WEB MINING

by Sharadindu Adhikari, 19BCE2105

K means Clustering, NAÏVE BAIYES

Kclustering

Question:7

Using K-means clustering algorithm to classify the Iris data into various classes/ clusters.
Consider both cases with K = 3 and 4;

- Use any of the Toolkits / Packages to perform the process
- Print out the Accuracy and Confusion Matrix of Classification
- Document the step by step process and upload with output and Code

Note: Dataset can be downloaded from the internet.

Please specify the source of the dataset in the documentation steps of this program.

Code:

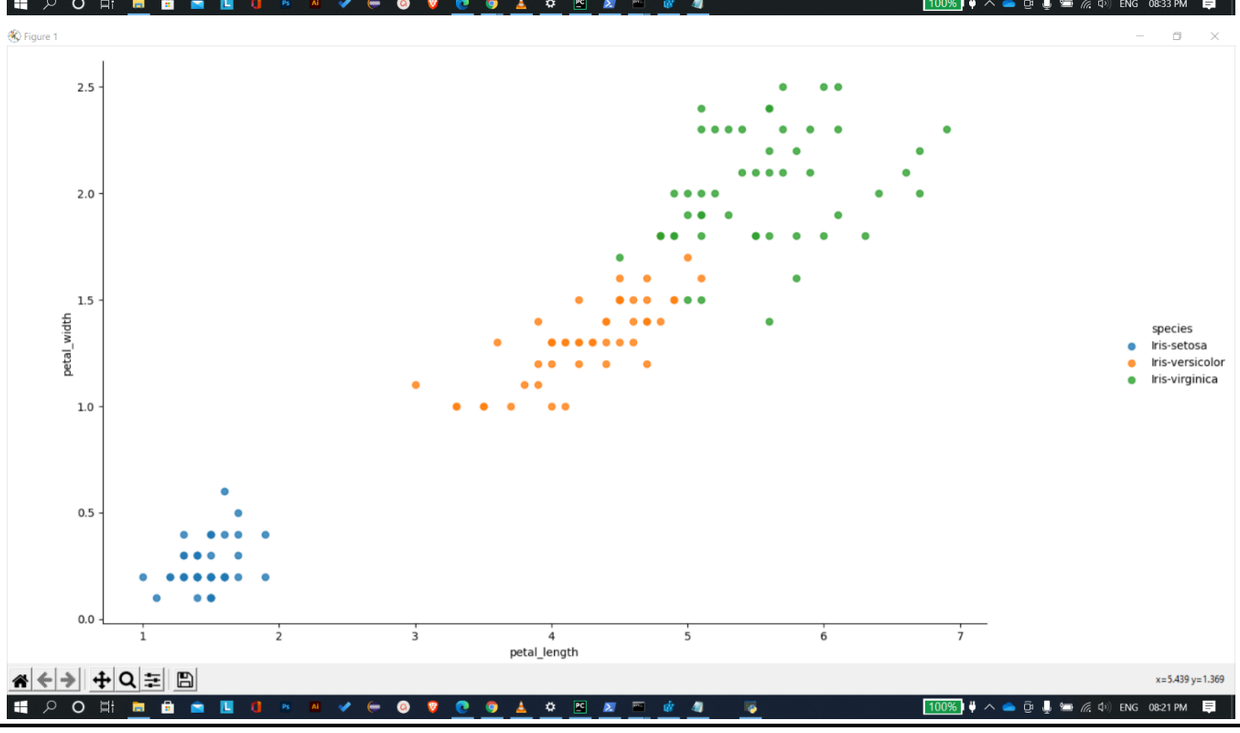
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, classification_report
warnings.filterwarnings('ignore')
df = pd.read_csv('C:\WM\IRIS.csv')#Iris dataset
sns.lmplot(x = 'petal_length', y = 'petal_width', data = df, fit_reg = False, hue =
'species', size = 6, aspect = 1)
plt.show()
from sklearn.cluster import KMeans
myKMC = KMeans(n_clusters =4 )#K-value(can be changed)
myKMC.fit(df.drop('species', axis = 1))
myKMC.cluster_centers_
df['Cluster'] = df['species'].apply(lambda x: 1 if x == 'Iris-setosa' else 0)
print("The confusion matrix is as follows:")
print(confusion_matrix(df['Cluster'], myKMC.labels_))
print(classification_report(df['Cluster'], myKMC.labels_))

