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

2. 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:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

- **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.
- **P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.
- **P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.
- **P(B) is Marginal Probability**: Probability of Evidence.

Multinomial: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.

The classifier uses the frequency of words for the predictors.

Formula Used:

Naive Bayes algorithm will make the classification based on the results it gets to these two equations below, where "w_1" is the first word, and $w_1, w_2, \ldots, w_n$ is the entire message:

$$P(\mathrm{Spam}|w_1, w_2, \ldots, w_n) \propto P(\mathrm{Spam}) \cdot \prod_{i=1}^{n} P(w_i|\mathrm{Spam})$$

$$P(\mathrm{Ham}|w_1, w_2, \ldots, w_n) \propto P(\mathrm{Ham}) \cdot \prod_{i=1}^{n} P(w_i|\mathrm{Ham})$$

If $P(\mathrm{Spam} \mid w_1, w_2, \ldots, w_n)$ is greater than $P(\mathrm{Ham} \mid w_1, w_2, \ldots, w_n)$, then the message is spam.

To calculate $P(w_i|\mathrm{Spam})$ and $P(w_i|\mathrm{Ham})$, we need to use separate equations:

$$P(w_i|\mathrm{Spam}) = \frac{N_{w_i|\mathrm{Spam}} + \alpha}{N_{\mathrm{Spam}} + \alpha \cdot N_{\mathrm{Vocabulary}}}$$

$$P(w_i|\mathrm{Ham}) = \frac{N_{w_i|\mathrm{Ham}} + \alpha}{N_{\mathrm{Ham}} + \alpha \cdot N_{\mathrm{Vocabulary}}}$$

$N_{w_i|\mathrm{Spam}}$ = the number of times the word w_i occurs in spam messages

$N_{w_i|\mathrm{Ham}}$ = the number of times the word w_i occurs in ham messages

N_{Spam} = total number of words in spam messages

N_{Ham} = total number of words in ham messages

$N_{\mathrm{Vocabulary}}$ = total number of words in the vocabulary

$\alpha = 1$ (α is a smoothing parameter)

Code:

```
import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score,classification_report

import seaborn as sns


# Importing the dataset

df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/emails.csv")

print("The dataset is as following : [5172 rows x 3002 columns]")

print(df)

print("\n")


# 0 - Non Spam

# 1 - Spam

df['Prediction'].value_counts(normalize=True)


# 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")
```

```
df_train, df_test =
train_test_split(df,test_size=0.009,train_size=0.991,random_state=0)

# Isolating spam and ham messages first
spam_messages = df_train[df_train['Prediction'] == 1]
ham_messages = df_train[df_train['Prediction'] == 0]

# P(Spam) and P(Ham)
p_spam = len(spam_messages) / len(df_train)
p_ham = len(ham_messages) / len(df_train)

df_train_n_of_count_spam = spam_messages.iloc[:,1:-1]
df_train_n_of_count_ham = ham_messages.iloc[:,1:-1]
spam_messages['No_of_words']= df_train_n_of_count_spam.sum(axis=1)
ham_messages['No_of_words']= df_train_n_of_count_ham.sum(axis=1)

# N_Spam
n_spam = spam_messages['No_of_words'].sum()

# N_Ham
n_ham = ham_messages['No_of_words'].sum()

# N_Vocabulary
n_vocabulary = len(df_train.columns) - 2

# Laplace smoothing
alpha = 1
```

```
# Initiate parameters

parameters_spam = {unique_word:0 for unique_word in df_train.columns[1:-1]}

parameters_ham = {unique_word:0 for unique_word in df_train.columns[1:-1]}


# Calculate parameters

for word in df_train.columns[1:-1]:

  n_word_given_spam = spam_messages[word].sum() # spam_messages already defined

  p_word_given_spam = (n_word_given_spam + alpha) / (n_spam + alpha*n_vocabulary)

  parameters_spam[word] = p_word_given_spam


  n_word_given_ham = ham_messages[word].sum() # ham_messages already defined

  p_word_given_ham = (n_word_given_ham + alpha) / (n_ham + alpha*n_vocabulary)

  parameters_ham[word] = p_word_given_ham


def classify(message):


  p_spam_given_message = p_spam

  p_ham_given_message = p_ham


  for word in message:

    if word in parameters_spam:

      p_spam_given_message *= parameters_spam[word]


    if word in parameters_ham:

      p_ham_given_message *= parameters_ham[word]
```

```
  if p_ham_given_message > p_spam_given_message:
    return 0
  elif p_ham_given_message < p_spam_given_message:
    return 1
  else:
    return 0

message_list_to_predict = []
Y_pred = []
itr = 0
while(itr<len(df_test)):
    message_list_to_predict = []
    columns = df_test.columns[1:-1]
    for column in columns:
        temp = df_test.iloc[itr,:][column]
        for i in range(temp):
            message_list_to_predict.append(column)
    Y_pred.append(classify(message_list_to_predict))
    itr = itr + 1

Y_test = df_test.iloc[:,-1]
```

 # Checking the accuracy of our model

```
print('Accuracy: ',accuracy_score(Y_test,Y_pred))
print('Precision: %.3f' % precision_score(Y_test, Y_pred))
print('Recall: %.3f' % recall_score(Y_test, Y_pred))
```

```
# Our Model Report
print('*************** Evaluation on Our Model ***************')
print('Accuracy Score: ', accuracy_score(Y_test,Y_pred))
# Look at classification report to evaluate the model
print(classification_report(Y_test, Y_pred))
print('-----------------------------------------------------')
print("")

# Confusion Matrix
cm = confusion_matrix(Y_test, Y_pred)
print(cm)

plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = "(Predicted and Actual Y_values)"
plt.title(all_sample_title, size = 15)

# likelihood Probabilities
print(parameters_spam)
print(parameters_ham)
```

