

CSE 3024

Web Mining

LAB ASSESSMENT - 6

NAME: Vibhu Kumar Singh

REG. NO: 19BCE0215

TEACHER: Mr. Hiteshwar Kumar Azad

1. Create a Python programme to implement the decision tree and prints the accuracy percentage, precision, recall and the predicted values.

Ans 1.

HANDWRITTEN CODE:

```
VIBHU KUMAR SINGH 19BCE0215
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
:
import io

uploaded = files.upload()

balance_data = pd.read_csv(io.BytesIO(uploaded['1.csv']))

print("Dataset Length: ", len(balance_data))
print("Dataset Shape: ", balance_data.shape)

print("Dataset: ", balance_data.head())

X = balance_data.iloc[:, 1:5].values
Y = balance_data.iloc[:, 5].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size
= 0.5, random_state = 0)

clf_entropy = DecisionTreeClassifier(criterion = "entropy",
random_state = 0)

clf_entropy.fit(X_train, Y_train)

y_pred = clf_entropy.predict(X_test)

print("Confusion Matrix: \n", confusion_matrix(Y_test, y_pred))

print("Accuracy: %", classification_report(Y_test, y_pred)*100)
print("Report: \n", (Y_test, y_pred))
print("Prediction: \n", y_pred)
```

CODE:

```
#NAME: VIBHU KUMAR SINGH
#ROLL NO: 19BCE0215
#WEB MINING

import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import io
from google.colab import files

uploaded = files.upload()

balance_data = pd.read_csv(io.BytesIO(uploaded['Iris.csv']))

print("Dataset Length: ", len(balance_data))
print("Dataset Shape: ", balance_data.shape)

print("Dataset: ", balance_data.head())

X = balance_data.iloc[:, 1:5].values
Y = balance_data.iloc[:, 5].values

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5, random_state=0)

clf_entropy = DecisionTreeClassifier(criterion="entropy", random_state=0)

clf_entropy.fit(X_train, y_train)

y_pred = clf_entropy.predict(X_test)

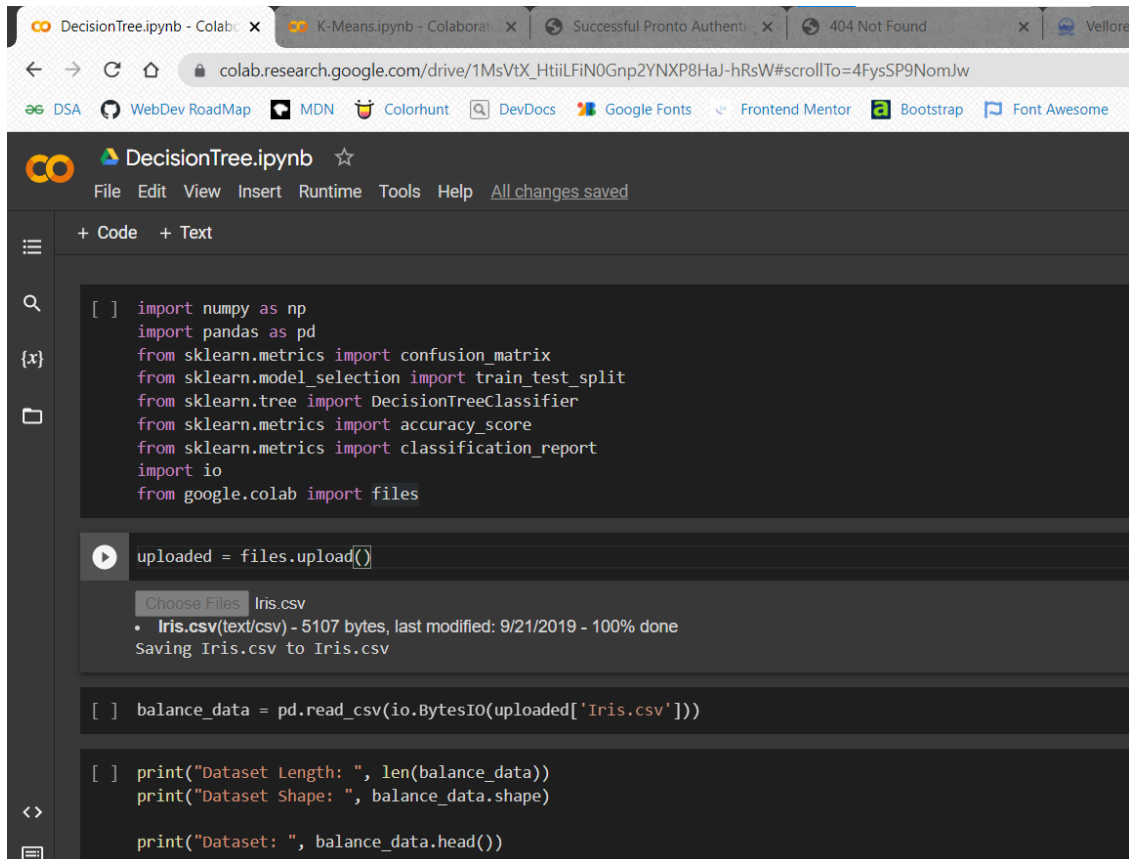
print("Confusion Matrix: \n",
      confusion_matrix(y_test, y_pred))

print("Accuracy : \n",
      accuracy_score(y_test, y_pred)*100)

print("Report : \n",
      classification_report(y_test, y_pred))

print("Prediction: \n", y_pred)
```