#Visualising the clusters

data = pd.read_csv('C:/WM/IRIS.csv')
data.head()
X = data[["petal_length","petal_width"]]
#Visualise data points
plt.scatter(X["petal_width"],X["petal_length"],c='black')
plt.xlabel('petal_width')
plt.ylabel('petal_length')
plt.show()
K=4#K-value(can be changed)

# Select random observation as centroids
Centroids = (X.sample(n=K))
plt.scatter(X["petal_width"],X["petal_length"],c='black')
plt.scatter(Centroids["petal_width"],Centroids["petal_length"],c='red')
plt.xlabel('petal_width')
plt.ylabel('petal_length')
plt.show()
diff = 1
j=0
while(diff!=0):
    XD=X
    i=1
    for index1,row_c in Centroids.iterrows():
        ED=[]
        for index2,row_d in XD.iterrows():
            d1=(row_c["petal_width"]-row_d["petal_width"])**2
            d2=(row_c["petal_length"]-row_d["petal_length"])**2
            d=np.sqrt(d1+d2)
            ED.append(d)
        X[i]=ED
        i=i+1

    C=[]
    for index,row in X.iterrows():
```

```
        min_dist=row[1]
        pos=1
        for i in range(K):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                pos=i+1
        C.append(pos)
    X["Cluster"]=C
    Centroids_new = X.groupby(["Cluster"]).mean()[["petal_length","petal_width"]]
    if j == 0:
        diff=1
        j=j+1
    else:
        diff = (Centroids_new['petal_length'] - Centroids['petal_length']).sum() +
(Centroids_new['petal_width'] - Centroids['petal_width']).sum()
    Centroids = X.groupby(["Cluster"]).mean()[["petal_length","petal_width"]]
    color=['blue','green','cyan','magenta']
for k in range(K):
    data=X[X["Cluster"]==k+1]
    plt.scatter(data["petal_width"],data["petal_length"],c=color[k])
plt.scatter(Centroids["petal_width"],Centroids["petal_length"],c='red')
plt.xlabel('petal_width')
plt.ylabel('petal_length')
plt.show()
```

OUTPUT:

- **For K = 3**

```
C:\Users\shara\WM\Lab7>python Kclustering.py
The confusion matrix is as follows:
[[62  0 38]
 [ 0 50  0]
 [ 0  0  0]]
              precision    recall  f1-score   support

           0       1.00      0.62      0.77       100
           1       1.00      1.00      1.00        50
           2       0.00      0.00      0.00         0

