CS 545 FINAL PROJECT REPORT

PREDICTION OF WINE QUALITY USING RANDOM FOREST AND DECISION TREE

Harie Vashini Dhamodharasamy Kalpana, Devyani Shrivastava

Winter term, 2019

Introduction

In the past few years, with the availability of lot of wine brands, it is difficult to identify the good quality wines. The quality of a wine is generally assessed by sensory and physiochemical methods. The traditional method of quality measurement in which we obtain results from the taste of trained panelist is time-consuming and expensive. In this project we chose to explore Decision tree and Random forest algorithms to predict the wine quality as perceived by a wine drinker on the Wine dataset. The project aims at training the models and experimentally determining the effects of various parameters and comparing the results of both the algorithms.

Method

We obtain the wine dataset from UCI Machine learning repository. There are two datasets, red wine dataset consisting of 1599 instances and white wine dataset consisting of 4898 instances. The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. The goal of this project is to determine white wine quality based on its chemical properties. The dataset contains 11 attributes and the 12^{th} column in the dataset is quality which is regarded as the label or target data. Quality is graded from 0 to 10, where 0 is the worst and 10 is the best quality. Based on the quality scores we classify wines as Good, Bad and Normal ones as follows:

- Scores under 5: Bad
- Scores above 6: Good
- Scores of 5 and 6: Normal

Table 1. The physiochemical data statistics of white wine

Attributes	Minimum	Maximum	Mean	Standard deviation
Fixed acidity	3.800	14.20	6.855	0.844
Volatile acidity	0.080	1.100	0.278	0.101
Citric acid	0.000	1.660	0.334	0.121
Residual sugar	0.600	65.80	6.391	5.072
Chlorides	0.009	0.346	0.046	0.022
Free sulfur dioxide	2.000	289.0	35.31	17.01
Total sulfur dioxide	9.000	440.0	138.4	42.50
Density	0.987	1.039	0.994	0.003
pH	2.720	3.820	3.188	0.151
Sulphates	0.220	1.080	0.490	0.114
Alcohol	8.000	14.20	10.51	1.231

Below Figure shows the importance of each item of physiochemical data in the Wine dataset UCI Repository.

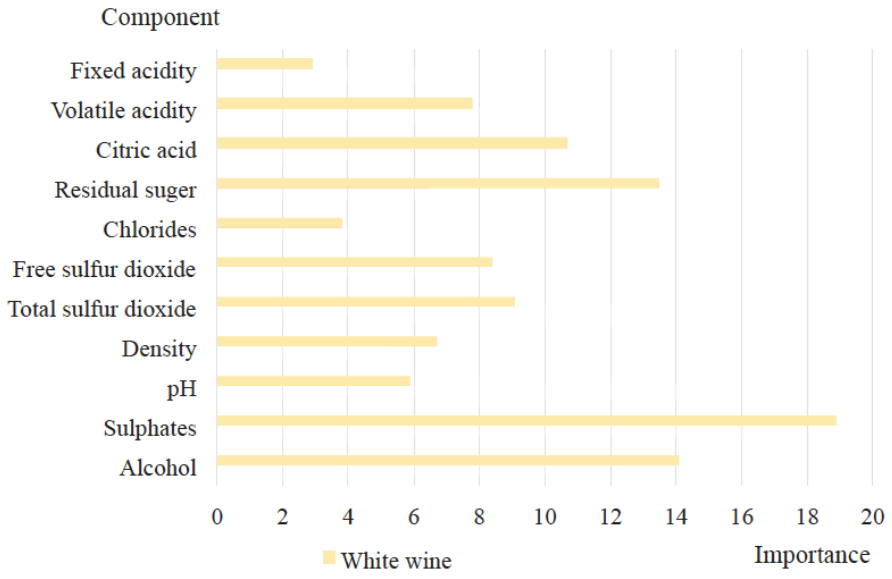

Figure 1. Importance of physiochemical indicators

Packages and libraries used:

1. Sklearn:
    - In python, sklearn is a machine learning package which include a lot of ML algorithms.
    - Here, we are using some of its modules like train_test_split, GridSearchCV, DecisionTreeClassifier, RandomForestClassifier etc
2. NumPy:
    - It is a numeric python module which provides fast math functions for calculations.
    - It is used to read data in numpy arrays and for manipulation purpose.
3. Pandas:
    - Used to read and write different files.
    - Data manipulation can be done easily with dataframes.
4. Matplotlib:
    - Matplotlib is a Python 2D plotting library which generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc.
5. Seaborn:
    - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
6. Export_graphviz:
    - Export a decision tree in DOT format.
    - This function generates a GraphViz representation of the decision tree, which is then written into out_file.

