Problem 2 (30 points): This problem illustrates the effect of the class imbalance of the accuracy of the decision trees. Download the red wine quality data from the UCI machine learning repository at: http://archive.ics.uci.edu/ml/datasets/Wine+Quality

1. Report how many classes (treat each quality level as a different class) are and what is the distribution of these classes for the red wine data is.

We load the dataset into python first –

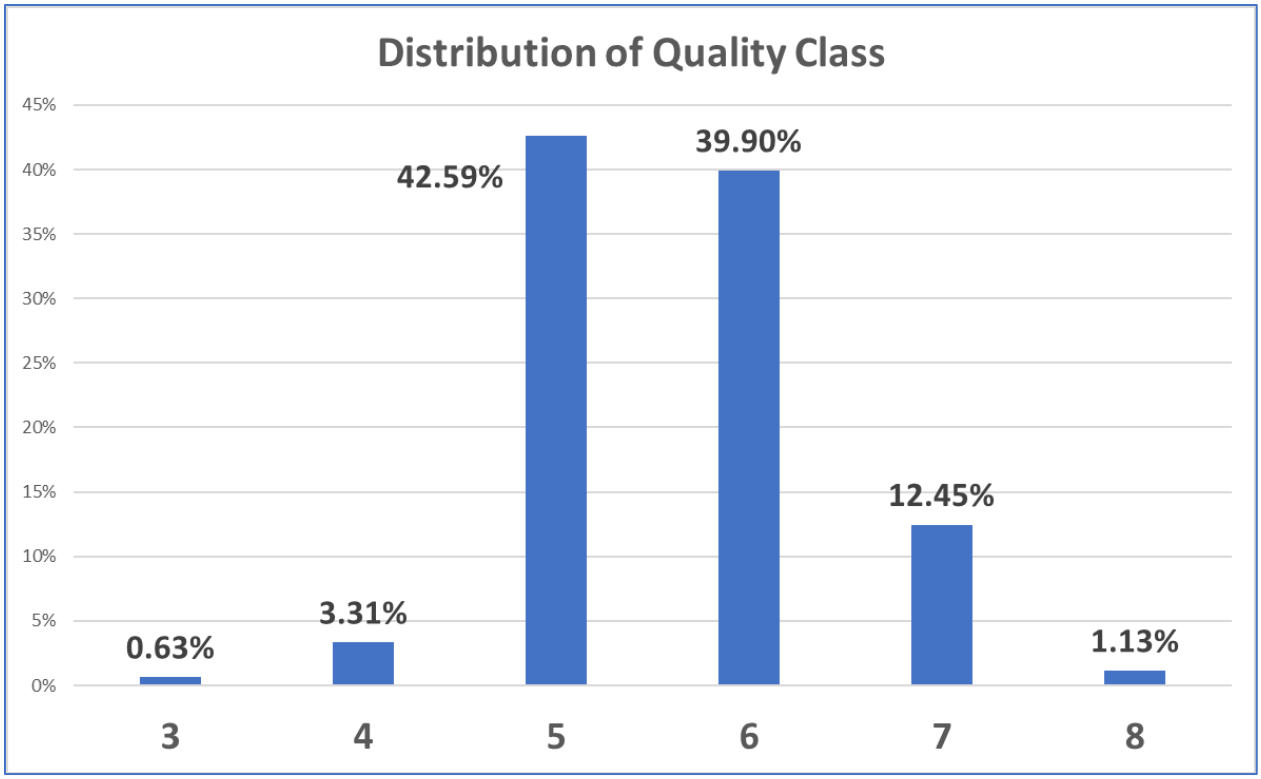
```
In [6]: dataset = pd.read_csv("C:\\Users\\Akanksha\\Downloads\\winequality-red.csv", sep = ";")
        dataset
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

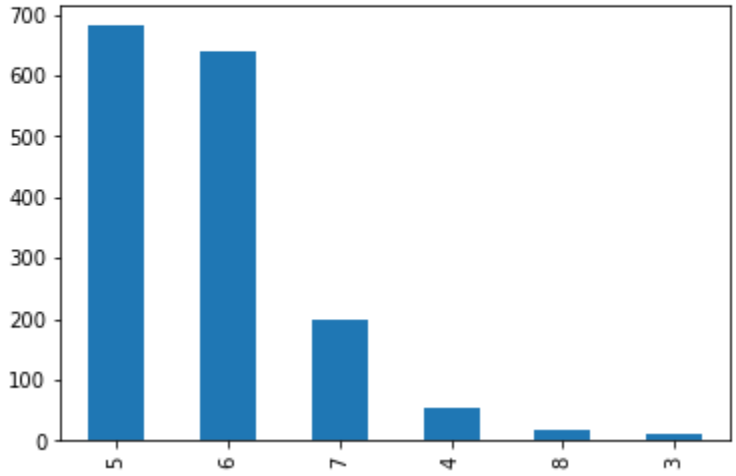
There are 6 classes (quality levels) in the dataset – 3, 4, 5, 6, 7, 8


Distribution of Quality Class
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
0.63%
3.31%
42.59%
39.90%
12.45%
1.13%
3
4
5
6
7
8

Distribution by counts (and sorted) –