CODE SCREENSHOT:



```
[ ] import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import io
from google.colab import files

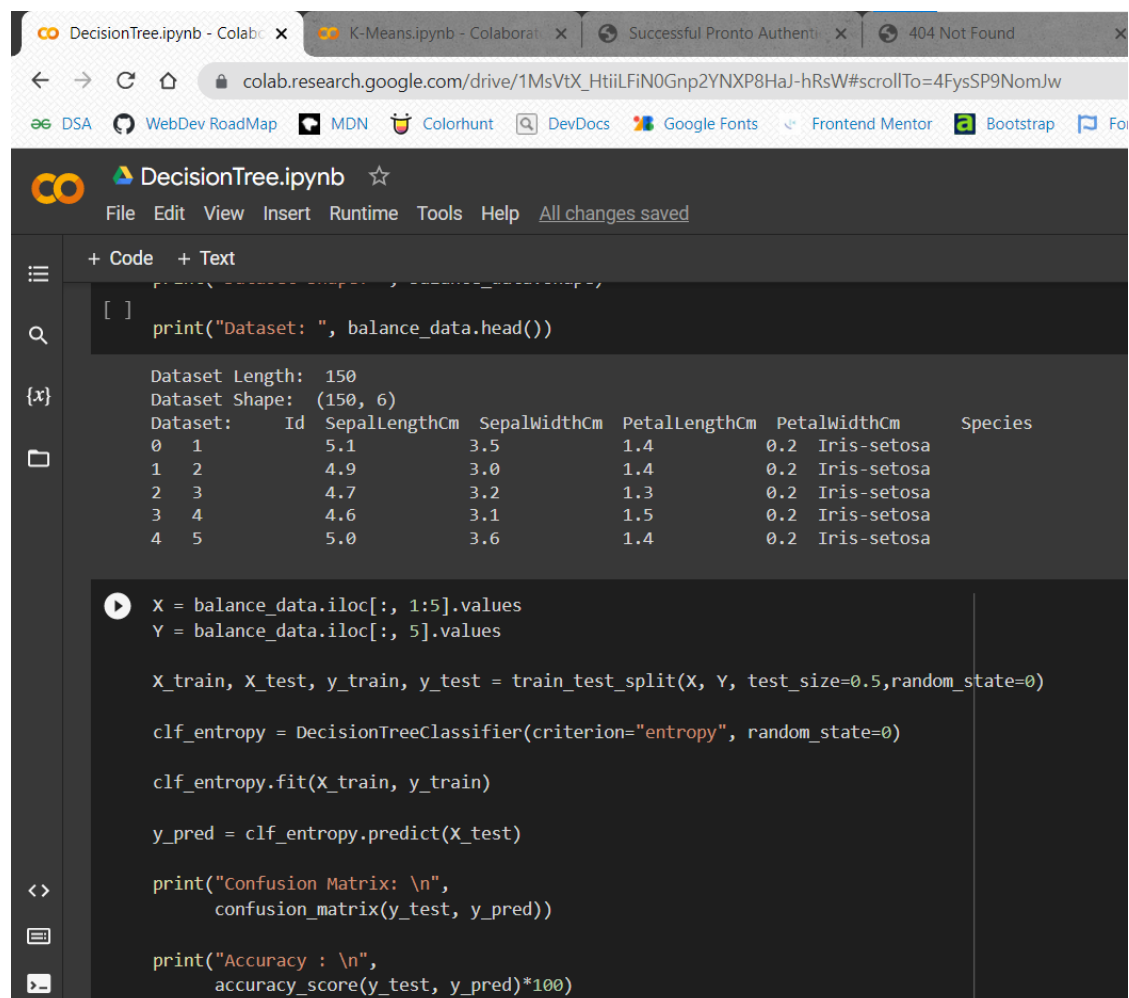
uploaded = files.upload()

Choose Files Iris.csv
• Iris.csv(text/csv) - 5107 bytes, last modified: 9/21/2019 - 100% done
Saving Iris.csv to Iris.csv

[ ] balance_data = pd.read_csv(io.BytesIO(uploaded['Iris.csv']))

[ ] print("Dataset Length: ", len(balance_data))
print("Dataset Shape: ", balance_data.shape)

print("Dataset: ", balance_data.head())
```



```
[ ] print("Dataset: ", balance_data.head())

Dataset Length: 150
Dataset Shape: (150, 6)
Dataset:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1             5.1             3.5             1.4             0.2  Iris-setosa
1   2             4.9             3.0             1.4             0.2  Iris-setosa
2   3             4.7             3.2             1.3             0.2  Iris-setosa
3   4             4.6             3.1             1.5             0.2  Iris-setosa
4   5             5.0             3.6             1.4             0.2  Iris-setosa

X = balance_data.iloc[:, 1:5].values
Y = balance_data.iloc[:, 5].values

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5, random_state=0)

clf_entropy = DecisionTreeClassifier(criterion="entropy", random_state=0)

clf_entropy.fit(X_train, y_train)

y_pred = clf_entropy.predict(X_test)

print("Confusion Matrix: \n",
      confusion_matrix(y_test, y_pred))

print("Accuracy : \n",
      accuracy_score(y_test, y_pred)*100)
```