Algorithms Used: We chose to implement our project using below two algorithms:

1) Decision Tree Algorithm
2) Random Forest Algorithm

Decision Tree Algorithm - Decision tree is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Additionally, it can be utilized for both classification and regression kind of problems. A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition based on the attribute value. It partitions the tree in recursive manner.

The basic idea behind decision tree algorithm is as follows:

- Select the best attribute using Attribute Selection Measures (ASM) to split the records.
- Make that attribute a decision node and breaks the dataset into smaller subsets.
- Starts tree building by repeating this process recursively for each child until one of the conditions will match:
    - All the tuples belong to the same attribute value.
    - There are no more remaining attributes.
    - There are no more instances.

Most popular selection measures are Information Gain (Entropy) and Gini Index.

Entropy:

if a random variable x can take N different value ,the i^{th} value x_i with probability $p(x_{ii})$,we can assocoate the folloeing entropy with x:

$$H(x) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i)$$

- Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy the more the information content.

Information Gain:

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of s with A = v and Values(A) is the set of all possible of A, then

$$Gain(S,A) = Entropy(S) - \sum_{v\ :va(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$|S|$ denotes the size of set S

- The entropy typically changes when we use a node in a decision tree to partition the training instances into smaller subsets. Information gain is a measure of this change in entropy.
- Sklearn supports “entropy” criteria for Information Gain.

Gini index:

$$\text{Gini Index} = 1 - \sum_{j} P_j^2$$

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower gini index should be preferred.
- Sklearn supports “gini” criteria for Gini Index and by default, it takes “gini” value

Random Forest Algorithm - This methodology uses a combination of tree predictors. Random forest or Random decision forest are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and

outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forest applies an ensemble algorithm called bagging to the decision trees, which helps reduce variance and overfitting. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Important Hyperparameters

The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster.

Increasing the predictive power:

1. n_estimators - The number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.
2. max_features - The maximum number of features Random Forest considers to split a node.
3. min_sample_leaf - The minimum number of leaves that are required to split an internal node.
4. min_samples_split -The minimum number of samples required to split an internal node.
5. max_depth -The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Increasing the model speed:

1. n_jobs – It tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of "-1" means that there is no limit.
2. random_state - makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.
3. oob_score – It is also called oob sampling, which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples.

Experiments: We began by obtaining the dataset of white wine from UCI machine learning repository. The data is read using pandas library. To have more insight of the data, a graph is plotted for different qualities of wine available.