```
dataset['quality'].value_counts().plot(kind='bar')
```

```
7]: <AxesSubplot:>
```



2. Repeat Problem 1 on the red wine data.

We create our initial decision tree using the Gini criterion without any limitations on depth and/or sample size.

This gives us a rather large/complex decision tree with 585 nodes and a depth of 19.

We have an accuracy of 60%. In our test dataset of 480 samples, our decision tree could predict 280 values correctly out of the 480 values resulting in a 60% accuracy.

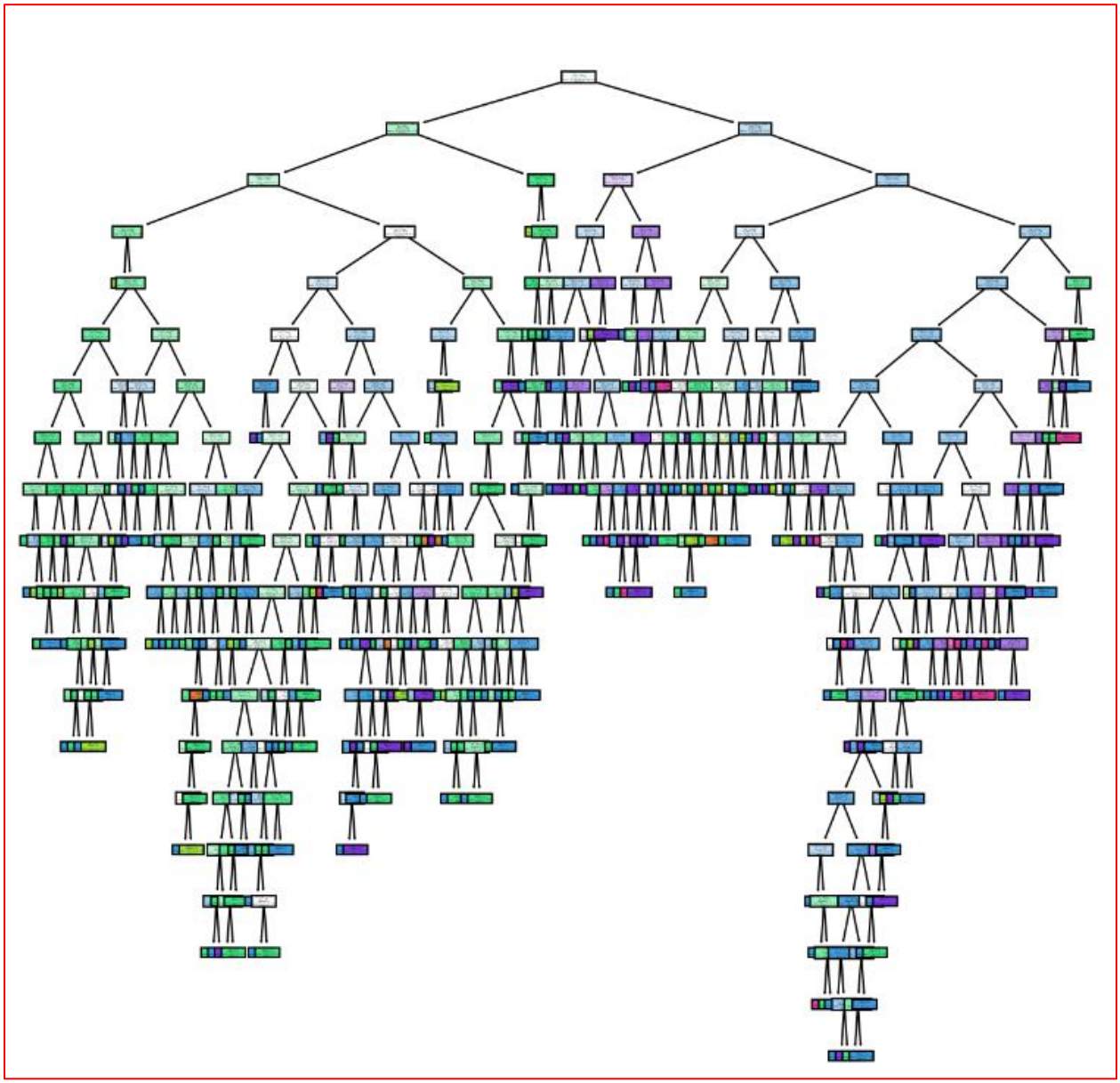
```
In [10]: #Seperating Label and Attributes
         X= dataset.drop('quality', axis =1)
         y = dataset['quality']

