

**naïve bayes and decision tree analysis (ASSESSMENT - 2)**

**CSE4020(MACHINE LEARNING)LAB:L49-L50**



**FEBURARY 11, 2022**

**ANIRUDH VADERA**

**20BCE2940**

**QUESTION:**

1. **Naïve Bayes Classification:**

Classify the mail as spam or not using Naïve Bayes classifier. Hard code it to learn the model.

**Expected Output**

**-----------------------**

1. **Likelihood probabilities**
2. **confusion matrix**
3. **accuracy**
4. **Precision, Recall**

**DATASET:** <https://www.kaggle.com/balaka18/email-spam-classification-dataset-csv>

1. **Decision Tree**

Classify the fruit by its type based on the fruit\_name, fruit\_subtype, mass, width, height, and color\_score. Construct CART tree. If possible prune it.

**Expected Output**

**-----------------------**

1. **Decision tree without pruning**
2. **Decision tree after pruning**
3. **Confusion matrix**
4. **Accuracy, precision, recall**

**DATASET:** <https://www.kaggle.com/mjamilmoughal/fruits-with-colors-dataset>

* **Naïve Bayes**

**Description:**

**Formula Used:**

**Code:**

**Code Snippets:**

**Output and Results:**

**Dataset Details:**

**Dataset:**

**As the missing values is none we can proceed further:**

**Dataset Details:**

**Regression Results:**

**Accuracy Analysis(Errors):**

**Inference:**

* **Decision Tree**

**Description:**

* Decision Tree is a **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where**internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
* In a Decision tree, there are two nodes, which are the **Decision Node** and**Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.
* ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
* In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
* A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
* Below diagram explains the general structure of a decision tree:



**Formula Used:**

**Information Gain:**

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
* It calculates how much information a feature provides us about a class.
* According to the value of information gain, we split the node and build the decision tree.
* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)

**Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

**Where,**

* **S= Total number of samples**
* **P(yes)= probability of yes**
* **P(no)= probability of no**

### Gini Index:

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
* An attribute with the low Gini index should be preferred as compared to the high Gini index.
* It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
* Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

## **Pruning:**

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning**technology used:

* **Cost Complexity Pruning**
* **Reduced Error Pruning.**

**Code:**

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.metrics import multilabel\_confusion\_matrix, accuracy\_score, precision\_score, recall\_score,classification\_report

import seaborn as sns

df = pd.read\_table("C:/Users/Anirudh/OneDrive/Desktop/fruit\_data\_with\_colors.txt")

df = pd.DataFrame(df)

print("The dataset is as following :")

print(df)

print("\n")

**# Check for missing values**

print("Checking for missing values :")

print(df.isnull().sum())

print("\n")

**# Printing the header of the dataset**

print("Dataset Header : ")

print(df.head())

print("\n")

**# Information regarding the columns**

print("Information regarding the columns : ")

print(df.info())

print("\n")

**# Information related to the dataset**

print("Dataset Details : ")

print(df.describe())

print("\n")

**# correlation matrix**

sns.heatmap(df.corr())

**# Dummy Variables**

**# The variable fruit\_subtype has many levels. We need to convert these levels into integer as well in order to predict**

**# For this, we will use something called dummy variables.**

**# Get the dummy variables for the feature 'fruit\_subtype' and store it in a new variable - 'status'**

status = pd.get\_dummies(df['fruit\_subtype'], drop\_first = True)

**# Now, you don't need all the columns.**

**# You can drop the fruit\_subtype column, as the fruit\_subtype can be identified with just the last 8 columns where encoding has already been done**

**# Add the results to the original dataframe**

df = pd.concat([df, status], axis = 1)

**# Drop 'fruit\_subtype' as we have created the dummies for it**

df.drop(['fruit\_subtype'], axis = 1, inplace = True)

**# Now let's see the head of our dataframe.**

print("After Trimming and correcting the dataset looks like follows : ")

print(df.head())

**# Extracting Independent and dependent Variable**

X = df.iloc[:, 2:14].values

Y = df.iloc[:, 0].values

**# Splitting the dataset into training and testing set**

X\_train, X\_test, Y\_train, Y\_test= train\_test\_split(X, Y, test\_size= 0.3, random\_state=0)

**#Fitting Decision Tree classifier to the training set**

classifier= DecisionTreeClassifier(criterion='gini', random\_state=0)

classifier.fit(X\_train, Y\_train)

**#Predicting the test set result**

Y\_pred = classifier.predict(X\_test)