Code Snippets:

ver.py × client1.py × ML_LAB_ASSESSMENT_2_ANIRUDH_VADERA(20BCE2940).PY ×

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,cla

# Importing the dataset
df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/emails.csv")
print("The dataset is as following : [5172 rows x 3002 columns]")
print(df)
print("\n")

# 0 - Non Spam
# 1 - Spam
df['Prediction'].value_counts(normalize=True)

# 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")

# 0 - 85

df_train, df_test = train_test_split(df,test_size=0.009,train_size=0.991,random_state=0)

# Isolating spam and ham messages first
spam_messages = df_train[df_train['Prediction'] == 1]
ham_messages = df_train[df_train['Prediction'] == 0]

# P(Spam) and P(Ham)
p_spam = len(spam_messages) / len(df_train)
p_ham = len(ham_messages) / len(df_train)

df_train_n_of_count_spam = spam_messages.iloc[:,1:-1]
df_train_n_of_count_ham = ham_messages.iloc[:,1:-1]
spam_messages['No_of_words']= df_train_n_of_count_spam.sum(axis=1)
ham_messages['No_of_words']= df_train_n_of_count_ham.sum(axis=1)
```

```
def classify(message):

    p_spam_given_message = p_spam
    p_ham_given_message = p_ham

    for word in message:
        if word in parameters_spam:
            p_spam_given_message *= parameters_spam[word]

        if word in parameters_ham:
            p_ham_given_message *= parameters_ham[word]

    if p_ham_given_message > p_spam_given_message:
        return 0
    elif p_ham_given_message < p_spam_given_message:
        return 1
    else:
        return 0
```

Output and Results:

Dataset Details:

Dataset:

```
In [36]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/ML_LAB_ASSESSMENT_2_ANIRUD
OneDrive/Desktop/python')
The dataset is as following : [5172 rows x 3002 columns]
       Email No.  the  to  ect  and  ...  military  allowing  ff  dry  Prediction
0        Email 1    0   0    1    0  ...         0         0   0    0           0
1        Email 2    8  13   24    6  ...         0         0   1    0           0
2        Email 3    0   0    1    0  ...         0         0   0    0           0
3        Email 4    0   5   22    0  ...         0         0   0    0           0
4        Email 5    7   6   17    1  ...         0         0   1    0           0
...          ...  ...  ..  ...  ...  ...       ...       ...  ..  ...         ...
5167  Email 5168    2   2    2    3  ...         0         0   0    0           0
5168  Email 5169   35  27   11    2  ...         0         0   1    0           0
5169  Email 5170    0   0    1    1  ...         0         0   0    0           1
5170  Email 5171    2   7    1    0  ...         0         0   1    0           1
5171  Email 5172   22  24    5    1  ...         0         0   0    0           0

[5172 rows x 3002 columns]
```

```
Checking for missing values :
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

As the missing values is none we can proceed further:

Dataset Details:

```
Dataset Header :
  Email No.  the  to  ect  and  ...  military  allowing  ff  dry  Prediction
0   Email 1    0   0    1    0  ...         0         0   0    0           0
1   Email 2    8  13   24    6  ...         0         0   1    0           0
2   Email 3    0   0    1    0  ...         0         0   0    0           0
3   Email 4    0   5   22    0  ...         0         0   0    0           0
4   Email 5    7   6   17    1  ...         0         0   1    0           0

[5 rows x 3002 columns]
```

Probabilities and calculating constants:

Calculating P(Spam) and P(Ham) i.e probability of spam and non-spam messages in dataset

```
In [41]: p_spam
Out[41]: 0.29092682926829266

In [42]: p_ham
Out[42]: 0.7090731707317073
```

Likelihood Probability:

The whole Formula:

$$P(\mathrm{Spam}|w_1, w_2, \ldots, w_n) \propto P(\mathrm{Spam}) \cdot \prod_{i=1}^{n} P(w_i|\mathrm{Spam})$$

$$P(\mathrm{Ham}|w_1, w_2, \ldots, w_n) \propto P(\mathrm{Ham}) \cdot \prod_{i=1}^{n} P(w_i|\mathrm{Ham})$$

The required Likelihood Probability:

$$P(w_i|\mathrm{Spam}) = \frac{N_{w_i|\mathrm{Spam}} + \alpha}{N_{\mathrm{Spam}} + \alpha \cdot N_{\mathrm{Vocabulary}}}$$

$$P(w_i|\mathrm{Ham}) = \frac{N_{w_i|\mathrm{Ham}} + \alpha}{N_{\mathrm{Ham}} + \alpha \cdot N_{\mathrm{Vocabulary}}}$$

These are calculated for every word that exists in our dataset:

The 2 of such word probabilities are:

For spam:

```
In [45]: parameters_spam["the"]
Out[45]: 0.004465945946446625

In [46]: parameters_spam["took"]
Out[46]: 1.3893848220823267e-05
```

For ham:

```
In [47]: parameters_ham["the"]
Out[47]: 0.006363027373405528

In [48]: parameters_ham["took"]
Out[48]: 1.4991632302075605e-05
```

Showing all such likelihood probabilities for spam as well as non-spam:

For spam:

```
'reflect': 4.1681544662469795e-06,
'assets': 1.2041335124713498e-05,
'lamadrid': 4.631282740274422e-07,
'general': 1.8988259235125132e-05,
'bridge': 1.620948959096048e-05,
'ability': 5.742790597940283e-05,
'oct': 6.946924110411634e-05,
'play': 7.085862592619865e-05,
'enrononline': 4.631282740274422e-07,
'compliance': 3.658713364816793e-05,
'spam': 4.2607801210524684e-05,
'availability': 2.6398311619564205e-05,
'king': 0.00031770599598282534,
'understanding': 6.946924110411633e-06,
'chance': 1.5283233042905593e-05,
'quick': 5.279662323912841e-05,
'effort': 2.732456816761909e-05,
'points': 2.315641370137211e-06,
'reliantenergy': 4.631282740274422e-07,
'fixed': 1.6672617864987918e-05,
'short': 3.28821074559484e-05,
'hill': 1.9914515783180014e-05,
'cheryl': 1.3893848220823265e-06,
'aepin': 4.631282740274422e-07,
'key': 4.816534049885399e-05,
'understand': 3.0103337811783744e-05,
'valign': 3.705026192219538e-05,
'capacity': 6.020667562356749e-06,
'game': 2.8713952989701416e-05,
'took': 1.3893848220823267e-05,
...}

In [50]:
```

For ham:

```
Console 12/A

'oasis': 2.7353153673962506e-05,
'reflect': 4.7078985650377775e-05,
'assets': 1.7621743232264307e-05,
'lamadrid': 2.2618953999622842e-05,
'general': 2.340798727867945e-05,
'bridge': 3.5769508650566360e-05,
'ability': 3.235036444132104e-05,
'oct': 0.00011651391420735953,
'play': 4.2607797069056984e-05,
'enrononline': 2.6827131487924765e-05,
'compliance': 2.893122023207573e-06,
'spam': 5.260221860377405e-07,
'availability': 8.153343883584978e-06,
'king': 0.00020094047506641688,
'understanding': 2.077787634849075e-05,
'chance': 1.8936798697358657e-05,
'quick': 1.9462820883396398e-05,
'effort': 2.20929318135851e-05,
'points': 2.4197020557736065e-05,
'reliantenergy': 2.104088744150962e-05,
'fixed': 1.2624532464905773e-05,
'short': 3.050928679018895e-05,
'hill': 7.101299511509497e-05,
'cheryl': 3.261337553433991e-05,
'aepin': 2.5775087115849287e-05,
'key': 2.4723042743773805e-05,
'understand': 4.076671941792489e-05,
'valign': 2.6301109301887027e-07,
'capacity': 2.051486525547188e-05,
'game': 2.20929318135851e-05,
'took': 1.4991632302075605e-05,
...}

In [51]:
```

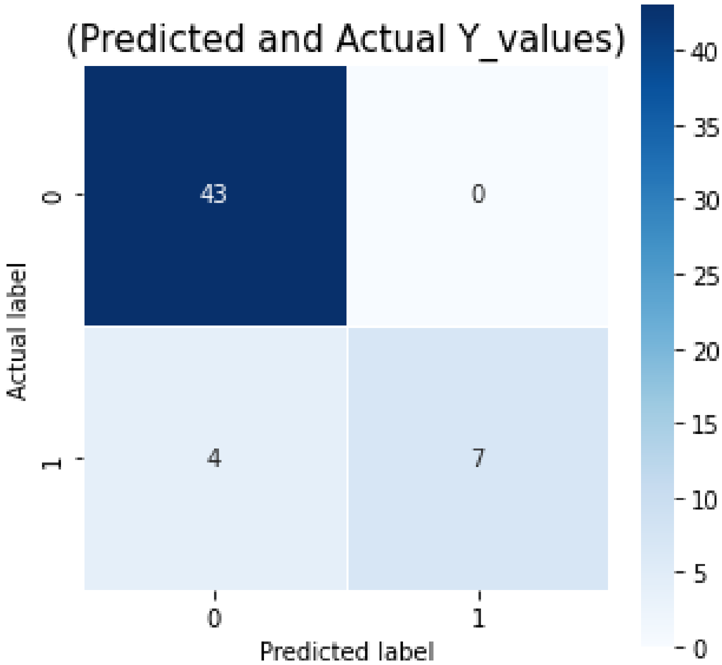
As there are around 3000 probabilities they can't be set in single screen.

Confusion Matrix:

Here 0 = Non-Spam

And 1 = Spam

```
[[43  0]
 [ 4  7]]
```



We get to know out 43 predictions were correct for non-spam and 7 predictions were correct for spam

Accuracy Analysis(Errors):

```
Accuracy:  0.9259
Precision: 1.000
Recall: 0.222
```

```
*************** Evaluation on Our Model ***************
Accuracy Score:  0.9259
              precision    recall  f1-score   support

           0       0.84      1.00      0.92        38
           1       1.00      0.22      0.36         9

    accuracy                           0.85        47
   macro avg       0.92      0.61      0.64        47
weighted avg       0.87      0.85      0.81        47


--------------------------------------------------------
```