In [12]: from sklearn.model_selection import train_test_split
         X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.30, random_state=3)

In [15]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier()
         clf.fit(X_trainset, y_trainset)

Out[15]: DecisionTreeClassifier()

In [29]: from sklearn import metrics
         import matplotlib.pyplot as plt
         y_pred = clf.predict(X_testset)
         print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))

         DecisionTrees's Accuracy:  0.6
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_testset, y_pred))
print(classification_report(y_testset, y_pred))
```

```
[[  0   1   2   0   1   0]
 [  1   1   7   4   2   0]
 [  2   7 146  41   6   0]
 [  0   9  53 110  22   2]
 [  0   1   5  20  28   5]
 [  0   0   0   1   0   3]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         4
           4       0.05      0.07      0.06        15
           5       0.69      0.72      0.70       202
           6       0.62      0.56      0.59       196
           7       0.47      0.47      0.47        59
           8       0.30      0.75      0.43         4

    accuracy                           0.60       480
   macro avg       0.36      0.43      0.38       480
weighted avg       0.61      0.60      0.60       480
```

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 19
Total number of nodes in the tree are 585
```



We can see that the top three most important features are Alcohol, Sulphates and Volatile Acidity in our feature set.

Feature	Importance
alcohol	0.18494352

sulphates	0.10265159
volatile acidity	0.10096897
total sulfur dioxide	0.09594696
residual sugar	0.08886918
density	0.08144391
citric acid	0.07506444
pH	0.07427905
fixed acidity	0.07413289
chlorides	0.06821411
free sulfur dioxide	0.05348538

When we use the Entropy criterion, without limiting depth or sample size, we get a slightly better decision tree than the Gini criterion used above. However, the depth of the tree and number nodes increase considerably as well. This can result in overfitting too.

```
clf = DecisionTreeClassifier(criterion = 'entropy')
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.63125
```

The accuracy of this decision tree is 63.125% which is ~3% more than the previous decision tree we made using the Gini criterion. Meaning, our tree classified 303 test samples accurately out of total of 480 in the test dataset.

```
[[  0   1   2   0   1   0]
 [  0   3   6   4   2   0]
 [  0   5 144  48   5   0]
 [  1   1  46 125  21   2]
 [  1   0   5  16  29   8]
 [  0   1   0   0   1   2]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         4
           4       0.27      0.20      0.23        15
           5       0.71      0.71      0.71       202
           6       0.65      0.64      0.64       196
           7       0.49      0.49      0.49        59
           8       0.17      0.50      0.25         4

    accuracy                           0.63       480
   macro avg       0.38      0.42      0.39       480
weighted avg       0.63      0.63      0.63       480
```

More Depth and Nodes than the Gini Tree.