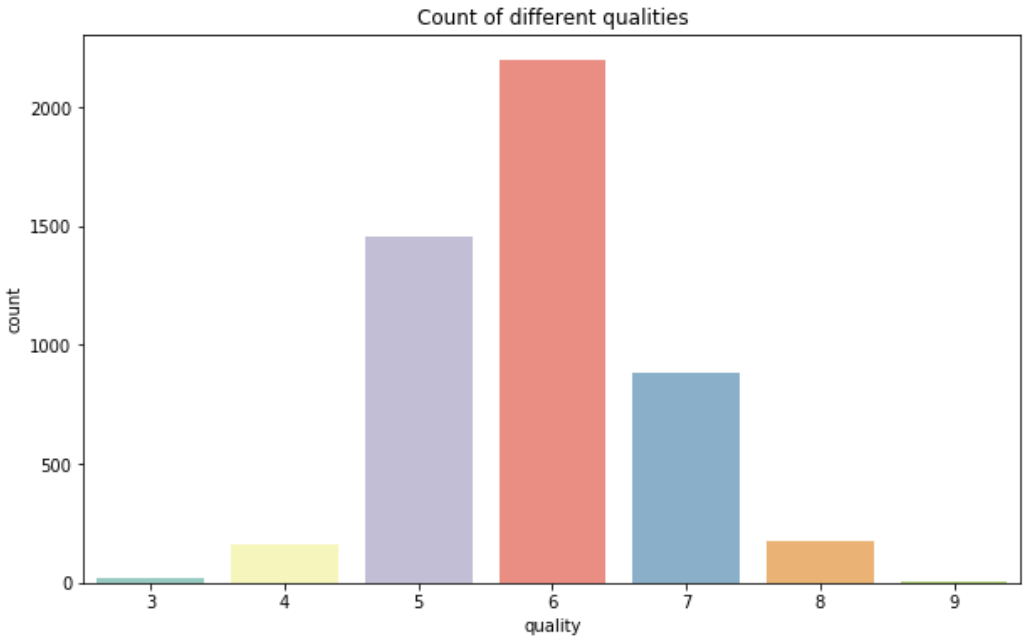

Figure 2. Frequency of different qualities

Thus, as per the graph the quality of wines varies from 3 to 9. However, our hypothesis categorizes the wines into good, bad and normal ones. So, we transformed the data and labelled it as good (quality > 6), bad (quality <5) and Normal (quality = 5 or 6). Now our data can be viewed as below:

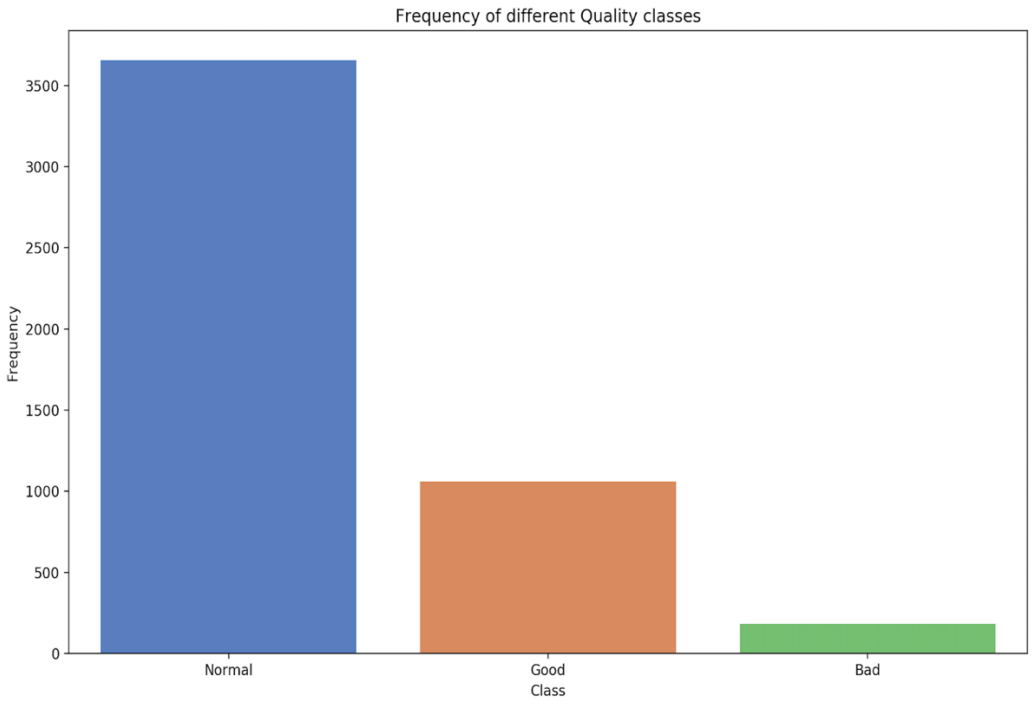

Figure 3. Frequency of the three quality classes

Now we divide our data into features and labels. After dividing, the data is split into test and training data as 30% and 70% respectively.

Classifying the data using decision tree: After dividing the data into test and train, the DecisionTreeClassifier function is called from sklearn and the features and labels of training data is fitted. The various parameters of decision tree classifier such as criteria, min_samples_leaf, min_samples_split, max_depth, are varied in the experiment giving us the different results. The test data is then predicted and confusion matrix, accuracy is drawn accordingly.

Classifying the data using random forest: After dividing the data into test and train, the RandomForestClassifier function is called from sklearn and the features and labels of training data is fitted. The parameter of RandomForestClassifier, Number of trees(n_estimator) is varied in the experiment giving us the different results. The classification results of Random forest are evaluated using 10-fold cross-validation and by giving different values for the hypermeters. The model takes the best parameters and fits the data accordingly. Number of features are also varied to see the results. Confusion matrix, classification report and accuracy are then computed for the predicted data.