    accuracy                           0.75       150
   macro avg       0.67      0.54      0.59       150
weighted avg       1.00      0.75      0.84       150
```



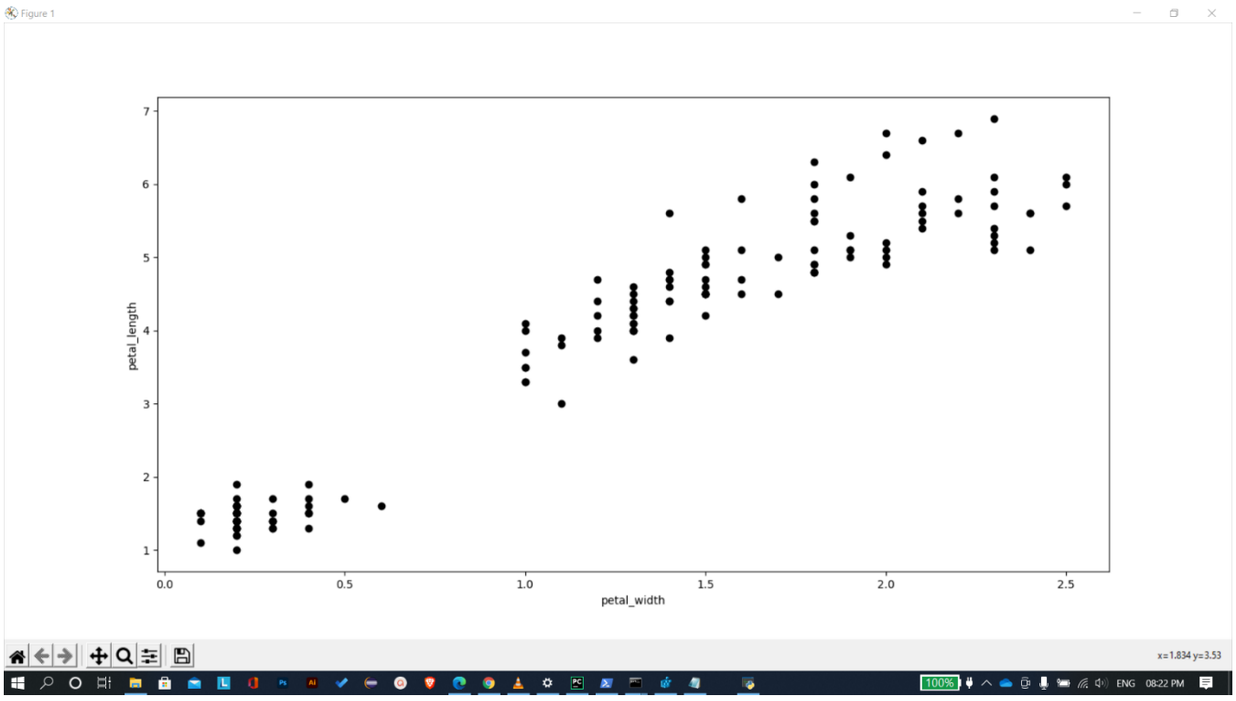