**#Creating the Confusion matrix**

c = multilabel\_confusion\_matrix(Y\_test,Y\_pred)

print(c)

**# Printing the Confusion Matrix**

i=0

for x in c:

plt.figure(figsize=(5,5))

sns.heatmap(data=x,linewidths=.5, annot=True,square = True, cmap = 'Blues')

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

all\_sample\_title = f'For Fruit\_Label {i+1}'

plt.title(all\_sample\_title, size = 15)

i = i + 1

**# Checking the accuracy of our model**

print('Accuracy: ',accuracy\_score(Y\_test,Y\_pred))

print('Precision: %.3f' % precision\_score(Y\_test, Y\_pred,average='micro'))

print('Recall: %.3f' % recall\_score(Y\_test, Y\_pred,average='micro'))

**# The decision tree**

print(tree.plot\_tree(classifier,filled=True,precision = 4))

plt.savefig('destination\_path.eps')

**# Our Model Report**

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Evaluation on Our Model \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

score\_te = classifier.score(X\_test, Y\_test)

print('Accuracy Score: ', score\_te)

# Look at classification report to evaluate the model

print(classification\_report(Y\_test, Y\_pred))

print('--------------------------------------------------------')

print("")

**# # Pre pruning**

max\_depth = []

acc\_gini = []

acc\_entropy = []

for i in range(1,6):

dtree = DecisionTreeClassifier(criterion='gini', max\_depth=i)

dtree.fit(X\_train, Y\_train)

pred = dtree.predict(X\_test)

acc\_gini.append(accuracy\_score(Y\_test, pred))

dtree = DecisionTreeClassifier(criterion='entropy', max\_depth=i)

dtree.fit(X\_train, Y\_train)

pred = dtree.predict(X\_test)

acc\_entropy.append(accuracy\_score(Y\_test, pred))

max\_depth.append(i)

d = pd.DataFrame({'acc\_gini':pd.Series(acc\_gini),

'acc\_entropy':pd.Series(acc\_entropy),

'max\_depth':pd.Series(max\_depth)})

**# visualizing changes in parameters**

plt.plot('max\_depth','acc\_gini', data=d, label='gini')

plt.plot('max\_depth','acc\_entropy', data=d, label='entropy')

plt.xlabel('max\_depth')

plt.ylabel('accuracy')

plt.legend()

dtree = DecisionTreeClassifier(criterion='gini', max\_depth=4)

dtree.fit(X\_train, Y\_train)

pred = dtree.predict(X\_test)

**#Creating the Confusion matrix**

c = multilabel\_confusion\_matrix(Y\_test,pred)

print(c)

**# Printing the Confusion Matrix**

i=0

for x in c:

plt.figure(figsize=(5,5))

sns.heatmap(data=x,linewidths=.5, annot=True,square = True, cmap = 'Blues')

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

all\_sample\_title = f'For Fruit\_Label {i+1}'

plt.title(all\_sample\_title, size = 15)

i = i + 1

**# Checking the accuracy of our model**

print('Accuracy: ',accuracy\_score(Y\_test,pred))

print('Precision: %.3f' % precision\_score(Y\_test, pred,average='micro'))

print('Recall: %.3f' % recall\_score(Y\_test, pred,average='micro'))

**# Our Model Report**

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Evaluation on Our Model \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

score\_te = dtree.score(X\_test, Y\_test)

print('Accuracy Score: ', score\_te)

# Look at classification report to evaluate the model

print(classification\_report(Y\_test, pred))

