

Project 8 Report

Team members: Manish Meshram, Supratik Chanda

Introduction

The idea behind our analysis is to use machine learning techniques and algorithms to determine which physiochemical properties make a wine 'good'! The dataset is related to red variants of the Portuguese "Vinho Verde" wine. We have used decision trees and neural networks to train the model on the data and predict the quality of wine. Both the models performed similar in the experiments. For decision tree, we have also analyzed the effect of gini feature importance and how they are correlated with finding node impurity.

Dataset

The dataset contains 1599 instances of physiochemical properties of wine. All the wine samples are related to red variants of Portuguese "Vinho Verde" wine. This dataset was pretty clean with no null values. Following are the independent variables we have in our dataset:

1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol

The dependent variable is 'quality' which is the score between 1 and 10 where 1 represents poor quality and 10 represents the highest quality of wine. The dataset was unbalanced between the different categories of 'quality', to make it a balanced dataset and to also deliver more value from our analysis, we have divided the data amongst two categories; 'good' and 'bad' and trained our models on the transformed data.

Analysis technique

Exploratory Data Analysis

To start with the analysis we first studied the data and checked if it is suitable for modelling and further analysis. We started with plotting out the frequencies of our dependent categories just to get an idea about the distribution of our dependent variable. Once we figured out that it is quite unbalanced, we modified the target variable and divided the data into two classes based on the

value of 'quality' of each wine. We have used a criteria of naming a wine 'bad' if its quality is $<=5$ and 'good' if its quality $>=6$. The other reason for dividing the data is because it seemed to be delivering more value to the user where we just tell him whether the quality of wine is 'good' or 'bad' rather than giving a quality point to the wine.

We have also performed correlations between the variables and plotted a heatmap to get more idea about correlations amongst all the variables.

We also found out the most important features from the independent variables. For this, we used the SelectKBest function of the feature Selection package of Scikit learn. We used chi2 as the score function and found out the p-values and scores of every column of X Feature. Then, we created a dataframe containing the name of the X Features columns, their p_values and the scores from highest p_values and score to the lowest. From the 10 features, according to the p_values we took the first 6 columns with highest p_values for the decisiontree classifier. Surprisingly when we counted the number of unique values of the Y Label, there is a high percentage of few Labels and extremely low number of other Labels like 1 and 2. For experimental purposes, we used the top 5 features obtained from SelectKBest feature selection technique, that have most significant p-values. Even though on using the features, the performance didn't increase. The F1 score was identical to the F1 score that we found out using all features. Hence, we used all the X Features. The below experiments use all X features.

Decision Trees

To kick off the analysis, we used the feature importance or gini importance of every column using the feature_importances_ function of decision tree classifier. For decision trees we have performed experiments on both kind of datasets; (1) where quality value ranges from 1 to 10 and (2) where we just have two classes for the data namely 'good' and 'bad'.

We used 3 types of tuning parameters of decisiontreeclassifier: 1) Minimum samples of leaf at each node(internal and leaf) 2) Maximum depth of the tree 3) Maximum leaf nodes

1) Minimum samples of leaf at each node(internal and leaf)

Part1 Analysis: Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 1000

Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 1000

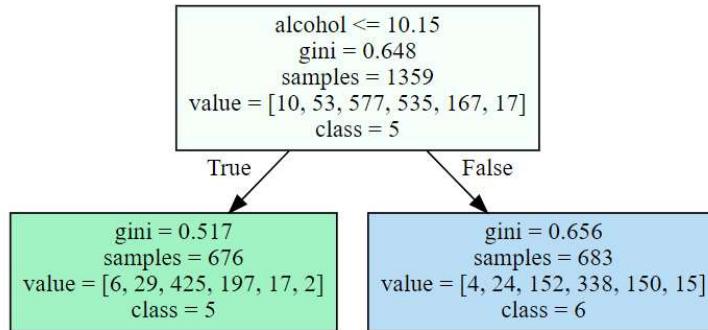
gini = 0.648
samples = 1359
value = [10, 53, 577, 535, 167, 17]
class = 5

This image provides a detailed analysis why although the maximum depth is 3, but there are no internal nodes as well as leaf nodes. There is only one node. This is because of the parameter known as minimum samples of leaf at each node. As we see that, the min_samples_leaf is equal

to 1000. So, if the leaf has to be splitted into two other leaf nodes, then the samples would have been less than 1000. Henceforth, although the max_depth is 3 , the leaf has no branch. The bias-variance tradeoff does depend on the depth of the tree. Since, the depth of the tree is 0 , the tree has high bias and low variance. That means, that the tree wil make huge assumptions in predicting the target Label while greatly change if we change few samples in training dataset. If we look at the gini parameter, it is 0.648. This means that the node is not at its purest level. As you can see, the root node (depth=0) didn't chose any X Feature as the splitting feature.

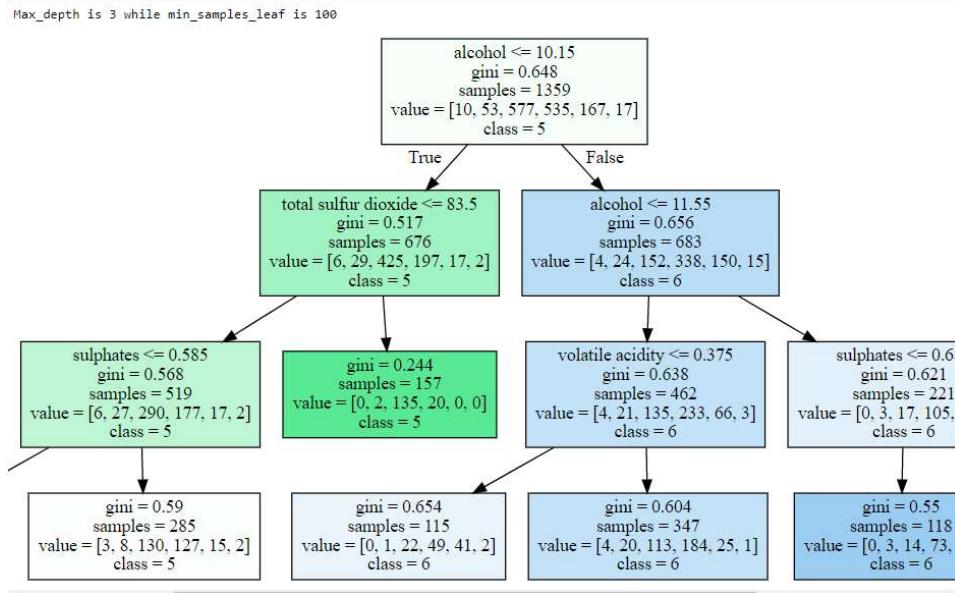
Part2 Analysis: Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 500

Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 500



As you can see in the figure, the maximum depth is 1 not 3. This is because of the minimum_samples_split. If the nodes were splitted further into leaf nodes, then the minimum no of samples of each leaf node would have been lesser than 500 and that contradicts the parameter min_samples_leaf which is 500. Therefore each sample at every node has a value greater than 500. Hence, proved that min_samples_leaf has a greater effect than max_depth. The root node (depth=0) chose "alcohol" as the splitting feature, and 10.15 as the threshold value. The bias-variance tradeoff does depend on the depth of the tree. Since, the depth of the tree is 1 , the tree has relatively less higher bias and a bit higher variance. That means, that the tree will make less huge assumptions in predicting the target Label while change to a certain amount if we change few samples in training dataset. The gini impurity is 0.648 and still high. It is still far away than its purest form . The two depth 1 nodes didn't chose any of the X Features as the their splitting feature, respectively. The two leaf nodes (depth=1) each predicts a class - one of them predict class_5 and one predicts class_6. Both of the leaf nodes have a large gini impurity (0.517 and 0.656). Hence, the tree suggests that it is not fitted properly and can be further broken down.

Part3 Analysis: Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 100



As it is seen, in this case the max_depth is 3 and not less. This is because even after reaching a depth of 3 , the leaf nodes' samples count is more than 100. The least sample leaf count is 103. The total leaf nodes count is exactly 10. If the max leaf nodes were 2 then although the min_samples_leaf would be more than 500, max_depth would be 1 This is because the max_leaf_nodes were attained and no more splitting would happen. The highest and most significant gini value is found to be 0.244 which determines a class of 5. For this particular node, the leaf has low bias and high variance. For the other cases, since the node is not pure, the node has high bias and low variance.

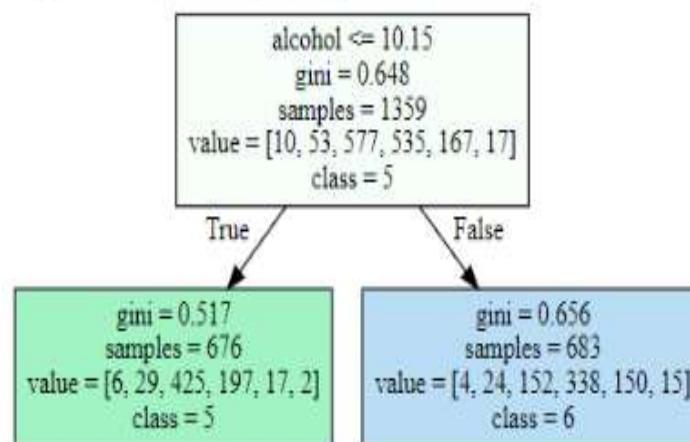
2) Maximum depth of the tree

Part 1 Analysis: Max_depth is 1 while min_samples_leaf is 10

```

1 dTree_Max_Depth_1= DecisionTreeClassifier(criterion='gini',max_depth=1,min_samples_leaf=10,random_state=4)
2 dTree_Max_Depth_1.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Depth_1,feature_names=list(FeatureDTFrame.Features[0:10]),
4                                         class_names=['3','4','5','6','7','8'],filled=True))
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 1 while min_samples_leaf is 10')
8 display(tempSVG)
  
```

Max_depth is 1 while min_samples_leaf is 10



This image provides a detailed analysis why depth is 1 although the minimum samples leaf is 10. As you can see, the root node (depth=0) chose "alcohol" as the splitting feature, and 10.15 as the threshold value. The two depth 1 nodes has gini value as 0.517 and 0.656 which means that the leaf node is not all homogenous and has a high bias. These attributes are explained below: In each node, "samples" represents how many training instances fall in that node. For example, there are 1359 samples that has alcohol ≤ 10.15 . The attribute "value" gives how many instances fall in each class for each node. For example, in the leftmost leaf node, there are 676 instances in class_5, and 683 instances in class_6. Each node is also assigned a "class" based on which class has the majority of the instances in that node. Therefore, even though the root node has comparable number of instances in each of the three classes, the root node's class label is "class_5" since $425 > 197 > 29$ To make a prediction, you keep going along the tree until you reach a leaf node. The probability of each class is the fraction of training samples of each class in a leaf node. For example, the bottom left leaf node predicts class_2 with a probability of $425/676 \approx 62.0\%$. This attribute can be obtained through DecisionTreeClassifier's predict_proba() function. Note that all samples that fall in the same leaf node share the same prediction probability.