Results

Decision tree results:

Table 2 presents the results of a decision tree algorithm using entropy as criteria. Our predictions for white wines have a precision of 55.3%, a recall of 46.0%, an f-measure of 48.0% on average and an accuracy of 76.19 %.

Here 0 represents Bad wines, 1 represents Good wines and 2 represents Normal wines.

Table 2. Classification report of decision tree algorithm using entropy as criteria

Quality	Precision	Recall	F-measure
0	0.29	0.05	0.09
1	0.56	0.45	0.50
2	0.80	0.88	0.84
Average	0.55	0.46	0.48

Confusion Matrix for entropy model:

Actual \ Predicted	bad	good	normal
bad	14	3	39
good	1	196	140
normal	35	131	911

From the confusion matrix, one can infer the ‘Normal wines’ are most accurately classified then the rest. About 140 of the wines which are actually Good, are predicted as Normal wines.

Table 3 presents the results of a decision tree algorithm using gini as criteria. Our predictions for white wines have a precision of 48.6%, a recall of 46.0%, an f-measure of 46.6% on average and an accuracy of 76.66 %.

Table 3. Classification report of decision tree algorithm using gini as criteria

Quality	Precision	Recall	F-measure
0	0.11	0.03	0.05
1	0.54	0.49	0.51
2	0.81	0.88	0.84
Average	0.48	0.46	0.46

```
Confusion Matrix for gini model :

  Predicted  bad  good  normal
Actual
bad         14     5      37
good         2   202     133
normal      40   136     901
```

From the confusion matrix, we see again the ‘Normal wines’ are most accurately classified then the rest. About 136 of the wines which are actually Normal, are wrongly predicted as Good wines. Thus, we can see that varying the criteria and using other parameters as default doesn’t seem to have much effect on accuracy of the model. Now we will vary the other different parameters to see their effects on accuracy.

Max_depth:

The first parameter to vary is max_depth. This indicates how deep the tree can be. The deeper the tree, the more splits it has and it captures more information about the data. Here, we fit a decision tree with depths 1 and 32.

Clf= DecisionTreeClassifier(max_depth=1)

The accuracy of the model is 0.7326530612244898

Clf= DecisionTreeClassifier(max_depth=32)

The accuracy of the model is 0.7693877551020408

Min_samples_split:

It represents the minimum number of samples required to split an internal node. This can vary between considering at least one sample at each node to considering all of the samples at each node. When we increase this parameter, the tree becomes more constrained as it has to consider more samples at each node. Here we will vary the parameter as 10% and 100% of the samples

clf=DecisionTreeClassifier(min_samples_split= 0.1)

The accuracy of the model is = 0.7639455782312925

clf=DecisionTreeClassifier(min_samples_split= 10)

The accuracy of the model is = 0.7448979591836735

Min_samples_leaf:

The minimum number of samples required to be at a leaf node. Here we have varied its value as 0.1 and 5.

Clf= DecisionTreeClassifier(min_samples_leaf=0.1)

The accuracy of the model is = 0.746938775510204

Clf= DecisionTreeClassifier(min_samples_leaf=5)

The accuracy of the model is = 0.7517006802721088

Figure 4 and Figure 5 shows the visualization of trees using entropy and gini index.