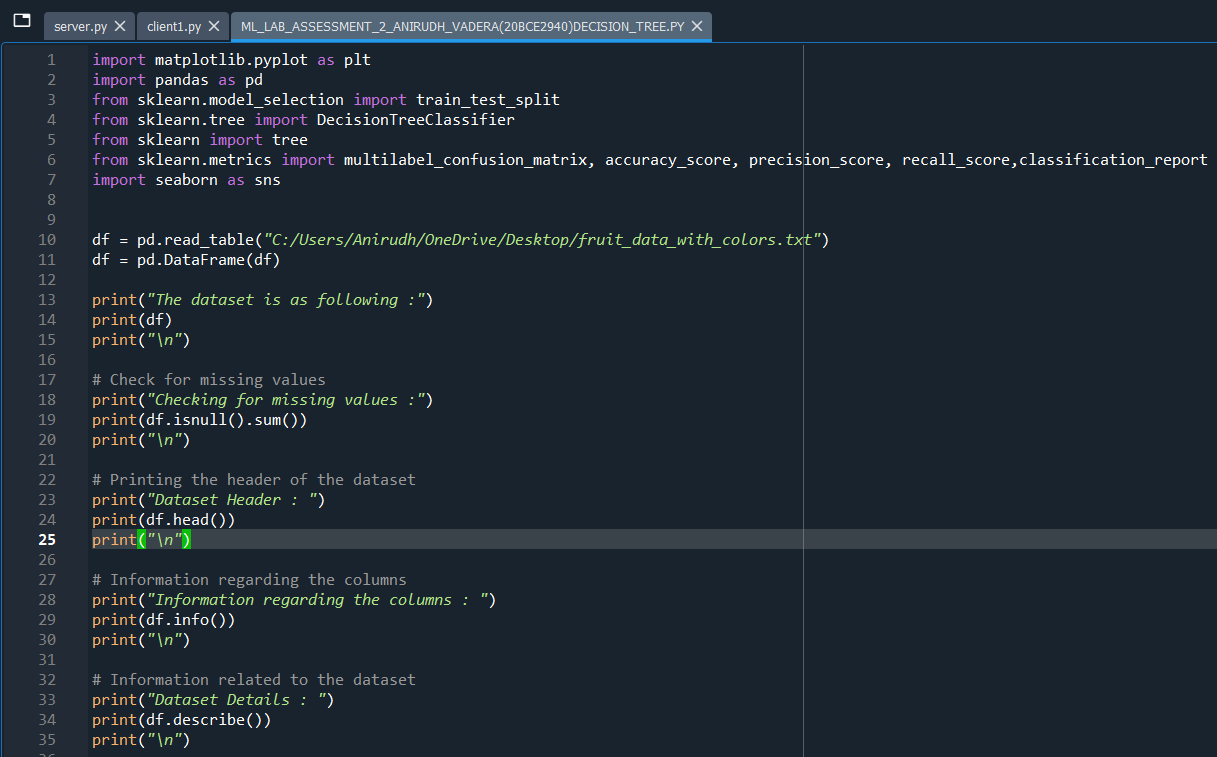
print('--------------------------------------------------------')

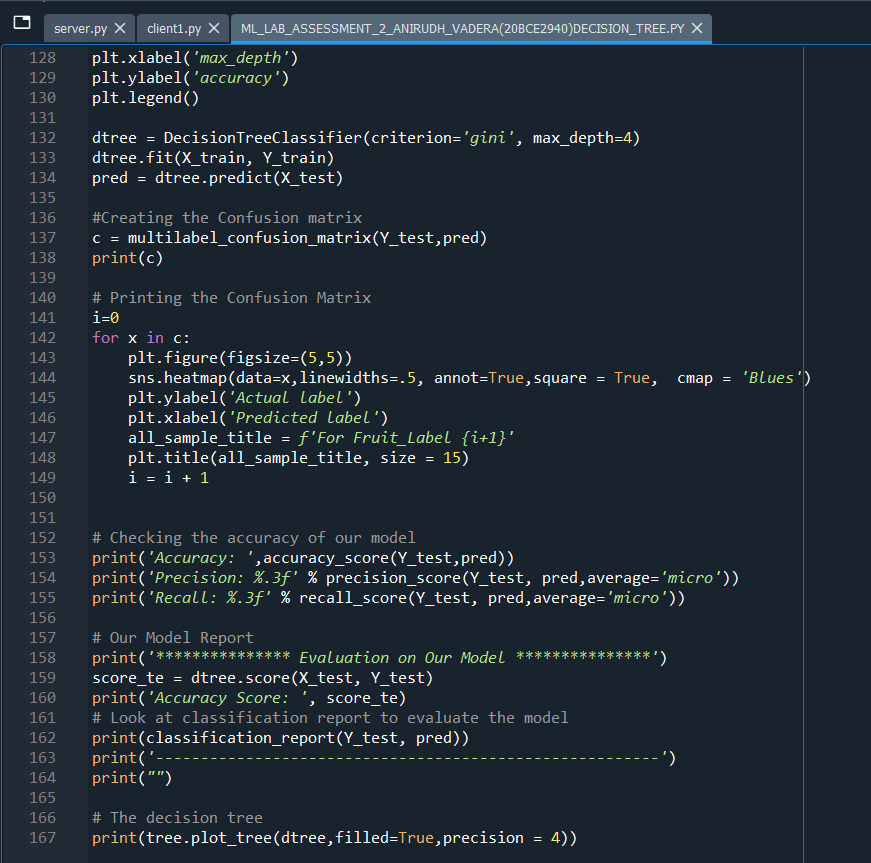
print("")

