Task 1

Build two decision tree classifiers with Gini index and entropy criteria for the given Wine.csv data set.

Decision tree classifier with Gini Index

https://colab.research.google.com/drive/1zhq8ZYNWiZyLgfVeY9xyf28yYW8pXhA6#scrollTo=VxvypxVE_af7

Decision tree classifier with Entropy

https://colab.research.google.com/drive/1Gqxn9VCqHmuZwcPiXEjfvatwAw_AoWBE#scrollTo=2ozjk7vKbG6E

Task 2

Demonstrate how decision trees deal with missing values.

To check missing values in variables, wine_df.isnull().sum() command can be used.

```
[23] wine_df.isnull().sum()

Alcohol                         0
Malic acid                      0
Ash                             0
Alcalinity of ash               0
Magnesium                       0
Total phenols                   0
Flavanoids                      0
Nonflavanoid phenols            0
Proanthocyanins                 0
Color intensity                 0
Hue                             0
OD280/OD315 of diluted wines    0
Proline                         0
Class                           0
dtype: int64
```

As showed above, there are no missing values in this dataset. But real-world data sets often have a lot of missing values. There are some methods used in handling missing values.

- Deleting Rows
  If the data set is large enough rows containing null values can be deleted. But if the data set is small, it's not a good option. It works poorly if the percentage of missing values is high compared to the whole dataset.
- Predicting missing values
  Using the features which do not have missing values, null values can be predicted with the help of machine learning algorithms. Unless a missing value is having a very high variance, this will result in better accuracy.
- Replacing with Mean/Median/Mode

If the attribute has a numeric value, this can be used. Mean, median or mode of the attribute can be calculated and replace with the missing values. This approximation can add variance to the data set.

Task 3

Evaluate the classifiers with suitable performance metrices

Accuracy

```
[35] print('Accuracy:', metrics.accuracy_score(y_test, y_pred))

     Accuracy: 0.3333333333333333
```

Confusion matrix

```
[37] from sklearn.metrics import confusion_matrix
     conf_mat = confusion_matrix(y_test, y_pred)
     conf_mat

     array([[11,  3,  4],
            [11,  4,  2],
            [ 7,  3,  0]])
```

Classification report

```
[ ] from sklearn.metrics import classification_report
    print(classification_report(y_test, y_pred))

                  precision    recall  f1-score   support

               1       0.38      0.61      0.47        18
               2       0.40      0.24      0.30        17
               3       0.00      0.00      0.00        10

        accuracy                           0.33        45
       macro avg       0.26      0.28      0.25        45
    weighted avg       0.30      0.33      0.30        45
```

For both decision tree classifiers with Gini index and entropy, same result was obtained for above evaluation criteria.

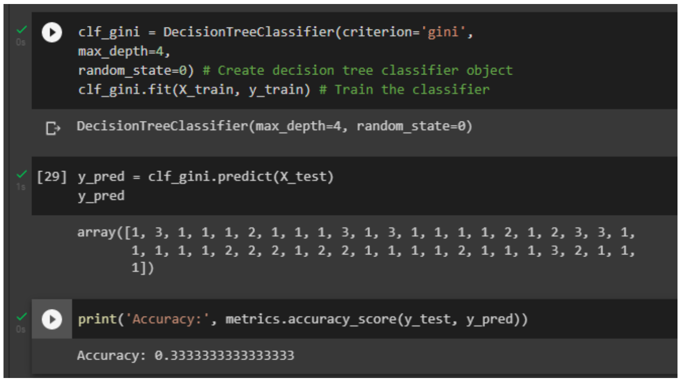
Task 4

Demonstrate how pruning can be applied to overcome overfitting of decision tree classifiers.

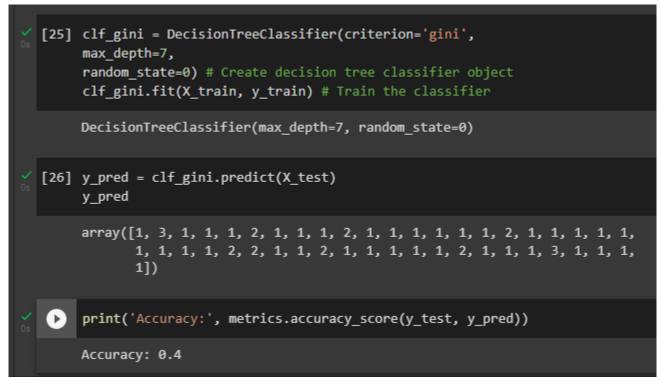
Pruning is a data compression technique that reduces the size of decision tree by removing sections of the tree that are non-critical and redundant to classify instances. There are two types of pruning namely pre-pruning and post-pruning. Here optimization of decision tree classifier is performed only using pre-pruning.