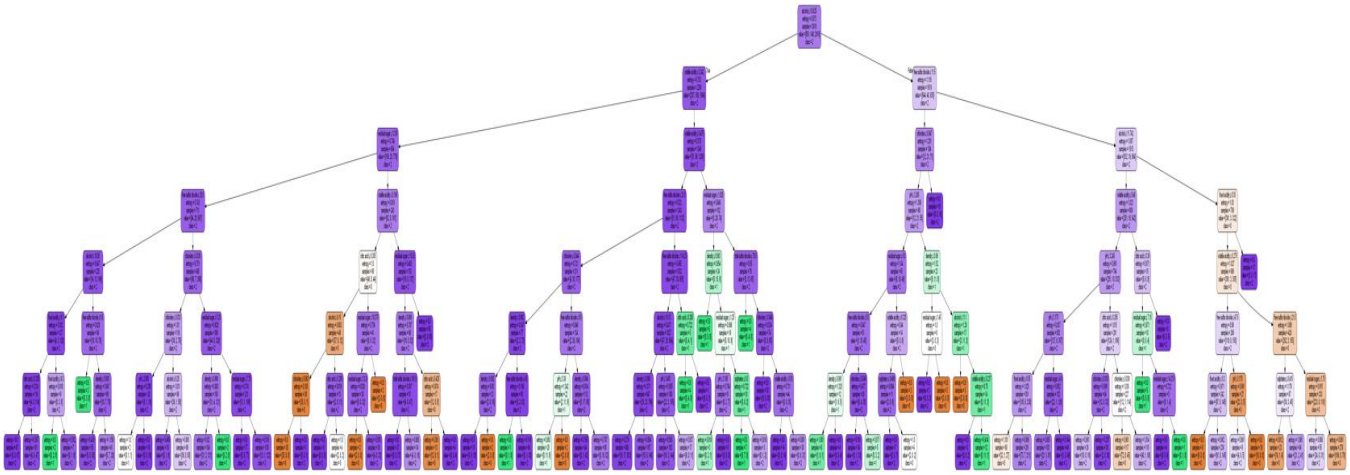

Figure 4. Tree with entropy as criteria

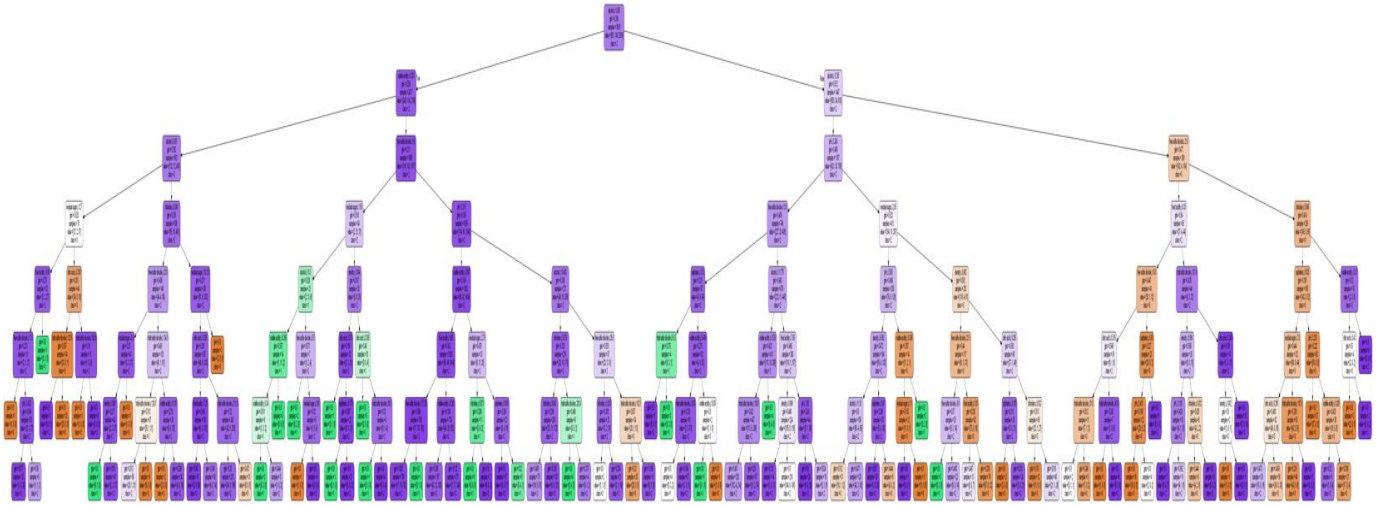

Figure 5. Tree with gini as criteria

(**For better visualization the images of trees are attached separately in the folder alongwith this pdf.)

Random forest results:

Table 4 presents the results of a random forest algorithm with default hyperparameters using 40 trees' prediction of taste preferences for wines. Our predictions for white wines have a precision of 87.6%, a recall of 51.0%, an f-measure of 53.3% on average and an accuracy of 83.87 %.

Here 0 represents Bad wines, 1 represents Good wines and 2 represents Normal wines.