**# The decision tree**

print(tree.plot\_tree(dtree,filled=True,precision = 4))

**Code Snippets:**

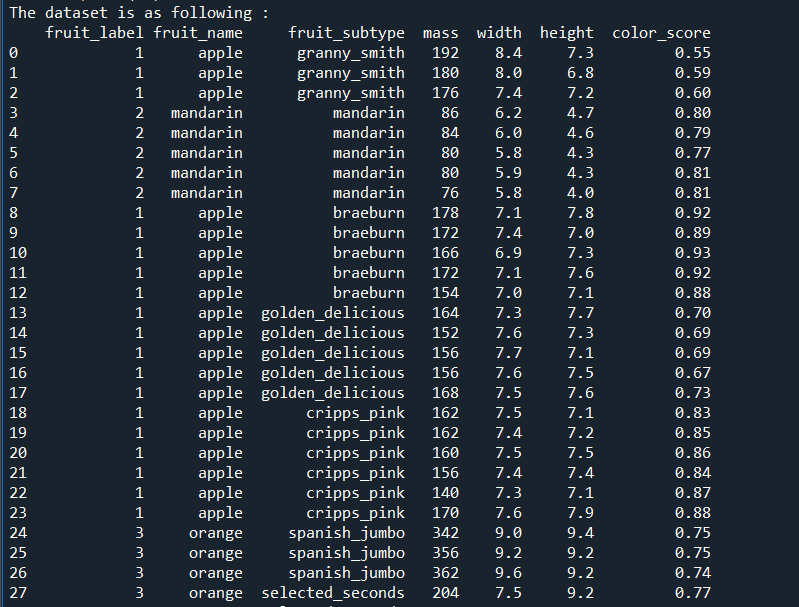
****

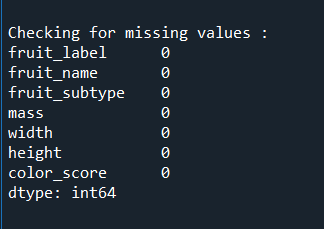
****

**Output and Results:**

**Dataset Details:**

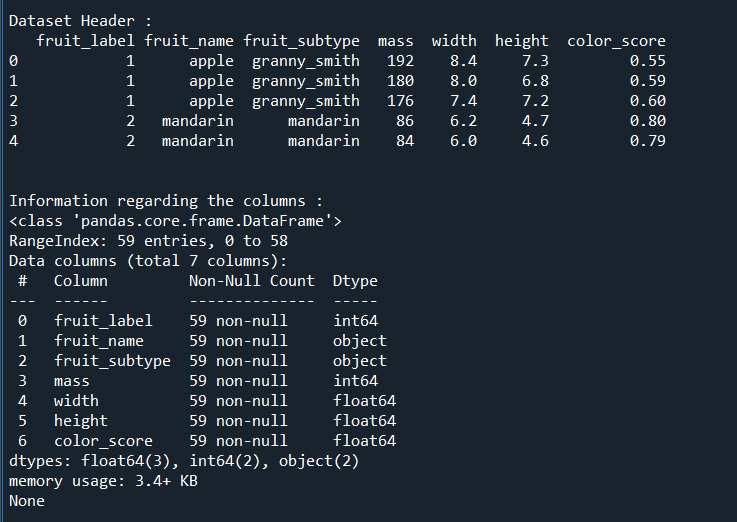
**Dataset:**

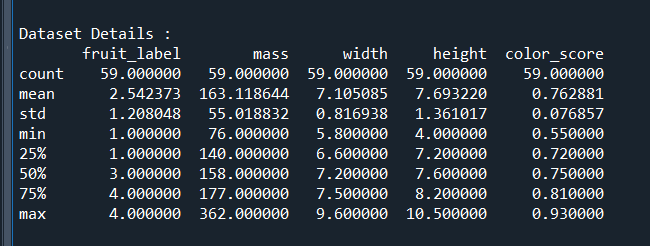
****

****

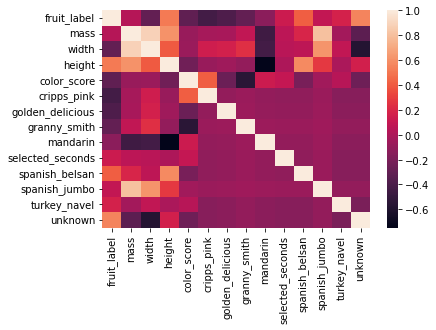
**As the missing values is none we can proceed further:**