Figure 1
petal_length
petal_width
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
x=1.834 y=3.53
100%
ENG
08:22 PM

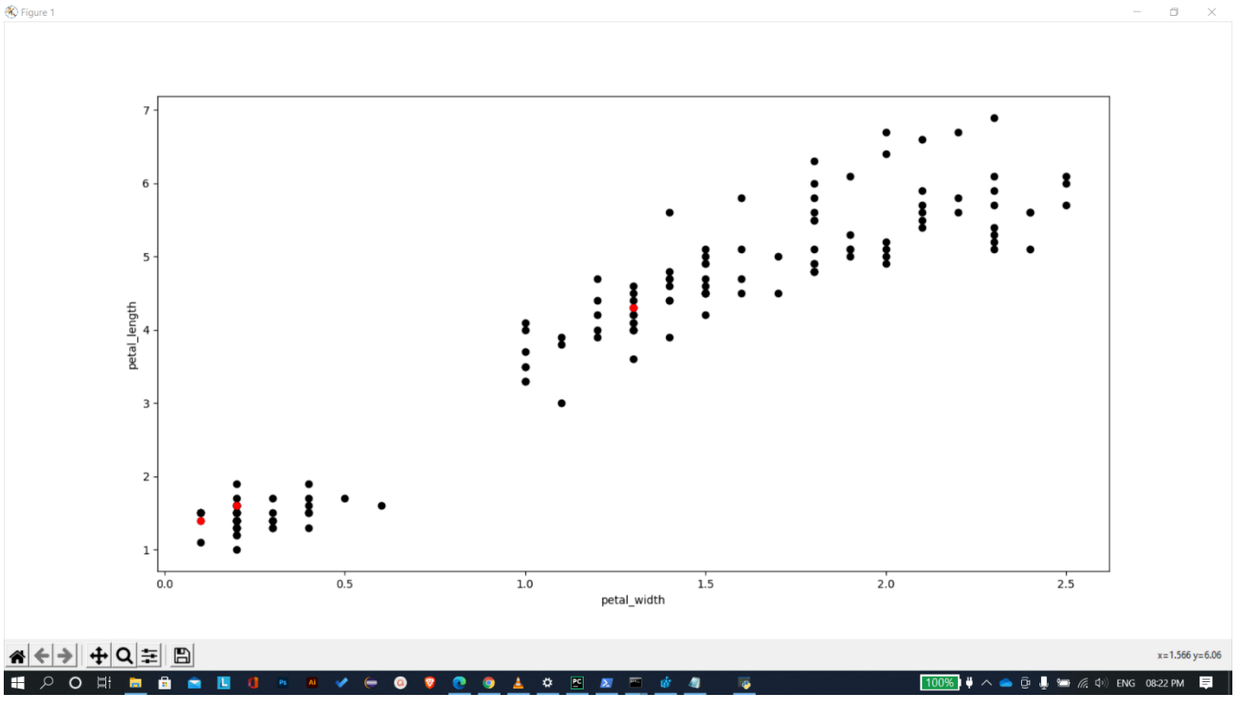


Figure 1
petal_length
