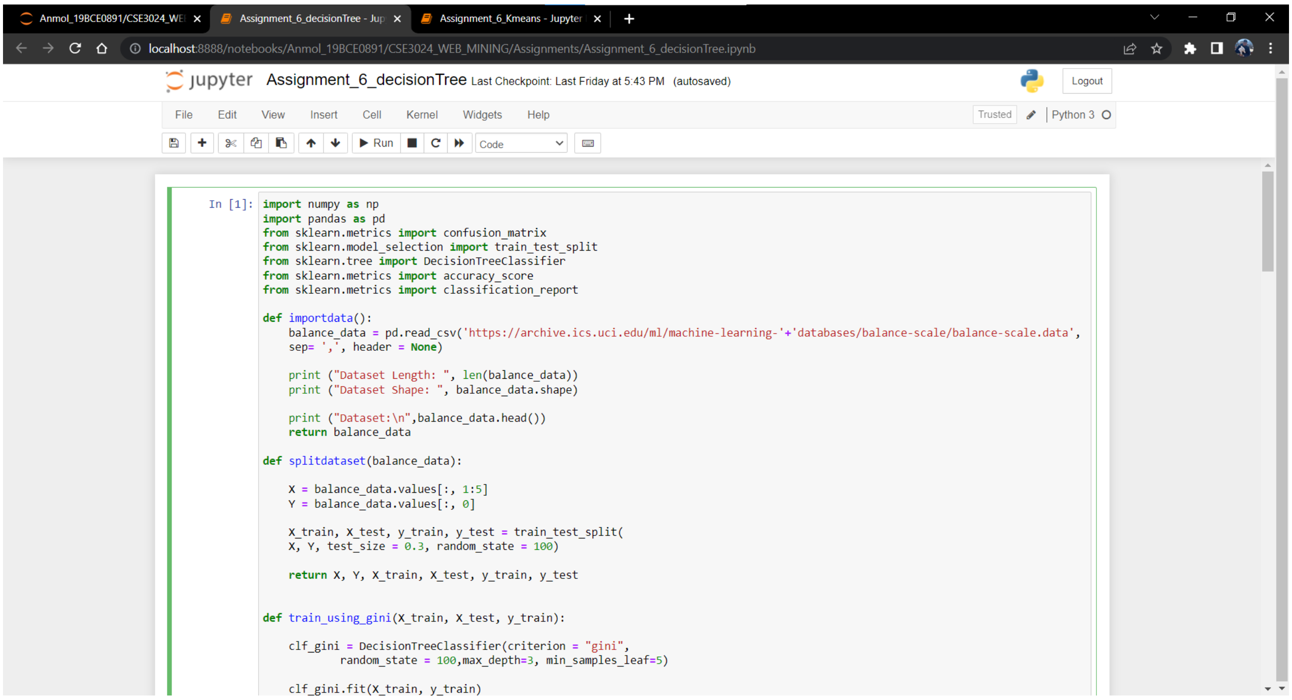
CSE3024 - WEB MINING (L41 + L42)