DecisionTree.ipynb - ColabK-Means.ipynb - ColaboratorySuccessful Pronto Authent404 Not Found

colab.research.google.com/drive/1MsVtX_HtiLFiN0Gnp2YNXP8HaJ-hRsW#scrollTo=4FysSP9NomJw

DSAWebDev RoadMapMDNColorhuntDevDocsGoogle FontsFrontend MentorBootstrapFo

DecisionTree.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
print("Report : \n",
      classification_report(y_test, y_pred))

print("Prediction: \n", y_pred)
```

Confusion Matrix:
[[21 0 0]
 [0 29 1]
 [0 2 22]]
Accuracy :
96.0
Report :

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	21
Iris-versicolor	0.94	0.97	0.95	30
Iris-virginica	0.96	0.92	0.94	24
accuracy			0.96	75
macro avg	0.96	0.96	0.96	75
weighted avg	0.96	0.96	0.96	75

Prediction:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor']

0s completed at 20:13

DecisionTree.ipynb - ColabK-Means.ipynb - ColaboratorySuccessful Pronto Authent404 Not Found

colab.research.google.com/drive/1MsVtX_HtiLFiN0Gnp2YNXP8HaJ-hRsW#scrollTo=4FysSP9NomJw

DSAWebDev RoadMapMDNColorhuntDevDocsGoogle FontsFrontend MentorBootstrapFo

DecisionTree.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
Iris-setosa      1.00      1.00      1.00      21
Iris-versicolor  0.94      0.97      0.95      30
Iris-virginica   0.96      0.92      0.94      24

accuracy         0.96
macro avg        0.96
weighted avg     0.96
```