The choice of the criterion is a way of pre-pruning. Here we have tried two of them; Gini index and entropy. But the results were same for both of them.

Another way is using the maximum depth of the tree as pre-pruning. If the maximum depth value is higher, it will cause overfitting.



```
clf_gini = DecisionTreeClassifier(criterion='gini',
max_depth=4,
random_state=0) # Create decision tree classifier object
clf_gini.fit(X_train, y_train) # Train the classifier

DecisionTreeClassifier(max_depth=4, random_state=0)

[29] y_pred = clf_gini.predict(X_test)
y_pred

array([1, 3, 1, 1, 1, 2, 1, 1, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1, 2, 3, 3, 1,
       1, 1, 1, 1, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 3, 2, 1, 1,
       1])

print('Accuracy:', metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.3333333333333333
```



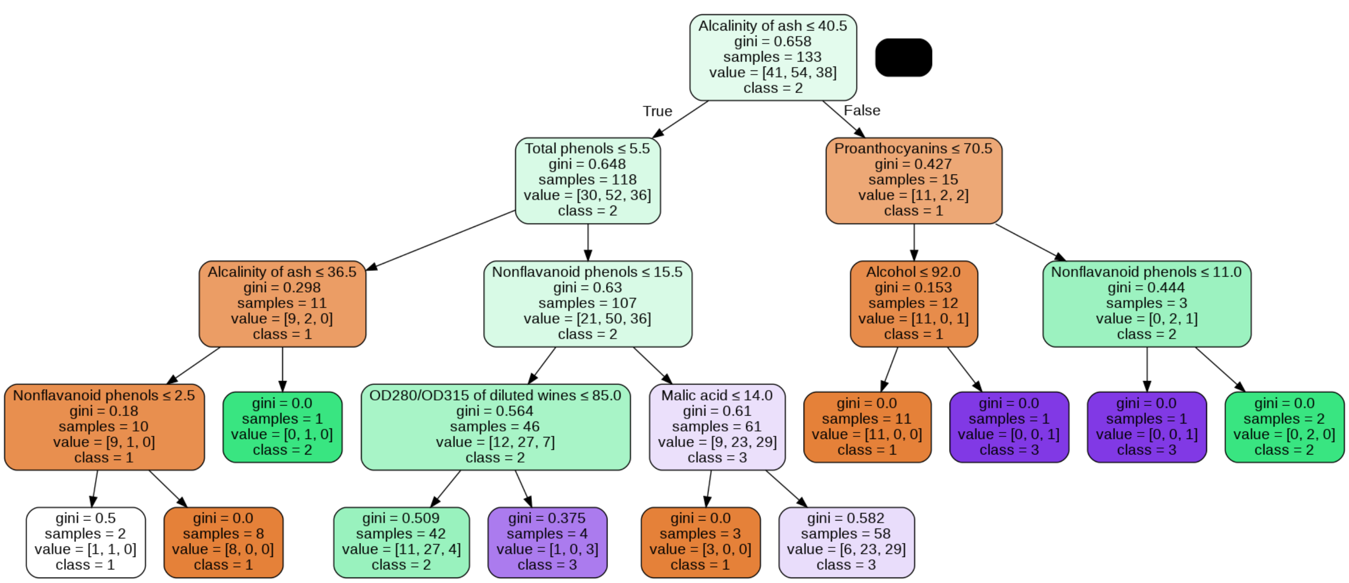
```
[25] clf_gini = DecisionTreeClassifier(criterion='gini',
max_depth=7,
random_state=0) # Create decision tree classifier object
clf_gini.fit(X_train, y_train) # Train the classifier

DecisionTreeClassifier(max_depth=7, random_state=0)

[26] y_pred = clf_gini.predict(X_test)
y_pred

array([1, 3, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 3, 1, 1, 1,
       1])

print('Accuracy:', metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.4
```

When maximum depth is set to 4, accuracy is 0.33 and when it is set to 7 accuracy has increased to 0.4

Task 5

Visualize decision trees.

With Gini index



With entropy