petal_width
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
x=1.566 y=6.06
100%
ENG
08:22 PM

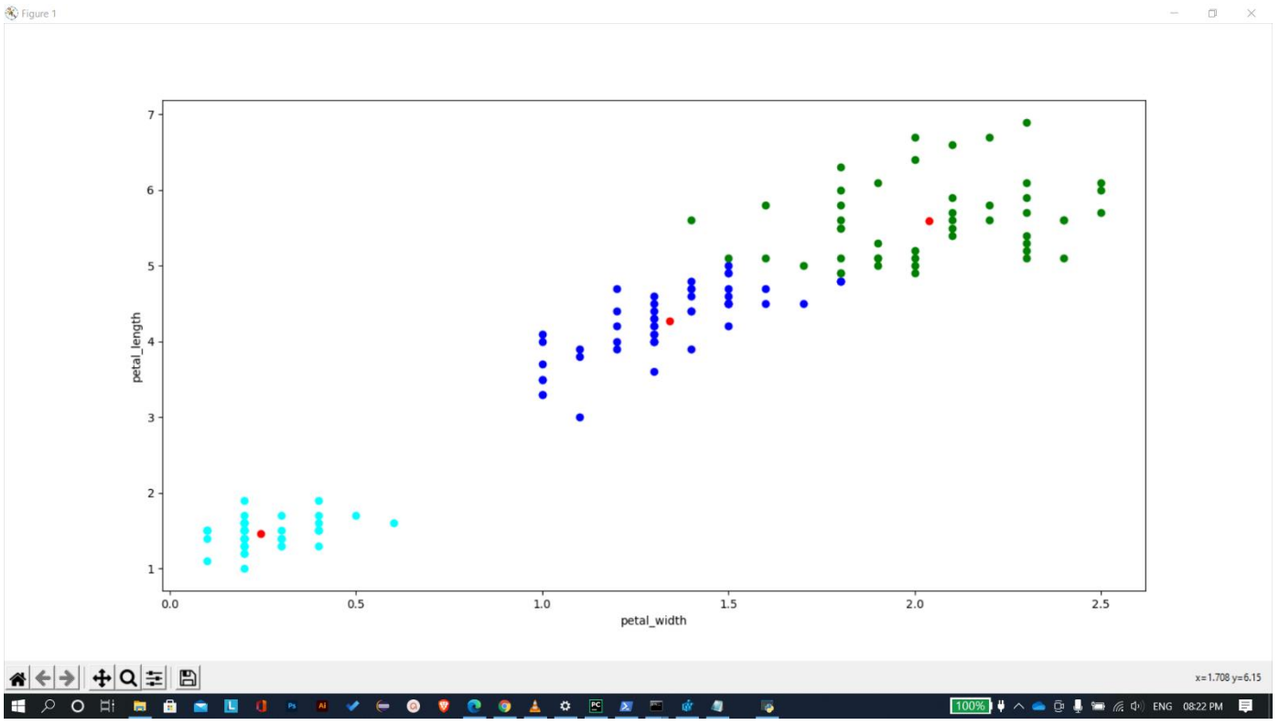
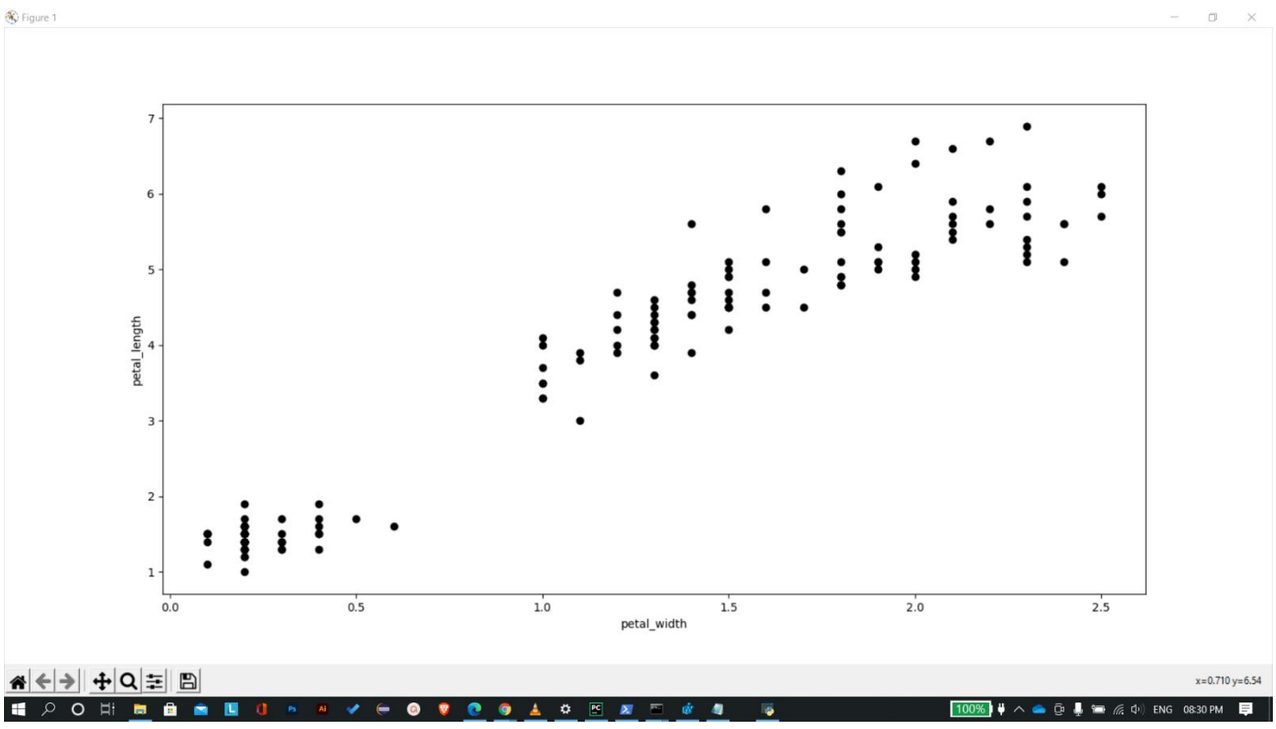
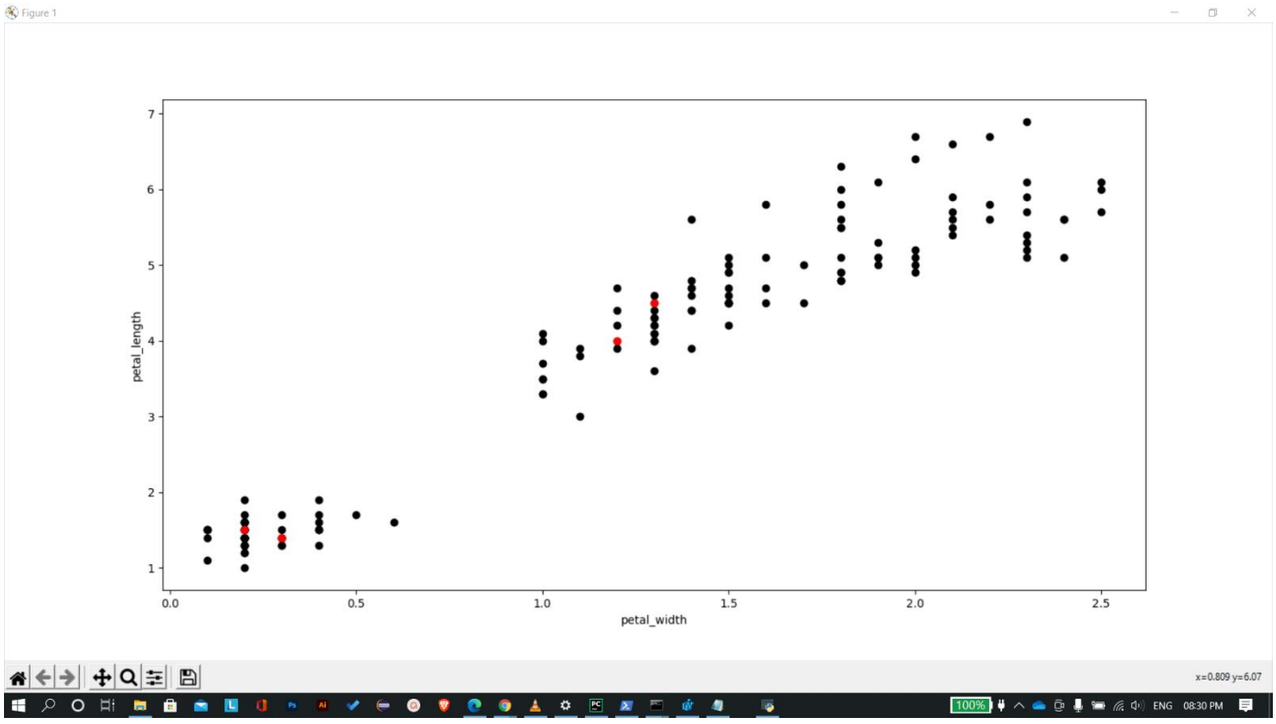


