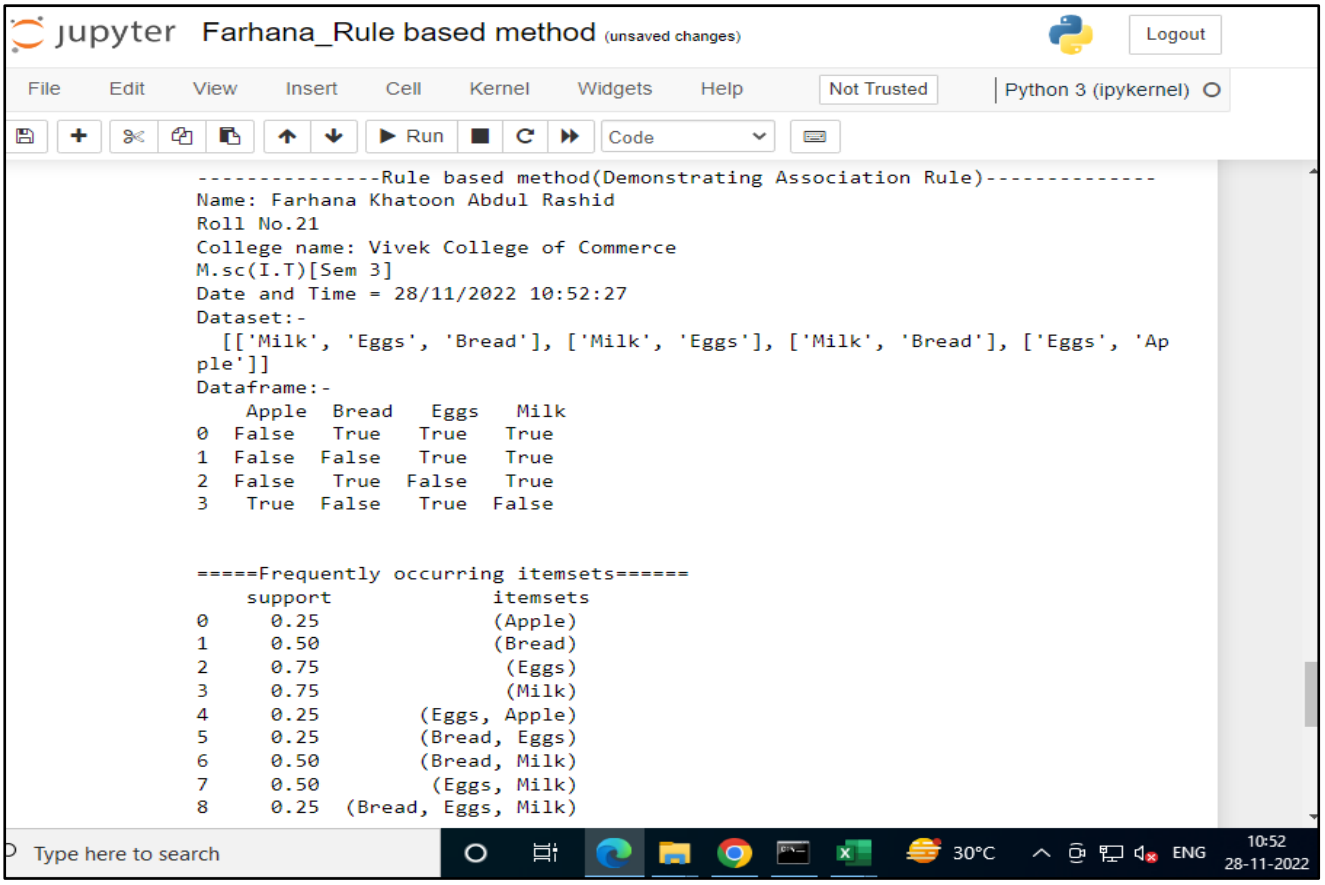


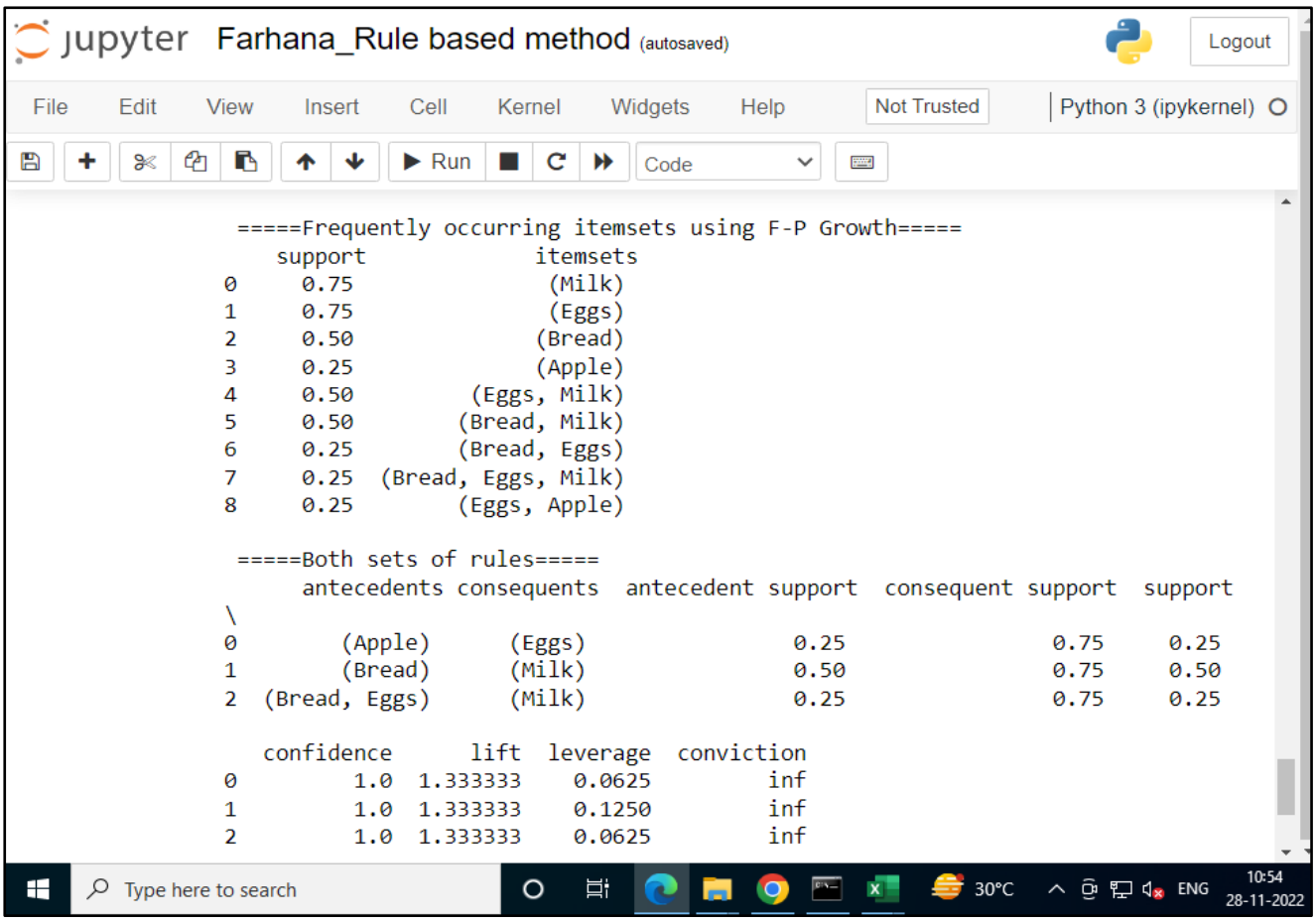
Source Code:

```
from datetime import datetime
print('---------------Rule based method(Demonstrating Association Rule)--------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
import pandas as pd
#Creating a list with the required data
dataset = [
    ["Milk", "Eggs", "Bread"],
    ["Milk", "Eggs"],
```

```
    ["Milk", "Bread"],
    ["Eggs", "Apple"],
]
print("Dataset:-\n ",dataset)
#Convert list to dataframe with boolean values
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_array = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_array, columns=te.columns_)
print("Dataframe:- \n", df)
print("\n")
#Find frequently occurring itemsets using Apriori Algorithm
from mlxtend.frequent_patterns import apriori
frequent_itemsets_ap = apriori(df, min_support=0.01, use_colnames=True)
print("=====Frequently occurring itemsets====== \n", frequent_itemsets_ap)
#Find frequently occurring itemsets using F-P Growth
from mlxtend.frequent_patterns import fpgrowth
frequent_itemsets_fp=fpgrowth(df, min_support=0.01, use_colnames=True)
print("\n =====Frequently occurring itemsets using F-P Growth===== \n",
frequent_itemsets_fp)
#Mine the Association Rules
from mlxtend.frequent_patterns import association_rules
rules_ap = association_rules(frequent_itemsets_ap, metric="confidence",
min_threshold=0.8)
rules_fp = association_rules(frequent_itemsets_fp, metric="confidence",
min_threshold=0.8)
print("\n =====Both sets of rules===== \n", rules_ap)
```