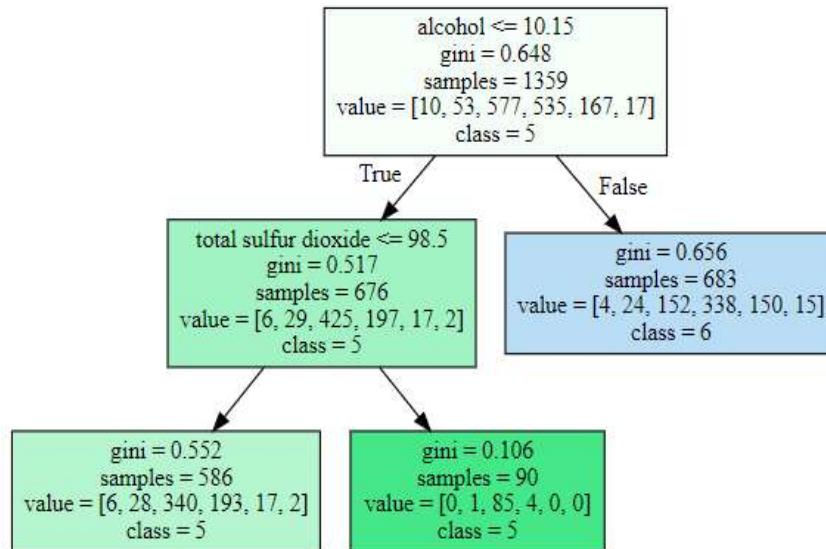
Now if we add max_leaf_nodes in the analysis:Max_depth is 1 ,max_leaf_nodes is 3 while min_samples_leaf is 10.

```

1 dTree_Max_Depth_1= DecisionTreeClassifier(criterion='gini',max_depth=1,max_leaf_nodes=3,min_samples_leaf=10,random_state=4)
2 dTree_Max_Depth_1.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Depth_1,feature_names=list(FeatureDTFrame.Features[0:10]),
4                                         class_names=['3','4','5','6','7','8'],filled=True))
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 1 and max_leaf_nodes is 3 while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 1 and max_leaf_nodes is 3 while min_samples_leaf is 10



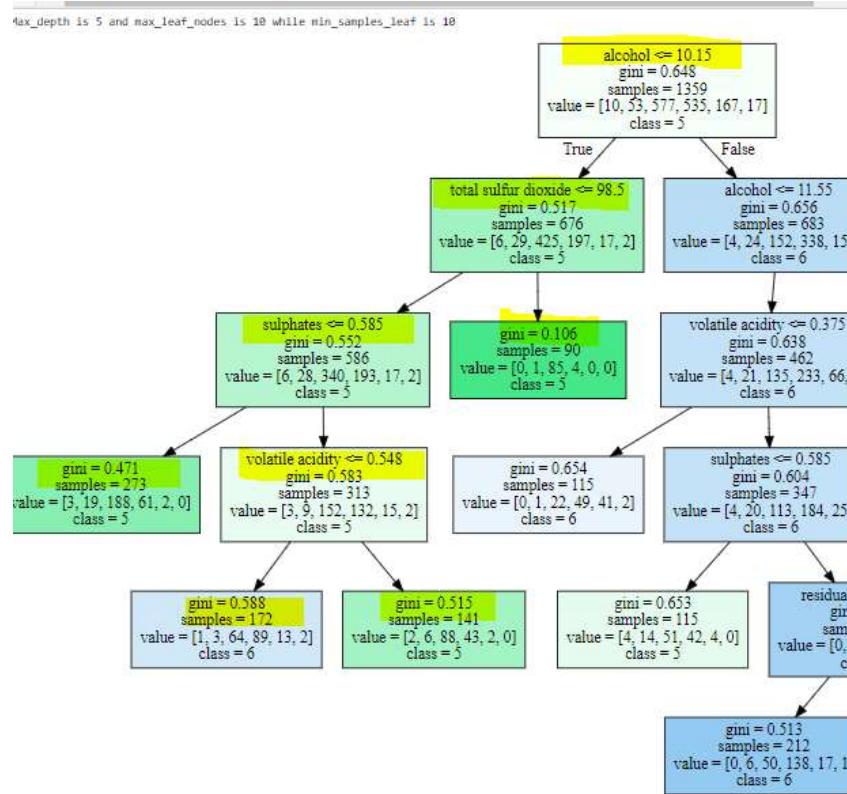
The first question that comes to mind is although the max depth is 1 , then why in the diagram the max depth is 2. This is because of the parameter max_leaf_nodes. Although the max depth is 1, the algorithm prints the tree till the max_leaf_nodes is not exhausted. In depth 1, the leaf nodes are 2 which is less than 3 . Hence the depth goes down to level 2 and then the max_leaf_nodes become 3. From this we can infer that max_leaf_nodes has a greater priority over max_depth. The gini impurity is still high and far away from giving homogenous nodes. In each node, "samples" represents how many training instances fall in that node. For example, there are 1359 samples that has alcohol ≤ 10.15 . So, the parent node is divided into two internal nodes, one with alcohol > 10.15 and other with alcohol ≤ 10.15 and total sulfur dioxide ≤ 98.5 .The internal node with total

sulfur dioxide is further divided into two leaf nodes. For Leaf Node 1: gini is 0.552, that implies that the leaf node has a high bias . Node 2 has a good gini though, 0.106 close to 0 and that tells that the leaf nodes is rather pure with a low bias. For the leaf node which has alcohol value > 10.15, we have a bad gini which is equal to 0.656.

Part 2 Analysis: Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10

In this case, the algorithm first goes to a depth of 5 and sees that leaf nodes produced by the tree is 10. So all the criteria is matched and hence no more depth. The best gini parameter that we got in this tree is of 0.106. The gini impurity is close to 0.0. Hence, for this node we can say that it is highly homogenous and have a class_5

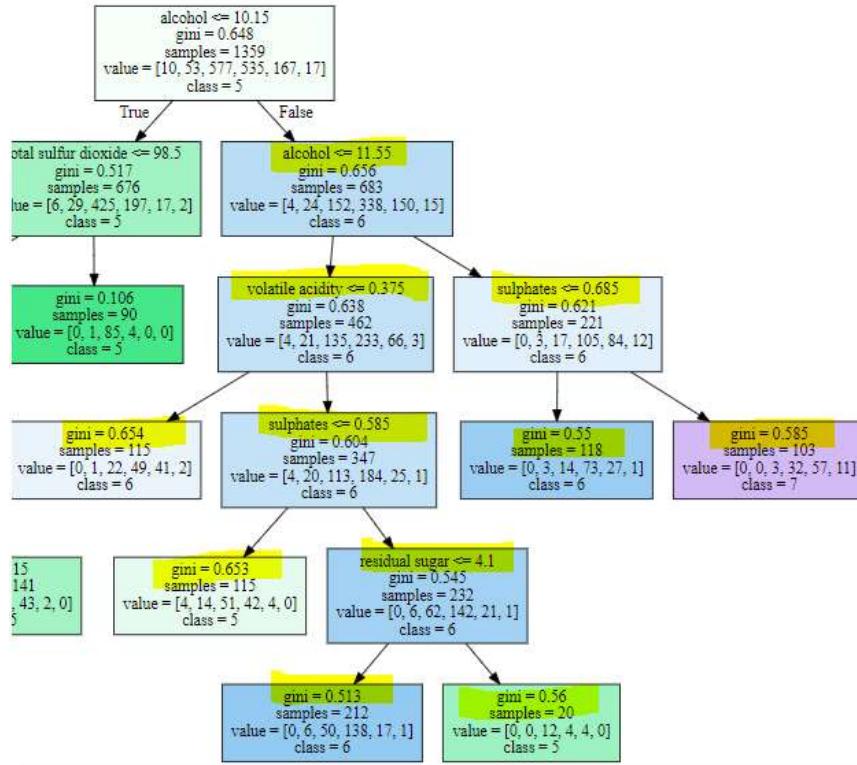
First branch(left side) details where all the nodes are highlighted:



First internal node(total sulfur dioxide) is formed by one condition: alcohol ≤ 10.15 . The next two internal nodes are formed by: total sulfur dioxide .One leaf node with total sulfur dioxide > 98.5 with a great gini value 0.106.The other internal node with total sulfur dioxide ≤ 98.5 and lower gini value 0.552. The next two internal nodes are formed by: sulphates.One leaf node with sulphates ≤ 0.585 .The other node with condition sulphates > 0.585 . The node with sulphates > 0.585 is further divided into two nodes. The condition is volatile acidity ≤ 0.548 The nodes formed by the condition volatile acidity ≤ 0.548 are the leaf nodes.

Second branch(right side) details where all the nodes are highlighted:

Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10



First internal node (alcohol ≤ 11.55) is formed by one condition: alcohol > 10.15 . The internal node alcohol ≤ 11.55 is broken into two nodes. One node with volatile acidity ≤ 0.375 with a gini value 0.638. The other internal node with sulphates ≤ 0.685 and has a gini value 0.621. The node containing volatile acidity ≤ 0.375 is further divided into two nodes. One is a leaf node and the other is an internal node. The internal node has a condition of sulphates ≤ 0.585 and is also divided into one leaf node and the other internal node with a condition of residual sugar ≤ 4.1 . This internal node is ultimately divided into two leaf nodes with bad gini impurity of 0.513 and 0.56. On the other hand, the node with sulphates ≤ 0.685 is further divided into two leaf nodes and is complete. The two leaf nodes have a moderate gini impurity of 0.55 and 0.585. Both of them are not homogeneous. The bias is high, that means the prediction will take a lot of assumption. And as the variance is low, any sample addition is not going to make a significant difference in the predicted result.

3) Maximum leaf nodes

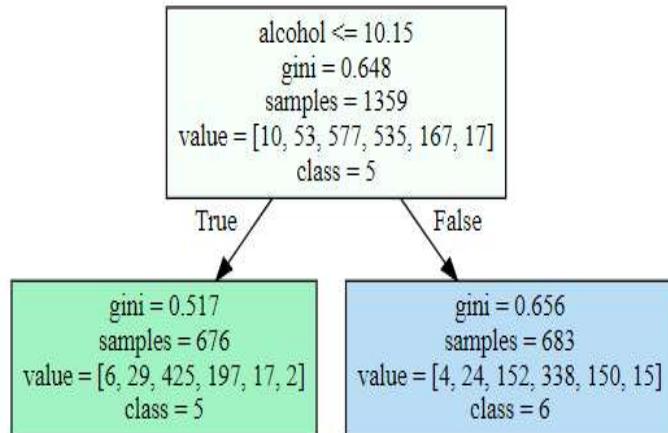
Part 1 Analysis: Max_depth = 5, max_leaf_nodes = 2, min_samples_leaf = 10

```

1 dTree_Max_Leaf_Nodes_1= DecisionTreeClassifier(criterion='gini',max_depth=5,max_leaf_nodes=2,min_samples_leaf=10,random_state=42)
2 dTree_Max_Leaf_Nodes_1.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Leaf_Nodes_1,feature_names=list(FeatureDTFrame.Features[0:10]),
4                                         class_names=['3','4','5','6','7','8'],filled=True))
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 2 while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 5 and max_leaf_nodes is 2 while min_samples_leaf is 10



This image provides a detailed analysis of why the tree having a `max_depth` of 5 only lands up to a depth of 1. The parameters given in making the decisiontree are `max_depth`, `max_leaf_nodes` and `min_samples_split`. As soon as the first parent node with a condition of `alcohol <= 10.15` splits into two leaf nodes, the parameter `max_leaf_nodes` is exhausted. We have two leaf nodes. One with a gini impurity of 0.517 and the other with a gini impurity of 0.656. Both of them are not homogeneous and have a high bias and low variance. This tree is therefore, not capable of producing a good prediction as the nodes are not in the purest form.