Prediction:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor']

OUTPUT:

Confusion Matrix:

```
[[21  0  0]
 [ 0 29  1]
 [ 0  2 22]]
```

Accuracy :

96.0

Report :

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	21
Iris-versicolor	0.94	0.97	0.95	30
Iris-virginica	0.96	0.92	0.94	24
accuracy		0.96		75
macro avg	0.96	0.96	0.96	75
weighted avg	0.96	0.96	0.96	75

Prediction:

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-setosa' 'Iris-versicolor']
```

OUTPUT SCREENSHOT:

```
Confusion Matrix:
[[21  0  0]
 [ 0 29  1]
 [ 0  2 22]]
Accuracy :
96.0
Report :
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	21
Iris-versicolor	0.94	0.97	0.95	30
Iris-virginica	0.96	0.92	0.94	24
accuracy			0.96	75
macro avg	0.96	0.96	0.96	75
weighted avg	0.96	0.96	0.96	75

```
Prediction:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-setosa' 'Iris-versicolor']
```

(P.T.O.)

2. Create a Python programme that uses the K-means clustering algorithm and displays all clusters in different colours.

Ans 2.

HANDWRITTEN CODE:

```

VIBHU KUMAR SINGH 19BCF0215

import numpy as np
...
import io

names = ['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
         'PetalWidthCm', 'Species']

upload = files.upload()
dataset = pd.read_csv(io.StringIO(upload['iris.csv']),
                      decode('utf-8'), names = names)

X = dataset[pd.read_csviloc[1:, [2,3]].values]

wcss_list = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++',
                    random_state=42)
    kmeans.fit(X)
    wcss_list.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss_list)
plt.title("The Elbow Graph Method")
plt.xlabel("Number of Clusters")
plt.ylabel("wcss_list")
plt.show()
:
:

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_
            centers_[:, 1], s=300, c="yellow", label='Centroid')

plt.title('clusters')
```