Output:


jupyter Farhana_Rule based method (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
----------Rule based method(Demonstrating Association Rule)----------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 28/11/2022 10:52:27
Dataset:-
[['Milk', 'Eggs', 'Bread'], ['Milk', 'Eggs'], ['Milk', 'Bread'], ['Eggs', 'Apple']]
Dataframe:-
Apple Bread Eggs Milk
0 False True True True
1 False False True True
2 False True False True
3 True False True False
=====Frequently occurring itemsets======
support itemsets
0 0.25 (Apple)
1 0.50 (Bread)
2 0.75 (Eggs)
3 0.75 (Milk)
4 0.25 (Eggs, Apple)
5 0.25 (Bread, Eggs)
6 0.50 (Bread, Milk)
7 0.50 (Eggs, Milk)
8 0.25 (Bread, Eggs, Milk)
Type here to search
30°C
ENG
10:52
28-11-2022

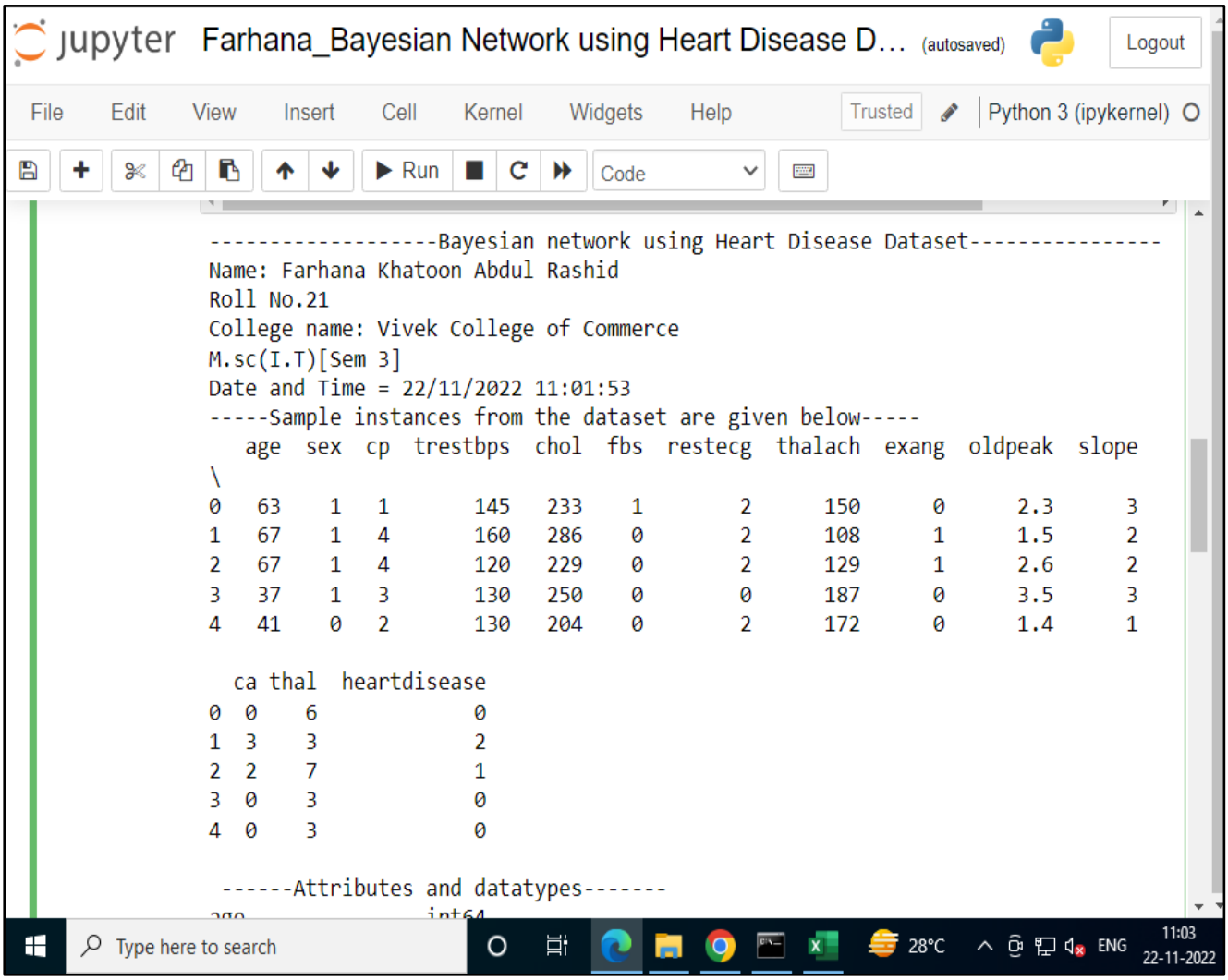

jupyter Farhana_Rule based method (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
=====Frequently occurring itemsets using F-P Growth=====
support itemsets
0 0.75 (Milk)
1 0.75 (Eggs)
2 0.50 (Bread)
3 0.25 (Apple)
4 0.50 (Eggs, Milk)
5 0.50 (Bread, Milk)
6 0.25 (Bread, Eggs)
7 0.25 (Bread, Eggs, Milk)
8 0.25 (Eggs, Apple)
=====Both sets of rules=====
antecedents consequents antecedent support consequent support support
\
0 (Apple) (Eggs) 0.25 0.75 0.25
1 (Bread) (Milk) 0.50 0.75 0.50
2 (Bread, Eggs) (Milk) 0.25 0.75 0.25
confidence lift leverage conviction
0 1.0 1.333333 0.0625 inf
1 1.0 1.333333 0.1250 inf
2 1.0 1.333333 0.0625 inf
Type here to search
30°C
ENG
10:54
28-11-2022

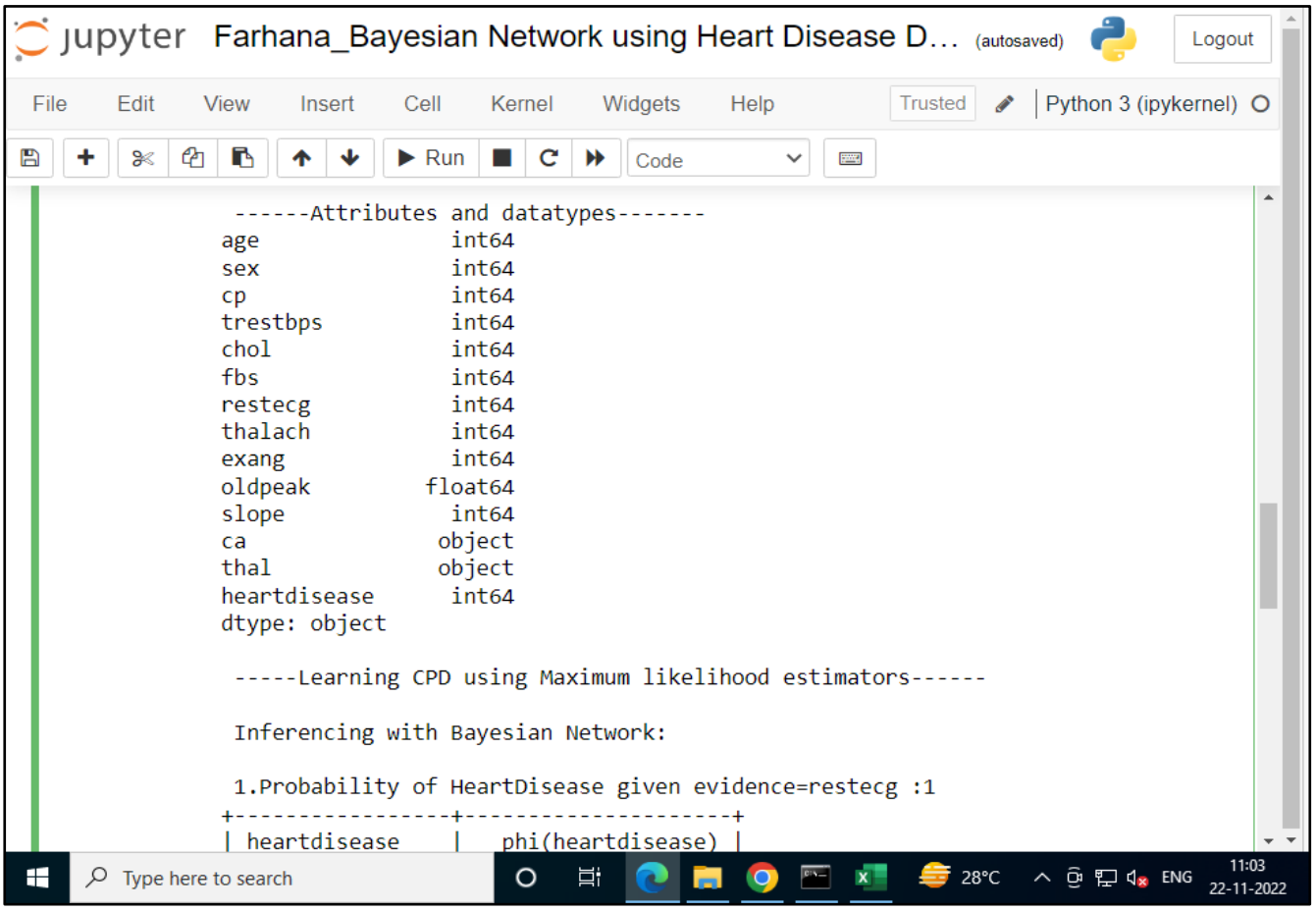
Source Code:

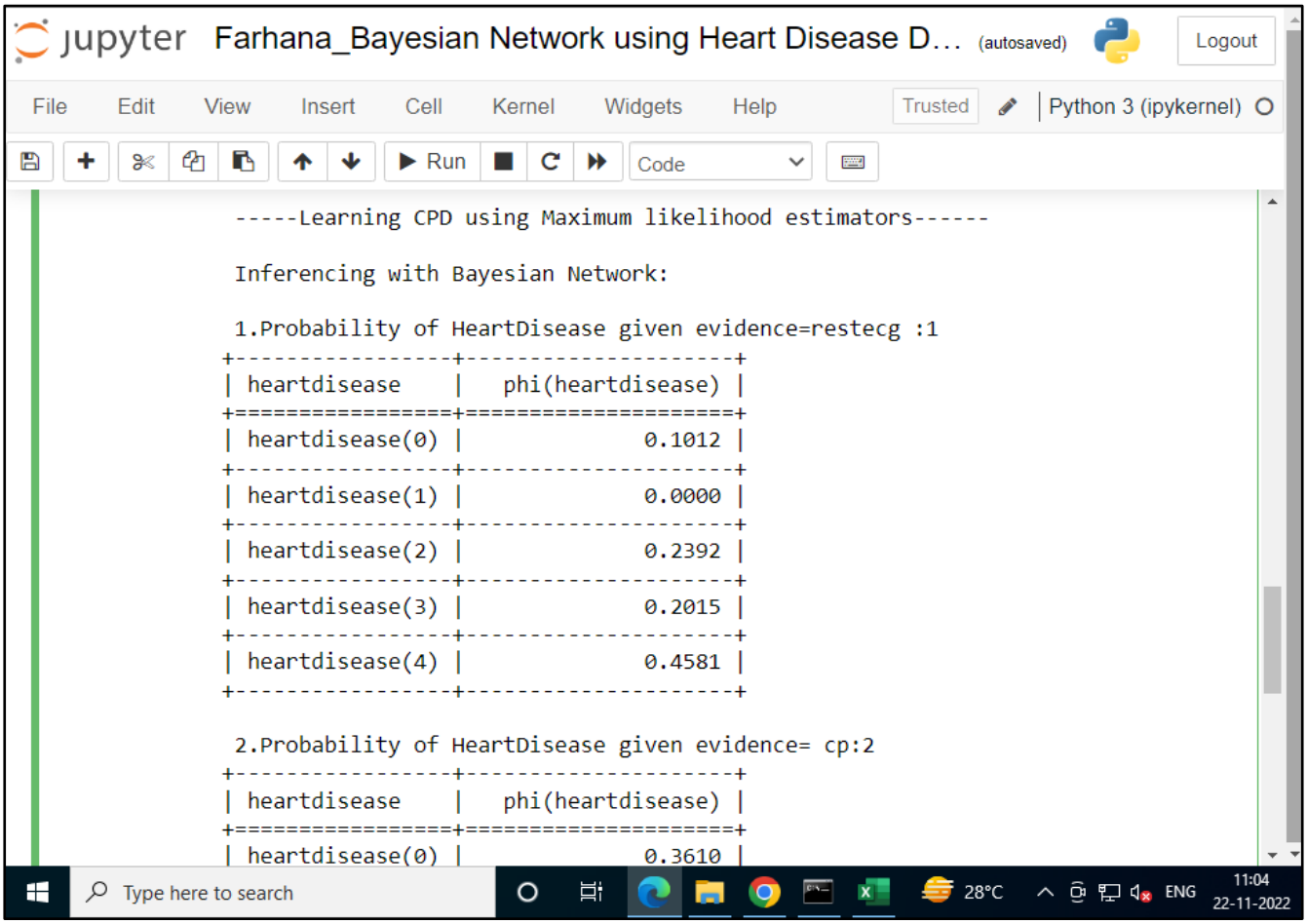
```
from datetime import datetime
print('------------------Bayesian network using Heart Disease Dataset----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
#read Cleveland Heart Disease data
heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)
#display the data
print('-----Sample instances from the dataset are given below-----')
print(heartDisease.head())
#display the Attributes names and datatyes
print('\n ------Attributes and datatypes-------')
print(heartDisease.dtypes)
#Creat Model- Bayesian Network
model =BayesianModel([('age','heartdisease'),('sex','heartdisease'),(
'exang','heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','chol')])
#Learning CPDs using Maximum Likelihood Estimators
print('\n -----Learning CPD using Maximum likelihood estimators------')
```