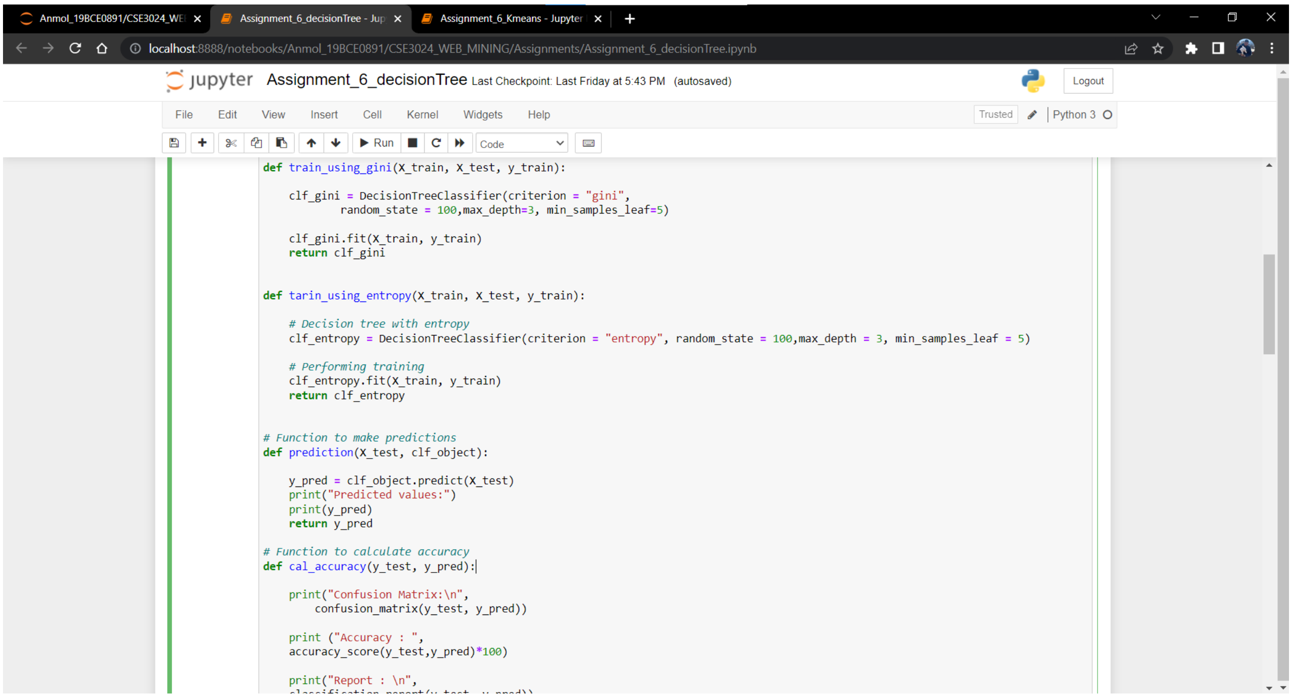
NAME – ANMOL
REG. NO. - 19BCE0891

DIGITAL ASSIGNMENT – 6

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

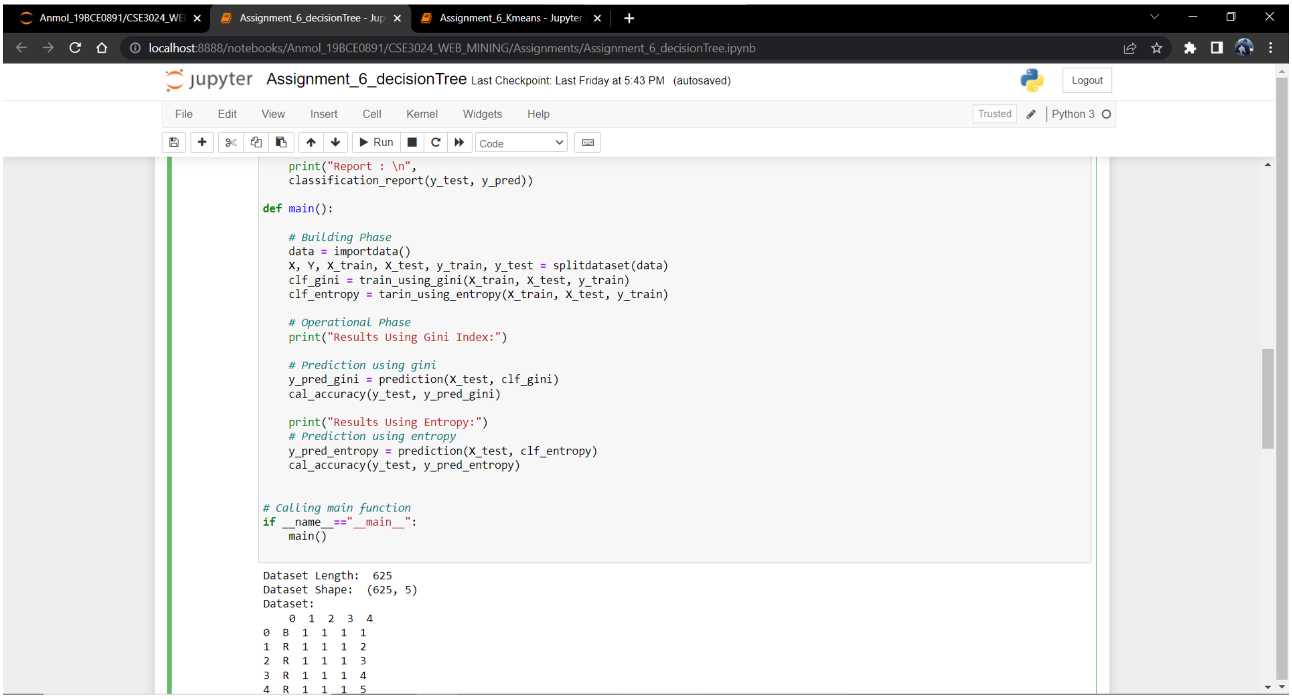

Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_decisionTree.ipynb
jupyter
Assignment_6_decisionTree
Last Checkpoint: Last Friday at 5:43 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
In [1]:

```python
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

def importdata():
    balance_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-'+'databases/balance-scale/balance-scale.data',
    sep= ',', header = None)

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

    print ("Dataset:\n",balance_data.head())
    return balance_data

def splitdataset(balance_data):

    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test


def train_using_gini(X_train, X_test, y_train):

    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    clf_gini.fit(X_train, y_train)
```


Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_decisionTree.ipynb
jupyter
Assignment_6_decisionTree
Last Checkpoint: Last Friday at 5:43 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