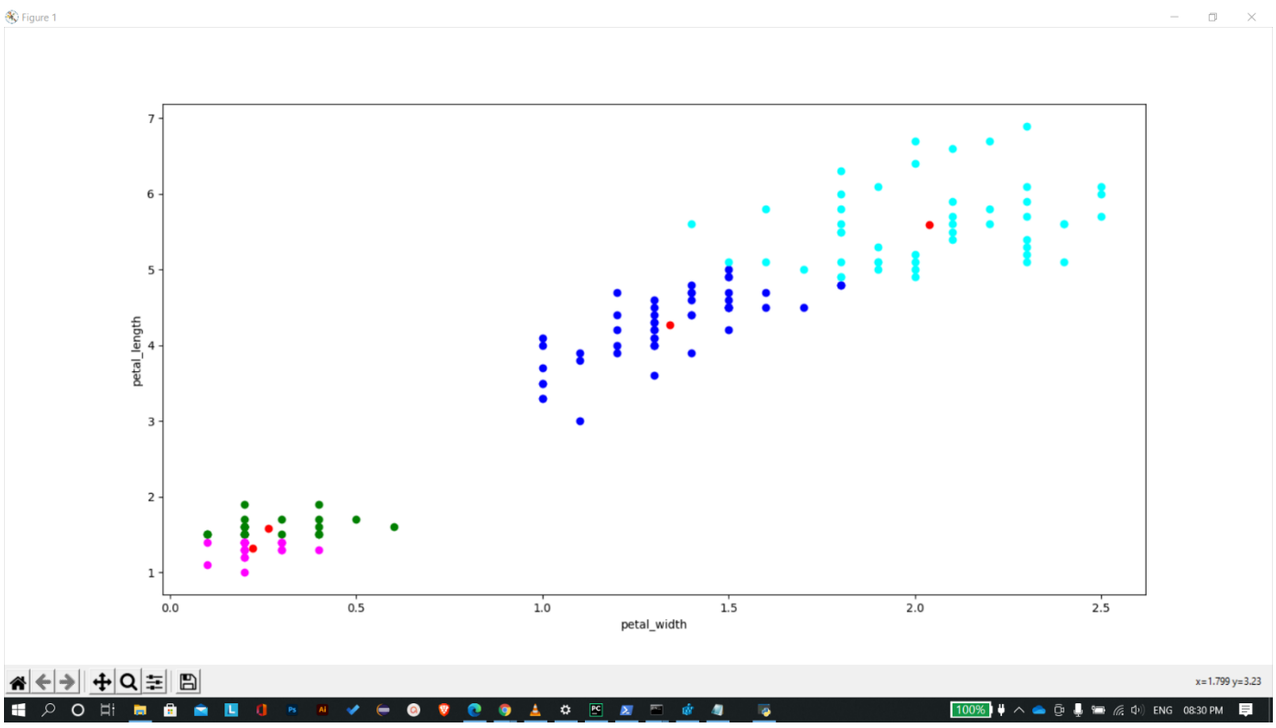
Figure 1
petal_length
petal_width
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
x=1.708 y=6.15
100%
ENG
08:22 PM

- **For K = 4**