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 27
Total number of nodes in the tree are 591
```

3. Now bin the class variable in such a way that data is not so imbalanced with respect to the class variable. Repeat Problem 1 but on the wine data with a smaller number of classes (the binned class variable). (Combine classes into a smaller number of classes to balance the data).

Older distribution:

```
quality
3      10
4      53
5     681
6     638
7     199
8      18
dtype: int64
```

We have combined the quality classes into the following new classes:

Quality Classes 3, 4, 7 and 8 have been classified as L.

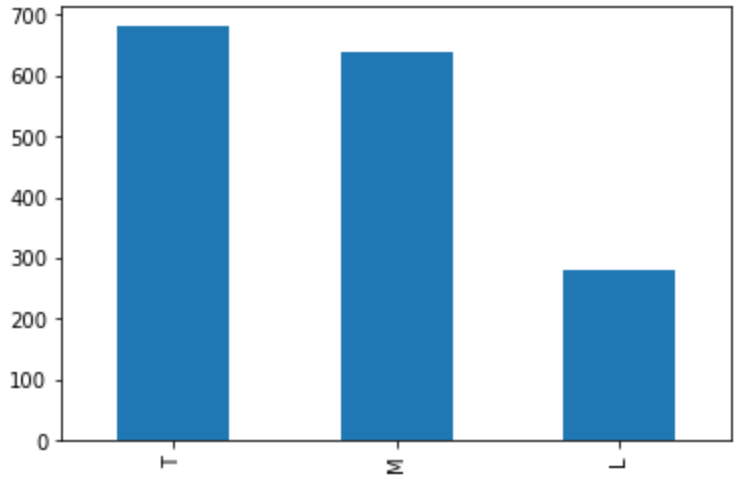
Quality Class 5 → T

Quality Class 6 -> M

New Distribution:

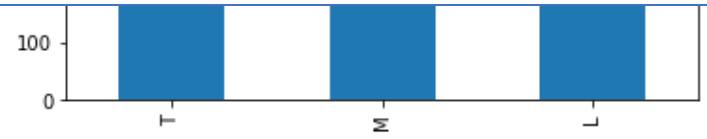
```
In [78]: dataset['quality'] = dataset['quality'].map({'3': "L", '4': "L",'5': "T", '6': "M",'7': "L", '8': "L"})

In [79]: dataset['quality'].describe()

Out[79]: count     1599
         unique       3
         top          T
         freq       681
         Name: quality, dtype: object

In [80]: dataset.groupby(['quality']).size()

Out[80]: quality
         L    280
         M    638
         T    681
         dtype: int64
```

```
In [81]: dataset['quality'].value_counts().plot(kind='bar')

Out[81]: <AxesSubplot:>
```



1. Entropy based decision tree

We first build it without setting a maximum depth and using the entropy criterion. We get an accuracy of 65.8% which is higher than our previous decision tree *(63.125% in the previous part of this question)* meaning the binning of the quality classes has made our decision tree better.