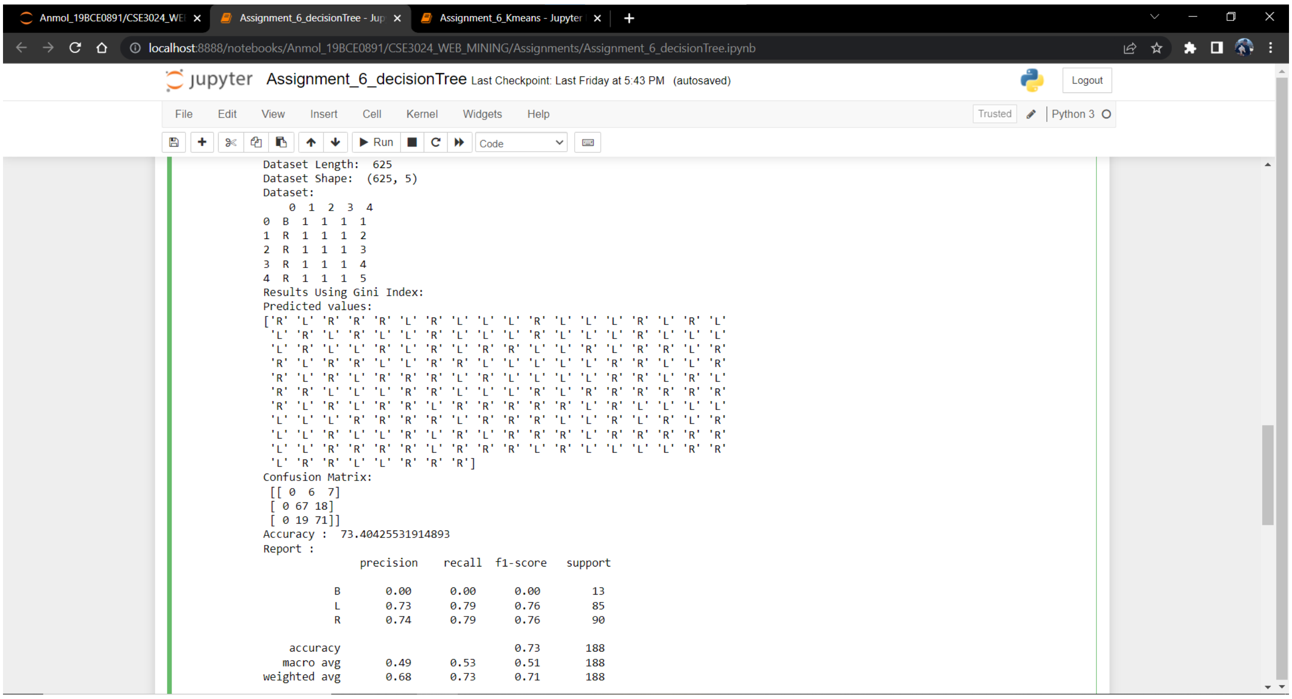
```python
def train_using_gini(X_train, X_test, y_train):

    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    clf_gini.fit(X_train, y_train)
    return clf_gini


def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,max_depth = 3, min_samples_leaf = 5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy


# Function to make predictions
def prediction(X_test, clf_object):

    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

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

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

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

CSE3024 - WEB MINING (L41 + L42)

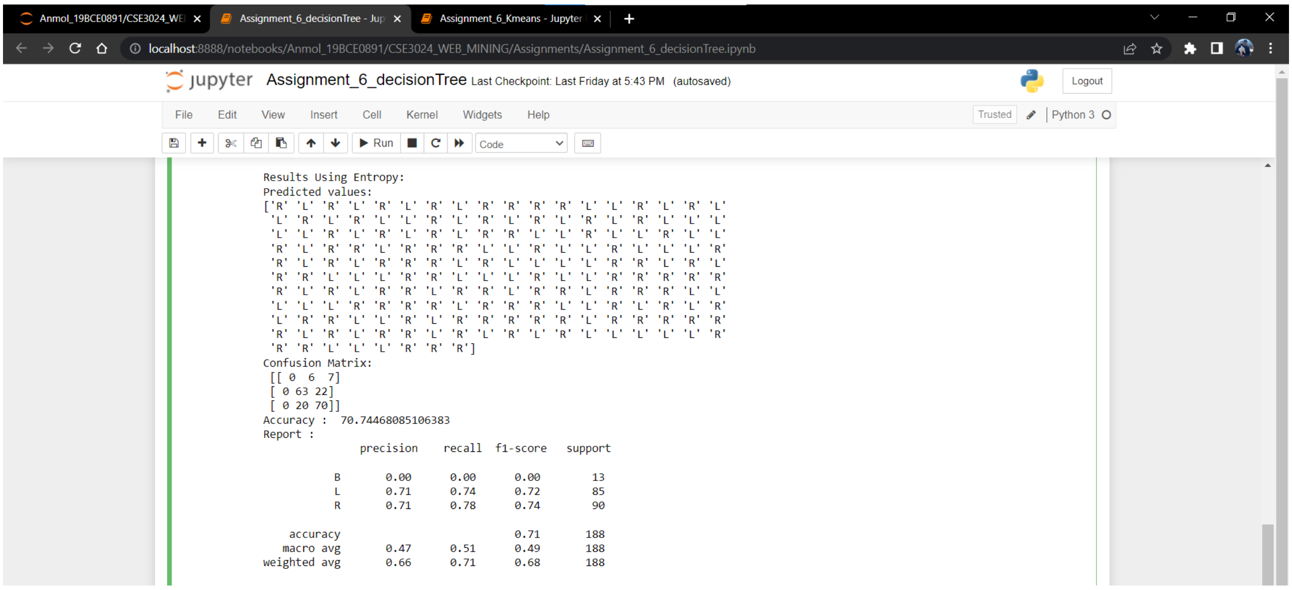

Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_decisionTree.ipynb
jupyter Assignment_6_decisionTree Last Checkpoint: Last Friday at 5:43 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

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

def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

    # Operational Phase
    print("Results Using Gini Index:")

    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)


# Calling main function
if __name__=="__main__":
    main()
```

```
Dataset Length:  625
Dataset Shape:  (625, 5)
Dataset:
    0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
```


Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_decisionTree.ipynb
jupyter Assignment_6_decisionTree Last Checkpoint: Last Friday at 5:43 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
Dataset Length:  625
Dataset Shape:  (625, 5)
Dataset:
    0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
Results Using Gini Index:
Predicted values:
['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R'
 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:
 [[ 0  6  7]
 [ 0 67 18]
 [ 0 19 71]]
Accuracy :  73.40425531914893
Report :
               precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.73      0.79      0.76        85
           R       0.74      0.79      0.76        90