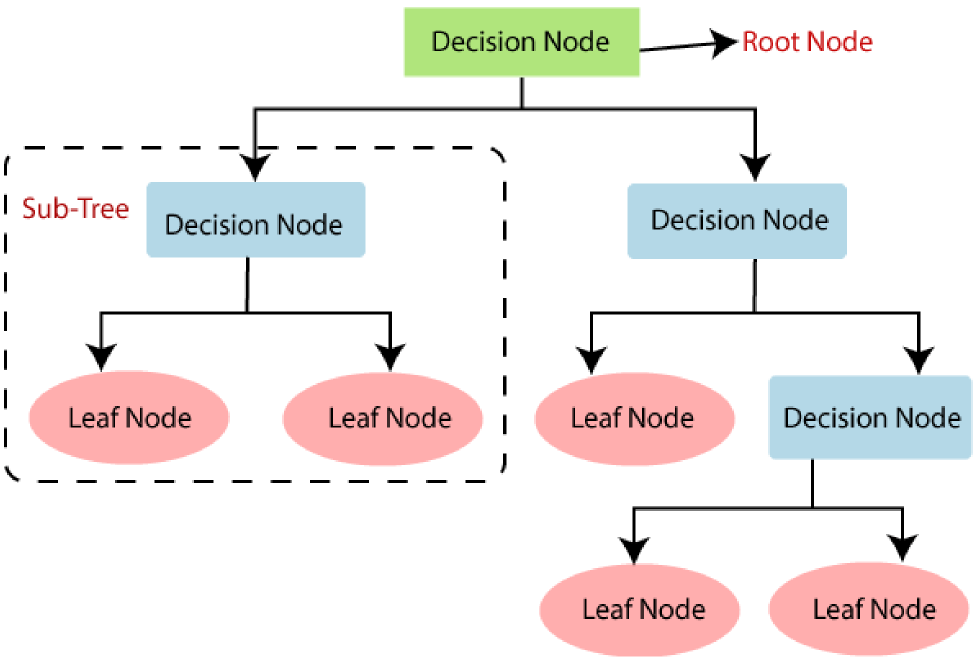
Inference:

- **From confusion matrix we get to know our model predicted 43 correct non spam emails and 7 correct spam emails whereas it predicted 4 spam emails as non-spam**
- **The accuracy of our model is 92.25 percent whereas precision is 100 percent and recall is 22.2 percent.**
- **The dataset was very large so we took the test dataset as only a small fraction of the dataset due to a huge training set our accuracy was above par.**

➔ 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= $1- \sum_j P_j^2$

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 plot_confusion_matrix, 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= 0.36, random_state=10)

#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 = confusion_matrix(Y_test,Y_pred)
print(c)

class_names = ["Aple","Mandarin","Orange","Lemon"]
sns.heatmap(c, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=class_names, yticklabels=class_names)
```

#Plotting the Confusion Matrix

```
plt.ylabel('Actual Label', fontsize=18)
plt.xlabel('Predicted Label', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```

 # 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))
```

 # 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 = []
var = []
for i in range(1,6):
    dtree = DecisionTreeClassifier(criterion='gini', random_state=0)
    dtree.fit(X_train, Y_train)
    pred = dtree.predict(X_test)
    var.append(accuracy_score(Y_test, pred))
    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),
  'var':pd.Series(var)
  })
```

 # visualizing changes in parameters

```
plt.plot('max_depth','var', data=d, label='pre pruned tree')
```

```
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='entropy', max_depth=4)
dtree.fit(X_train, Y_train)
pred = dtree.predict(X_test)
plt.plot('max_depth','var', data=d, label='pre pruned tree')
plt.plot('max_depth','acc_entropy', data=d, label='entropy')
plt.plot('max_depth','acc_gini', data=d, label='gini')
plt.xlabel('max_depth')
plt.ylabel('accuracy')
plt.legend()

# The decision tree
print(tree.plot_tree(dtree,filled=True,precision = 4))



#Creating the Confusion matrix
c = confusion_matrix(Y_test,pred)
print(c)
class_names = ["Aple","Mandarin","Orange","Lemon"]
sns.heatmap(c, square=True, annot=True, fmt='d', cbar=False,
        xticklabels=class_names, yticklabels=class_names)
#Plotting the Confusion Matrix
plt.ylabel('Actual Label', fontsize=18)
plt.xlabel('Predicted Label', fontsize=18)
```

plt.title('Confusion Matrix', fontsize=18)

plt.show()

 # 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("")

Code Snippets:

server.py | client1.py | ML_LAB_ASSESSMENT_2_ANIRUDH_VADERA(20BCE2940)DECISION_TREE.PY

```
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")
```

server.py × client1.py × ML_LAB_ASSESSMENT_2_ANIRUDH_VADERA(20BCE2940)DECISION_TREE.PY ×

```python
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))
```

Output and Results:

Dataset Details:

Dataset:

```
The dataset is as following :
    fruit_label fruit_name     fruit_subtype  mass  width  height  color_score
0             1      apple      granny_smith   192    8.4     7.3         0.55
1             1      apple      granny_smith   180    8.0     6.8         0.59
2             1      apple      granny_smith   176    7.4     7.2         0.60
3             2   mandarin          mandarin    86    6.2     4.7         0.80
4             2   mandarin          mandarin    84    6.0     4.6         0.79
5             2   mandarin          mandarin    80    5.8     4.3         0.77
6             2   mandarin          mandarin    80    5.9     4.3         0.81
7             2   mandarin          mandarin    76    5.8     4.0         0.81
8             1      apple          braeburn   178    7.1     7.8         0.92
9             1      apple          braeburn   172    7.4     7.0         0.89
10            1      apple          braeburn   166    6.9     7.3         0.93
11            1      apple          braeburn   172    7.1     7.6         0.92
12            1      apple          braeburn   154    7.0     7.1         0.88
13            1      apple  golden_delicious   164    7.3     7.7         0.70
14            1      apple  golden_delicious   152    7.6     7.3         0.69
15            1      apple  golden_delicious   156    7.7     7.1         0.69
16            1      apple  golden_delicious   156    7.6     7.5         0.67
17            1      apple  golden_delicious   168    7.5     7.6         0.73
18            1      apple       cripps_pink   162    7.5     7.1         0.83
19            1      apple       cripps_pink   162    7.4     7.2         0.85
20            1      apple       cripps_pink   160    7.5     7.5         0.86
21            1      apple       cripps_pink   156    7.4     7.4         0.84
22            1      apple       cripps_pink   140    7.3     7.1         0.87
23            1      apple       cripps_pink   170    7.6     7.9         0.88
24            3     orange     spanish_jumbo   342    9.0     9.4         0.75
25            3     orange     spanish_jumbo   356    9.2     9.2         0.75
26            3     orange     spanish_jumbo   362    9.6     9.2         0.74
27            3     orange  selected_seconds   204    7.5     9.2         0.77
```

```
Checking for missing values :
fruit_label      0
fruit_name       0
fruit_subtype    0
mass             0
width            0
height           0
color_score      0
dtype: int64
```