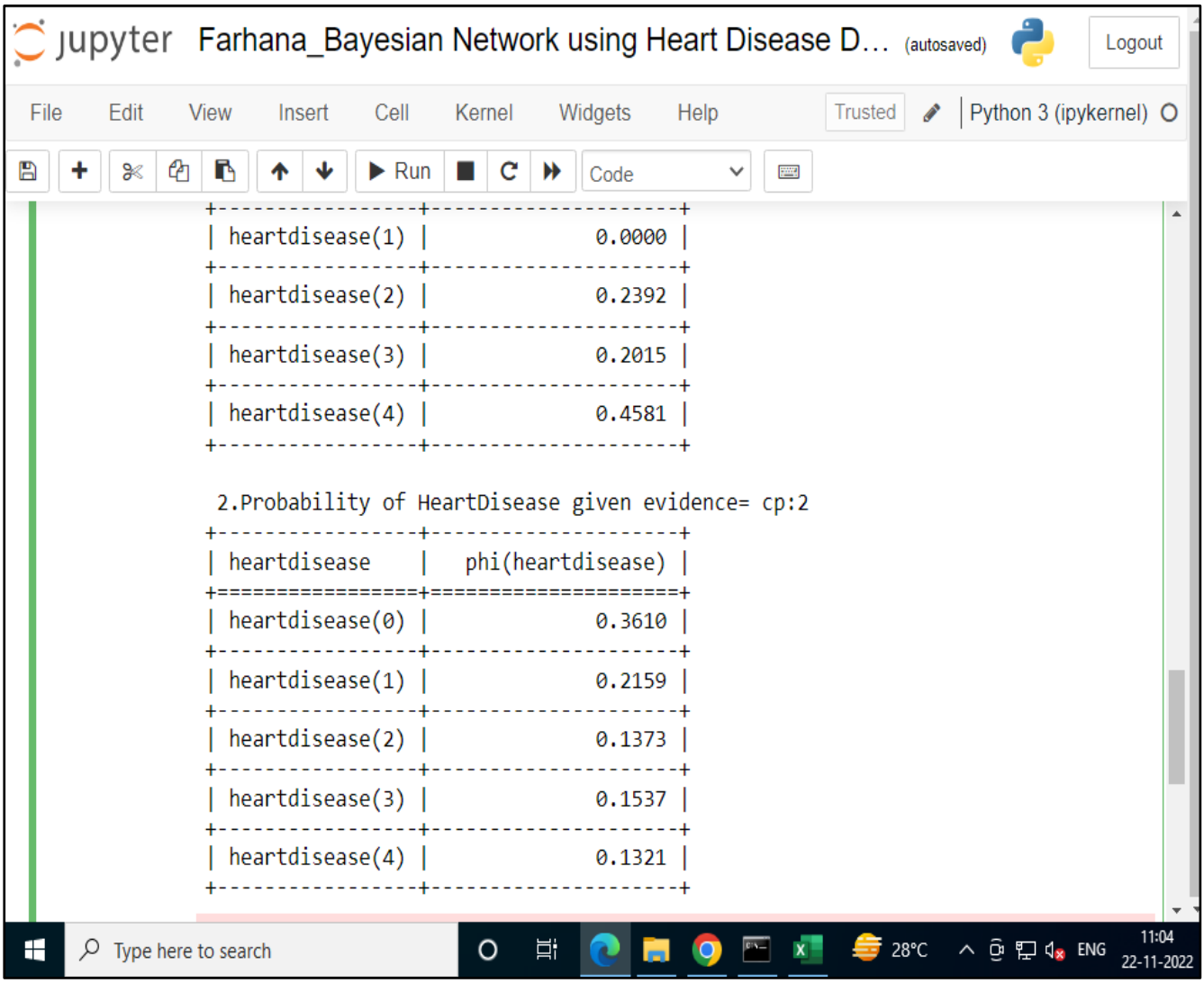
```
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)
# Inferencing with Bayesian Network
print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
#computing the Probability of HeartDisease given restecg
print('\n 1.Probability of HeartDisease given evidence=restecg :1')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)
#computing the Probability of HeartDisease given cp
print('\n 2.Probability of HeartDisease given evidence= cp:2 ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

Output:




jupyter Farhana_Bayesian Network using Heart Disease D... (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
------Attributes and datatypes-------
age int64
sex int64
cp int64
trestbps int64
chol int64
fbs int64
restecg int64
thalach int64
exang int64
oldpeak float64
slope int64
ca object
thal object
heartdisease int64
dtype: object
-----Learning CPD using Maximum likelihood estimators------
Inferencing with Bayesian Network:
1.Probability of HeartDisease given evidence=restecg :1
| heartdisease | phi(heartdisease) |
Type here to search
28°C
ENG
11:03
22-11-2022


jupyter Farhana_Bayesian Network using Heart Disease D... (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
-----Learning CPD using Maximum likelihood estimators------
Inferencing with Bayesian Network:
1.Probability of HeartDisease given evidence=restecg :1
heartdisease	phi(heartdisease)
heartdisease(0)	0.1012
heartdisease(1)	0.0000
heartdisease(2)	0.2392
heartdisease(3)	0.2015
heartdisease(4)	0.4581
2.Probability of HeartDisease given evidence= cp:2
heartdisease	phi(heartdisease)
heartdisease(0)	0.3610
Type here to search
28°C
ENG
11:04
22-11-2022


jupyter Farhana_Bayesian Network using Heart Disease D... (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
heartdisease(1)	0.0000
heartdisease(2)	0.2392
heartdisease(3)	0.2015
heartdisease(4)	0.4581
2.Probability of HeartDisease given evidence= cp:2
heartdisease	phi(heartdisease)
heartdisease(0)	0.3610
heartdisease(1)	0.2159
heartdisease(2)	0.1373
heartdisease(3)	0.1537
heartdisease(4)	0.1321
Type here to search
28°C
ENG
11:04
22-11-2022

Source Code:

```
from datetime import datetime
print('--------------non-parametric Locally Weighted Regression--------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
from numpy import *
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np1
import numpy.linalg as np
from scipy.stats.stats import pearsonr
from numpy.linalg import inv, det
def kernel(point,xmat, k):
    m,n = np1.shape(xmat)
    weights = np1.mat(np1.eye((m)))
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np1.exp(diff*diff.T/(-2.0*k**2))
    return weights
def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W
```

```
def localWeightRegression(xmat,ymat,k):
    m,n = np1.shape(xmat)
    ypred = np1.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred
# load data points
data = pd.read_csv('tips.csv')
print('\n Dataset:-\n', data)
bill = np1.array(data.total_bill)
tip = np1.array(data.tip)
#preparing and add 1 in bill
mbill = np1.mat(bill)
mtip = np1.mat(tip) # mat is used to convert to n dimensional to 2 dimensional array form
m= np1.shape(mbill)[1]
# print(m) 244 data is stored in m
one = np1.mat(np1.ones(m))
X= np1.hstack((one.T,mbill.T)) # create a stack of bill from ONE
#print(X)
#set k here
ypred = localWeightRegression(X,mtip,0.5)
SortIndex = X[:,1].argsort(0)
xsort = X[SortIndex][:,0]
#Plotting
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(bill,tip, color='green')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)
plt.xlabel('Total bill')
plt.ylabel('Tip')
plt.show()
```

Output:

jupyter Farhana_non-parametric Locally Weighted Reg… (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

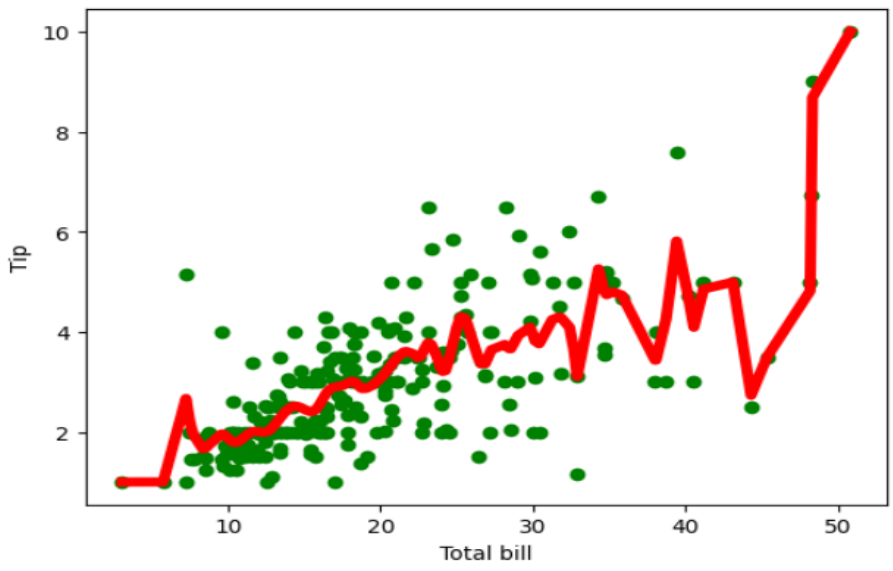
```
--------------non-parametric Locally Weighted Regression--------------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 23/11/2022 09:07:29

 Dataset:-
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

Type here to search 27°C ENG 09:08 23-11-2022

jupyter Farhana_non-parametric Locally Weighted Regres… (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code



In [ ]:

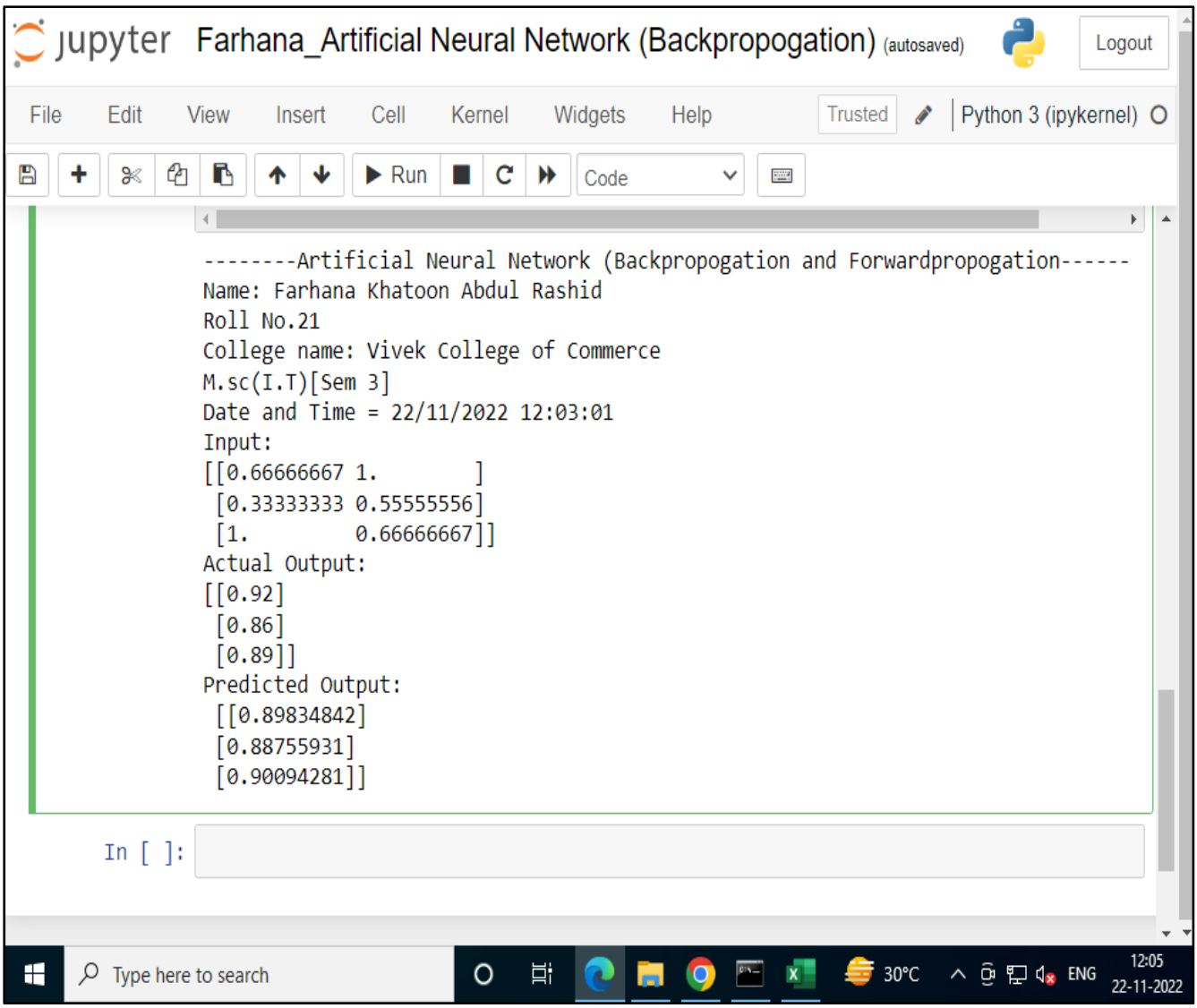
Type here to search 27°C ENG 09:08 23-11-2022