    accuracy                           0.73       188
   macro avg       0.49      0.53      0.51       188
weighted avg       0.68      0.73      0.71       188
```

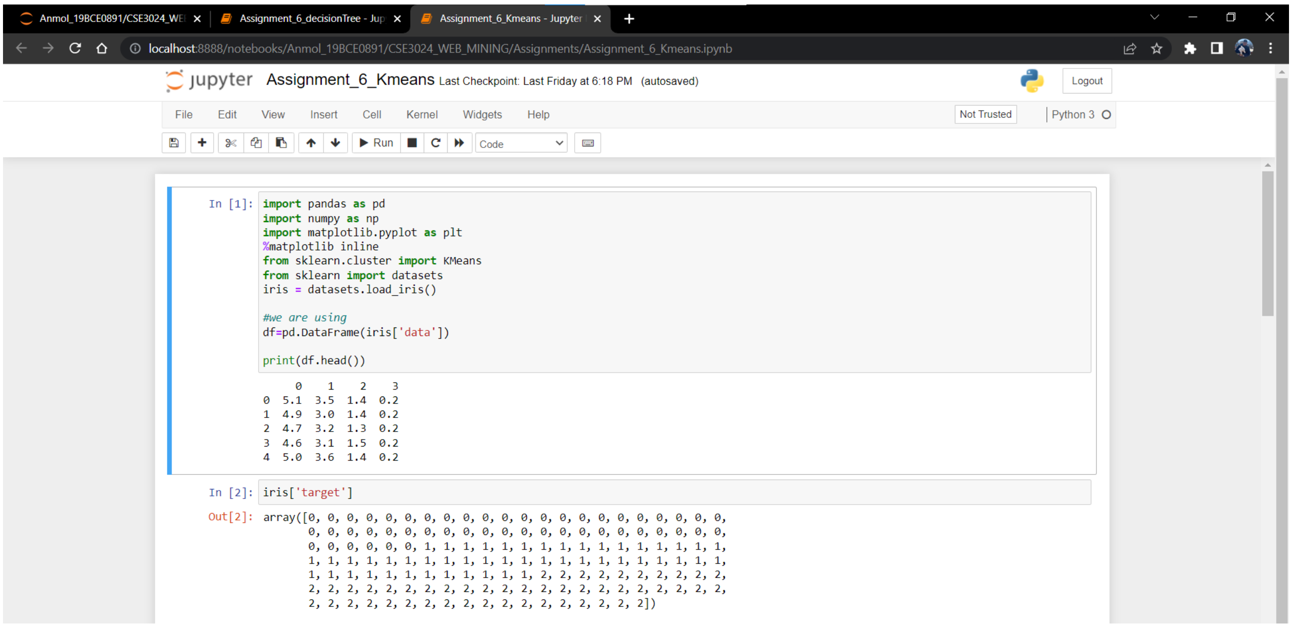

Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_decisionTree.ipynb
jupyter
Assignment_6_decisionTree Last Checkpoint: Last Friday at 5:43 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
Results Using Entropy:
Predicted values:
['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:
 [[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy :  70.74468085106383
Report :
              precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.71      0.74      0.72        85
           R       0.71      0.78      0.74        90

    accuracy                           0.71       188
   macro avg       0.47      0.51      0.49       188
weighted avg       0.66      0.71      0.68       188
```

