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

Dimitrios Megkos - 210034034

1. Description and Motivation

The purpose of this exercise is to build and optimize two supervised machine learning algorithms in order to compare their performance on a binary classification problem. For this task, the White Wine Quality Data Set [1] was used, downloaded from UCI.

3. Machine Learning Models Summary

3.1. Random Forest [2]
A supervised machine learning algorithm that uses multiple Decision Trees to obtain better predictive performance. For classification problems, the class selected by most trees is the Random Forest output.

3.1.1. Pros
- Can be used for both classification and regression problems
- Strong against overfitting [4] because of ”bagging”
- Deals with outliers and missing values internally, requiring less data pre-processing

3.1.2. Cons
- Trade-off between higher accuracy and lower interpretability
- Training hundreds of trees can be time consuming and computationally heavy

3.2. K-Nearest Neighbour [5]
A supervised machine learning algorithm that makes predictions based on the distance between the test data and all the training points.

3.2.1. Pros
- Can be used for both classification and regression problems
- Fairly simple and easy to understand and implement, having fewer hyperparameters compared to other algorithms

3.2.2. Cons
- Data must be standardized beforehand, since K-NN is classifying samples based on calculated distances
- Training is affected by class imbalance
- A small value of k could lead to overfitting, a big value of k can lead to underfitting

4. Hypothesis Statement

- Both algorithms will deliver similar results, due to the size of the dataset and the number of classes.
- The imbalanced class (class “1”) will be the one with the most misclassifications.
- It will take significantly less time to train and optimize K-Nearest Neighbour compared to Random Forest due to Random Forest’s method of making predictions.
- Random Forest will be more versatile due to the higher number of available hyperparameters, compared to K-Nearest Neighbour.

5. Training and Evaluation Methodology

- Both algorithms are trained on the class balanced training set (80% of the original dataset), using the remaining 20% for the final testing of the optimized algorithms.
- Models are trained using 10-Fold cross validation for more accurate out-of-sample metrics.
- Performance metrics are obtained for each model, like K-Fold Accuracy, confusion matrix, precision, recall, and F1 Score [3].
- Optimal hyperparameters are obtained using a custom search function and F1 Scores.
- Models are trained on the training set using the optimal hyperparameters and tested on the testing set.
- Confusion matrices and F1 Scores are obtained and compared for both models to find the better algorithm for the wine classification problem.

8. Lessons Learned and Future Work

8.1. Lessons Learned
- The importance of hyperparameter tuning when it comes to creating good predictive models
- Some algorithms require extra data pre-processing while others work straight out-of-the-box
- Higher accuracy does not mean better results. Some problems require extra performance metrics for evaluation.

8.2. Future Work
- Implement a more elegant and cost and time efficient method of hyperparameter optimization
- Try different ways to tackle class imbalance like specifying a misclassification cost
- Use Random Forest as a tool to find feature importance and compare with the results of chi-square tests method

References:

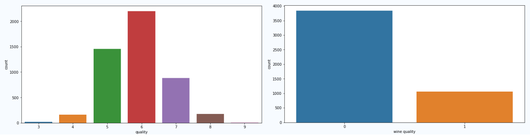
1. *P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. Available at: [Link]*
2. *Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). [Link]*
3. *Powers, David. (2008). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Mach. Learn. Technol.. 2. Available at: [Link]*
4. *Cawley, Gavin C.; Talbot, Nicola L. C. (2010). "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation" **11**. Journal of Machine Learning Research: 2079–2107.*
5. *Bishop, C.M. 2006, Nearest-neighbour methods. Pattern recognition and machine learning, Springer, New York.*

2. Initial Analysis and Pre-Processing

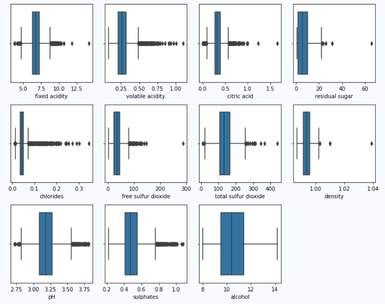
- The dataset contains 4898 white variants of the Portuguese "Vinho Verde" wine, having 12 columns, 11 physicochemical continuous features and 1 discrete output representing the quality score.

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6
6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5
6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6
5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7
6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6

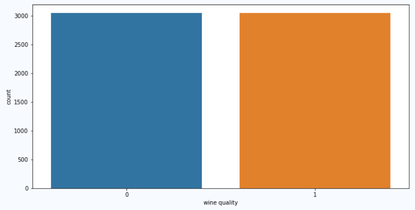
- Initially there were 7 unique quality classes, with scores ranging from 3 (worst quality) to 9 (best quality). However, for the purpose of this exercise, they are transformed to “0” as “not good quality” (scores 3 to 6), and “1” as “good quality” (scores 7 to 9).



- A search for missing values was performed returning no results.
- Using a series of boxplots for each feature, several outliers are identified in the dataset. There are 4864 samples left after the removal of outliers.



- Class “0” has 3805 wine samples while class “1” has 1059 samples.
- There is a big class imbalance in the dataset, which can have a great impact on the training process of some algorithms. This problem can be fixed by applying a technique known as SMOTE (Synthetic Minority Oversampling Technique) to the training set.
- The dataset is split into training/testing sets with a ratio of 80/20.
- The new training set contains a total of 6092 wine samples, after applying SMOTE, equally split between the two classes.

