# A Comparison of Random Forest and K-Nearest Neighbour in Predicting Wine Quality

*Dimitrios Megkos - 210034034*

## 1. Glossary

| Abbreviation/Term | Definition |
|---|---|
| Physicochemical features | Features that can describe a wine sample (e.g., pH, Alcohol, chlorides) |
| SMOTE/Synthetic Minority Oversampling Technique | Sampling technique applied to training set to overcome class imbalance problems. |
| Training set | Part of the original data that is used for training the machine learning algorithm |
| Testing set | Part of the original data that is used for the final testing of the macine learning algorithm |
| 10-Fold cross validation | A method that splits the dataset in 10 folds giving a chance to every observation in the dataset to be used for training and testing. |
| Hyperparameter | A parameter whose value controls the learning process. |
| ML/Machine Learning | A system that builds/trains a predictive model from input data. |
| DT/Decision Tree | Machine learning algorithm used for classification and regression tasks. |
| Boosting | A technique that iteratively combines a set of "weak" classifiers into a classifier with high accuracy by upweighting the examples that the model is currently misclassifying. |
| RF/Random Forest | Machine learning algorithm that uses multiple Decision Trees to make predictions. |
| NumSplit/maximum number of splits | The maximal number of decision splits (or branch nodes) for the ML tree. |
| LeafSize/minimum leaf size | The Minimum number of leaf node observations for the ML tree |
| NumPred/number of predictors | Number of predictors to select at random for each split of the ML tree. |
| PredSel/Predictor selection | Algorithm used to select the best split predictor at each node of the ML tree. |
| LnCycles/number of learning cycles | The number of ensembles learning cycles |
| KNN/K-Nearest Neighbour | Machine Learning algorithm that makes predictions based on the distance between the test data and all the training points. |
| Knum/neighnum/k-neighbour | Number of nearest neighbors to find for classifying each point when predicting. |
| DistFunc/dist/Distance | Distance metric/function used to calculate distance between test data and training points. |
| Confusionmat/Confusion matrix | A table that summarizes how successful a classification model's predictions were. |

## 2. Intermediate results including any negative results

### 2.1. Data pre-processing

Initially the plan was to use the dataset in its original form, without fixing the class imbalance by adding synthetic data. However, the performance of the Machine Learning algorithms, especially that of K-Nearest Neighbours, was poor.

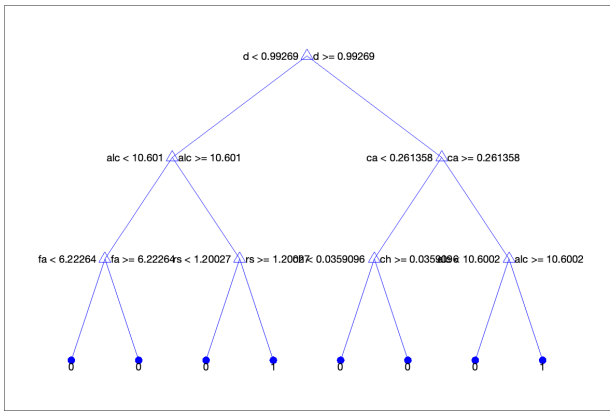
The first method that was tested in order to improve performances, was K-Fold cross-validation using stratify option. The performance improved a bit, though the results were not satisfactory. Adding synthetic data, using the SMOTE technique, improved the results by a significant amount (accuracy increased from low 60s percentages to high 70s percentages for both models). For this reason, it was decided to add synthetic data to the training set.

### 2.2. Random Forest

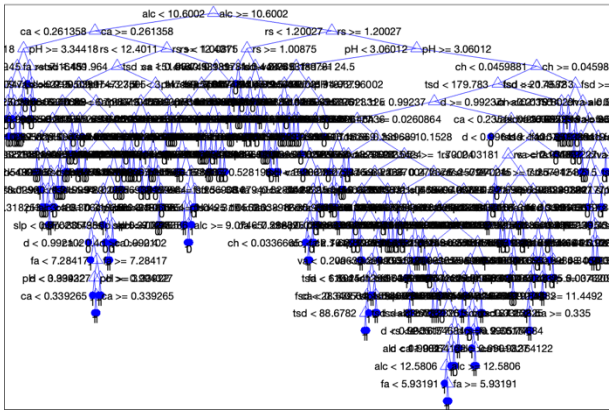
Regarding hyperparameter optimization for Random Forest algorithm, the original idea was to include a great number of different values for maximum number of splits, minimum leaf size, number of predictors, and number of learning cycles since the goal was to highlight the versatility Random Forest has. However, this proved to be somewhat idealistic since the number of iterations the optimization algorithm had to go through initially was too great and the searching for the best combination of parameters never ended. For this reason, the number of different parameter values to be tested was reduced drastically.

Special attention was given to maximum number of splits and minimum leaf size as they are the most important parameters, having a direct control on the depth of the trees. More specifically, numbers that were too small and too large were excluded from the searching function in order to avoid underfitting and overfitting.