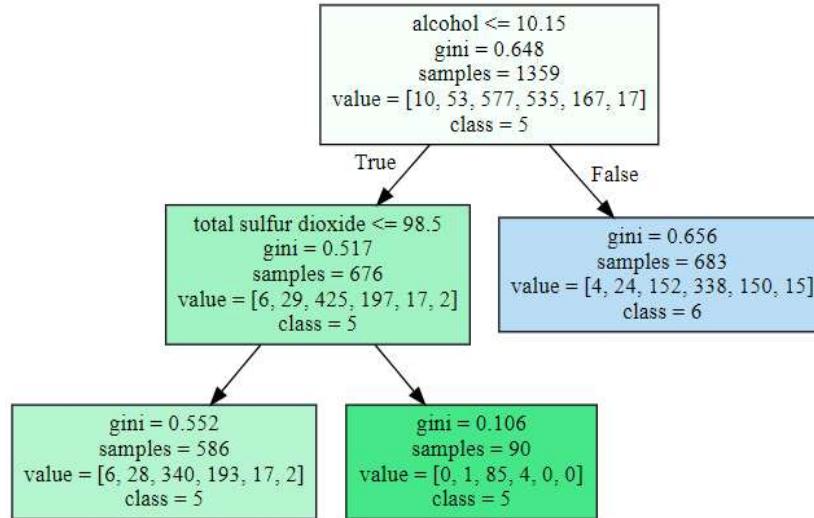
Part 2 Analysis: Max_depth = 5,max_leaf_nodes = 3,min_samples_leaf=10

```

1 dTree_Max_Leaf_Nodes_2= DecisionTreeClassifier(criterion='gini',max_depth=5,max_leaf_nodes=3,min_samples_leaf=10,random_state=42)
2 dTree_Max_Leaf_Nodes_2.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Leaf_Nodes_2,feature_names=list(FeatureDTFrame.Features[0:10]),
4                                         class_names=['3','4','5','6','7','8'],filled=True))
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 3 while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 5 and max_leaf_nodes is 3 while min_samples_leaf is 10



This image provides a detailed analysis of why the tree having a `max_depth` of 5 only lands up to a depth of 2. The parameters given in making the decisiontree are `max_depth`, `max_leaf_nodes` and `min_samples_split`. The first parent node is divided on the condition `alcohol <= 10.15`. One of them with a condition of `alcohol > 10.15` is a leaf node, with a bad gini value of 0.656 which is not all homogeneous. The other internal node that is formed with the condition `alcohol <= 10.15` is divided into two leaf nodes. The condition for splitting is `total sulfur dioxide`. One of them has `total sulfur dioxide <= 98.5` and the other has `total sulfur dioxide > 98.5`. Both of them are leaf nodes. One of them is not though in its purest forms. Gini value is high and close to 0.6. While the other one is more closer to its purest form. The gini value is 0.106 which is more nearer to 0.0. The bias for this node is relatively much lower and has a high variance. This is definitely good. So the total leaf nodes is already 3 but the `max_depth` is not 5. Now, if any of the tree node gets divided into to leaf nodes, the `max_leaf_nodes` would be higher than 3 and that violates the parameter `max_leaf_nodes`. Henceforth, the `max_depth` is retained at 2.

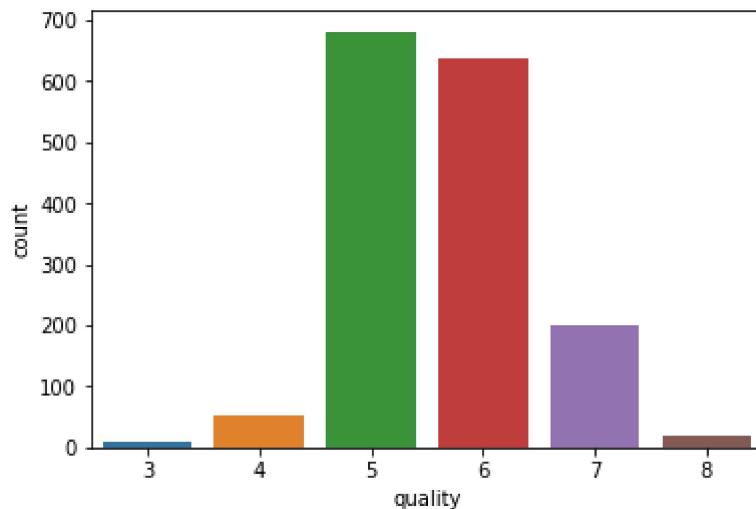
Neural Networks

Neural networks seemed to be a good bet after decision trees. We have used sklearn's Multi Layer Perceptron library for our analysis. We have followed a train-test split of 75-25 and also scaled the data based on mean and standard deviation before feeding it to the training. We have trained 3 different neural networks where we tried a bunch of different parameters while training and selected the best ones based on the performance. During this exercise we tried various hidden layers with different number of neurons, 'logistic' & 'relu' activation functions and 'adam' & 'sgd' as our solvers. We have observed that 'relu' activation function with 'adam' solver gives best results if all the other parameters kept constant. We have also tried different values of maximum iterations while training the models ranging from 500 to 10000.

Results

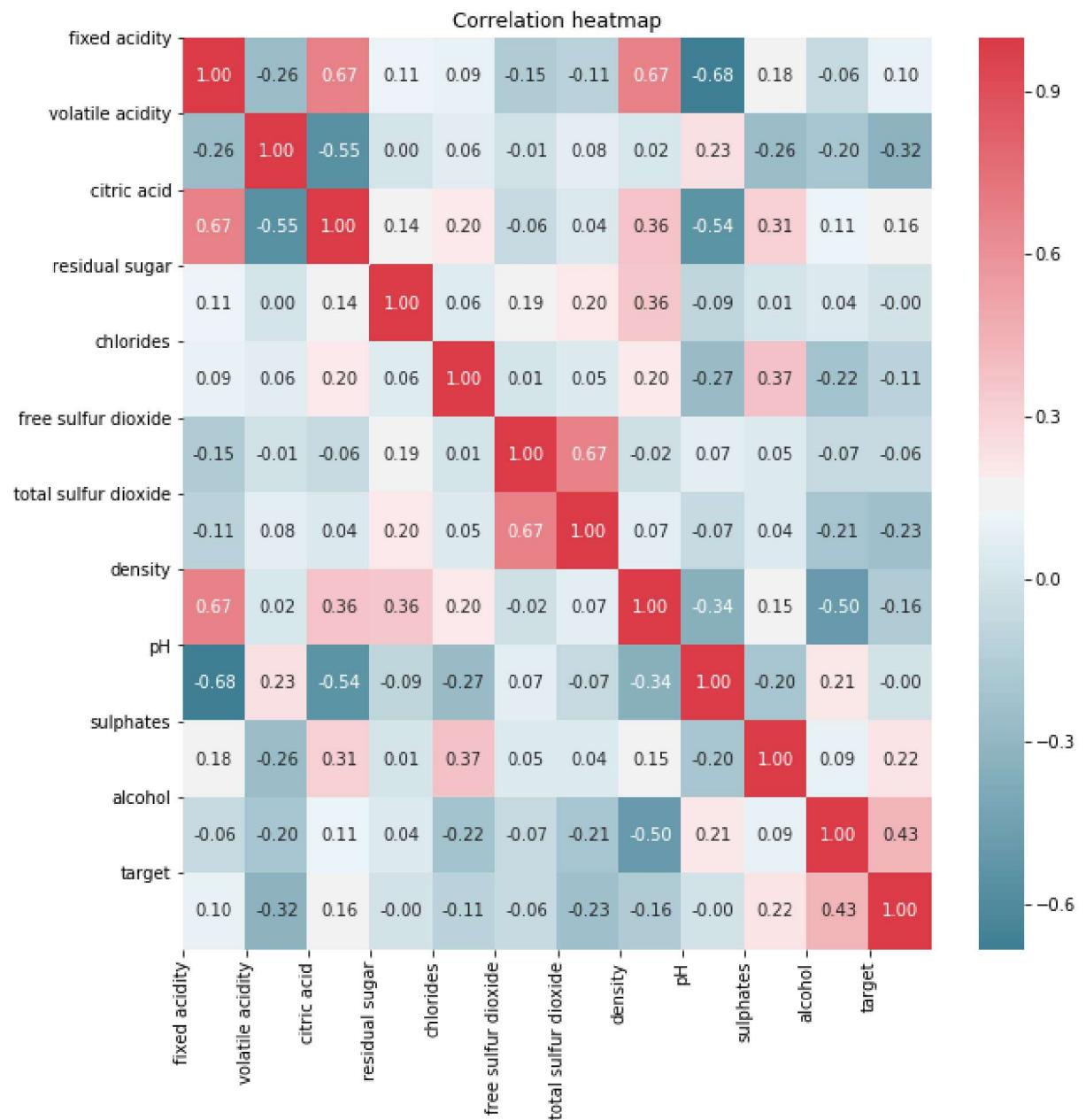
Exploratory Data Analysis

The following is the frequency plot of each 'quality' category in the data:



From the plot it can be seen that the data is highly unbalanced across the different categories and also we don't have very poor or very high quality wine in our dataset. To make it more balanced we divided the data into two categories namely 'good' and 'bad' based on the criteria explained in the Analysis section. After breaking it down we got a quite balanced data with 744 instances in Bad quality(class=0) and 855 instances in Good quality(class=1) for wine.

We also checked the correlations amongst different variables in the data and here are the results:



From the correlation heatmap we can see that there is no strong relationship between independent variables and our target variable, although we can see there is a slight positive relationship between alcohol and our target variable which implies the quality of wine tends to get better with increasing concentration of alcohol. There is also a good negative correlation between fixed acidity and pH values, which makes complete sense as we know alkalinity increases as the pH value increases.

Decision Trees

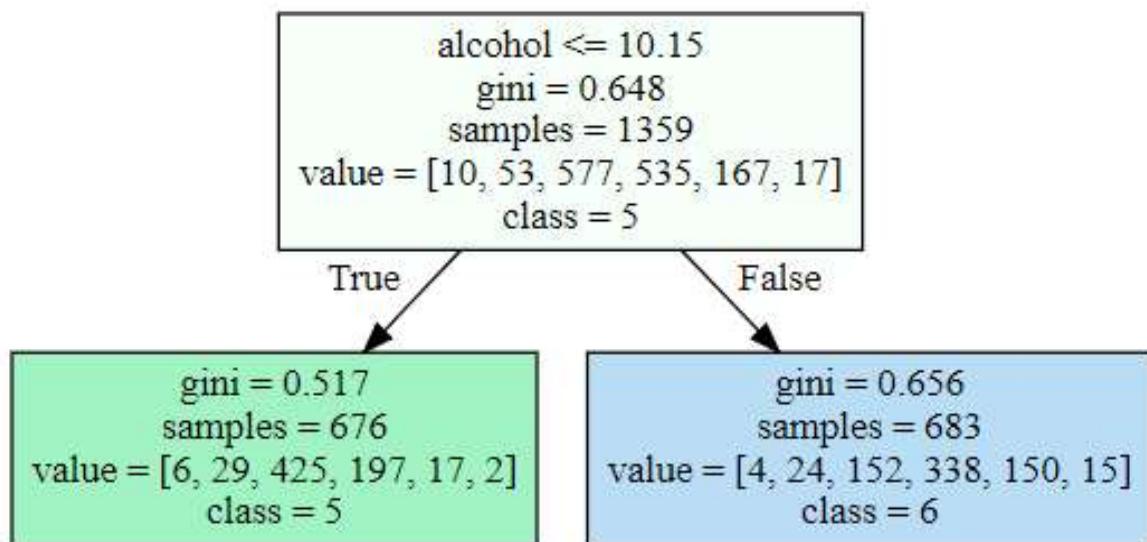
Min_Samples_Leaf:

```

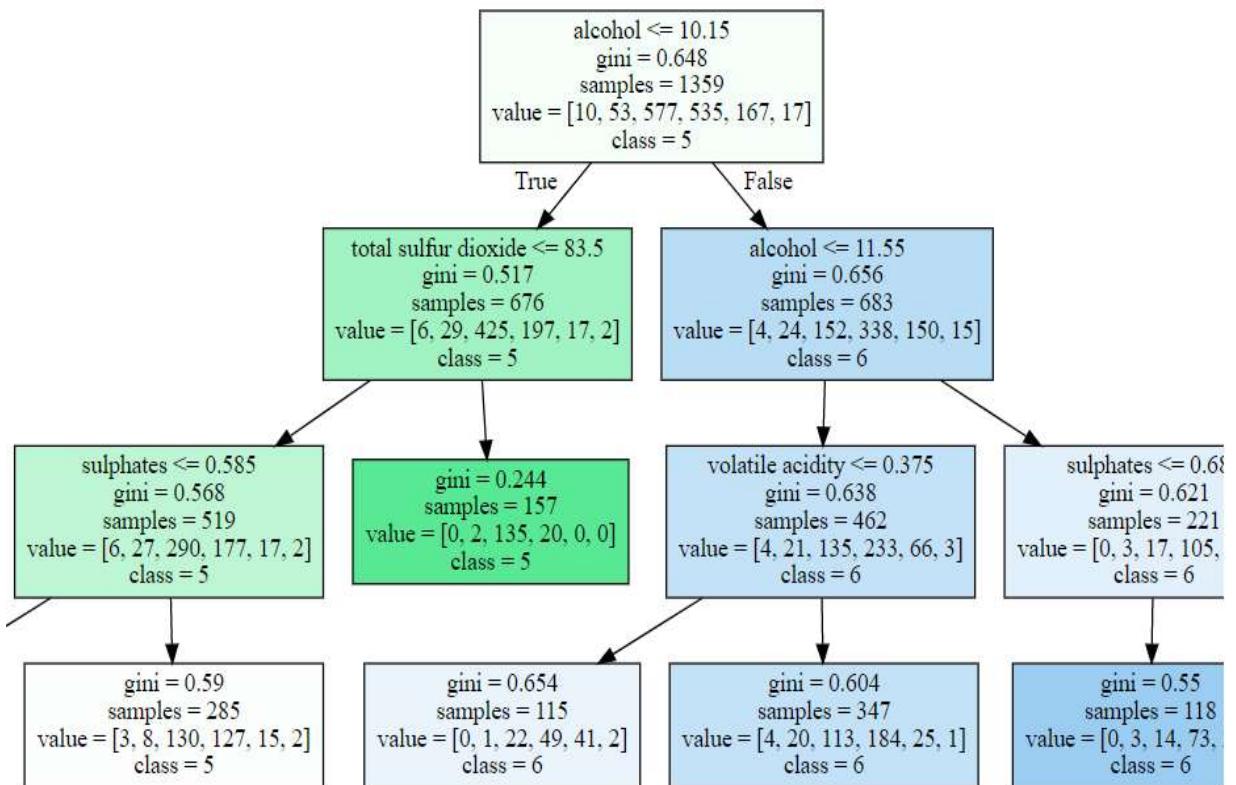
gini = 0.648
samples = 1359
value = [10, 53, 577, 535, 167, 17]
class = 5

```

The parameters given are Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 1000. Since the minimum number of samples in each leaf is given as 1000, if the parent node(depth=0) is divided into leaf nodes, then the samples size would have been lesser than 1000. This violates the parameter min_samples_leaf. Hence, although the max_depth is not 3, the tree is stopped, there is no breakage of internal nodes to leaf nodes and the gini parameter of the only node becomes 0.648. The gini impurity result implies that the only node is not at all homogeneous and the bias is high. Therefore, the results or predictions are going to be hugely assumed. This tree is not all in its purest form.

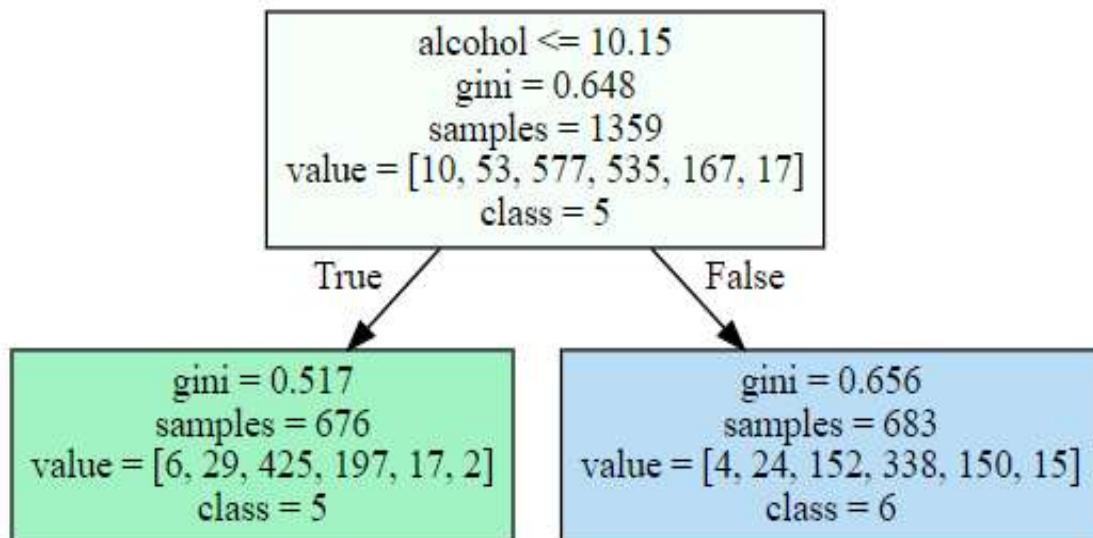


In contrast to the previous diagram, this decision tree has more leaf nodes and has lesser gini impurity values. The parameters given for making this tree are Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 500. The primary reason for getting depth 1 leaf nodes is because of the minimum samples. This time, the minimum samples have been reduced from 1000 to 500. So, the tree can break into two leaf nodes and have a depth of 1. Though the gini impurity has decreased a bit yet it has a high value closer to 1.0 which is not at all pure. Hence, again the bias-variance trade off is not justified to the fullest. This tree has relatively higher bias and lower variance. The depth is 1 instead of 3.

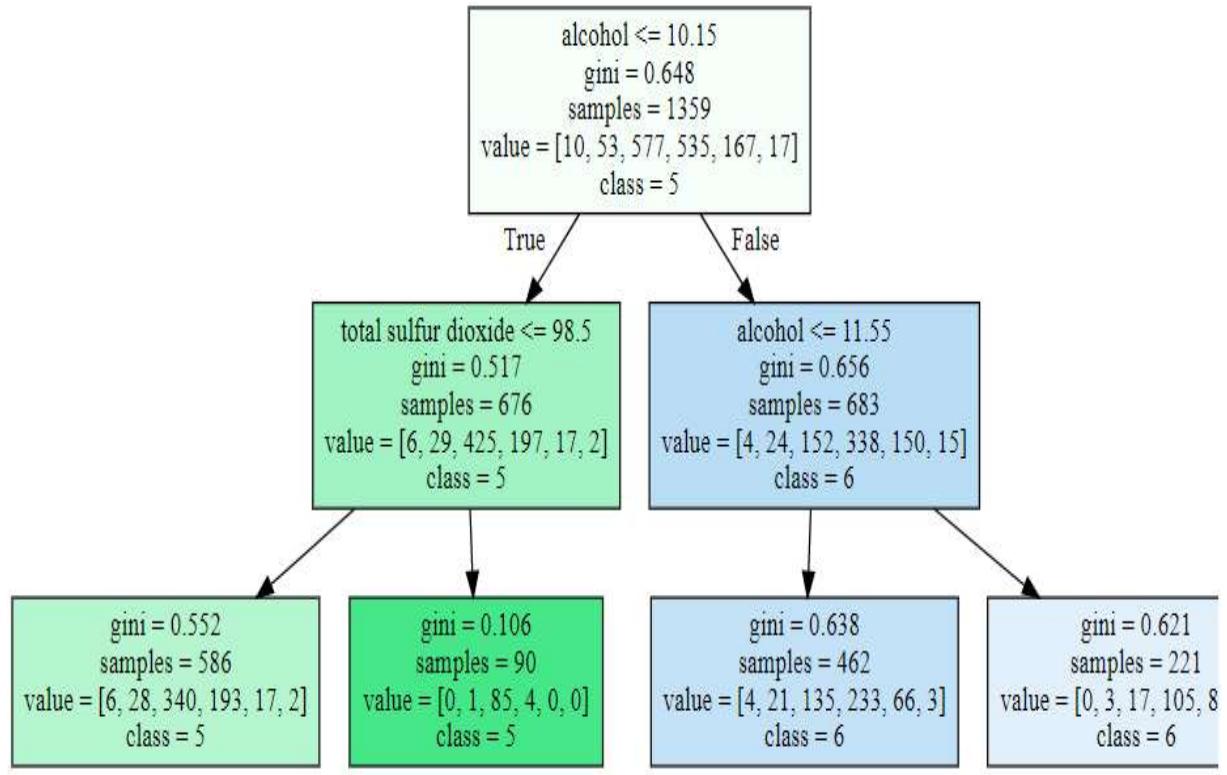


Among all of the three diagrams, this is the best tree in terms of gini impurity and bias-variance importance. The depth of the tree is 3 which is fully shown in the diagram. The tree is broken into internal and leaf nodes on the basis of few features: alcohol, total sulfur dioxide, sulphates, volatile acidity. The leaf node that is formed from "total sulfur dioxide <= 83.5 =False" has the most significant gini impurity value ,0.244. This is closer to 0.0 . This leaf node has a low bias and a high variance. Any sample replacement will have a deep impact on the predicted class due to high variance. The gini values for class=6 is more than 0.5 . This is not pure and therefore, we can assume that any prediction of class_6 will be based on some assumption. But this time, the min_samples_leaf was 100 ,so when the depth of the tree becomes 3, the sampels in each node is still above 100. Thus making the depth ultimately attain a value of 3.

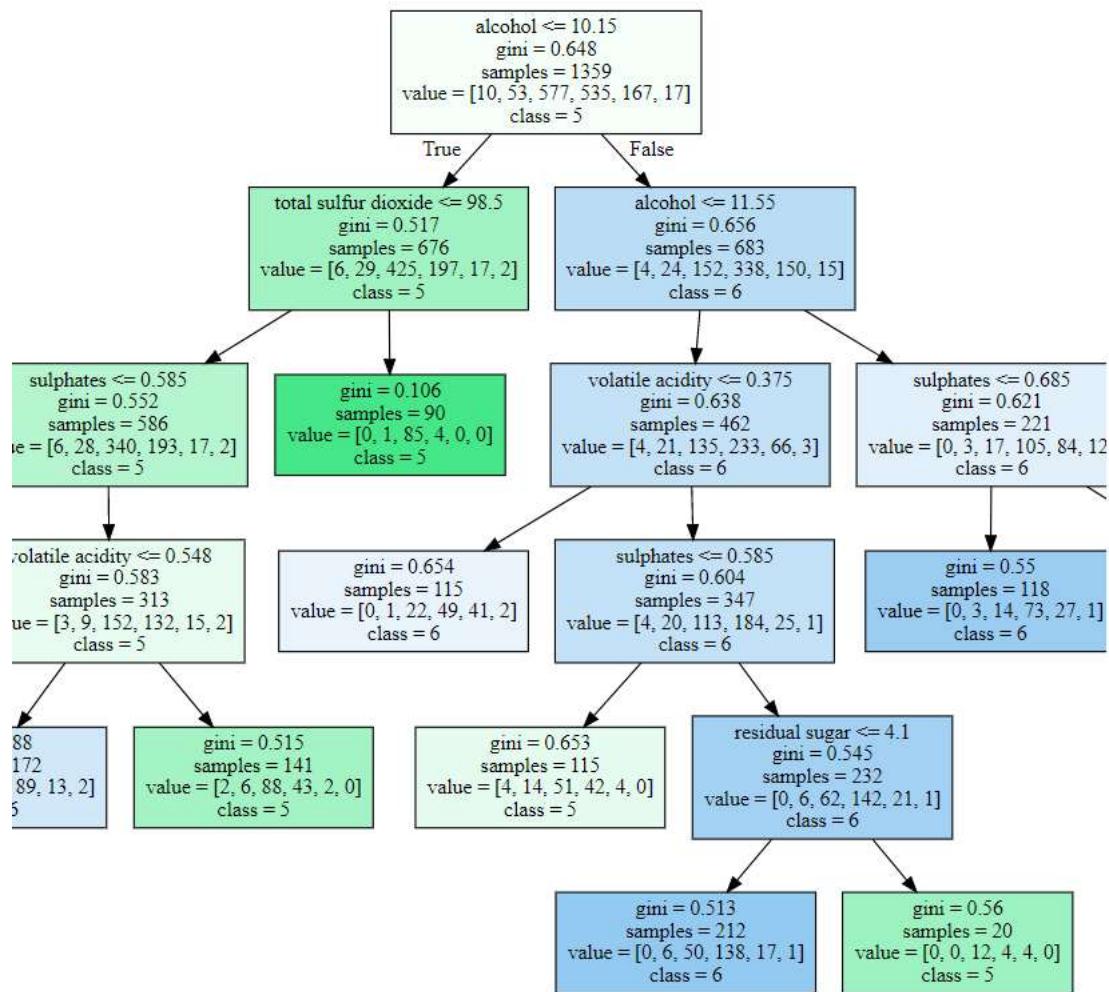
Max_depth



The parameters are : Max_depth is 1 and max_leaf_nodes is 3 while min_samples_leaf is 10. The maximum depth parameter given in this diagram is 1 which is fully shown in the diagram. This picture shows the effectiveness of max_depth parameter. The algorithm produces a tree of 2 leaf nodes . This means that there is another room for 1 more leaf nodes. But as we see that max_depth is 1, the tree is already divided into two branches and therefore has depth of 1. Since the max_depth parameter is exhausted, no more leaf nodes is formed. Although, the main problem with the decision tree is its gini values. They are closer to 1.0 . Therefore, the tree is not at all in its purest form.

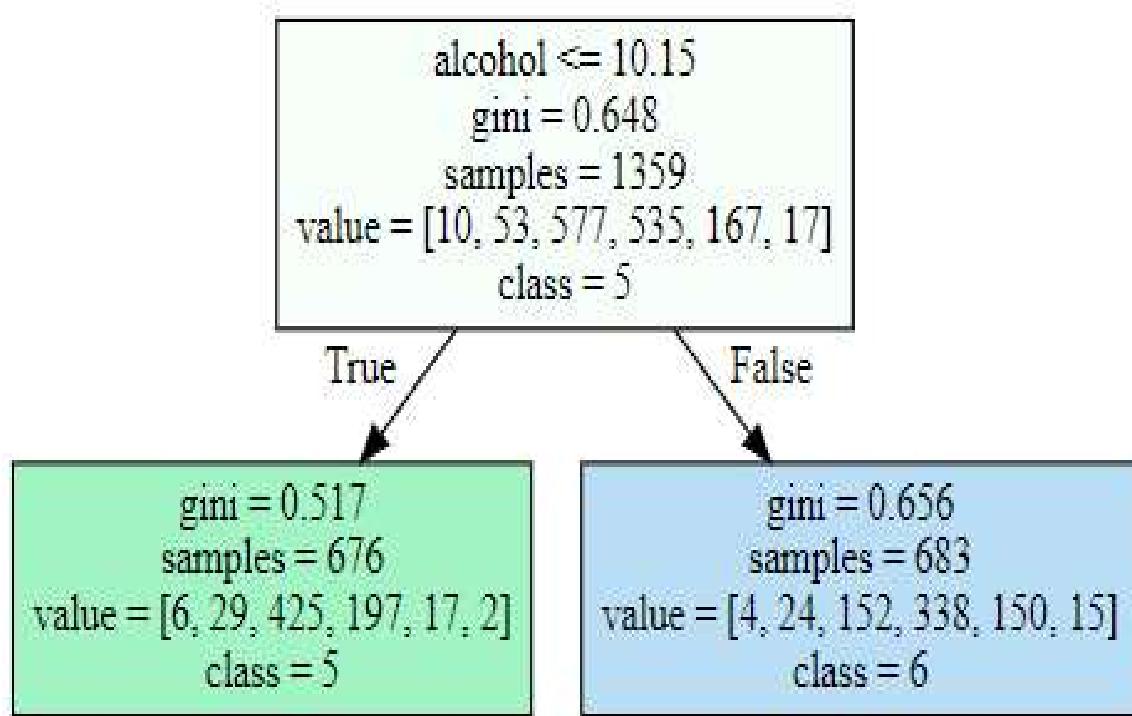


The parameters are : Max_depth is 2 while min_samples_leaf is 10. Like the previous diagram, this tree has also adhered to its depth parameter, that is 2. The attributes used for splitting the nodes are: alcohol, total sulfur dioxide. This time although the minimum samples leaf is way greater than 10, yet the tree does not have more depth than 2 due to max_depth parameter. The gini value is still closer to 0.6 for most of the nodes. There is one node that's relatively pure and has a gini value of 0.106. This is good since this node is almost pure. This means that this node has a lower bias and a higher variance. The sample size is 90 where sample 2 consists of 85.

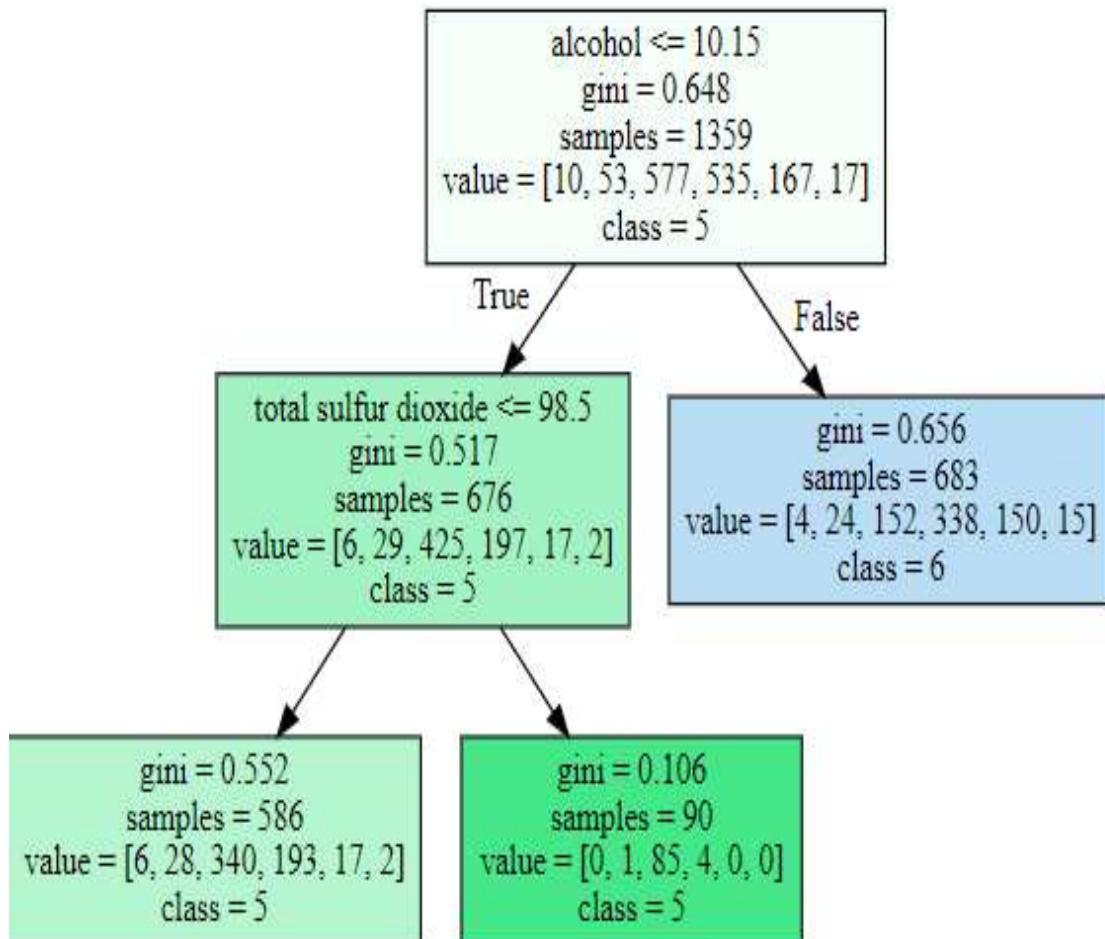


This is most complex tree with parameters :Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10. On reaching the depth of 5, the leaf nodes are exactly 10 and the min_samples_leaf is greater than 10. So, all the points are covered and all the parameters are taken care of. If the max_leaf_nodes would have been 20, then the parameter max_leaf_nodes would not have reached 20. This is because at a depth of 5, max leaf nodes is 10. So in order to produce more leaf nodes , depth has to increase . But the depth parameter is 5 which is already exhausted. Thus, the tree would have been exactly like this one even though the max_leaf_nodes was less than 20. There is no significant improvement in the gini value . All of them are nearer to 0.6. This leads to the fact that these tree nodes are not pure and prediction will be based on assumption which is not reliable.