As the missing values is none we can proceed further:

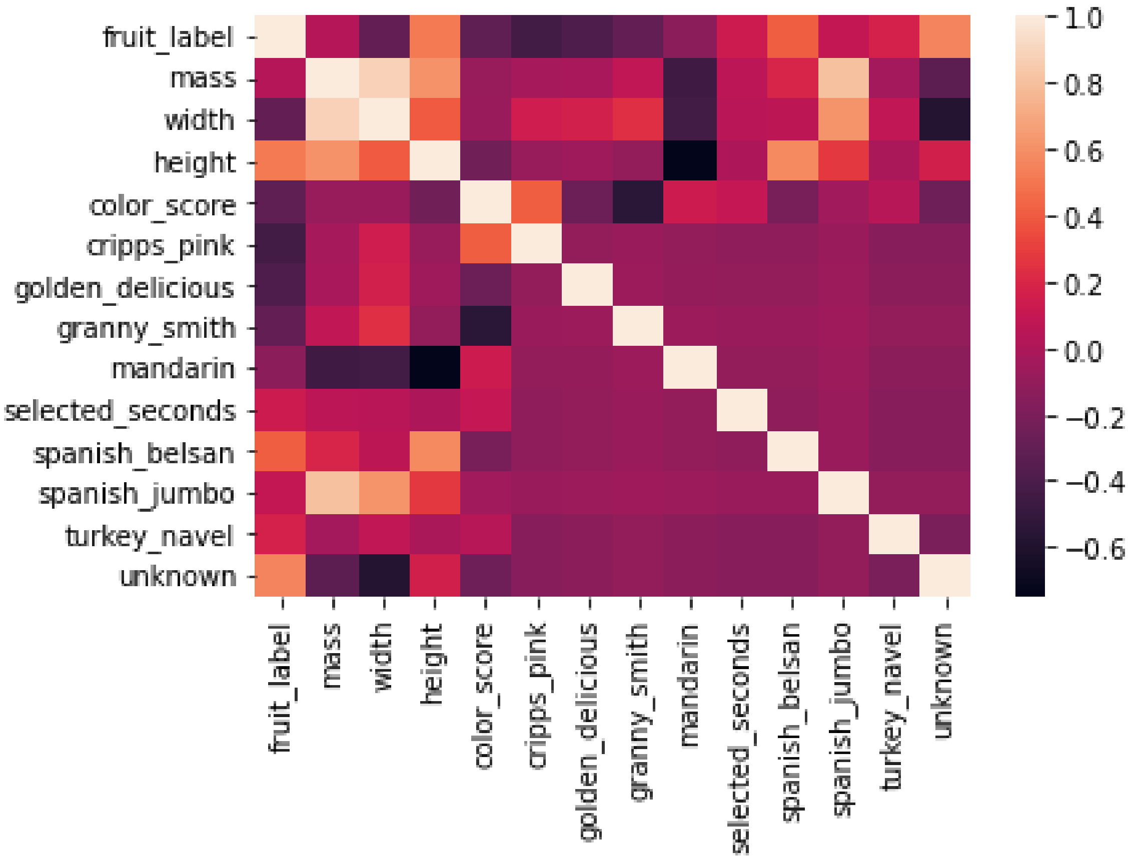
Dataset Details:

```
Dataset Header :
   fruit_label fruit_name fruit_subtype  mass  width  height  color_score
0            1      apple  granny_smith   192    8.4     7.3         0.55
1            1      apple  granny_smith   180    8.0     6.8         0.59
2            1      apple  granny_smith   176    7.4     7.2         0.60
3            2   mandarin      mandarin    86    6.2     4.7         0.80
4            2   mandarin      mandarin    84    6.0     4.6         0.79


Information regarding the columns :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   fruit_label    59 non-null     int64
 1   fruit_name     59 non-null     object
 2   fruit_subtype  59 non-null     object
 3   mass           59 non-null     int64
 4   width          59 non-null     float64
 5   height         59 non-null     float64
 6   color_score    59 non-null     float64
dtypes: float64(3), int64(2), object(2)
memory usage: 3.4+ KB
None
```

```
Dataset Details :
       fruit_label        mass      width     height  color_score
count    59.000000   59.000000  59.000000  59.000000    59.000000
mean      2.542373  163.118644   7.105085   7.693220     0.762881
std       1.208048   55.018832   0.816938   1.361017     0.076857
min       1.000000   76.000000   5.800000   4.000000     0.550000
25%       1.000000  140.000000   6.600000   7.200000     0.720000
50%       3.000000  158.000000   7.200000   7.600000     0.750000
75%       4.000000  177.000000   7.500000   8.200000     0.810000
max       4.000000  362.000000   9.600000  10.500000     0.930000
```