CODE:

```
#NAME: VIBHU KUMAR SINGH
#ROLL NO: 19BCE0215
#WEB MINING

import numpy as np
import pandas as pd
import matplotlib.pyplot as mtp
from sklearn.cluster import KMeans
from google.colab import files
import io

names = ['Id', 'SepalLengthCm', 'SepalWidthCm',
         'PetalLengthCm', 'PetalWidthCm', 'Species']
upload = files.upload()
dataset = pd.read_csv(io.StringIO(upload['Iris.csv'].decode('utf-8')), names=names)

x = dataset.iloc[1:, [2, 3]].values

wcss_list = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)

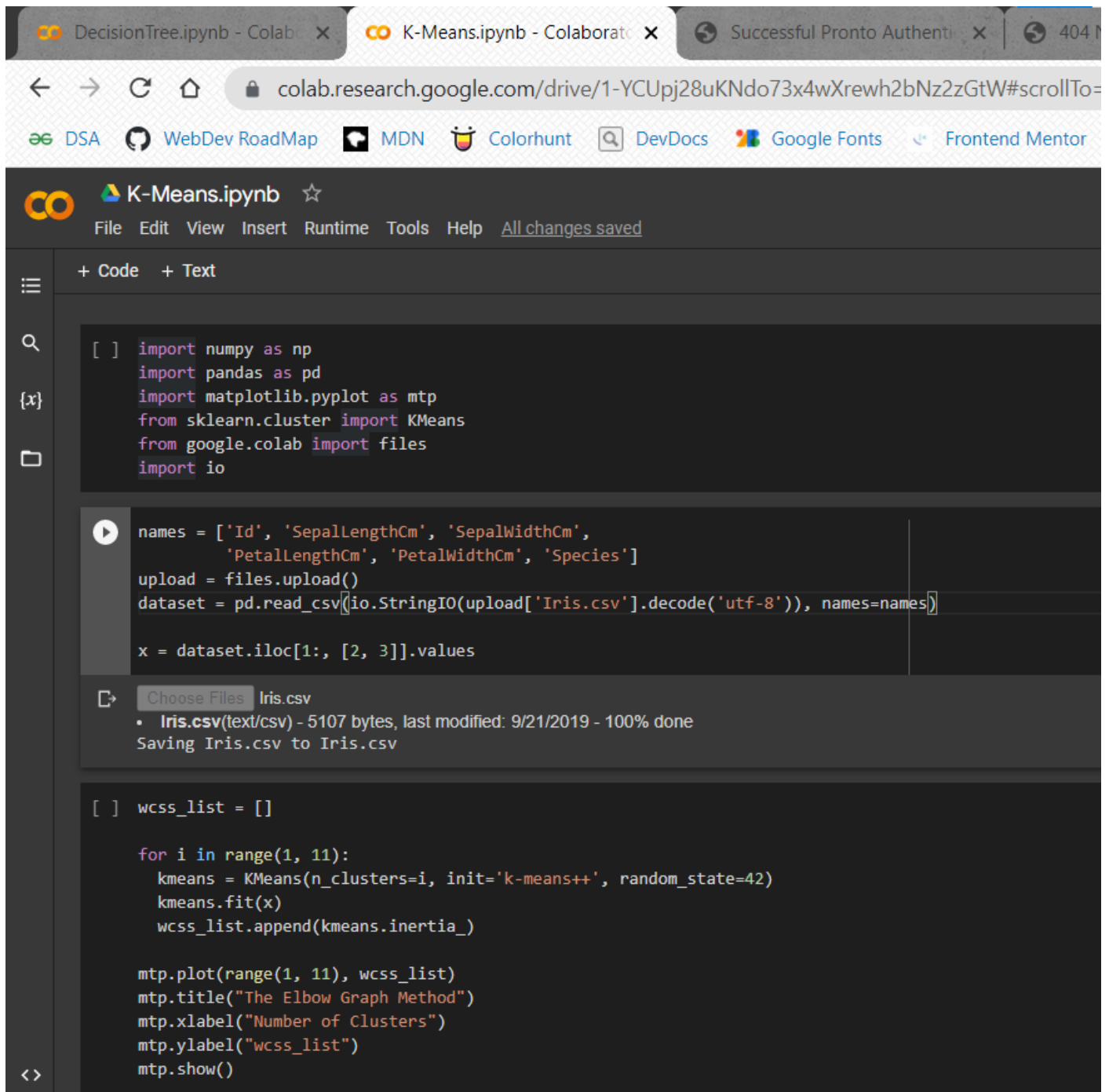
mtp.plot(range(1, 11), wcss_list)
mtp.title("The Elbow Graph Method")
mtp.xlabel("Number of Clusters")
mtp.ylabel("wcss_list")
mtp.show()

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
y_predict = kmeans.fit_predict(x)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1],
            s=100, c='magenta', label='Cluster1')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1],
            s=100, c='cyan', label='Cluster2')
mtp.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1],
            s=100, c='red', label='Cluster3')
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1],
            s=100, c='blue', label='Cluster4')

mtp.scatter(kmeans.cluster_centers_[0, 0], kmeans.cluster_centers_[
    0, 1], s=300, c='yellow', label='Centroid')
mtp.title('Clusters')
```

CODE SCREENSHOT:



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'DecisionTree.ipynb - Colab', 'K-Means.ipynb - Colaborat', and 'Successful Pronto Authenti'. The address bar shows the URL 'colab.research.google.com/drive/1-YCUpj28uKNdo73x4wXrewh2bNz2zGtW#scrollTo='. The notebook title is 'K-Means.ipynb' with a star icon. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status 'All changes saved'. The left sidebar has icons for a menu, search, code editor, and file explorer. The code editor shows the following code:

```
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as mtp
from sklearn.cluster import KMeans
from google.colab import files
import io

names = ['Id', 'SepalLengthCm', 'SepalWidthCm',
         'PetalLengthCm', 'PetalWidthCm', 'Species']
upload = files.upload()
dataset = pd.read_csv(io.StringIO(upload['Iris.csv'].decode('utf-8')), names=names)

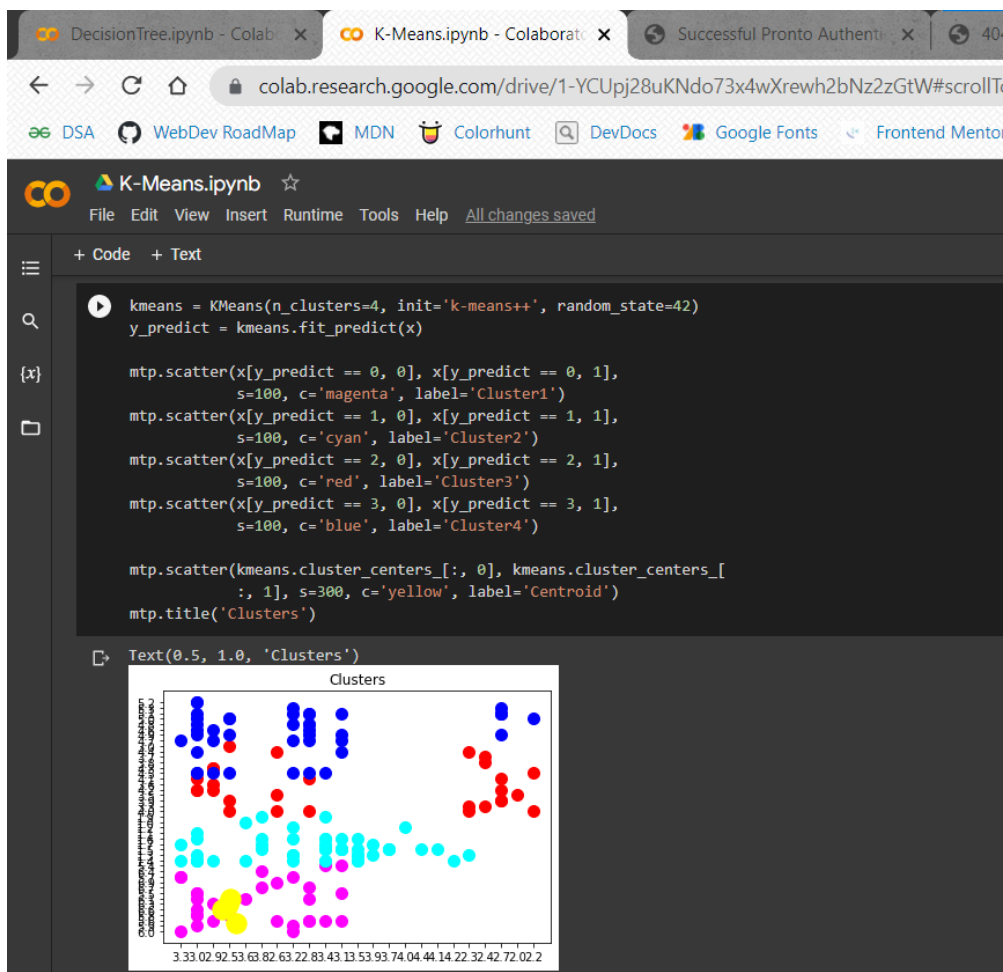
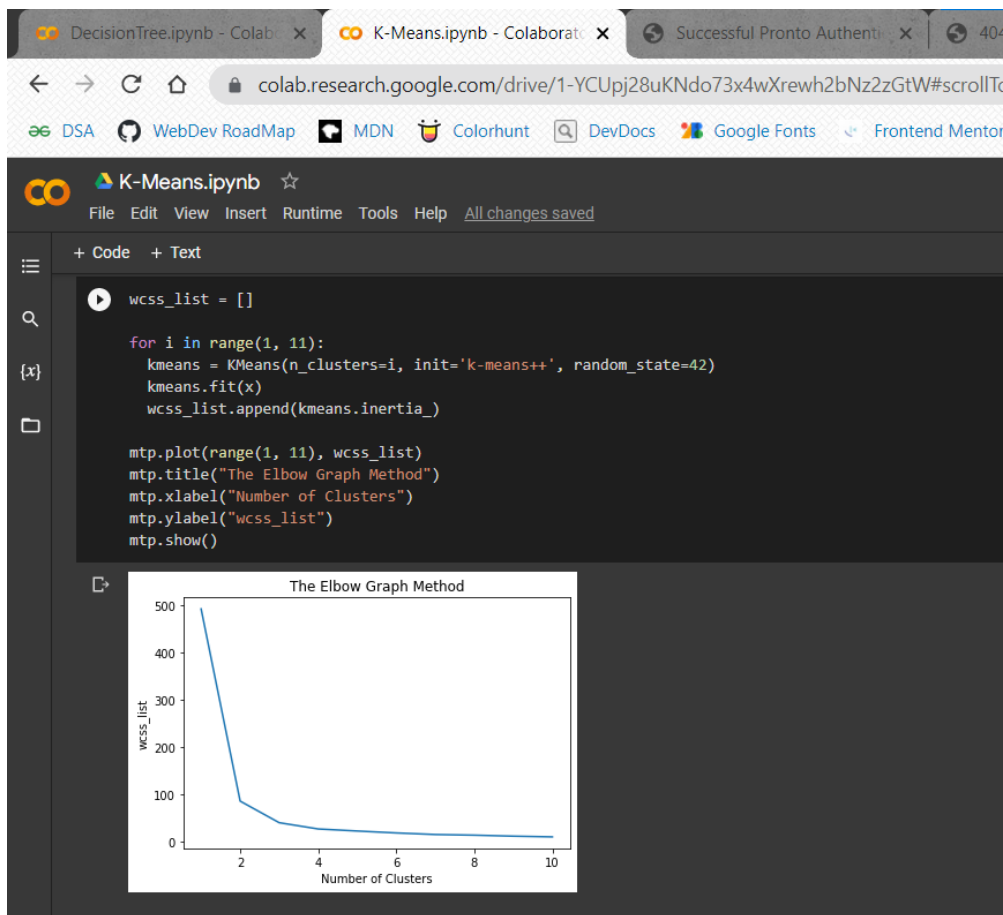
x = dataset.iloc[1:, [2, 3]].values

[ ] wcss_list = []

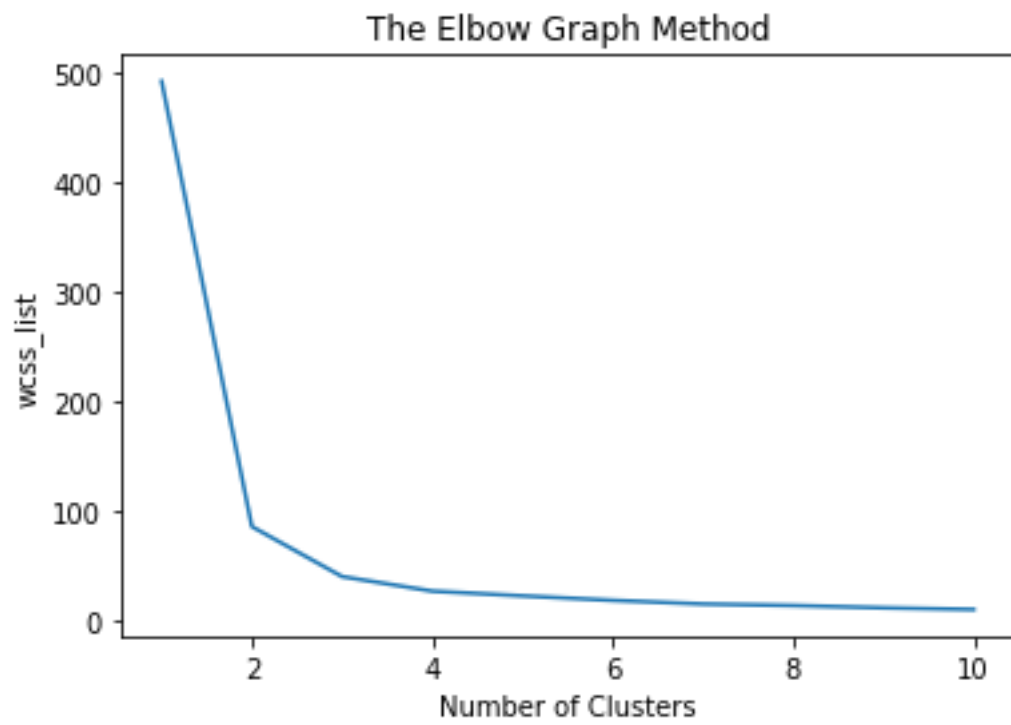
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)
mtp.title("The Elbow Graph Method")
mtp.xlabel("Number of Clusters")
mtp.ylabel("wcss_list")
mtp.show()
```