**Dataset Details:**





**Correlation Matrix:**



**We Infer Height has a great impact on predicting fruit\_labels**

**Data Preparation**

**# Dummy Variables**

**# The variable fruit\_subtype has many levels. We need to convert these levels into integer as well in order to predict**

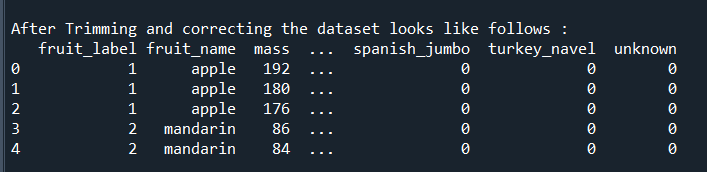
**# For this, we will use something called dummy variables.**

**# Get the dummy variables for the feature 'fruit\_subtype' and store it in a new variable - 'status'**

**# Now, you don't need all the columns.**

**# You can drop the fruit\_subtype column, as the fruit\_subtype can be identified with just the last 8 columns where encoding has already been done**

**# Drop 'fruit\_subtype' as we have created the dummies for it**

****

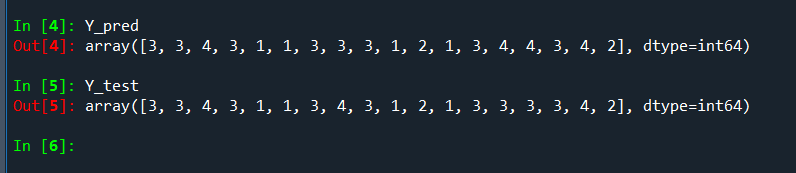
**Fitting the Decision Tree Model:**

**70% data for training and 30% for testing:**

**We use the gini criterion to train our model**

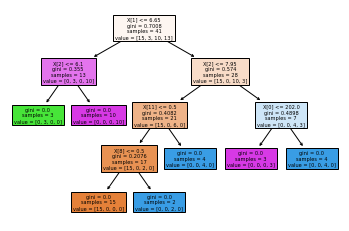
**Prediction Results:**

**The Predicted Y\_Values And The Actual Y\_Values are:**

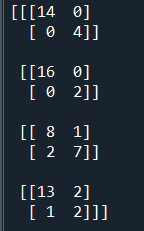
****

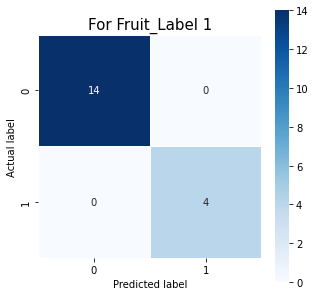
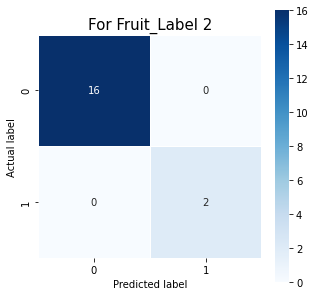
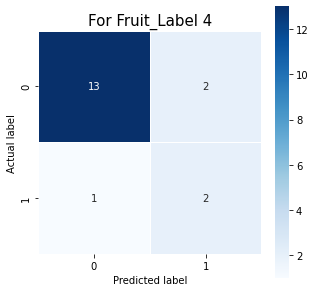
**Before Pruning:**