Table 4. Classification report of predicted taste preferences using random forest

Quality	Precision	Recall	F-measure
0	1.00	0.04	0.07
1	0.78	0.53	0.63
2	0.85	0.96	0.90
Average	0.88	0.51	0.53

```
confusion matrix
Predicted  bad  good  normal
Actual
bad          1     0      27
good         0   113     100
normal       0    31     708
```

From the confusion matrix, we see the 'Normal wines' are most accurately classified then the rest. About 100 of the wines which are actually Good, are wrongly predicted as normal wines.

Default hyperparameters - {'warm_start': False, 'oob_score': False, 'n_jobs': None, 'min_impurity_decrease': 0.0, 'verbose': 0, 'max_leaf_nodes': None, 'bootstrap': True, 'min_samples_leaf': 1, 'n_estimators': 40, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'criterion': 'gini', 'random_state': None, 'min_impurity_split': None, 'max_features': 'auto', 'max_depth': None, 'class_weight': None}

Table 5 presents the results of a random forest algorithm with optimization using 40 trees' prediction of taste preferences for wines. A ten-fold grid search cross validation of wine prediction with best hyperparameters are presented. Our predictions for white wines have a precision of 51.0%, a recall of 42%, an f-measure of 43.3% on average and an accuracy of 79.49 %.

Table 5. Classification report of taste preferences using random forest with optimization

Quality	Precision	Recall	F-measure
0	0.00	0.00	0.00
1	0.73	0.30	0.42
2	0.80	0.97	0.88
Average	0.51	0.42	0.43

Confusion matrix

Predicted	bad	good	normal
Actual			
bad	0	0	0
good	0	63	23
normal	28	150	716

Again, normal wines show high prediction rate.

The hyperparameters given for cross validation are: {'max_depth':[2,3,4], 'bootstrap':[True,False], 'max_features':['auto','sqrt','log2',None], 'criterion':['gini','entropy']}

Best parameters using grid search cross validation: {'max_features': None, 'bootstrap': True, 'criterion': 'gini', 'max_depth': 4}

Below Figure shows the normalized weights of the top five most predictive features.

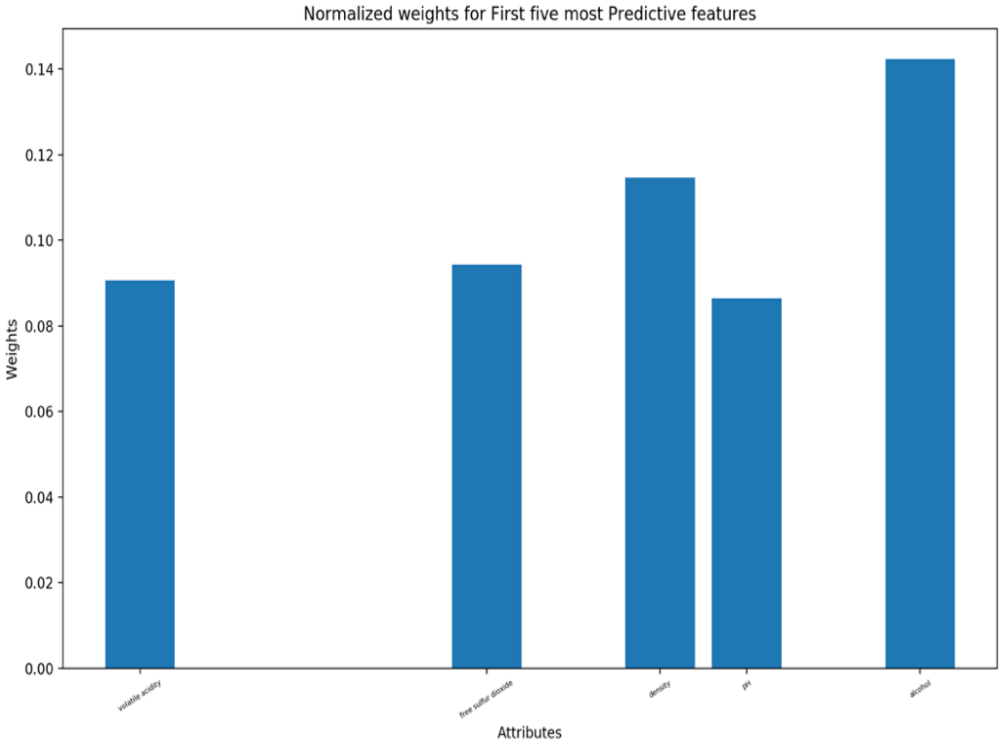

Figure 6. Top five features and their weights

Table 6 presents the results of a random forest algorithm using the top five most predictive features and 100 trees. Our predictions for white wines have a precision of 32.0%, a recall of 33.0%, an f-measure of 29.3% on average and an accuracy of 74.59 %.