Source Code:

```
from datetime import datetime
print('--------Artificial Neural Network (Backpropagation and Forward Propagation------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
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) # maximum of X array longitudinally
y = y/100
#Sigmoid Function
def sigmoid (x):
 return 1/(1 + np.exp(-x))
#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
 return x * (1 - x)
#Variable initialization
epoch=5000 #Setting training iterations
lr=0.1 #Setting learning rate
inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization
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))

#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):

#Forward Propogation
 hinp1=np.dot(X,wh)
 hinp=hinp1 + bh
 hlayer_act = sigmoid(hinp)
 outinp1=np.dot(hlayer_act,wout)
 outinp= outinp1+ bout
 output = sigmoid(outinp)

#Backpropagation
 EO = y-output
 outgrad = derivatives_sigmoid(output)
 d_output = EO* outgrad
 EH = d_output.dot(wout.T)

#how much hidden layer wts contributed to error
 hiddengrad = derivatives_sigmoid(hlayer_act)
 d_hiddenlayer = EH * hiddengrad
# dotproduct of nextlayererror and currentlayerop
wout += hlayer_act.T.dot(d_output) *lr
wh += X.T.dot(d_hiddenlayer) *lr
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

Output:

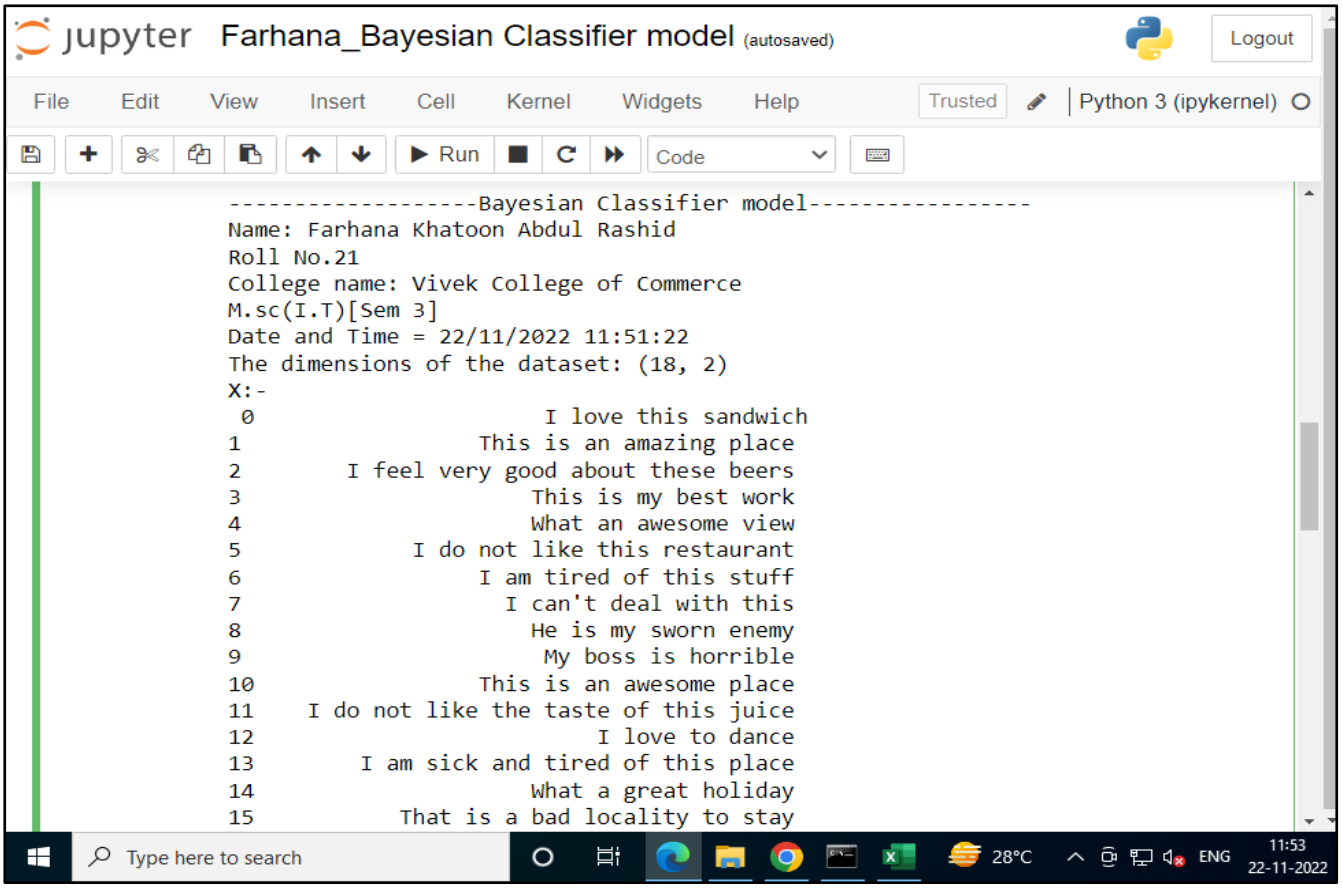

jupyter Farhana_Artificial Neural Network (Backpropogation) (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
--------Artificial Neural Network (Backpropogation and Forwardpropogation------
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 22/11/2022 12:03:01
Input:
[[0.66666667 1.]
[0.33333333 0.55555556]
[1. 0.66666667]]
Actual Output:
[[0.92]
[0.86]
[0.89]]
Predicted Output:
[[0.89834842]
[0.88755931]
[0.90094281]]
In [ ]:
Type here to search
30°C
ENG
12:05
22-11-2022

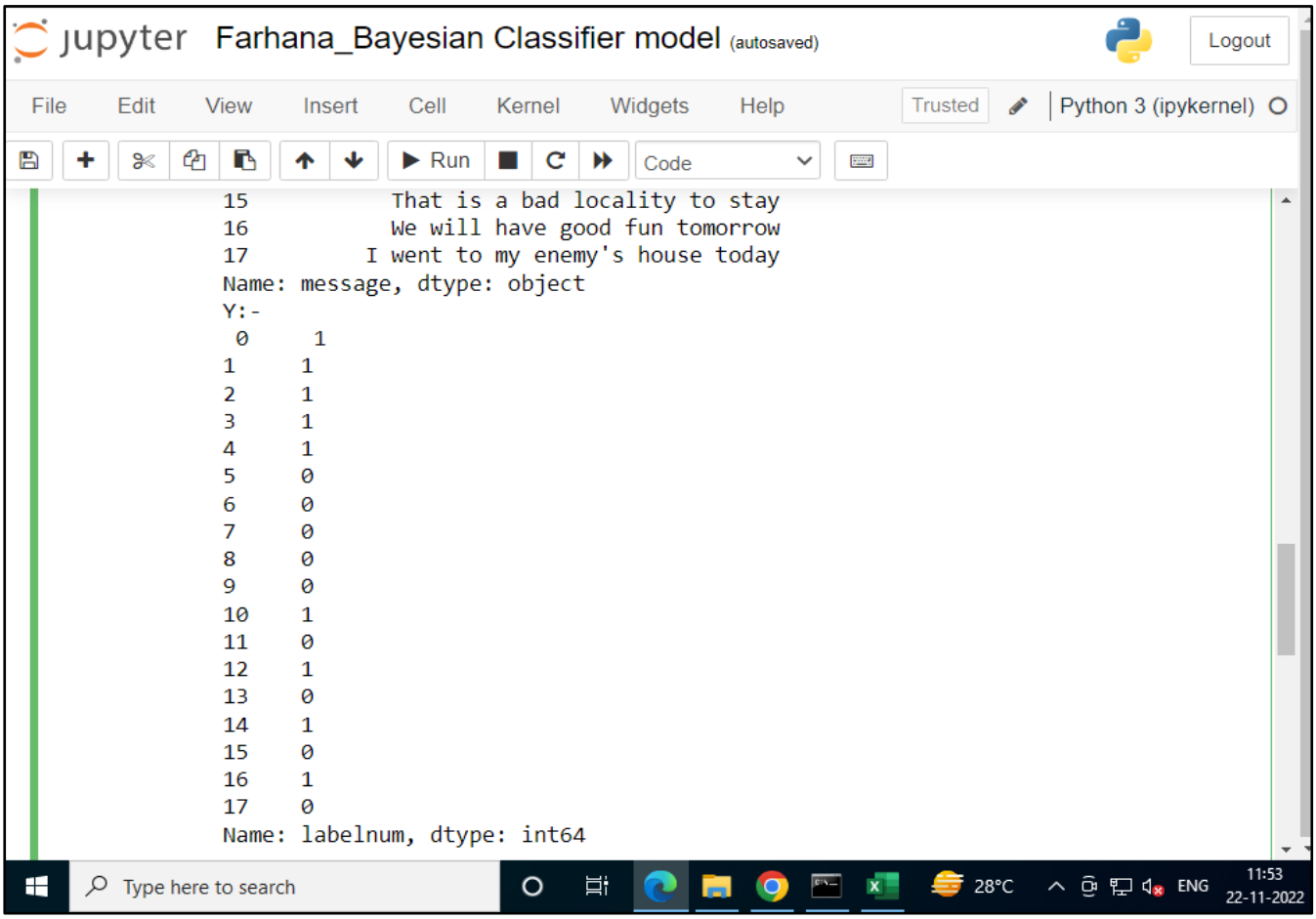
Source Code:

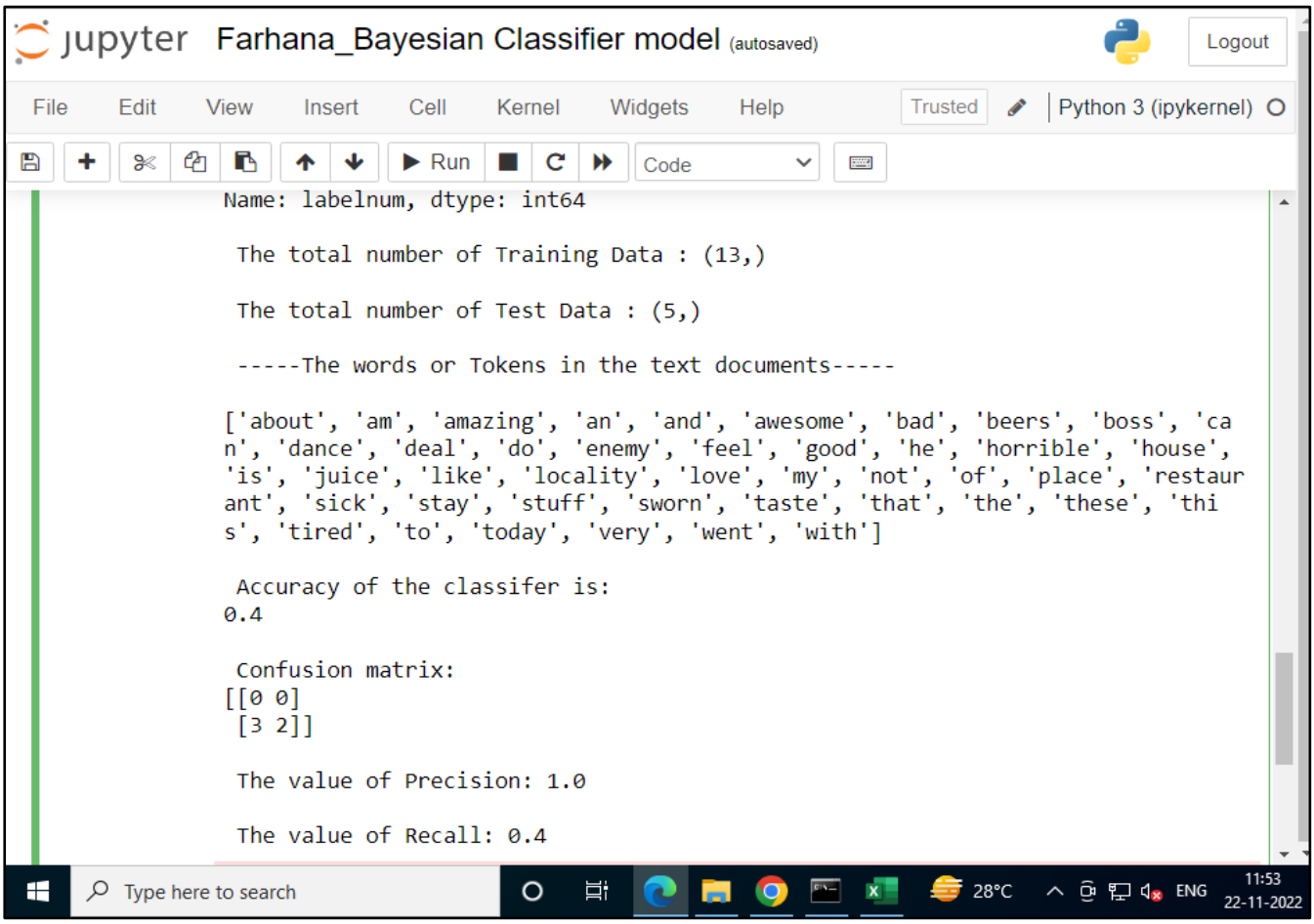
```
from datetime import datetime
print('------------------Bayesian Classifier model----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
# Import the required libraries
import pandas as pd
#To import dataset
msg=pd.read_csv('naivetext.csv',names=['message','label'])
print('The dimensions of the dataset:',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
print("X:-\n",X)
print("Y:-\n",y)
#splitting the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print ('\n The total number of Training Data :',ytrain.shape)
print ('\n The total number of Test Data :',ytest.shape)
#output of count vectoriser is a sparse matrix
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print('\n -----The words or Tokens in the text documents----- \n')
```