```
#Seperating Label and Attributes
X= dataset.drop('quality', axis =1)
y = dataset['quality']
```

```
from sklearn.model_selection import train_test_split
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.30, random_state=3)
```

```
clf = DecisionTreeClassifier(criterion = 'entropy')
clf.fit(X_trainset, y_trainset)
```

```
1]: DecisionTreeClassifier(criterion='entropy')
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.6583333333333333
```

Our new Entropy based decision tree is predicting 316 correct classifications on the test dataset our of 480 values which is a 65.8% accuracy. See below-

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_testset, y_pred))
print(classification_report(y_testset, y_pred))
```

```
[[ 40  23  19]
 [ 26 125  45]
 [  8  43 151]]
              precision    recall  f1-score   support

           L       0.54      0.49      0.51        82
           M       0.65      0.64      0.65       196
           T       0.70      0.75      0.72       202

    accuracy                           0.66       480
   macro avg       0.63      0.62      0.63       480
weighted avg       0.66      0.66      0.66       480
```

2. Gini Based new Tree with new binned data–

Accuracy is 63.75%, with and 18 depth. It has 553 nodes total in the tree.

```
clf = DecisionTreeClassifier()
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier()
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.6375
```

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 18
Total number of nodes in the tree are 553
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_testset, y_pred))
print(classification_report(y_testset, y_pred))
```

```
[[ 43  23  16]
 [ 30 115  51]
 [ 10  44 148]]
              precision    recall  f1-score   support

           L       0.52      0.52      0.52        82
           M       0.63      0.59      0.61       196
           T       0.69      0.73      0.71       202

    accuracy                           0.64       480
   macro avg       0.61      0.61      0.61       480
weighted avg       0.64      0.64      0.64       480
```

Comparison of new decision tree with the old one (Gini and Entropy both)

Previous Tree (Entropy):

Node Count - 591

Depth – 27

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 27
Total number of nodes in the tree are 591
```

New Tree: Lesser number of nodes and depth

Node Count - 527

Depth -17

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 17
Total number of nodes in the tree are 527
```

Previous Tree (Using Gini):

Node Count - 585

Depth - 19

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 19
Total number of nodes in the tree are 585
```

New Tree (Using Gini): Lesser number of nodes and depth

Node Count - 553

Depth -18

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 18
Total number of nodes in the tree are 553
```

4.How does the performance of the best classification model on the original class variable compares with the accuracy of the best classification model on the binned classification variable?

The performance of the new tree is better with the new binning/bucketing of the quality classes. This is true regardless of the criterion we chose to make the tree (i.e., Gini or Entropy). Below I have compared the accuracy of the Old Tree Vs New Tree (with new bins) for both Gini and Entropy based trees. We can see that the new trees *(with less classes/new bins)* perform better in both cases.

Better Accuracy -

	Old	New (with new bins)	Difference/Improvement
Gini	60%	63.75%	3.75%
Entropy	63.13%	65.80%	2.68%

Although after the new binning, the distribution is still not equal but it is still less sparse than earlier.

Reduced Complexity

In both the cases, Entropy and Gini, we also saw that the complexity of the Tree has reduced (as seen in previous answer) with reduced number of nodes and depth for the new tree.

Previous Tree (Entropy):
Node Count - 591
Depth - 27

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 27
Total number of nodes in the tree are 591
```

New Tree (using Entropy): Lesser number of nodes and depth
Node Count - 527
Depth -17

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 17
Total number of nodes in the tree are 527
```

Previous Tree (using Gini):
Node Count - 585
Depth - 19

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 19
Total number of nodes in the tree are 585
```

New Tree (Using Gini): Lesser number of nodes and depth
Node Count - 553
Depth -18

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 18
Total number of nodes in the tree are 553
```

We can always remove low importance features and check how that helps with the model performance.

5. Do you have any other ideas on how you can improve the results further? Showing that your idea will actually work will be graded with five extra credit points.

We can try a few methods to check if they increase the accuracy/performance of the model –

1. Set minimum sample size required to be at a leaf node.
2. Limit maximum depth of the tree
3. Feature Selection

For the sake of experimentation we will do this on the tree with the highest accuracy, i.e. the Entropy tree.

1. Set minimum sample size required to be at a leaf node.

I tried four combinations of minimum samples for leaf node – 20,50,100 and 250

For minimum samples at leaf node = 20, the accuracy went down to 55% which is worse than the previous tree.