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

```
After Trimming and correcting the dataset looks like follows :
   fruit_label fruit_name  mass  ...  spanish_jumbo  turkey_navel  unknown
0            1      apple   192  ...              0             0        0
1            1      apple   180  ...              0             0        0
2            1      apple   176  ...              0             0        0
3            2   mandarin    86  ...              0             0        0
4            2   mandarin    84  ...              0             0        0
```

Fitting the Decision Tree Model:

70% data for training and 30% for testing:

We use the gini criterion to train our model

Prediction Results:

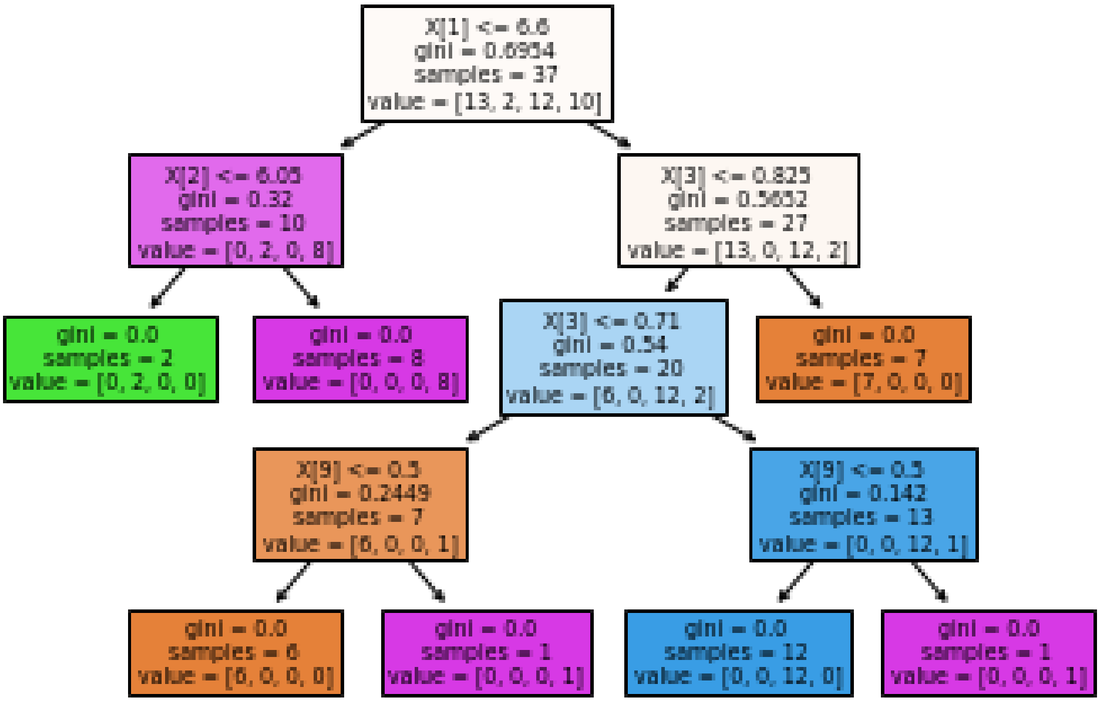
The Predicted Y_Values And The Actual Y_Values are:

```
In [4]: Y_pred
Out[4]:
array([3, 2, 3, 3, 1, 3, 4, 1, 1, 3, 3, 3, 1, 3, 2, 2, 4, 4, 4, 1, 4, 4],
      dtype=int64)

In [5]: Y_test
Out[5]:
array([3, 2, 3, 3, 1, 1, 4, 1, 1, 3, 3, 3, 1, 3, 2, 2, 4, 4, 4, 1, 4, 4],
      dtype=int64)
```

Before Pruning:

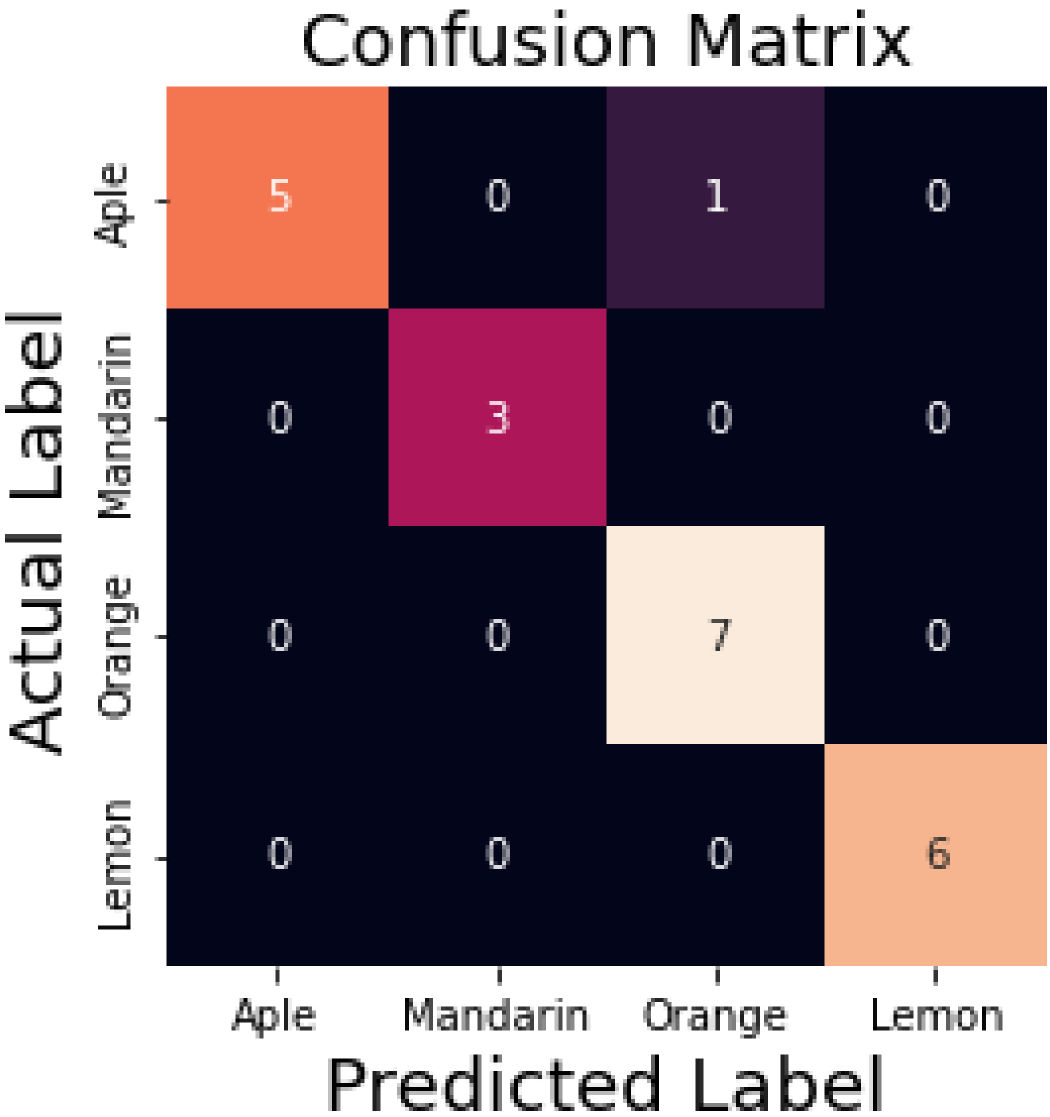
Decision Tree Before Pruning:



Confusion Matrix:

```
[[5 0 1 0]
 [0 3 0 0]
 [0 0 7 0]
 [0 0 0 6]]
```

For Individual Labels:



Accuracy Analysis(Errors):

Checking the Accuracy of the model:

```
Accuracy:  0.9545454545454546
Precision: 0.955
Recall: 0.955
```

```
*************** Evaluation on Our Model ***************
Accuracy Score:  0.9545454545454546
              precision    recall  f1-score   support

           1       1.00      0.83      0.91         6
           2       1.00      1.00      1.00         3
           3       0.88      1.00      0.93         7
           4       1.00      1.00      1.00         6

    accuracy                           0.95        22
   macro avg       0.97      0.96      0.96        22
weighted avg       0.96      0.95      0.95        22

--------------------------------------------------------
```

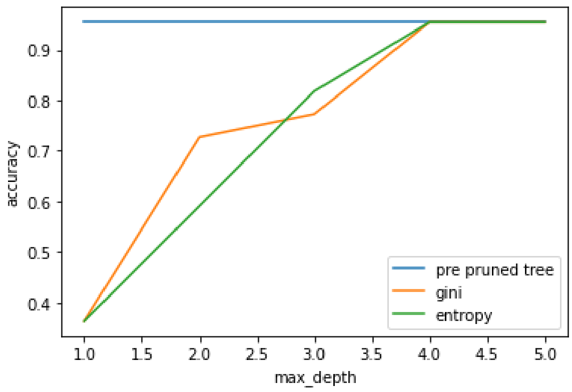
Inference:

Label1 – Apple : Label2 – Mandarin : Label3 – Orange : Label4 - Lemon

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 2, label 3 and label 4 while there is some error in predicting label 1. It classified a sing;le label 1 as label 3.
4. The accuracy of model without pruning is 95.45 percent which is quite good
5. The macro average recall is 96 percent and macro average precision is 97 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 entropy and gini both give same performance at depth = 4



Now all the points below the blue line (The pre pruned tree) is not required as it will prune the tree but the accuracy will decrease

So we chose depth 4 at which accuracy remains same for both pre and post pruned tree

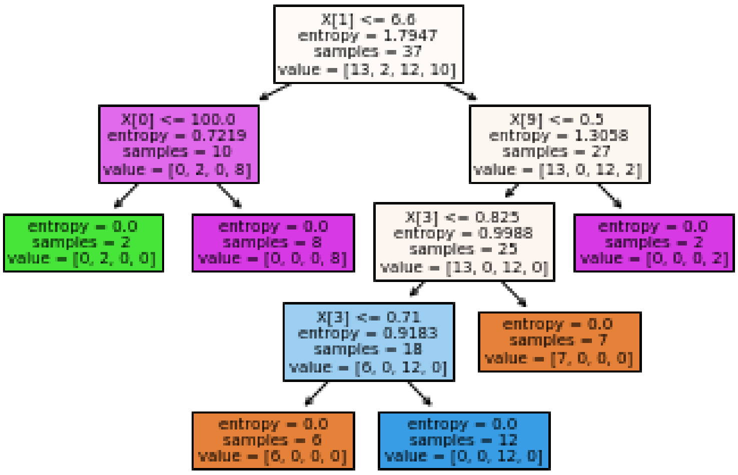
This graph tells us that at depth = 4 the both methods works best for our case that means reducing the depth to 4 will give us same accuracy and also it will reduce the number of nodes in our decision tree and the max_depth will be equal to 4 instead of 5