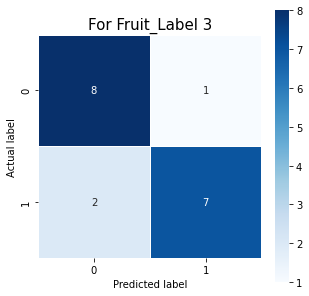
**Decision Tree Before Pruning:**



**Confusion Matrix:**

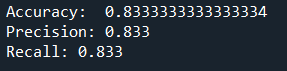
****

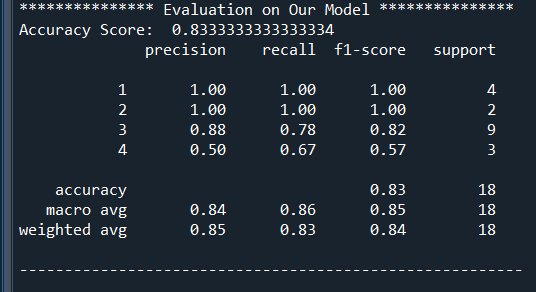
**For Individual Labels:**



**Accuracy Analysis(Errors):**

**Checking the Accuracy of the model:**

****

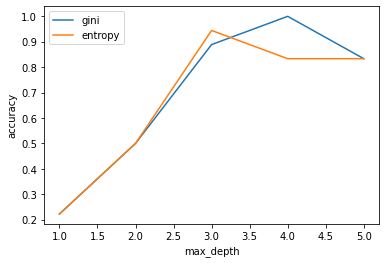
****

**Inference:**

1. **From the correlation matrix we get to know that height has a great impact on predicting the fruit labels**
2. **The decision Tree has currently 13 nodes and a depth of 5**
3. **The confusion matrix tells us that there is no error in predicting the label 1 and label 2 while there is some error in predicting label 3 and 4**
4. **The accuracy of model without pruning is 83.34 percent which is quite good**
5. **The macro average recall is 86 percent and macro average precision is 84 percent which can be seen from evaluation**
6. **Individual scores are also given in evaluation**

**After Pruning:**

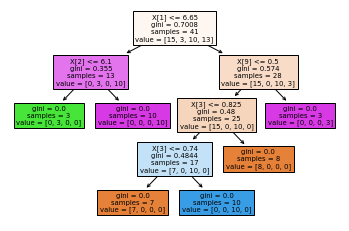
**First we check which type is better gini or entropy from below graph we get to know gini is better**



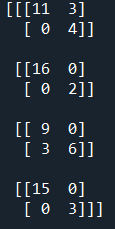
**This graph tells us that at depth = 4 the gini algo works best for our case that means reducing the depth to 4 will give us same accuracy and also it will reduce the number of nodes in our decision tree and the max\_depth will be equal to 4 instead of 5**

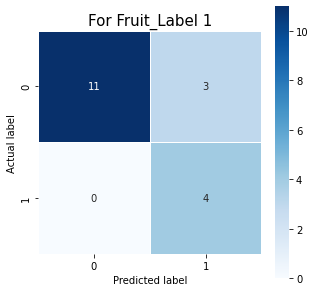
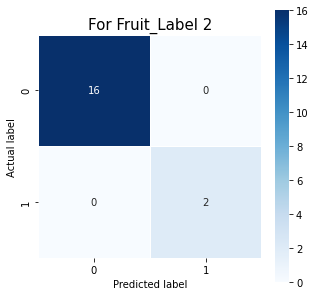
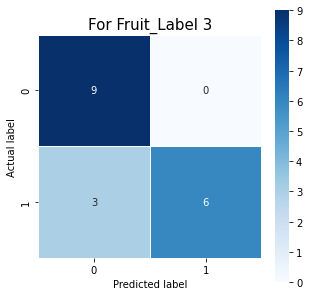
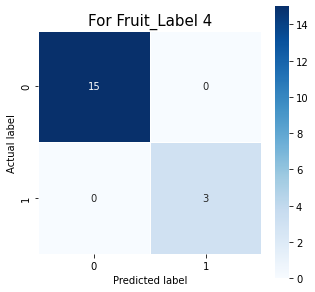
**Decision Tree After Pruning:**

**Reduced Nodes:**



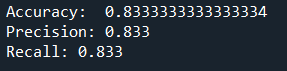
**Confusion Matrix:**

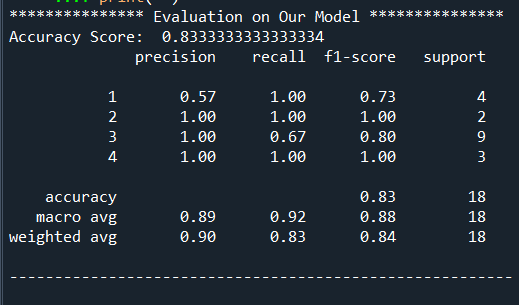
****

**For Individual Labels:**

**Accuracy Analysis(Errors):**

**Checking the Accuracy of the model:**

****

****

**Inference:**

1. **The decision Tree after pruning has 11 nodes and a depth of 4**
2. **The confusion matrix tells us that there is no error in predicting the label 4 and label 2 and label1 while there is some error in predicting label 3**
3. **The accuracy of model without pruning is 83.34 percent which is quite good**
4. **There are some changes in recall and precision scores**
5. **The macro average recall is 92 percent and macro average precision is 89 percent which can be seen from evaluation**
6. **Individual scores are also given in evaluation**