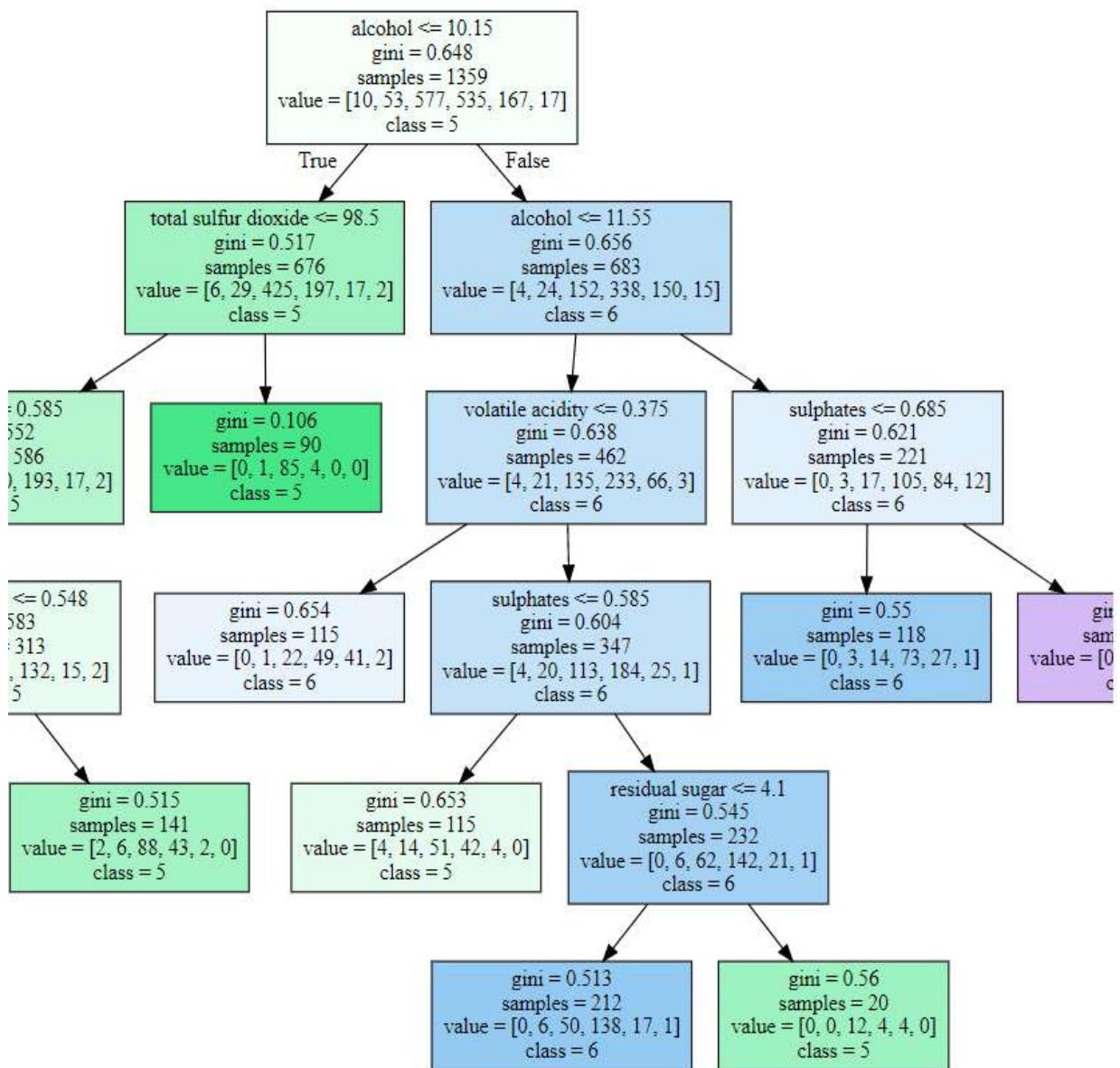
Maximum leaf nodes



The parameters for this diaagram are:Max_depth is 5 and max_leaf_nodes is 2 while min_samples_leaf is 10. Here we see that although the max_depth is 5 and min_samples_leaf is 10, the tree has only 1 depth. This is because of the parameter: max_leaf_nodes. After getting a depth of 1, we see that the tree has already reached 2 leaf nodes. Therefore, if the tree had expanded to depth 2, then the total number of leaf_nodes would have exceeded 2. This would have violated the parameter max_leaf_nodes which says 2. At deoth 1 , the 2 leaf_nodes is attained. Both the gini values are higher. So, we can safely assume that this tree can have more nodes and is not in its pure or homogeneous form. Moreover, the tree has higher bias which makes it vulnerable for assumption of unseen values.



For this diagram, the parameters that have been used are: `Max_depth` is 5 and `max_leaf_nodes` is 3 while `min_samples_leaf` is 10. Here we again see that the tree abided by the parameter: `max_leaf_nodes` which is 3. Although the depth is in its basic stage : 2, the tree is not getting more depth. This is because, if the tree would have got depth:4, then the `leaf_nodes` would have surpassed 3 . This doesn't follow the parameter:`max_leaf_nodes = 3` which is already attained. However, we find out the one leaf node has come closer to its purest form and has a gini value of 0.106. This means that the `class_value` is not based on assumption and has a low bias tradeoff error.



At last in this diagram we see that the parameters:Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10 is fully utilized. The depth is 5 , max leaf nodes are 10 and samples are way above 10. Unfortunately, the gini value doesn't get better. This tree although is deep and has a lot more nodes, the bias is not all good. This is because the gini values are still on a higher range and henceforth the tree is not good for prediction .

Finally the prediction results:

We first found out the f1 and the accuracy score with the target variable as 3,4,5,6,7,8.

The accuracy score is 0.6029411764705882

	precision	recall	f1-score	support
3	0.00	0.00	0.00	2
4	0.00	0.00	0.00	11
5	0.61	0.87	0.72	116
6	0.59	0.44	0.51	107
7	0.57	0.48	0.52	33
8	0.00	0.00	0.00	3
micro avg	0.60	0.60	0.60	272
macro avg	0.30	0.30	0.29	272
weighted avg	0.56	0.60	0.57	272

Shockingly, we had a disastrous f1 score let alone accuracy. Henceforth , we decided to add a column which says that any quality under 5 would be considered as 'bad' or 0 . While anything having a quality of 6 or above will be 'good' or 1. Then we found out the below statistics.

The accuracy score is 0.796875

	precision	recall	f1-score	support
0	0.78	0.79	0.78	149
1	0.82	0.80	0.81	171
micro avg	0.80	0.80	0.80	320
macro avg	0.80	0.80	0.80	320
weighted avg	0.80	0.80	0.80	320

The tree shows far better accuracy and f1_score. The f1_score has jumped 30% after the modification

Neural Networks

We have tried various neural network architectures and trained the models by changing the number of hidden layers, neurons in hidden layers, different activation functions and solvers and finalized the following three architectures that performed well:

Note: We have trained all the neural networks for classification into two classes namely 'good' and 'bad' quality of wine.

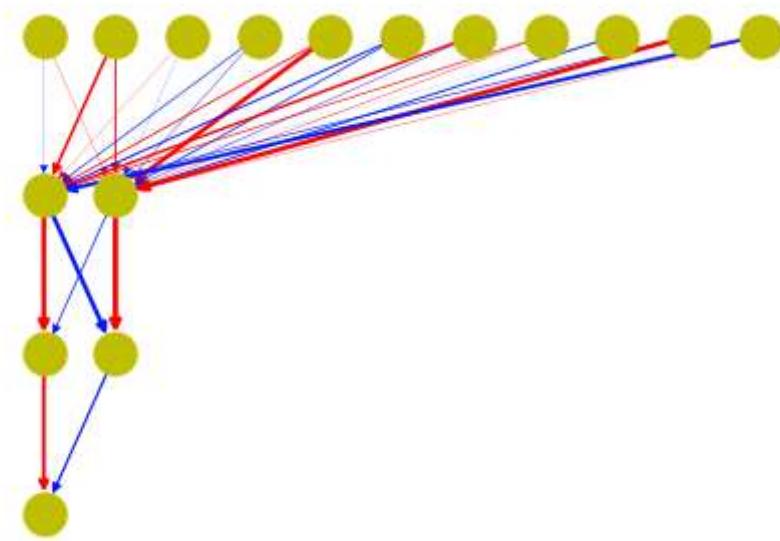
Neural Network 1

Architecture: 11 x 2 x 2 x 1 (2 hidden layers with 2 neurons in each layer)

Solver: Adam

Activation function: ReLU

Maximum iterations: 500



Results:

	precision	recall	f1-score	support
0.0	0.67	0.73	0.70	178
1.0	0.77	0.71	0.74	222
avg / total	0.72	0.72	0.72	400

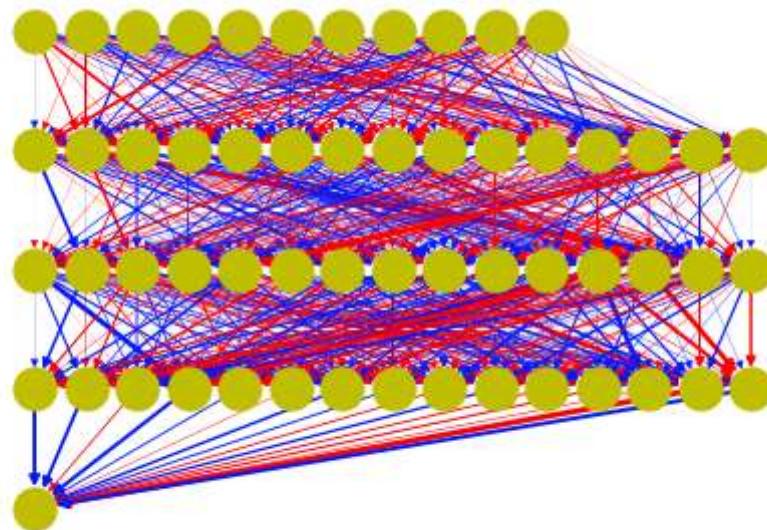
Neural Network 2

Architecture: $11 \times 15 \times 15 \times 15 \times 1$ (3 hidden layers with 15 neurons in each layer)

Solver: Adam

Activation function: ReLU

Maximum iterations: 10000



Results:

	precision	recall	f1-score	support
0.0	0.75	0.75	0.75	178
1.0	0.80	0.80	0.80	222
avg / total	0.78	0.78	0.78	400

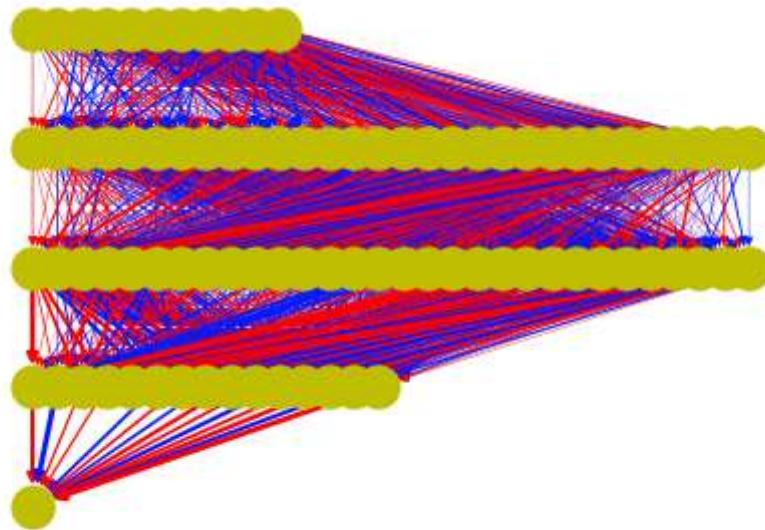
Neural Network 3

Architecture: $11 \times 30 \times 30 \times 15 \times 1$ (3 hidden layers with 30, 30 and 15 neurons in each layer respectively)

Solver: Adam

Activation function: ReLU

Maximum iterations: 10000



Results:

	precision	recall	f1-score	support
0.0	0.78	0.79	0.79	178
1.0	0.83	0.82	0.83	222
avg / total	0.81	0.81	0.81	400

Our best decision tree has F1 score of 0.8 as against the best neural network which is 0.81. So finally for this dataset we have observed that both decision trees and neural networks are similar in case of performance.

Project 8 Code