```
print(count_vect.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())
# Training Naive Bayes (NB) classifier on training data.
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
#printing accuracy, Confusion matrix, Precision and Recall
from sklearn import metrics
print('\n Accuracy of the classifer is:')
print(metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix:')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision:' ,
metrics.precision_score(ytest,predicted))
print('\n The value of Recall:' ,
metrics.recall_score(ytest,predicted))
```

Output:




jupyter Farhana_Bayesian Classifier model (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
15 That is a bad locality to stay
16 We will have good fun tomorrow
17 I went to my enemy's house today
Name: message, dtype: object
Y:-
0 1
1 1
2 1
3 1
4 1
5 0
6 0
7 0
8 0
9 0
10 1
11 0
12 1
13 0
14 1
15 0
16 1
17 0
Name: labelnum, dtype: int64
Type here to search
28°C
ENG
11:53
22-11-2022


jupyter Farhana_Bayesian Classifier model (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Name: labelnum, dtype: int64
The total number of Training Data : (13,)
The total number of Test Data : (5,)
-----The words or Tokens in the text documents-----
['about', 'am', 'amazing', 'an', 'and', 'awesome', 'bad', 'beers', 'boss', 'ca
n', 'dance', 'deal', 'do', 'enemy', 'feel', 'good', 'he', 'horrible', 'house',
'is', 'juice', 'like', 'locality', 'love', 'my', 'not', 'of', 'place', 'restaur
ant', 'sick', 'stay', 'stuff', 'sworn', 'taste', 'that', 'the', 'these', 'thi
s', 'tired', 'to', 'today', 'very', 'went', 'with']
Accuracy of the classifer is:
0.4
Confusion matrix:
[[0 0]
[3 2]]
The value of Precision: 1.0
The value of Recall: 0.4
Type here to search
28°C
ENG
11:53
22-11-2022

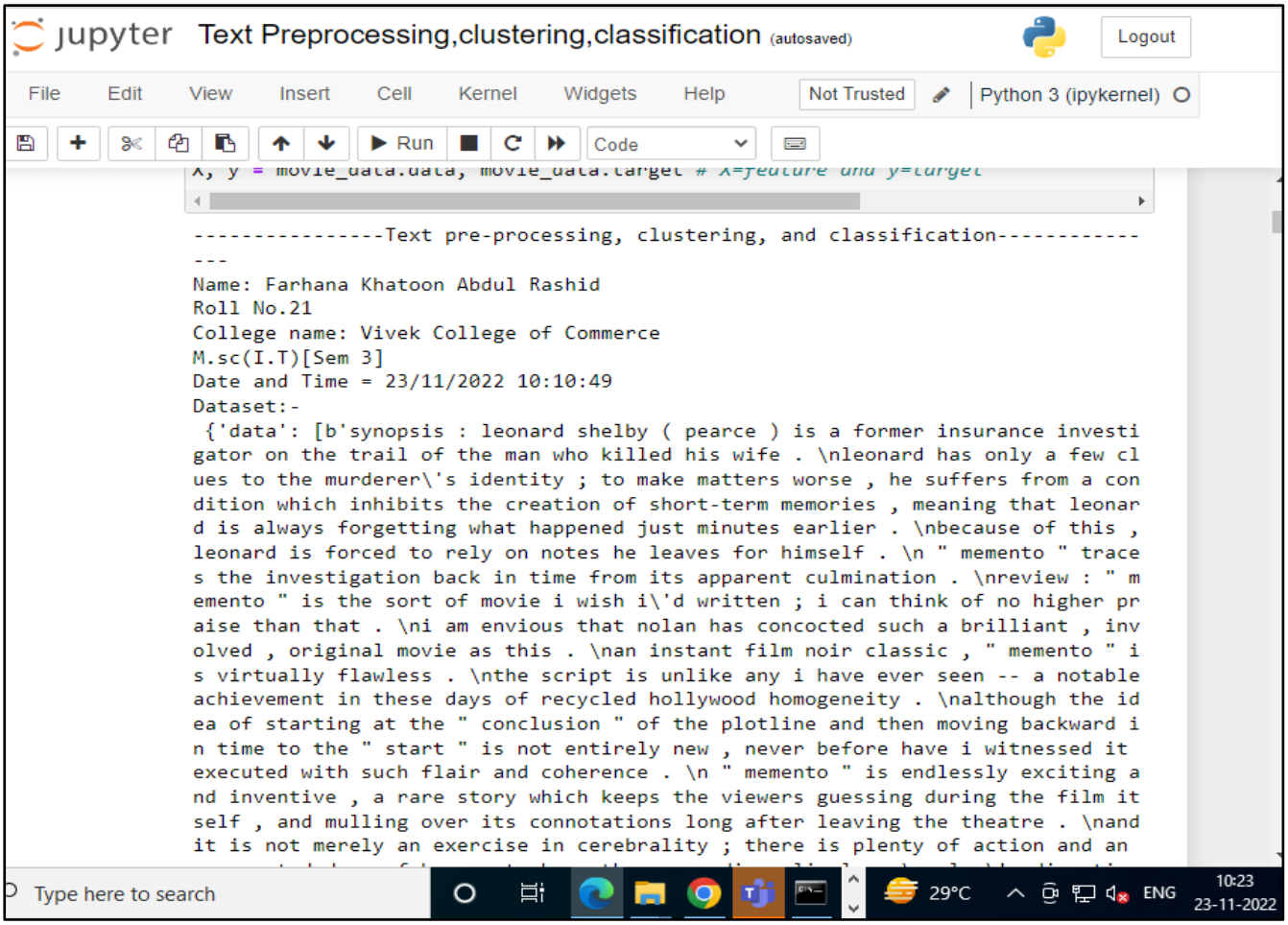
Source Code:

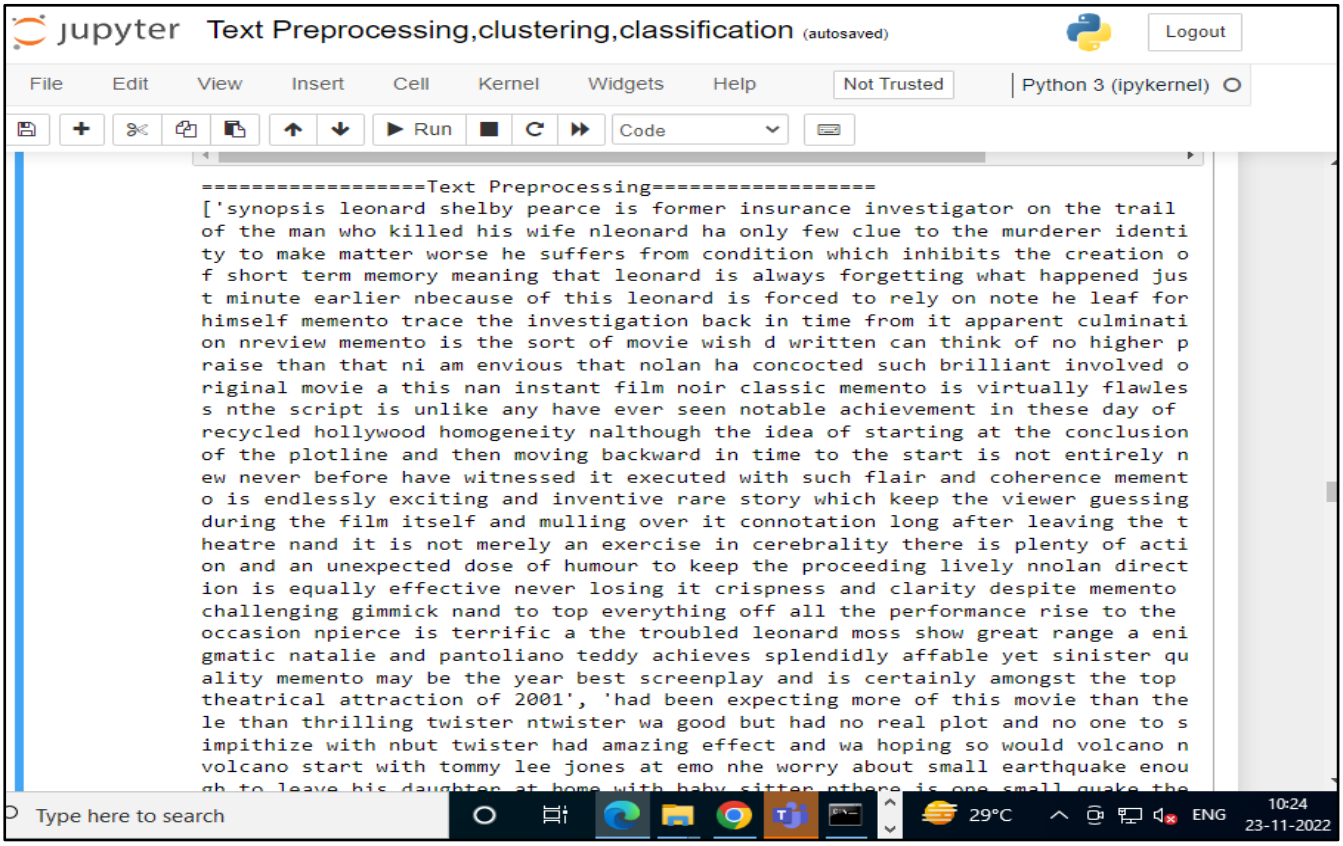
```
from datetime import datetime
print('----------------Text pre-processing, clustering, and classification----------------')
print('Name: Farhana Khatoon Abdul Rashid')
print('Roll No.21')
print('College name: Vivek College of Commerce')
print('M.sc(I.T)[Sem 3]')
now = datetime.now()
# dd/mm/YY H:M:S
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("Date and Time =", dt_string)
#Importing Libraries
import numpy as np
import re
import nltk
from sklearn.datasets import load_files
nltk.download('stopwords')
import pickle
from nltk.corpus import stopwords
#Importing the Dataset
movie_data = load_files(r"C:\Users\Mscit6\Documents\txt_sentoken\txt_sentoken")
print("\n =========Before text preprocessing========= \n")
print(movie_data)
print("\n")
X, y = movie_data.data, movie_data.target
#Text Preprocessing
documents = []
from nltk.stem import WordNetLemmatizer
stemmer = WordNetLemmatizer()
for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))
    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)
    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)
    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)
    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
```