```
clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf=20)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', min_samples_leaf=20)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.55625
```

For minimum samples at leaf node = 50, the accuracy went down to 59.58% which is worse than the previous tree.

```
clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf=50)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', min_samples_leaf=50)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.5958333333333333
```

For minimum samples at leaf node = 100, the accuracy went down to 60.62% which is worse than the previous tree.

```
clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf=100)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', min_samples_leaf=100)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.60625
```

For minimum samples at leaf node = 150, the accuracy went down to 55.62% which is worse than the previous tree.

```
clf = DecisionTreeClassifier(criterion = "entropy", min_samples_leaf=150)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', min_samples_leaf=150)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.55625
```

2. Limit maximum depth of the tree-

Max Depth = 10

The accuracy improves from the previous tree but is still lower than the tree in Q2. The new accuracy when we limit depth to 10 is 63.75% which is less than the accuracy of the tree in Q2.

```
clf = DecisionTreeClassifier(criterion = "entropy", max_depth = 10)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.6375
```

Max Depth = 15

The accuracy further improves from the previous but is still slightly lower than the tree in Q2. The new accuracy when we limit depth to 15 is 64.375% which is slightly less than the accuracy of the tree in Q2.

```
clf = DecisionTreeClassifier(criterion = "entropy", max_depth = 15)
clf.fit(X_trainset, y_trainset)
```

```
]: DecisionTreeClassifier(criterion='entropy', max_depth=15)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.64375
```

Max Depth = 7

The accuracy decreased from the previous tree but is still slightly lower than the tree in Q2. The new accuracy when we limit depth to 15 is 64.375% which is slightly less than the accuracy of the tree in Q2.

```
clf = DecisionTreeClassifier(criterion = "entropy", max_depth = 7)
clf.fit(X_trainset, y_trainset)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.6208333333333333
```

3. Feature Selection

Finally, I chose to look at feature selection and check if we can remove a few variables which are least important. For this we used the *feature_importance* method to get feature importance score.

```
clf.feature_importances_
```

```
array([0.01900315, 0.10479492, 0.05465905, 0.03440136, 0.02684032,
       0.05096721, 0.16793969, 0.05074355, 0.03327093, 0.16137887,
       0.29600096])
```

We can see that the lowest important feature is 'Fixed Acidity' (first feature) and so we create a tree after removing that variable/feature.

```
#Seperating Label and Attributes
dataset = dataset.drop(['fixed acidity'], axis=1)
```

```
dataset
```

]:

	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	T
1	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	T
2	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	T
3	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	M
4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	T
...	...	...	...	...	...	...	...	...	...	...	...
1594	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	T
1595	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	M
1596	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	M
1597	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	T
1598	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	M

When we create the tree again, we can see that at max_depth =15 and removing an unimportant feature , our accuracy went up slightly even from the best tree we made in the B part of this question. The new accuracy is 67.5% which is the highest accuracy we ever received in all our combinations.

```
clf = DecisionTreeClassifier(criterion = "entropy", max_depth = 15)
clf.fit(X_trainset, y_trainset)
```

```
]: DecisionTreeClassifier(criterion='entropy', max_depth=15)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_testset)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, y_pred))
```

```
DecisionTrees's Accuracy:  0.675
```

The complexity of the tree also reduces as the number of nodes reduces to 513 which is lower than previous trees.

```
print("Depth of the Decision tree is:",clf.tree_.max_depth)
print("Total number of nodes in the tree are",clf.tree_.node_count)
```

```
Depth of the Decision tree is: 15
Total number of nodes in the tree are 513
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_testset, y_pred))
print(classification_report(y_testset, y_pred))
```

```
[[ 45  22  15]
 [ 26 122  48]
 [  8  37 157]]
              precision    recall  f1-score   support

           L       0.57      0.55      0.56        82
           M       0.67      0.62      0.65       196
           T       0.71      0.78      0.74       202

    accuracy                           0.68       480
   macro avg       0.65      0.65      0.65       480
weighted avg       0.67      0.68      0.67       480
```