*Underfitting example*
*Maxnumsplits = 8, minleafsize = 1*




*Overfitting example*
*Maxnumsplits = 500, minleafsize = 10*

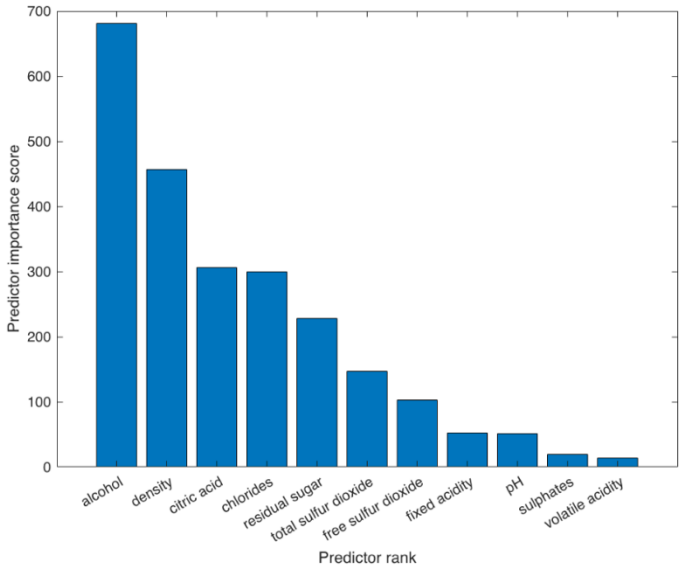


### 2.3. K-Nearest Neighbour

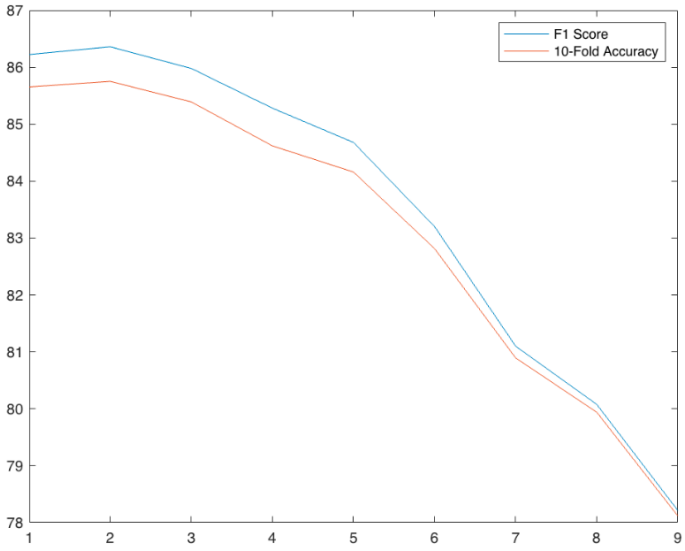
The first implementation of the algorithm did not include all the Physicochemical features of the dataset. Initial training and testing were performed using only the top 4 features according to univariate feature ranking for classification using chi-square tests. The decision was taken based solely on these results, without exploring how classification loss was changing when removing least important features. It was later discovered that K-NN's predictions had gotten worse by including only the top 4 features.

A minor accuracy improvement was observed by removing only the least important feature; however, it was decided in the end to keep all features in order to have a fair comparison with Random Forest.

*Feature Importance*




*Metrics when removing least important features*



# 3. Implementation details including a brief description of main implementation choices

## 3.1. K-Fold cross-validation

Cross-validation using K-Fold, specifically 10-fold, was used for both models, giving a chance to every observation in the dataset to be used for training and testing, in order to have better metrics.

Regarding Random Forest, since the algorithm is using bootstrap aggregation, using K-Fold cross validation does not really add any extra value. However, because K-Fold cross validation is used for K-Nearest Neighbour algorithm, in order to have a fair comparison between the two classification algorithms, it is used in Random Forest also.

## 3.2. Classification ensemble and templateTree

There is more than one way to implement a Random Forest algorithm in Matlab. In this exercise, a classification ensemble is used with method “Bag” for bootstrap aggregation. A templateTree is passed as a learner taking advantage of the flexibility the template gives in choosing and changing hyperparameters.

## 3.3. Custom functions

For the purpose of this exercise, six custom functions were created, three for each of the two algorithms, which are called from the main matlab file. The purpose of this approach is to have cleaner code and to automate training, testing and metrics calculation.

| | |
|---|---|
| CustomRF/CustomKNN | Receives dataset with hyperparameter values, trains/tests model using 10-Fold cross-validation and calculates performance metrics. |
| OptimiseCRF/OptimiseCKNN | Receives dataset with a range of hyperparameter values and calls CustomRF/KNN going through iterations to find best parameters that give the best F1 Score. |
| FinalCRF/FinalCKNN | Receives train dataset, an unknown test set, best hyperparameter values, trains/tests model and returns performance metrics with confusion matrix. FinalCRF also returns a random tree from the Random Forest. |

## 3.4. Custom optimization function

Matlab has a built-in technique that does automatic hyperparameter optimization for machine learning algorithms. While this is a strong tool to have, it has certain limitations since there is not much control in the selection and values of the hyperparameters that are being tested.
The custom optimization function that is being used for both algorithms, though the way it is searching for the best combination of hyperparameters can be considered unconventional and time consuming, is more versatile since it gives the option of which hyperparameters to change and which values to go through.