```
    document = document.lower()
    # Lemmatization
    document = document.split()
    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)
    documents.append(document)
print("\n =========After text preprocessing========= \n")
print(documents)
print("\n")

#Converting Text to Numbers
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7,
stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()
#Finding TFIDF
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(max_features=1500, min_df=5, max_df=0.7,
stop_words=stopwords.words('english'))
X = tfidfconverter.fit_transform(documents).toarray()
#Training and Testing Sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

print("\n =========Text Classification========= \n")
#Training Text Classification Model and Predicting Sentiment
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

#Evaluating the Model
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print("\n Confusion Matrix \n")
print(confusion_matrix(y_test,y_pred))
print("\n Classification Report \n")
print(classification_report(y_test,y_pred))
print("\n Accuracy Score \n")
print(accuracy_score(y_test, y_pred))

print("\n =========Text Clustering========= \n")
# Import KMeans Model
from sklearn.cluster import KMeans
```

```
# Create Kmeans object and fit it to the training data
kmeans = KMeans(n_clusters=2).fit(X)

# Get the labels using KMeans
pred_labels = kmeans.labels_
#Evaluate Clustering Performance
from sklearn import metrics
print("\n The accuracy of KMeans Text clustering is:-", metrics.accuracy_score(y,
pred_labels))
cm=metrics.confusion_matrix(y, pred_labels)
print("\n Confusion Matrix:-", cm)
# Compute DBI score
dbi = metrics.davies_bouldin_score(X, pred_labels)
# Compute Silhoutte Score
ss = metrics.silhouette_score(X, pred_labels , metric='euclidean')
# Print the DBI and Silhoutte Scores
print("DBI Score: ", dbi, "\nSilhoutte Score: ", ss)

#Visualize Text clustering
# reduce the features to 2D
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
pca = PCA(n_components=2, random_state=0)
reduced_features = pca.fit_transform(X)
# reduce the cluster centers to 2D
reduced_cluster_centers = pca.transform(kmeans.cluster_centers_)
plt.scatter(reduced_features[:,0], reduced_features[:,1], c=kmeans.predict(X))
plt.scatter(reduced_cluster_centers[:, 0], reduced_cluster_centers[:,1], marker='x', s=150,
c='b')
plt.title("Text clustering using KMeans with PCA")
```

Output:


jupyter Text Preprocessing,clustering,classification (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
----------Text pre-processing, clustering, and classification----------
---
Name: Farhana Khatoon Abdul Rashid
Roll No.21
College name: Vivek College of Commerce
M.sc(I.T)[Sem 3]
Date and Time = 23/11/2022 10:10:49
Dataset:-
{'data': [b'synopsis : leonard shelby (pearce) is a former insurance investi
gator on the trail of the man who killed his wife . \nleonard has only a few cl
ues to the murderer\'s identity ; to make matters worse , he suffers from a con
dition which inhibits the creation of short-term memories , meaning that leonar
d is always forgetting what happened just minutes earlier . \nbecause of this ,
leonard is forced to rely on notes he leaves for himself . \n " memento " trace
s the investigation back in time from its apparent culmination . \nreview : " m
emento " is the sort of movie i wish i\'d written ; i can think of no higher pr
aise than that . \ni am envious that nolan has concocted such a brilliant , inv
olved , original movie as this . \nan instant film noir classic , " memento " i
s virtually flawless . \nthe script is unlike any i have ever seen -- a notable
achievement in these days of recycled hollywood homogeneity . \nalthough the id
ea of starting at the " conclusion " of the plotline and then moving backward i
n time to the " start " is not entirely new , never before have i witnessed it
executed with such flair and coherence . \n " memento " is endlessly exciting a
nd inventive , a rare story which keeps the viewers guessing during the film it
self , and mulling over its connotations long after leaving the theatre . \nand
it is not merely an exercise in cerebrality ; there is plenty of action and an
Type here to search
29°C
ENG
10:23
23-11-2022


jupyter Text Preprocessing,clustering,classification (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
==================Text Preprocessing==================
['synopsis leonard shelby pearce is former insurance investigator on the trail
of the man who killed his wife nleonard ha only few clue to the murderer identi
ty to make matter worse he suffers from condition which inhibits the creation o
f short term memory meaning that leonard is always forgetting what happened jus
t minute earlier nbecause of this leonard is forced to rely on note he leaf for
himself memento trace the investigation back in time from it apparent culminati
on nreview memento is the sort of movie wish d written can think of no higher p
raise than that ni am envious that nolan ha concocted such brilliant involved o
riginal movie a this nan instant film noir classic memento is virtually flawles
s nthe script is unlike any have ever seen notable achievement in these day of
recycled hollywood homogeneity nalthough the idea of starting at the conclusion
of the plotline and then moving backward in time to the start is not entirely n
ew never before have witnessed it executed with such flair and coherence mement
o is endlessly exciting and inventive rare story which keep the viewer guessing
during the film itself and mulling over it connotation long after leaving the t
heatre nand it is not merely an exercise in cerebrality there is plenty of acti
on and an unexpected dose of humour to keep the proceeding lively nnolan direct
ion is equally effective never losing it crispness and clarity despite memento
challenging gimmick nand to top everything off all the performance rise to the
occasion npierce is terrific a the troubled leonard moss show great range a eni
gmatic natalie and pantoliano teddy achieves splendidly affable yet sinister qu
ality memento may be the year best screenplay and is certainly amongst the top
theatrical attraction of 2001', 'had been expecting more of this movie than the
le than thrilling twister ntwister wa good but had no real plot and no one to s
impithize with nbut twister had amazing effect and wa hoping so would volcano n
volcano start with tommy lee jones at emo nhe worry about small earthquake enou
Type here to search
29°C
ENG
10:24
23-11-2022

jupyter Farhana_Text Last Checkpoint: 38 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

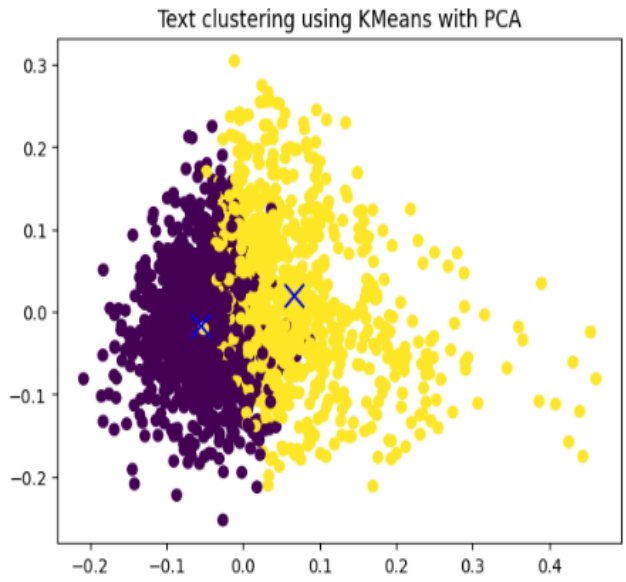
Run Code

```
=========Text Classification=========


 Confusion Matrix

[[180  28]
 [ 30 162]]

 Classification Report

              precision    recall  f1-score   support

           0       0.86      0.87      0.86       208
           1       0.85      0.84      0.85       192

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400


 Accuracy Score

0.855

 =========Text Clustering=========


 The accuracy of KMeans Text clustering is:- 0.389

 Confusion Matrix:- [[432 568]
 [654 346]]
DBI Score:  13.321745962719735
Silhoutte Score:  0.00500618387381189
```

In [ ]:

Type here to search 26°C ENG 08:36 24-11-2022

Out[15]: Text(0.5, 1.0, 'Text clustering using KMeans with PCA')



In [ ]:

Type here to search 27°C ENG 08:48 24-11-2022