Table 6. Classification report of predicted taste preferences using top five features

Quality	Precision	Recall	F-measure
0	0.00	0.00	0.00
1	0.21	0.01	0.03
2	0.75	0.99	0.85
Average	0.32	0.33	0.29

Confusion matrix

Predicted	bad	good	normal
Actual			
bad	0	0	0
good	0	3	11
normal	28	210	728

This infers even with the top important features, Number of features required to train and test matters. It should not be less, it might cause overfitting.

Varying the n_estimators:

The number of trees is varied from 30 to 500. Below figure shows the variation in accuracy for different number of trees.

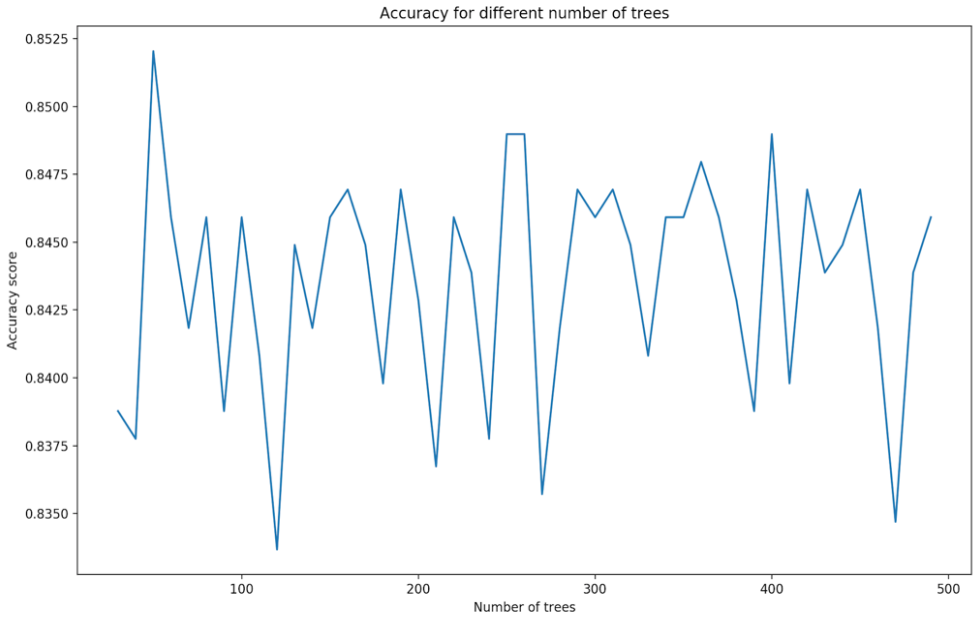

Figure 7. Variation in accuracy for different number of trees

This shows the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably. Therefore, a parameter search has to be done.

Discussions

Comparing both decision tree and random forest, Decision tree shows an accuracy of 76.66 % with gini criteria and with other default hyperparameter. Random forest shows an accuracy of 83.87 % with default hyperparameters (criteria- gini). From this, we can conclude random forest has higher accuracy. This is because of the advantage of random forest that it's default hyperparameters often produce a good prediction result and it is an ensemble of decision trees. In random forest, since by averaging several trees, there is a significantly lower risk of overfitting. Whereas, in decision tree overfitting might occur when one creates over-complex trees that do not generalize the data well. Decision trees can be unstable because small variations in the data might result in a completely different tree being generated but Random forests make a wrong prediction only when more than half of the base classifiers are wrong.

Conclusions and Future work

We have proposed a new way of predicting taste preferences for wines using a decision tree and random forest. And evaluated it using the Wine Quality Data Set from the UCI Machine Learning Repository. Results suggest that these models could offer untutored consumers a better chance of selecting a high-quality wine. For future work, we want to develop a more optimized model of random forest to predict the wine quality and research some other optimization algorithm.