```
Command Prompt

C:\Users\shara\WM\Lab7>python Kclustering.py
The confusion matrix is as follows:
[[32  0 28 40]
 [ 0 50  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
              precision    recall  f1-score   support

           0       1.00      0.32      0.48       100
           1       1.00      1.00      1.00        50
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0

    accuracy                           0.55       150
   macro avg       0.50      0.33      0.37       150
weighted avg       1.00      0.55      0.66       150
```







Documentation:

1. The libraries are imported first, followed by the dataset. Thereafter target names were identified. Scatter plot between petal length and petal width are plotted, applying K mean algorithm for the iris dataset.
2. In the code above, only lines 13, 31 and 72 needs changes on transforming from K = 3 to K = 4. Except for them, everything remains same.
3. In both K = 3 & 4,
   - Graph 1 represents the scatter plot of the CSV file;
   - Graph 2 denotes scatter plot without highlighting cluster centres;
   - Graph 3 represents scatter plot with highlighted cluster centres;
   - Graph 4 is the final representation of the clusters.
4. Iris dataset: https://gist.github.com/shara-d/f208ddda6d82695f90a90d3038e0aaec

NAÏVE BAIYES

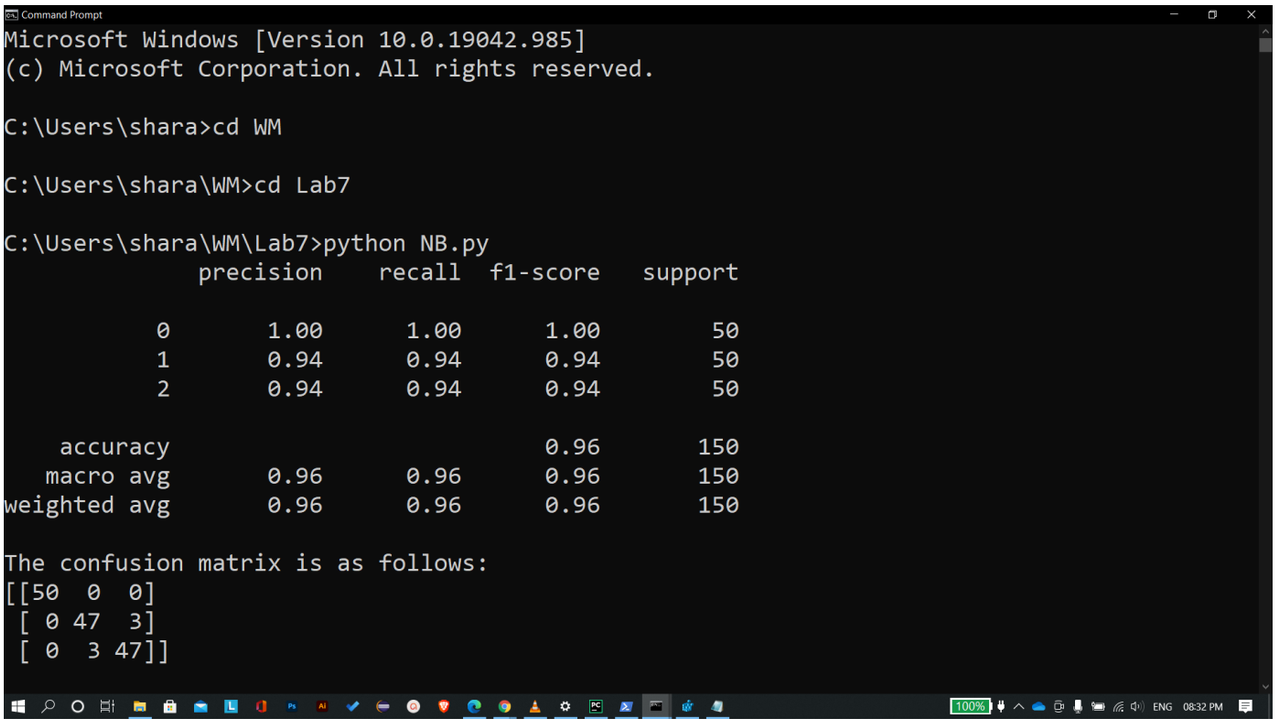
Write a code to build a Navie Bayes Classifier for categorising the flowers collected from Iris Data Set into

a) Use any of the Toolkit / Package to perform the process
b) Print out the Accuracy and Confusion Matrix of Classification
c) Document the step by step process and upload with output and Code

Code:

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
import pandas as pd
import numpy as np
import random as rd
import matplotlib.pyplot as plt
dataset = datasets.load_iris()#using the iris dataset
model = GaussianNB()#applying gaussian probability density function
model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))#accuracy report
print("The confusion matrix is as follows:")
print(metrics.confusion_matrix(expected, predicted))#confusion matrix
```

Output:



Documentation:

1. All the required libraries (`numpy, pandas, matplotib`) are imported, followed by the dataset.
2. The dataset thus has been split, feature scaled (using `sklearn`), followed by training the naïve bayes classification model.
3. Finally predicting the test dataset results, formation of confusion matrix, and accuracy of the model (here, 96%).