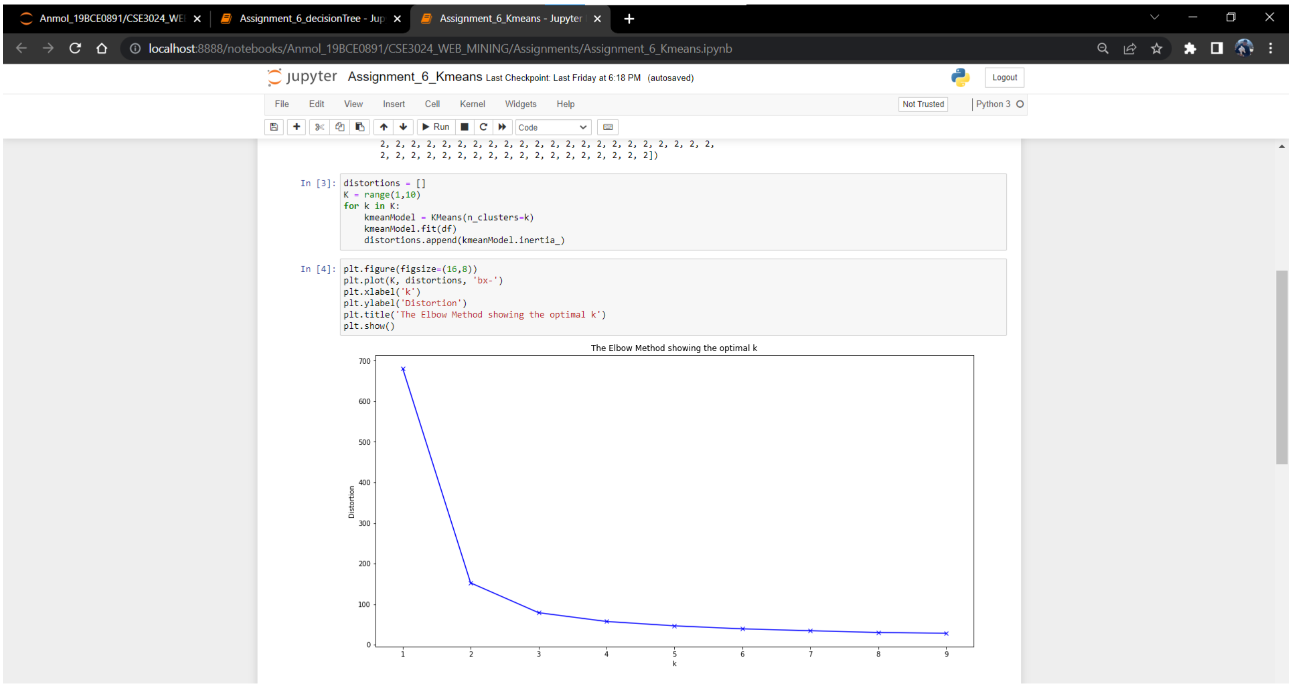
CODE

```
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

def importdata():
    balance_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-'+'databases/
balance-scale/balance-scale.data',
    sep= ',', header = None)

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

    print ("Dataset:\n",balance_data.head())
    return balance_data

def splitdataset(balance_data):

    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test


def train_using_gini(X_train, X_test, y_train):
    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    clf_gini.fit(X_train, y_train)
    return clf_gini


def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,max_depth = 3,
min_samples_leaf = 5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy


# Function to make predictions
def prediction(X_test, clf_object):

    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

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

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

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

def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

    # Operational Phase
    print("Results Using Gini Index:")
    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)


# Calling main function
if __name__=="__main__":
    main()
```

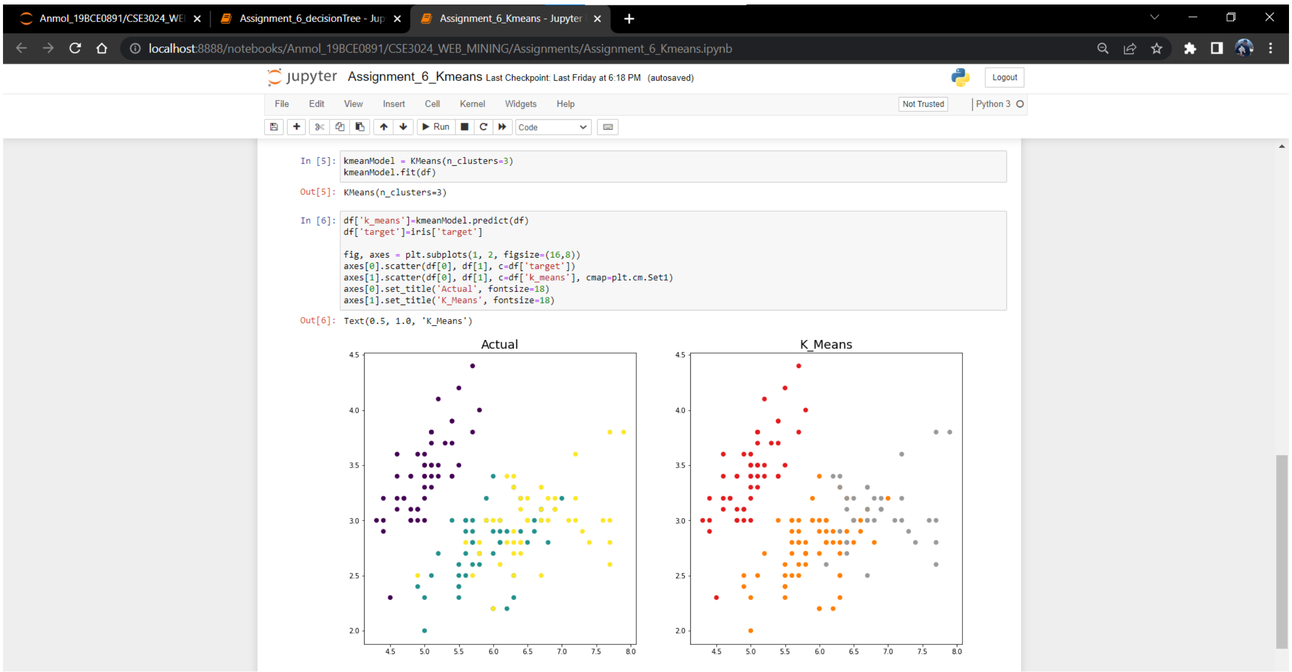
2. Create a Python programme that uses the K-means clustering algorithm and displays all clusters in different colours.


Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_Kmeans.ipynb
jupyter
Assignment_6_Kmeans Last Checkpoint: Last Friday at 6:18 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn.cluster import KMeans
        from sklearn import datasets
        iris = datasets.load_iris()

        #we are using
        df=pd.DataFrame(iris['data'])

        print(df.head())

             0    1    2    3
        0  5.1  3.5  1.4  0.2
        1  4.9  3.0  1.4  0.2
        2  4.7  3.2  1.3  0.2
        3  4.6  3.1  1.5  0.2
        4  5.0  3.6  1.4  0.2

In [2]: iris['target']

Out[2]: array([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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```


Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_Kmeans.ipynb
jupyter
Assignment_6_Kmeans Last Checkpoint: Last Friday at 6:18 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
The Elbow Method showing the optimal k
700
600
500
400
300
200
100
0
Distortion
1
2
3
4
5
6
7
8
9
k

```
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

In [3]: distortions = []
        K = range(1,10)
        for k in K:
            kmeanModel = KMeans(n_clusters=k)
            kmeanModel.fit(df)
            distortions.append(kmeanModel.inertia_)

In [4]: plt.figure(figsize=(16,8))
        plt.plot(K, distortions, 'bx-')
        plt.xlabel('k')
        plt.ylabel('Distortion')
        plt.title('The Elbow Method showing the optimal k')
        plt.show()
```


Anmol_19BCE0891/CSE3024_WE
Assignment_6_decisionTree - Jup
Assignment_6_Kmeans - Jupyter
localhost:8888/notebooks/Anmol_19BCE0891/CSE3024_WEB_MINING/Assignments/Assignment_6_Kmeans.ipynb
jupyter
Assignment_6_Kmeans Last Checkpoint: Last Friday at 6:18 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code
Actual
K_Means
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

```
In [5]: kmeanModel = KMeans(n_clusters=3)
        kmeanModel.fit(df)

Out[5]: KMeans(n_clusters=3)

In [6]: df['k_means']=kmeanModel.predict(df)
        df['target']=iris['target']

        fig, axes = plt.subplots(1, 2, figsize=(16,8))
        axes[0].scatter(df[0], df[1], c=df['target'])
        axes[1].scatter(df[0], df[1], c=df['k_means'], cmap=plt.cm.Set1)
        axes[0].set_title('Actual', fontsize=18)
        axes[1].set_title('K_Means', fontsize=18)

Out[6]: Text(0.5, 1.0, 'K_Means')
```

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn import datasets
iris = datasets.load_iris()

#we are using
```

```
df=pd.DataFrame(iris['data'])

print(df.head())


iris['target']


distortions = []
K = range(1,10)
for k in K:
   kmeanModel = KMeans(n_clusters=k)
   kmeanModel.fit(df)
   distortions.append(kmeanModel.inertia_)


plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()


kmeanModel = KMeans(n_clusters=3)
kmeanModel.fit(df)


df['k_means']=kmeanModel.predict(df)
df['target']=iris['target']

fig, axes = plt.subplots(1, 2, figsize=(16,8))
axes[0].scatter(df[0], df[1], c=df['target'])
axes[1].scatter(df[0], df[1], c=df['k_means'], cmap=plt.cm.Set1)
axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('K_Means', fontsize=18)
```