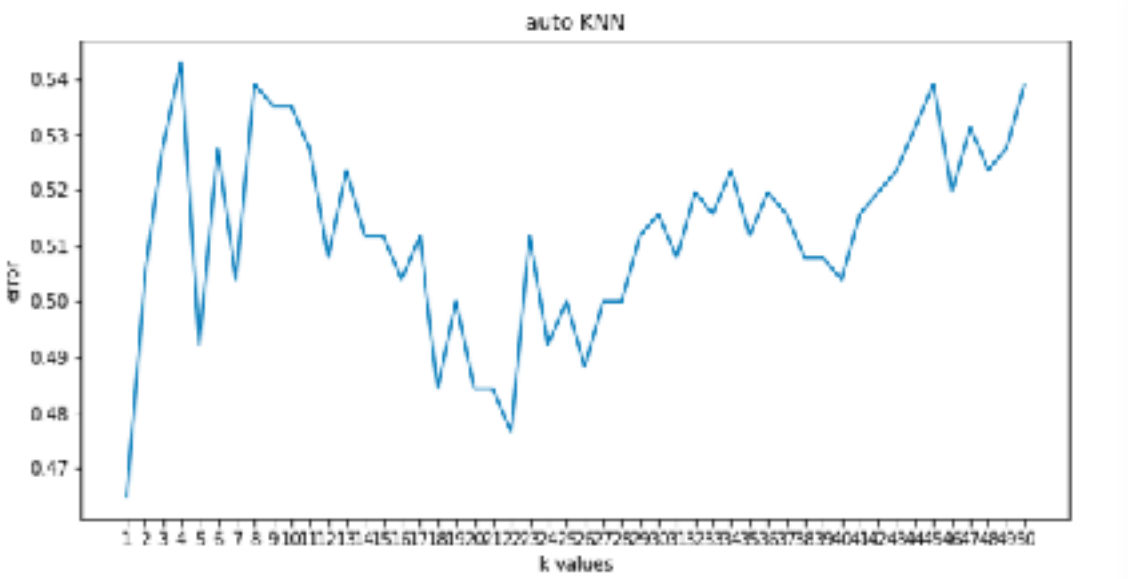
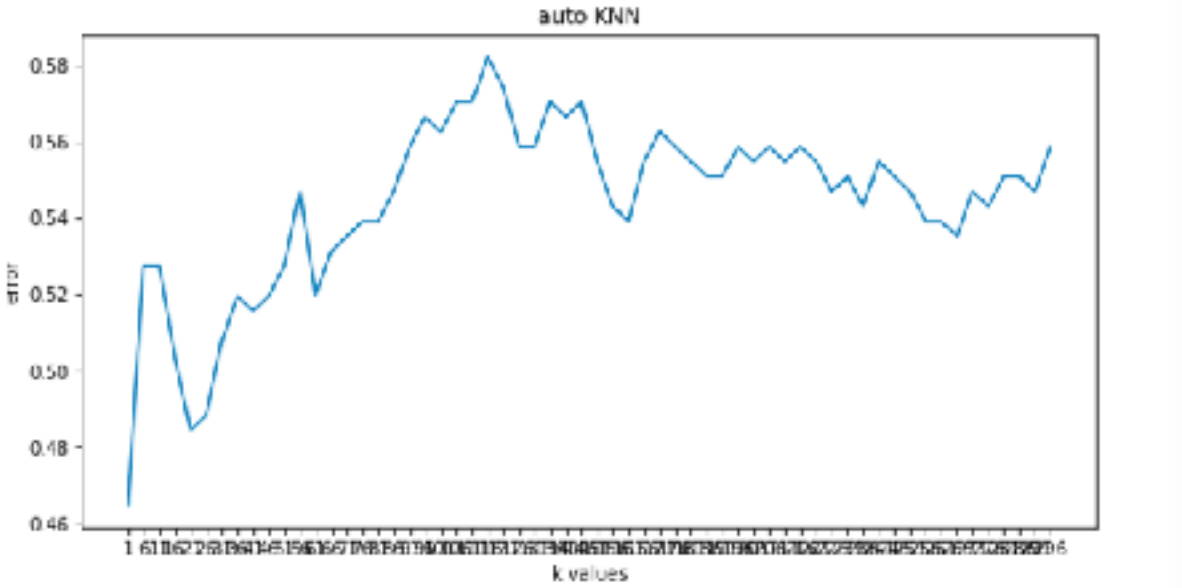
Report for three basic models:
Jialin Cui
First, k-nearest neighbors algorithm on red wine quality problem.
By changing the hyper-parameter n_neighbors, which is the number of neighbors to use by the k-neighbors queries. I got the k-value vs error plot shown as below:

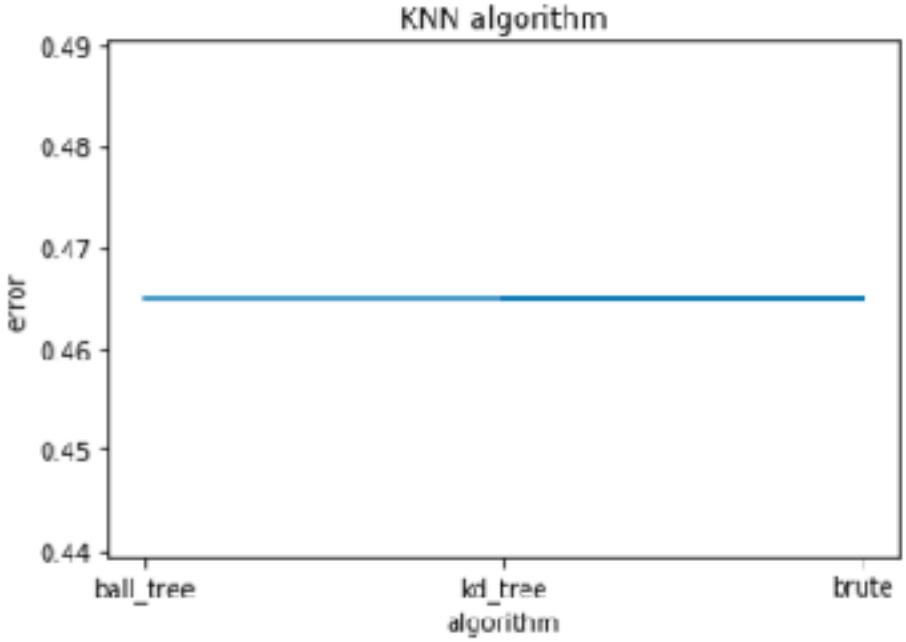


It is easy to see that the error grows as k-value grows. Here k-value grows from 1 to 51, to ensure that error won't drop as k grow even larger, the graph below is k-value grows from 1 to 300 with step size 5:



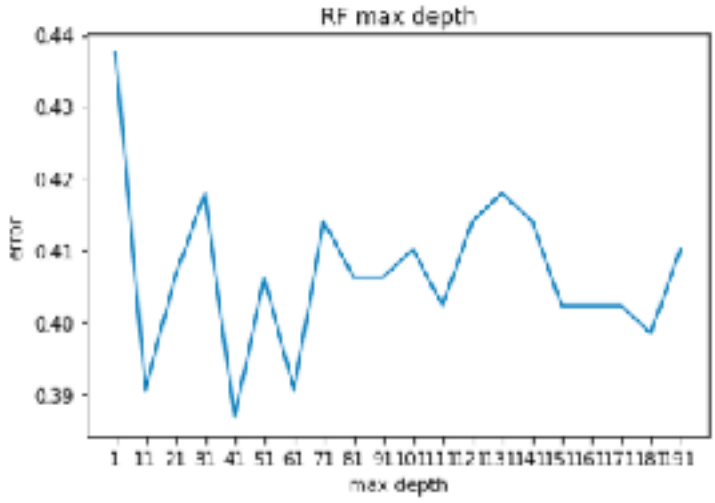
As shown in the graph, we can conclude that the global minimum is k-value = 1.

For the algorithm used by KNN, there is really no difference between ball-tree, kd-tree, and brute as shown in the graph:



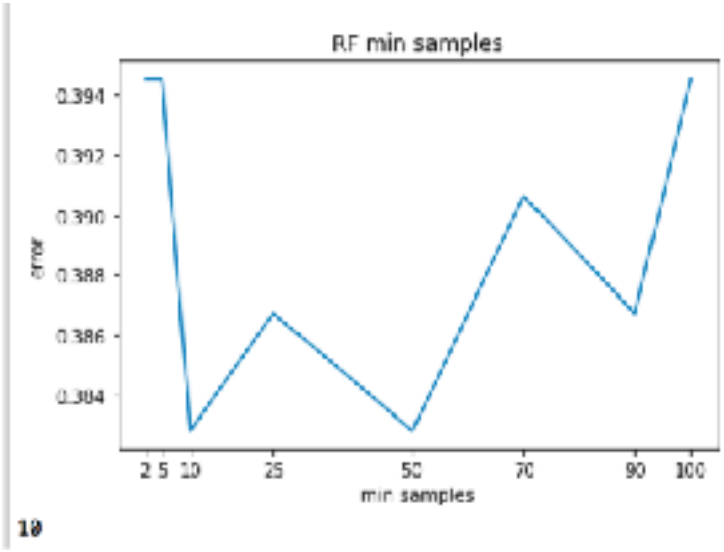
I tried different k-value combined with different algorithm, the error stay the same.
By picking k-value = 1 and auto pick algorithm, the best model only give a 54.68% accuracy, but there is a improvement compare to the default model, which only gives 51.875% accuracy.I assume that K-nearest neighbor classier is not that appropriate for the red wine classification problem.

Second, Decision tree classier on red wine quality problem.
By changing the hyper-parameter max_depth, which is the maximum depth of the tree, the effect is shown as below:



As you can see, the max depth varies a lot, in the graph, max depth goes from 1 to 200 with step size 10, and the best depth in this case is 41.

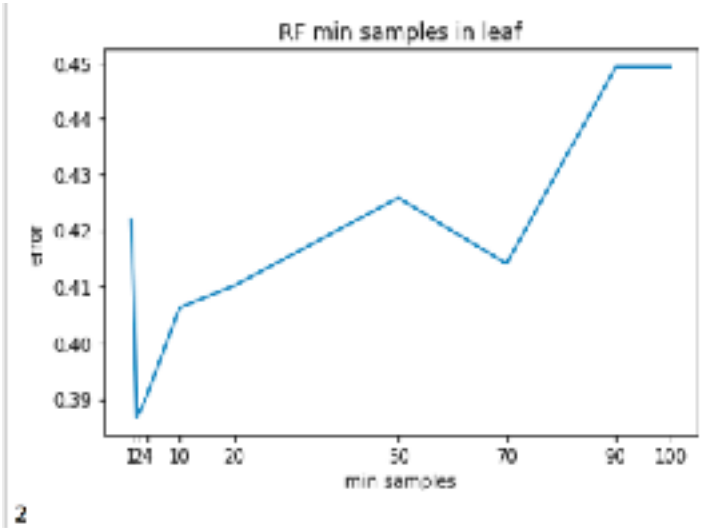
By changing the hyper-parameter min_samples_split, which is the minimum number of samples required to split an internal node, the effect is shown as below:




```
10
```

I tried some typical minimum number here, 2,5,10,25,50,70,90,100, since the number goes too high is useless, the best number is 10. Increase the number at beginning really helps to drop the error, but when the number become large, the effect is not the obvious.

By changing the hyper-parameter min_samples_leaf, which is the minimum number of samples required to be at a leaf node, the effect is shown as below:




```
2
```

The min_samples_leaf pick from 1,2,4,10,20,50,70,90,100, is show that the minimum error value is reached pretty early, which is min_samples_leaf = 2. When the number become larger, the error seems go high and do not drop any more.

I build three models with respect to best_depth, best min_sample_split, and best min_sample_leaf. the original default model gives me accuracy 57.5%

With best min_sample_leaf = 2, I got 58.44% accuracy
With best min_sample_split = 10, I got 58.125% accuracy
With best depth = 41, I got 57.5% accuracy

```
0.575
0.584375
0.58125
0.575
```

These results shows that, we do get some performance improvement from changing sample split size, sample leaf size. For depth, it shows a improvement during validation, but no improvement during test. In my other trails I got the same problem, sometimes some hyper-parameters can improve the validation performance, but not necessary the test performance. However, at least 2 out of 3 the modified model will show a better performance in test.

Another thing that worth noting is that I get the feature importance of this model as shown below:

```
fixed acidity 0.0825265177344
volatile acidity 0.109803005197
citric acid 0.0995746436533
residual sugar 0.0507289561843
chlorides 0.0985816017378
free sulfur dioxide 0.0550985290622
total sulfur dioxide 0.104740151884
density 0.0867586164844
pH 0.0445837566314
sulphates 0.0971725454056
alcohol 0.170431676026
```

And then I drop every single feature, rerun the model error with the same set of training data and validation data got the below result:

As you can see, drop some of the feature actually gives you a better accuracy, the results are shown below:

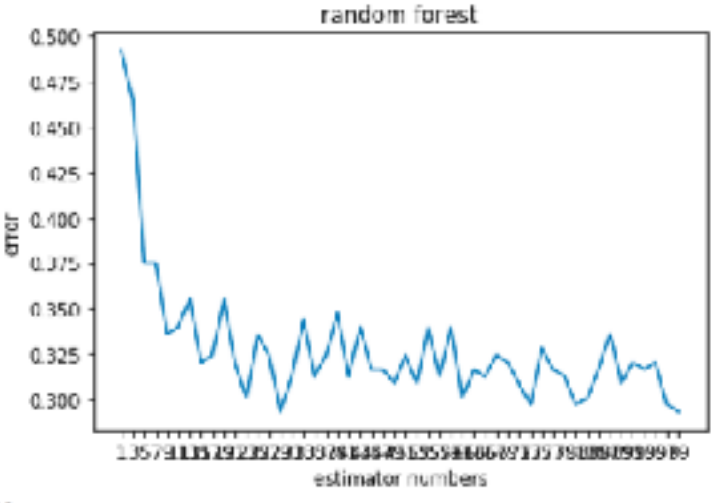
```
alcohol
Accuracy Score: 0.555

chlorides
Accuracy Score: 0.617

citric acid
Accuracy Score: 0.598

density
Accuracy Score: 0.605

fixed acidity
Accuracy Score: 0.590

free sulfur dioxide
Accuracy Score: 0.582

pH
Accuracy Score: 0.586

residual sugar
Accuracy Score: 0.602

sulphates
Accuracy Score: 0.570

total sulfur dioxide
Accuracy Score: 0.539

volatile acidity
Accuracy Score: 0.621
```

Every feature is listed corresponds to the accuracy after it is dropped. The original accuracy is 60.55%

```
0.60546875
```

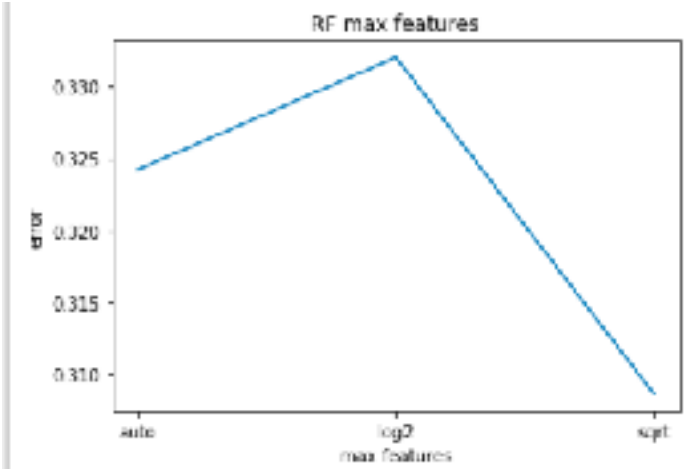
Clearly, after dropping some of the features, we achieve a better accuracy.

Third, random forest classier on red wine quality problem.
By changing the hyper-parameter n_estimators, which is the number of trees in the forest, the effect is shown as below:



As the number of estimators increase, the prediction error decrease in general, but the best estimator number does not guaranteed to be the largest one. I tried estimator number goes from 1 through 100. Here, 29 estimators give the best result.

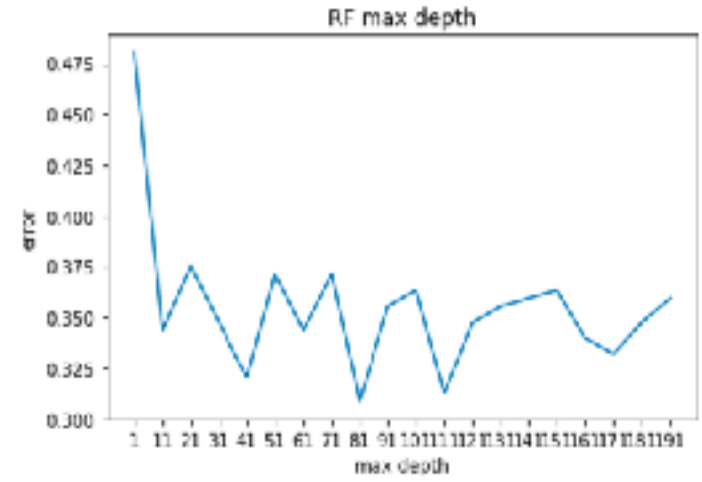
By changing the hyper-parameter max_features, which is the number of features to consider when looking for the best split, the effect is shown as below:




```
sqrt
```

The best number of features to be considered varies between data set to data set, here the square root of the total number of features give the best result.