```
In [1]: 1 import graphviz
```

In [2]:

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn import tree
5 import random
6 from sklearn.tree import export_graphviz
7 from IPython.display import SVG
8 from graphviz import Source
9 import matplotlib.pyplot as plt
10 import os
11 os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin'

```

In [3]:

```

1 #df=pd.read_csv('https://github.com/SupChanda/Wine-Type-Prediction-Using-Dec
2 df=pd.read_csv('data/winequality-red.csv')
3 df.head()

```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4



In [4]:

```
1 df = df.applymap(lambda x : str(x).replace('$',''))
```

In [5]:

```
1 df.head()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.0	2.6	0.098	25.0	67.0	0.9968	3.2	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.997	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.998	3.16	0.58	9.8
4	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4



In [6]:

```
1 df.shape
```

Out[6]:

(1599, 12)

```
In [7]: 1 df.isnull().sum()
```

```
Out[7]: fixed acidity      0
volatile acidity      0
citric acid          0
residual sugar        0
chlorides            0
free sulfur dioxide  0
total sulfur dioxide 0
density              0
pH                   0
sulphates            0
alcohol              0
quality              0
dtype: int64
```

```
In [8]: 1 df.dtypes
```

```
Out[8]: fixed acidity      object
volatile acidity      object
citric acid          object
residual sugar        object
chlorides            object
free sulfur dioxide  object
total sulfur dioxide object
density              object
pH                   object
sulphates            object
alcohol              object
quality              object
dtype: object
```

In [9]: 1 df[df.duplicated(keep=False)]

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar		chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	7.4	0.7	0.0	1.9		0.076	11.0	34.0	0.9978	3.51	
4	7.4	0.7	0.0	1.9		0.076	11.0	34.0	0.9978	3.51	
9	7.5	0.5	0.36	6.1		0.071	17.0	102.0	0.9978	3.35	
11	7.5	0.5	0.36	6.1		0.071	17.0	102.0	0.9978	3.35	
22	7.9	0.43	0.21	1.6		0.106	10.0	37.0	0.9966	3.17	
27	7.9	0.43	0.21	1.6		0.106	10.0	37.0	0.9966	3.17	
39	7.3	0.45	0.36	5.9	0.07400000000000001	12.0	87.0	0.9978	3.33		
40	7.3	0.45	0.36	5.9	0.07400000000000001	12.0	87.0	0.9978	3.33		
64	7.2	0.725	0.05	4.65		0.086	4.0	11.0	0.9962	3.41	
65	7.2	0.725	0.05	4.65		0.086	4.0	11.0	0.9962	3.41	
75	8.8	0.41	0.64	2.2		0.093	9.0	42.0	0.9986	3.54	
76	8.8	0.41	0.64	2.2		0.093	9.0	42.0	0.9986	3.54	
86	8.6	0.49	0.28	1.9		0.11	20.0	136.0	0.9972	2.93	
87	7.7	0.49	0.26	1.9		0.062	9.0	31.0	0.9966	3.39	
91	8.6	0.49	0.28	1.9		0.11	20.0	136.0	0.9972	2.93	
93	7.7	0.49	0.26	1.9		0.062	9.0	31.0	0.9966	3.39	
99	8.1	0.545	0.18	1.9		0.08	13.0	35.0	0.9972	3.3	
102	8.1	0.545	0.18	1.9		0.08	13.0	35.0	0.9972	3.3	
103	8.1	0.575	0.22	2.1		0.077	12.0	65.0	0.9967	3.29	
105	8.1	0.575	0.22	2.1		0.077	12.0	65.0	0.9967	3.29	
110	7.8	0.56	0.19	1.8		0.104	12.0	47.0	0.9964	3.19	
114	7.8	0.56	0.19	1.8		0.104	12.0	47.0	0.9964	3.19	
118	8.8	0.55	0.04	2.2		0.119	14.0	56.0	0.9962	3.21	
121	8.8	0.55	0.04	2.2		0.119	14.0	56.0	0.9962	3.21	
131	5.6	0.5	0.09	2.3		0.049	17.0	99.0	0.9937	3.63	
132	5.6	0.5	0.09	2.3		0.049	17.0	99.0	0.9937	3.63	
135	8.4	0.745	0.11	1.9		0.09	16.0	63.0	0.9965	3.19	
136	8.3	0.715	0.15	1.8	0.08900000000000001	10.0	52.0	0.9968	3.23		
140	8.4	0.745	0.11	1.9		0.09	16.0	63.0	0.9965	3.19	
141	8.3	0.715	0.15	1.8	0.08900000000000001	10.0	52.0	0.9968	3.23		
...	
1468	7.3	0.48	0.32	2.1		0.062	31.0	54.0	0.99728	3.3	
1474	9.9	0.5	0.5	13.8		0.205	48.0	82.0	1.00242	3.16	

	fixed acidity	volatile acidity	citric acid	residual sugar		chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
1476	9.9	0.5	0.5	13.8		0.205	48.0	82.0	1.00242	3.16	
1479	8.2	0.28	0.6	3.0		0.104	10.0	22.0	0.99828	3.39	
1481	8.2	0.28	0.6	3.0		0.104	10.0	22.0	0.99828	3.39	
1488	5.6	0.54	0.04	1.7		0.049	5.0	13.0	0.9942	3.72	
1491	5.6	0.54	0.04	1.7		0.049	5.0	13.0	0.9942	3.72	
1493	7.7	0.54	0.26	1.9	0.089000000000000001	23.0	147.0	0.99636	3.26		
1496	7.7	0.54	0.26	1.9	0.089000000000000001	23.0	147.0	0.99636	3.26		
1497	6.9	0.74	0.03	2.3	0.054000000000000006	7.0	16.0	0.99508	3.45		
1499	6.9	0.74	0.03	2.3	0.054000000000000006	7.0	16.0	0.99508	3.45		
1504	7.5	0.38	0.57	2.3		0.106	5.0	12.0	0.99605	3.36	
1507	7.5	0.38	0.57	2.3		0.106	5.0	12.0	0.99605	3.36	
1516	6.1	0.32	0.25	2.3		0.071	23.0	58.0	0.99633	3.42	
1517	6.5	0.53	0.06	2.0		0.063	29.0	44.0	0.99489	3.38	
1520	6.5	0.53	0.06	2.0		0.063	29.0	44.0	0.99489	3.38	
1522	6.1	0.32	0.25	2.3		0.071	23.0	58.0	0.99633	3.42	
1554	6.6	0.855	0.02	2.4		0.062	15.0	23.0	0.99627	3.54	
1557	6.6	0.855	0.02	2.4		0.062	15.0	23.0	0.99627	3.54	
1559	7.8	0.6	0.26	2.0		0.08	31.0	131.0	0.99622	3.21	
1560	7.8	0.6	0.26	2.0		0.08	31.0	131.0	0.99622	3.21	
1561	7.8	0.6	0.26	2.0		0.08	31.0	131.0	0.99622	3.21	
1562	7.2	0.695	0.13	2.0		0.076	12.0	20.0	0.99546	3.29	
1563	7.2	0.695	0.13	2.0		0.076	12.0	20.0	0.99546	3.29	
1564	7.2	0.695	0.13	2.0		0.076	12.0	20.0	0.99546	3.29	
1567	7.2	0.695	0.13	2.0		0.076	12.0	20.0	0.99546	3.29	
1579	6.2	0.56	0.09	1.7		0.053	24.0	32.0	0.99402	3.54	
1581	6.2	0.56	0.09	1.7		0.053	24.0	32.0	0.99402	3.54	
1592	6.3	0.51	0.13	2.3		0.076	29.0	40.0	0.99574	3.42	
1596	6.3	0.51	0.13	2.3		0.076	29.0	40.0	0.99574	3.42	

460 rows × 12 columns



In [10]: 1 df.drop_duplicates(inplace=True)

In [11]: 1 df.shape

Out[11]: (1359, 12)

In [12]: 1 df.reset_index(drop=True,inplace=True)

In [13]: 1 df.index.values

Out[13]: array([0, 1, 2, ..., 1356, 1357, 1358], dtype=int64)

In [14]: 1 np.set_printoptions()
 2 from sklearn.feature_selection import SelectKBest,SelectPercentile,f_classif
 3 SKBest = SelectKBest(chi2,k=11).fit(df.iloc[:, :-1],df.iloc[:, -1])
 4 p_List =pd.Series(SKBest.pvalues_)
 5 Score_List =pd.Series(SKBest.scores_)
 6 print(SKBest.get_support(indices=True))
 7 Selected_features = pd.Series(df.iloc[:, :-1].columns[SKBest.get_support()])

[0 1 2 3 4 5 6 7 8 9 10]

In [15]: 1 FeatureDTFrame =pd.concat([p_List,Score_List,Selected_features],axis=1)
 2 FeatureDTFrame.columns=['P_Value_List','Score_Value_List','Features']
 3 FeatureDTFrame = FeatureDTFrame.sort_values(by='P_Value_List',ascending=True)

In [16]: 1 FeatureDTFrame

	P_Value_List	Score_Value_List	Features
0	0.000000e+00	2538.395288	total sulfur dioxide
1	3.198315e-33	162.301786	free sulfur dioxide
2	5.494598e-08	42.149269	alcohol
3	1.455892e-02	14.171001	volatile acidity
4	5.493333e-02	10.826370	citric acid
5	9.878418e-02	9.269539	fixed acidity
6	5.312526e-01	4.127193	residual sugar
7	5.712271e-01	3.849872	sulphates
8	9.809745e-01	0.735314	chlorides
9	9.995088e-01	0.156992	pH
10	1.000000e+00	0.000216	density

In [17]: 1 df.quality.value_counts()

Out[17]: 5 577
 6 535
 7 167
 4 53
 8 17
 3 10
 Name: quality, dtype: int64

In [18]:

```

1 print(FeatureDTFrame.Features[0:5])
2 X=df.loc[:,list(FeatureDTFrame.Features[0:5])]
3 Y=df.loc[:,['quality']]
```

```

0    total sulfur dioxide
1    free sulfur dioxide
2            alcohol
3      volatile acidity
4        citric acid
Name: Features, dtype: object
```

In [19]:

```

1 dTree_1= DecisionTreeClassifier(criterion='gini',max_depth=3,max_leaf_nodes=
2 dTree_1.fit(X,Y)
```

Out[19]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3, max_features=None, max_leaf_nodes=10, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=30, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=4, splitter='best')

In [20]:

```

1 feature_importance = pd.DataFrame({'feature':X.columns, 'Gini_importance':dT
2 feature_importance.sort_values(by='Gini_importance',ascending=True,inplace=True)
3 feature_importance
```

Out[20]:

	feature	Gini_importance
1	free sulfur dioxide	0.022182
4	citric acid	0.030431
0	total sulfur dioxide	0.105668
3	volatile acidity	0.176173
2	alcohol	0.665546

In [21]:

```
1 (FeatureDTFrame.Features[0:8])
```

Out[21]:

```

0    total sulfur dioxide
1    free sulfur dioxide
2            alcohol
3      volatile acidity
4        citric acid
5        fixed acidity
6        residual sugar
7        sulphates
Name: Features, dtype: object
```

```
In [22]: 1 print(Y.iloc[[5],:])
2 dTree_1.predict(X.iloc[[5]])
3 #X.iloc[[5,6],:].shape
```

```
    quality
5      5
```

```
Out[22]: array(['5'], dtype=object)
```

MIN_SAMPLES_LEAFS ANALYSIS

```
In [23]: 1 X=df.loc[:,list(FeatureDTFrame.Features[0:10])]
2 Y=df.loc[:,['quality']]
3 dTree_min_samples_nodes_1= DecisionTreeClassifier(criterion='gini',max_depth=3)
4 dTree_min_samples_nodes_1.fit(X,Y)
5 graph = Source(tree.export_graphviz(dTree_min_samples_nodes_1,feature_names=FeatureDTFrame.Features[0:10],class_names=['3','4','5','6','7','8'],filled=True))
6 tempGraph = graph.pipe(format='svg')
7 tempSVG=SVG(tempGraph)
8 print('Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 100')
9 display(tempSVG)
```

```
Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 100
```

```
<IPython.core.display.SVG object>
```

```
In [24]: 1 dTree_min_samples_nodes_2= DecisionTreeClassifier(criterion='gini',max_depth=3)
2 dTree_min_samples_nodes_2.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_min_samples_nodes_2,feature_names=FeatureDTFrame.Features[0:10],class_names=['3','4','5','6','7','8'],filled=True))
4 tempGraph = graph.pipe(format='svg')
5 tempSVG=SVG(tempGraph)
6 print('Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 500')
7 display(tempSVG)
```

```
Max_depth is 3 and max_leaf_nodes is 10 while min_samples_leaf is 500
```

```
<IPython.core.display.SVG object>
```

```
In [25]: 1 dTree_min_samples_nodes_3= DecisionTreeClassifier(criterion='gini',max_depth=3)
2 dTree_min_samples_nodes_3.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_min_samples_nodes_3,feature_names=FeatureDTFrame.Features[0:10],class_names=['3','4','5','6','7','8'],filled=True))
4 tempGraph = graph.pipe(format='svg')
5 tempSVG=SVG(tempGraph)
6 print('Max_depth is 3 while min_samples_leaf is 100')
7 display(tempSVG)
```

```
Max_depth is 3 while min_samples_leaf is 100
```

```
<IPython.core.display.SVG object>
```

MAX DEPTH ANALYSIS

In [26]:

```

1 dTree_Max_Depth_1= DecisionTreeClassifier(criterion='gini',max_depth=1,min_s
2 dTree_Max_Depth_1.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Depth_1,feature_names=list(Fea
4                                         class_names=['3','4','5','6','7','8']),fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 1 and max_leaf_nodes is 3 while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 1 and max_leaf_nodes is 3 while min_samples_leaf is 10

<IPython.core.display.SVG object>

In [27]:

```

1 dTree_Max_Depth_2= DecisionTreeClassifier(criterion='gini',max_depth=2,min_s
2 dTree_Max_Depth_2.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Depth_2,feature_names=list(Fea
4                                         class_names=['3','4','5','6','7','8']),fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 2 and while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 2 and while min_samples_leaf is 10

<IPython.core.display.SVG object>

In [28]:

```

1 dTree_Max_Depth_3= DecisionTreeClassifier(criterion='gini',max_depth=5,max_l
2 dTree_Max_Depth_3.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Depth_3,feature_names=list(Fea
4                                         class_names=['3','4','5','6','7','8']),fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10')
8 display(tempSVG)

```

Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10

<IPython.core.display.SVG object>

MAX_LEAF_NODES_ANALYSIS

In [29]:

```
1 dTree_Max_Leaf_Nodes_1= DecisionTreeClassifier(criterion='gini',max_depth=5,
2 dTree_Max_Leaf_Nodes_1.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Leaf_Nodes_1,feature_names=lis
4                                     class_names=['3','4','5','6','7','8'],fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 2 while min_samples_leaf is 10')
8 display(tempSVG)
```

Max_depth is 5 and max_leaf_nodes is 2 while min_samples_leaf is 10
<IPython.core.display.SVG object>

In [30]:

```
1 dTree_Max_Leaf_Nodes_2= DecisionTreeClassifier(criterion='gini',max_depth=5,
2 dTree_Max_Leaf_Nodes_2.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Leaf_Nodes_2,feature_names=lis
4                                     class_names=['3','4','5','6','7','8'],fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 3 while min_samples_leaf is 10')
8 display(tempSVG)
```

Max_depth is 5 and max_leaf_nodes is 3 while min_samples_leaf is 10
<IPython.core.display.SVG object>

In [31]:

```
1 dTree_Max_Leaf_Nodes_3= DecisionTreeClassifier(criterion='gini',max_depth=5,
2 dTree_Max_Leaf_Nodes_3.fit(X,Y)
3 graph = Source(tree.export_graphviz(dTree_Max_Leaf_Nodes_3,feature_names=lis
4                                     class_names=['3','4','5','6','7','8'],fi
5 tempGraph = graph.pipe(format='svg')
6 tempSVG=SVG(tempGraph)
7 print('Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10')
8 display(tempSVG)
```

Max_depth is 5 and max_leaf_nodes is 10 while min_samples_leaf is 10
<IPython.core.display.SVG object>

In [33]:

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.metrics import classification_report
4 X_train,X_test,y_train,y_test = train_test_split(df.loc[:,list(FeatureDTFrame
5                                         df.loc[:,['quality']]],test_
6 cls= DecisionTreeClassifier(criterion='gini',max_leaf_nodes=10,min_samples_l
7 cls.fit(X_train,y_train)
8 print('The accuracy score is {}'.format(cls.score(X_test,y_test)))
9 print('\n',classification_report(y_test,cls.predict(X_test)))

```

The accuracy score is 0.6029411764705882

	precision	recall	f1-score	support
3	0.00	0.00	0.00	2
4	0.00	0.00	0.00	11
5	0.61	0.87	0.72	116
6	0.59	0.44	0.51	107
7	0.57	0.48	0.52	33
8	0.00	0.00	0.00	3
micro avg	0.60	0.60	0.60	272
macro avg	0.30	0.30	0.29	272
weighted avg	0.56	0.60	0.57	272

C:\Users\supratik chanda\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)

In [42]:

```

1 df=pd.read_csv('data/winequality-red.csv')
2 df.head()

```

Out[42]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

◀ ▶

In [43]:

```

1 df['Target'] = df.quality.apply(lambda x : 0 if x <=5 else 1)

```

In [44]: 1 df.head(1)

Out[44]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [45]: 1 df.drop(columns=['quality'], inplace=True)
2 df.head(1)

Out[45]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [46]: 1 list(FeatureDTFrame.Features[0:5])

Out[46]: ['total sulfur dioxide',
'free sulfur dioxide',
'alcohol',
'volatile acidity',
'citric acid']

In [47]: 1 df.columns

Out[47]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
'pH', 'sulphates', 'alcohol', 'Target'],
dtype='object')

In [48]:

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.metrics import classification_report
4 sc=StandardScaler()
5 X_train,X_test,y_train,y_test = train_test_split(df.loc[:,list(FeatureDTFrame
6 df.loc[:,['Target']],test_s
7 X_train = sc.fit_transform(X_train)
8 X_test = sc.transform(X_test)
9 cls= DecisionTreeClassifier(criterion='gini',random_state=16)
10 cls.fit(X_train,y_train)
11 print('The accuracy score is {}'.format(cls.score(X_test,y_test)))
12 print('\n',classification_report(y_test,cls.predict(X_test)))
```

The accuracy score is 0.796875

	precision	recall	f1-score	support
0	0.78	0.79	0.78	149
1	0.82	0.80	0.81	171
micro avg	0.80	0.80	0.80	320
macro avg	0.80	0.80	0.80	320
weighted avg	0.80	0.80	0.80	320

In [49]:

```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.metrics import classification_report,confusion_matrix
6 import seaborn as sns
7 import matplotlib.pyplot as plt
8 import networkx as nx
9 import colorsys
10 from collections import Counter
```

Function to graphically draw neural network

In [50]:

```

1 def show_ann(mlp):
2     hidden_layers_n = len(mlp.coefs_)-1
3     layers_n = hidden_layers_n + 2
4     input_neurons_n = len(mlp.coefs_[0])
5     hidden_neurons_n = [len(mlp.coefs_[i+1]) for i in range(hidden_layers_n)]
6     output_neurons_n = len(mlp.coefs_[-1][0])
7
8     G = nx.DiGraph()
9     pos = {}
10
11    # Create the neurons of the input layer
12    for i in range(input_neurons_n):
13        pos['Layer0_{}'.format(i)] = (i,layers_n-1)
14
15    for j in range(hidden_layers_n):
16        # Create the neurons of the j'th hidden layer
17        prev_layer = j
18        cur_layer = j+1
19        if (j == 0):
20            prev_size = input_neurons_n
21        else:
22            prev_size = hidden_neurons_n[j-1]
23        for i in range(hidden_neurons_n[j]):
24            pos['Layer{}_{}'.format(cur_layer,i)] = (i,layers_n-1-cur_layer)
25            for k in range(prev_size):
26                w = mlp.coefs_[prev_layer][k][i]
27                G.add_edge('Layer{}_{}'.format(prev_layer,k), 'Layer{}_{}'.format
28
29    # Create the neurons of the output layer
30    prev_layer = hidden_layers_n
31    cur_layer = hidden_layers_n+1
32    for i in range(output_neurons_n):
33        pos['Layer{}_{}'.format(cur_layer,i)] = (i,layers_n-1-cur_layer)
34        for k in range(hidden_neurons_n[-1]):
35            w = mlp.coefs_[prev_layer][k][i]
36            G.add_edge('Layer{}_{}'.format(prev_layer,k), 'Layer{}_{}'.format
37
38    edges = G.edges()
39    colors = [colorsys.hsv_to_rgb(0 if G[u][v]['weight'] < 0 else 0.65,
40                                1, #min(1, abs(G[u][v]['weight'])),
41                                1) for u,v in edges]
42    weights = [abs(G[u][v]['weight'])*2 for u,v in edges]
43
44    nx.draw(G, pos, node_color='y', node_size=450, width=weights, edge_color

```

Loading wine data

In [51]:

```

1 wine_data = pd.read_csv("data/winequality-red.csv")
2 display(wine_data.head())
3 display(len(wine_data))
4
5 wine_data = wine_data.dropna()
6 display(len(wine_data))

```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

◀ ▶

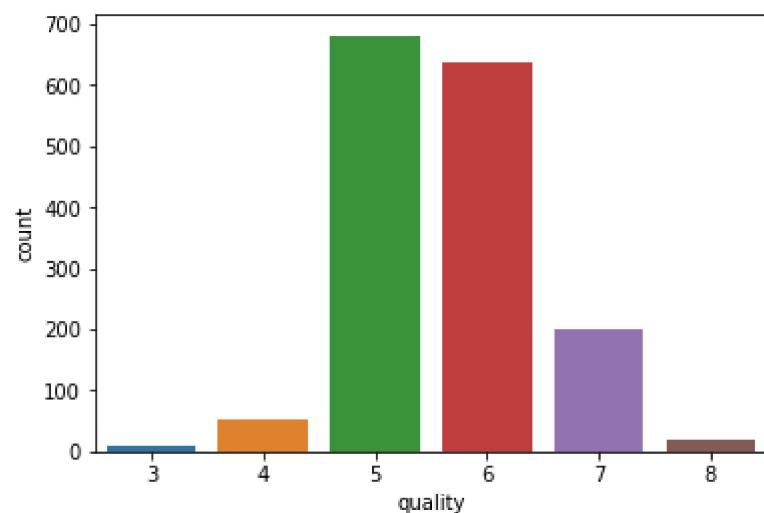
1599
1599

EDA

In [52]:

```
1 sns.countplot(x='quality', data=wine_data)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x20128b66a20>



In [53]:

```
1 # Changing the target variable to 0=> Bad, 1=> Good
2 # So, it is either good quality or bad quality wine
3 # It will also help in balancing the dataset
4 wine_data.loc[wine_data.quality <= 5, 'target'] = 0
5 wine_data.loc[wine_data.quality >= 6, 'target'] = 1
6
7 display(wine_data.head())
8 display(Counter(wine_data['target']))
9
10 wine_data = wine_data.drop(['quality'],axis=1)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

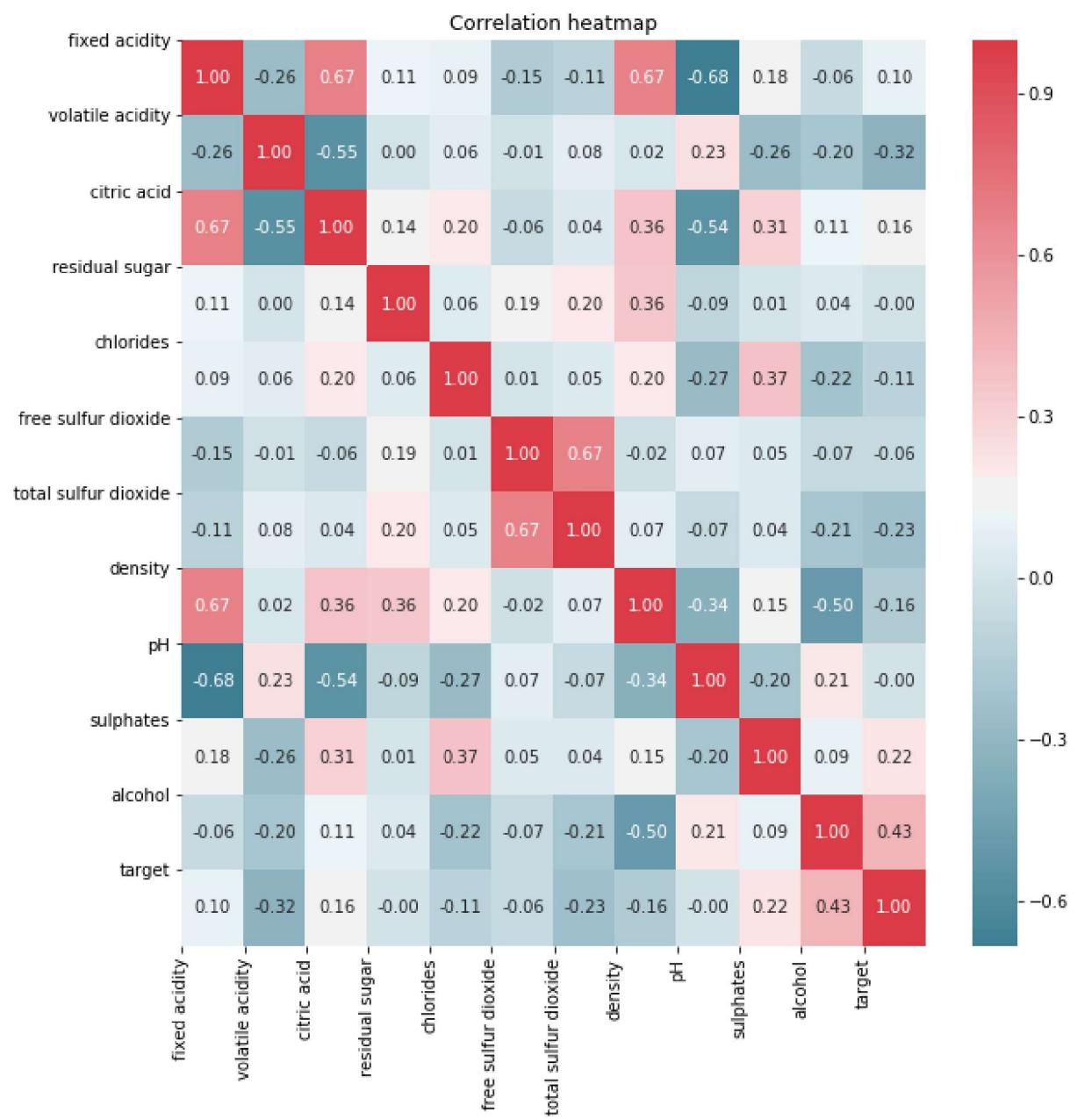
Counter({0.0: 744, 1.0: 855})

In [54]:

```

1 corr_df = wine_data
2 corr = corr_df.corr()
3 #Plot figsize
4 fig, ax = plt.subplots(figsize=(10, 10))
5 #Generate Color Map, red & blue
6 colormap = sns.diverging_palette(220, 10, as_cmap=True)
7 #Generate Heat Map, allow annotations and place floats in map
8 sns.heatmap(corr, cmap=colormap, annot=True, fmt=".2f")
9 #Add title
10 plt.title('Correlation heatmap')
11 #Apply xticks
12 plt.xticks(range(len(corr.columns)), corr.columns);
13 #Apply yticks
14 plt.yticks(range(len(corr.columns)), corr.columns)
15 #show plot
16 plt.show()

```



Train Test split

In [56]:

```

1 X = wine_data.drop(['target'],axis=1)
2 y = wine_data['target']
3 display(X.head())
4 display(y.head())
5
6 X_train, X_test, y_train, y_test = train_test_split(X, y)

```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
0	0.0										
1	0.0										
2	0.0										
3	1.0										
4	0.0										

Name: target, dtype: float64

Scaling the data

In [57]:

```

1 scaler = StandardScaler()
2
3 # Fit only to the training data
4 scaler.fit(X_train)
5
6 # Now apply the transformations to the data:
7 X_train = scaler.transform(X_train)
8 X_test = scaler.transform(X_test)

```

Neural network 1

```
In [58]: 1 mlp1 = MLPClassifier(hidden_layer_sizes=(2, 2),max_iter=500)
2 mlp1.fit(X_train,y_train)
3 predictions = mlp1.predict(X_test)
4 print(confusion_matrix(y_test, predictions))
5 print(classification_report(y_test,predictions))
```

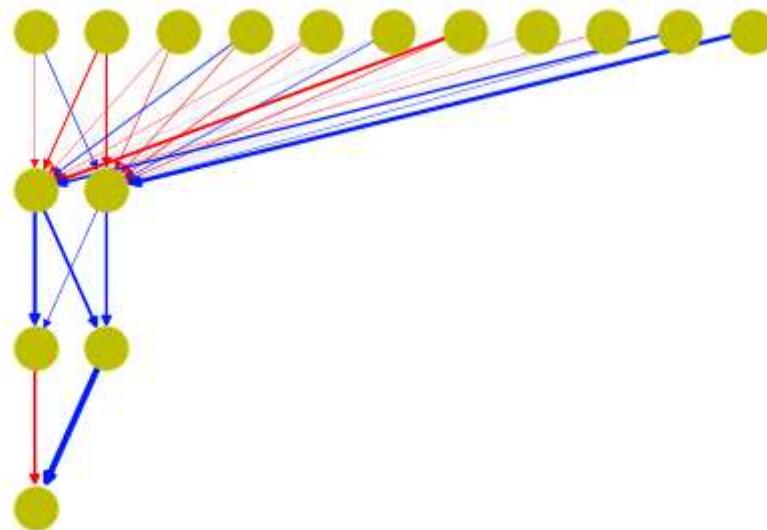
```
[[132  45]
 [ 65 158]]
          precision    recall   f1-score   support
          0.0       0.67     0.75      0.71      177
          1.0       0.78     0.71      0.74      223

   micro avg       0.72     0.72      0.73      400
   macro avg       0.72     0.73      0.72      400
weighted avg       0.73     0.72      0.73      400
```

```
In [59]: 1 print(len(mlp1.coefs_))
2 print(mlp1.coefs_)
3
4 print(len(mlp1.coefs_[0]))
5
6 print(len(mlp1.intercepts_[0]))
```

```
3
[array([[-0.17659428,  0.3193236 ],
       [-0.47447198, -0.53532713],
       [-0.15963346, -0.25496052],
       [ 0.38926941, -0.25627953],
       [-0.17153734, -0.28886129],
       [ 0.05982215,  0.30132202],
       [-0.81804841, -0.26407436],
       [ 0.06486539, -0.06368905],
       [-0.16398425,  0.04056372],
       [ 0.65274032,  0.19876795],
       [ 0.02326972,  1.06865128]]), array([[0.97533394, 0.79216119],
       [0.2589594 , 0.69038475]]), array([[-0.7304783 ],
       [ 1.59675147]])]
11
2
```

```
In [60]: 1 show_ann(mlp1)
```



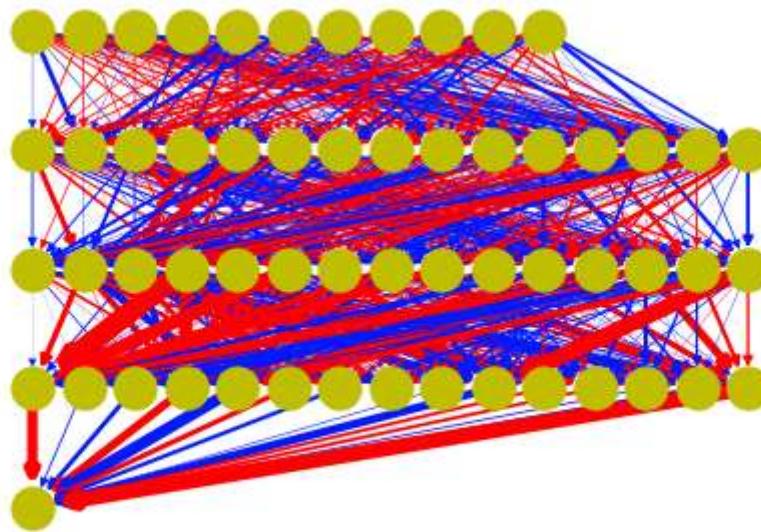
Neural Network 2

```
In [61]: 1 mlp2 = MLPClassifier(hidden_layer_sizes=(15, 15, 15),max_iter=10000)
 2 mlp2.fit(X_train,y_train)
 3 predictions = mlp2.predict(X_test)
 4 print(confusion_matrix(y_test, predictions))
 5 print(classification_report(y_test,predictions))
```

```
[[125  52]
 [ 54 169]]
```

	precision	recall	f1-score	support
0.0	0.70	0.71	0.70	177
1.0	0.76	0.76	0.76	223
micro avg	0.73	0.73	0.73	400
macro avg	0.73	0.73	0.73	400
weighted avg	0.74	0.73	0.74	400

```
In [62]: 1 show_ann(mlp2)
```



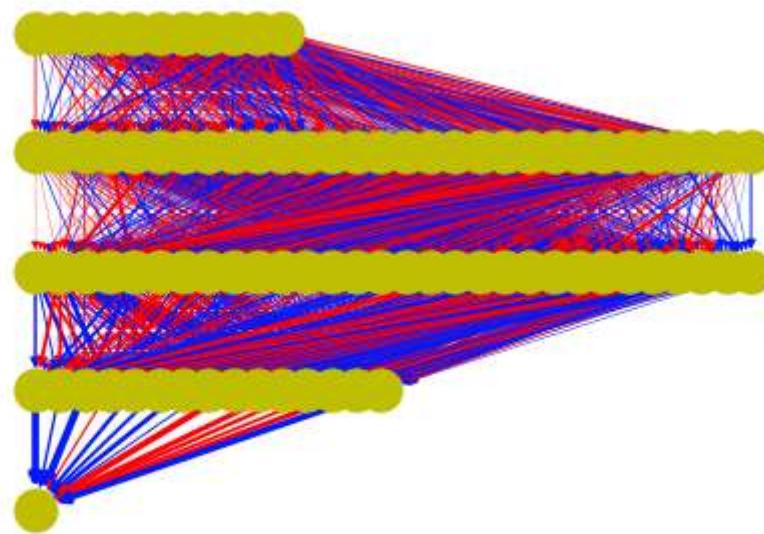
Neural Network 3

```
In [63]: 1 mlp3 = MLPClassifier(hidden_layer_sizes=(30, 30, 15),max_iter=10000)
 2 mlp3.fit(X_train,y_train)
 3 predictions = mlp3.predict(X_test)
 4 print(confusion_matrix(y_test, predictions))
 5 print(classification_report(y_test,predictions))
```

```
[[127 50]
 [ 45 178]]
```

	precision	recall	f1-score	support
0.0	0.74	0.72	0.73	177
1.0	0.78	0.80	0.79	223
micro avg	0.76	0.76	0.76	400
macro avg	0.76	0.76	0.76	400
weighted avg	0.76	0.76	0.76	400

In [64]: 1 show_ann(mlp3)



In []: 1