No of Nodes Before Pruning: 13

No of Nodes After Pruning: 11

Decision Tree After Pruning:

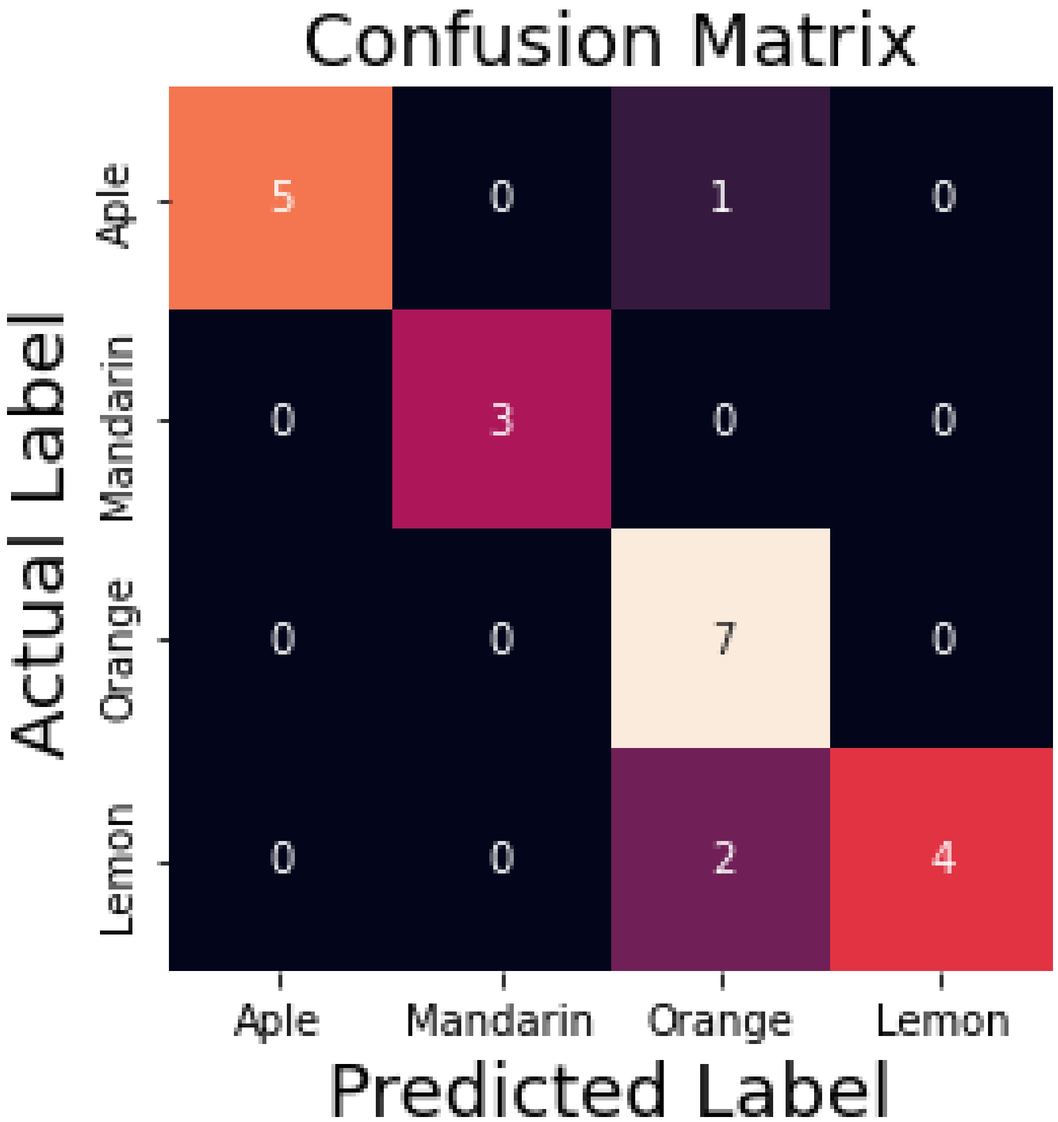
Reduced Nodes:



Confusion Matrix:

```
[[5 0 1 0]
 [0 3 0 0]
 [0 0 7 0]
 [0 0 0 6]]
```

For Individual Labels:



Accuracy Analysis(Errors):

Checking the Accuracy of the model:

```
Accuracy:  0.9545454545454546
Precision: 0.955
Recall: 0.955
```

```
*************** Evaluation on Our Model ***************
Accuracy Score:  0.9545454545454546
              precision    recall  f1-score   support

           1       1.00      0.83      0.91         6
           2       1.00      1.00      1.00         3
           3       0.88      1.00      0.93         7
           4       1.00      1.00      1.00         6

    accuracy                           0.95        22
   macro avg       0.97      0.96      0.96        22
weighted avg       0.96      0.95      0.95        22

-------------------------------------------------------
```

Inference:

Label1 – Apple : Label2 – Mandarin : Label3 – Orange : Label4 - Lemon

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 2, label 3 and label 4 while there is some error in predicting label 1. It classified a single label 1 as label 3.**
3. **Accuracy without pruning and with pruning is same 95.45 percent which means our pruning is correct**
4. **Individual scores for precision and recall are also given in evaluation and are same as before**
5. **Therefore, we successfully have reduced the decision tree size without damaging the accuracy**