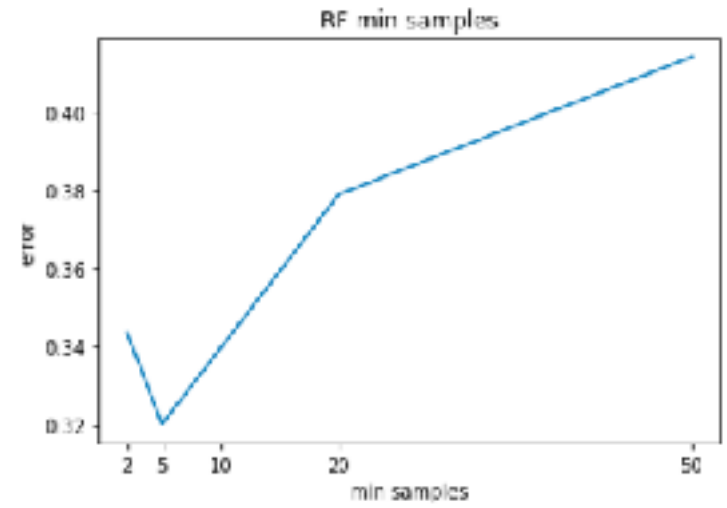
By changing the hyper-parameter max_depth, which is the maximum depth of the tree, the effect is shown as below:




```
81
```

In general, the model error decrease as the max depth increase, but not largest is not necessary the best. Here, max depth equals to 81 gives the best result.

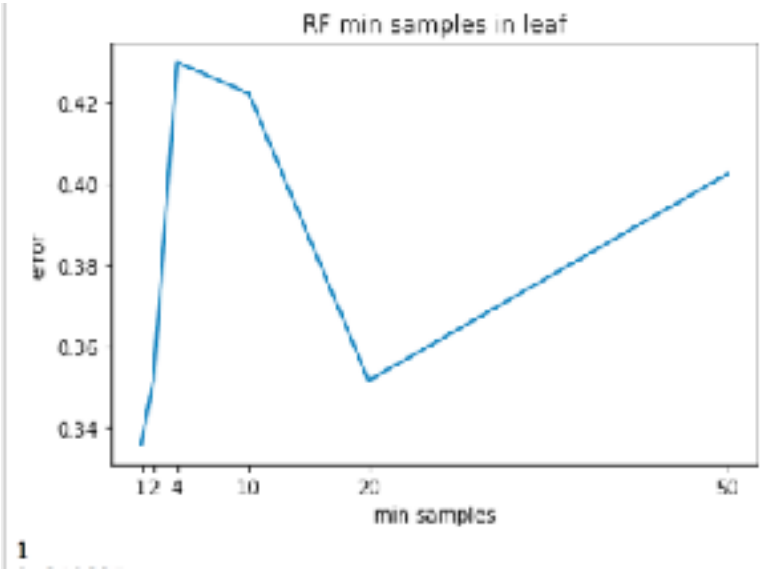
By changing the hyper-parameter min_samples_split, which is the minimum number of samples required to split an internal node, the effect is shown as below:



5

Only typical numbers are tried here, the model error keep increasing when the min_samples_split goes ridiculously high, 5 is the best choice here.

By changing the hyper-parameter min_samples_leaf, which is the minimum number of samples required to be at a leaf node, the effect is shown as below:



1

There is really not any obvious trend shown here, but 1 give the best result.

By building models corresponding to every best hyper-parameters I got result as shown below:

0.640625	The default model give a 64% accuracy.
0.678125	With best n_estimators = 29, I got 67.81% accuracy
0.678125	With best max_features = sqrt, I got 67.81% accuracy
0.6875	With best max_depth = 81, I got 68.75% accuracy
0.6375	With best min_sample_split = 1, I got 63.75% accuracy
0.65625	With best min_sample_leaf = 5, I got 65.63% accuracy
	Clearly, most of them give a improvement on the test accuracy.

When I combine all the best hyper-parameters to build a model, it gives me 70.31% accuracy. This is the only time I achieve a accuracy that is higher than 70% on this problem.

```
0.703125
```

Two things are worth noting here. First, a best parameter on the validation data does not guarantee to be the best parameter for test data. Second, the combination of all the best parameters does not guarantee a best model, there are a lot of trade off.

All in all, random forest appear to be the best model for this red wine classification problem.