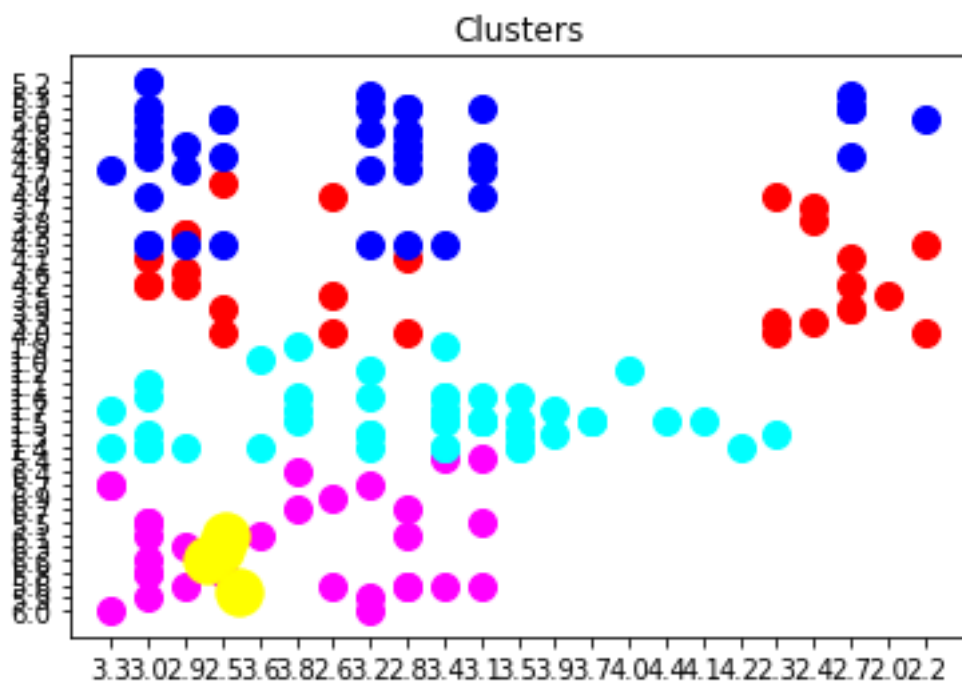
Below the code, a file upload dialog is shown with the title 'Choose Files Iris.csv'. It lists the file 'Iris.csv(text/csv) - 5107 bytes, last modified: 9/21/2019 - 100% done' and shows the status 'Saving Iris.csv to Iris.csv'.



OUTPUT SCREENSHOT:



(The Elbow Graph Method)



(Clusters)