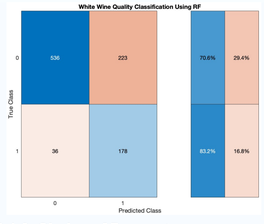


6. Experimental Results and Choice of Parameters

6.1. Random Forest

A custom templateTree is passed as a learner to Matlab’s fitcensemble function, using the method “Bag”. This approach provides flexibility when it comes to the choice of hyperparameter optimization. The parameters that are used for optimization are maximum number of splits, minimum leaf size, number of predictors to select randomly on each split, predictor selector, and number of learning cycles. Gini’s impurity is used for splitting and pruning.

6.1.1. Experimental Results
Initial call of Random Forest classification ended in 1.15 seconds, with scores F1 Score = 0.73 and Accuracy = 77%. The model classified correctly 70.6% of class “0” samples, and 83.2% of class “1” samples as shown in the confusion matrix on the right. These are decent out-of-the-box results; however, there is still room for improvement.



NumSplit	LeafSize	NumPred	PredSel	LnCycles	F1Scrore	TestAcc
10	1	5	“allsplits”	100	0.73	77%

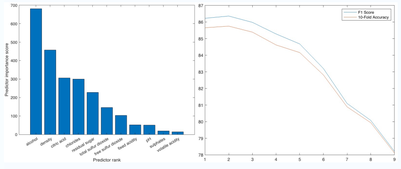
The above table shows the hyperparameter values that were used for these results.

NumSplit	LeafSize	NumPred	PredSel	LnCycles	F1Scrore	KFoldAcc
180	5	4	“allsplits”	100	0.87	86.5%

6.1.2. Choice of Parameters
Optimal parameters are found using a custom optimization function. Several different hyperparameter values are passed as an input. The function trains and tests Random Forest using these parameters and returns the combination that gives the best F1 Score. The results can be found in the table on the left.

6.2. K-Nearest Neighbour

The algorithm by its nature has fewer optimization options. The parameters that are used for optimization are the k-number of neighbours and the distance function that is used to calculate the distance. The importance of each feature on the classification was checked using chi-square tests. Removing the least important feature improved the scores a bit. However, in order to have a fair comparison with RF, all features are going to be used for KNN.



6.2.1. Experimental Results
Initial call of KNN classification ended in 0.20 seconds, with scores F1 Score = 0.75 and Accuracy = 78%. The model classified correctly 78.8% of class “0” samples, and 77.6% of class “1” samples as shown in the confusion matrix on the right. The above scores were returned by training and testing the model using k-number = 5 and Distance function = “euclidean”. Decent numbers overall but they can be improved.

k-num	DistFunc
5	”euclidean”

k-number	DistanceFunc	F1Score	KFoldAcc
3	“cityblock”	0.87	86%

6.2.2. Choice of Parameters
Same optimization method is used to obtain optimal parameters. The function trains and tests K-Nearest Neighbour using different k-number values and distance functions and returns the combination that gives the best F1 Score. The results can be found in the table on the left.

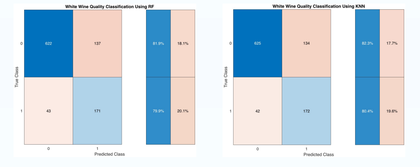
7. Analysis and Critical Evaluation of Results

- Both Random Forest and K-Nearest Neighbour did a fairly good job in predicting a good quality wine. In order to have a fair comparison, same techniques were used during data pre-processing, training and evaluating for both models.
- This classification problem can be considered simple, given it is a binary classification problem having a few thousand samples. Performances are expected to differ when more classes and samples are introduced, with RF possibly gaining the upper hand due to its versatility.

Random Forest
Random Forest does a better job in handling both categorical and numerical data, class imbalance, outliers, and missing values compared to K-Nearest Neighbour. It can almost work out of the box with minimum pre-processing. Furthermore, it is more versatile due to the higher number of available optimizable hyperparameters (i.e., maximum number of splits, minimum leaf size, number of variables to sample, predictor selector and number of learning cycles) compared to K-Nearest Neighbour (number of k-neighbours, distance function). Number of splits and minimum leaf size are the most important parameters since they have direct control on the depth of the trees. However, Random Forest is harder to fine tune and more time consuming since it must go through a much higher number of iterations to find the best combination of hyperparameters.

K-Nearest Neighbour
K-Nearest Neighbour might not have the same flexibility that Random Forest has regarding hyperparameter tuning, yet it requires significantly less amount of time to find the best combination of hyperparameters. It is an easy to use and understand algorithm, however, unlike Random Forest, it cannot handle both categorical and numerical predictors, class imbalance, outliers, and missing values internally thus, the data need to be cleaned accordingly beforehand.

Conclusion
The below confusion matrices show that K-Nearest Neighbour returns better results in finding both the **not good (0)** wines and the **good (1)** wines, compared to Random Forest. More specifically, Random Forest correctly classified **81.9% not good (0)** and **79.9% good (1)**, while K-Nearest Neighbour correctly classified **82.3% not good (0)** and **80.4% good (1)**.



Considering all the above, the better algorithm to use for the given binary classification problem, is **K-Nearest Neighbour**.