DATA7703 Group Project

Prediction of Wine Quality Using Machine Learning Models

By: Group 14

Dae Jong Han, Xinyuan Peng, Beatrice Carrol Que, Yicheng Liang

Abstract

Wine quality is crucial for the wine industry. However, the traditional methods to determine wine quality is labour intensive, time-consuming, and highly subjective. Application of machine learning techniques to predict wine quality scores based on compiled data of physicochemical properties and their obtained scores has widely been investigated as a potential innovative solution to save time, money, and effort for the wine industry. Machine learning models such as decision trees (DT), k-means nearest neighbours (KNN), Support Vector Machines (SVM), and Artificial Neural Network (ANN) have been explored to some extent in literature. In this project, we extend our investigation into ensemble and boosting methods such as Random Forest (RF), AdaBoost, XGBoost, and LightGBM. Furthermore, we will assess model performance and investigate significant features that affect model performance.

We give consent for this to be used as a teaching resource.

Table of Contents

1. Introduction

1.1. Wine Quality Prediction Applications

Wine is a popular beverage globally with many producers and consumers, making it an important part of the food and beverages industry. Given the significance of the market, the assessment and certification of quality, for the purpose of producing and selling, is crucial in the wine industry (Cortez, 2009). However, the assessment process involving human experts to determine wine quality is expensive, time-consuming, and labour intensive (Mor, 2022).

The application of machine learning techniques has been widely studied for the purpose of industrial quality control process (Hoseini, 2021). The wine industry is no exception and has moved to develop and utilise various modelling techniques to predict the quality of wine. Traditionally, wine quality has been tested in the lab, however machine learning models provides a method to predict wine quality based on historical data. This has many benefits such as identifying and eliminating the need to perform unnecessary tests, reducing the number of samples tested, save cost on laboratory reagents and consumables, save time by anticipating the results and failing samples early and thereby taking an earlier action (Rodriguez, 2022). Early detection of low-quality products will prevent spending resources on further testings and corrective actions for a product and enable focus on more relevant tests to help reduce operational cost of the production line (Rodriguez, 2022).

1.2. Aim

Various models have been explored in past literature such as decision trees, k-means nearest neighbours (KNN), Random Forest (RF), Support Vector Machines (SVM), and Artificial Neural Network (ANN) (Kumar, 2020; Dahal, 2021; Kothawade, 2021). For this project, we aim to extend further to explore more complex models such as Gradient Boosting, AdaBoost, XGBoost, and LightGBM. The ensemble and boosting methods turn weak learners into stronger learners to allow for more accurate results. This project will explore and assess the differences between these models and determine the best performing models for this application.

2. Methodology

2.1. Data Source and Pre-processing

For this project, we obtained our wine quality dataset from the publicly available UCL Machine Learning Repository (Cortez, 2009). There are two types of wine quality dataset, red and white wine. The datasets contain 11 physiochemical properties of wine: ***fixed acidity, volatile acidity, total sulphur dioxide, chlorides, pH level, free sulphur dioxide, density, residual sugar, citric acid, sulphates,*** and ***alcohol***. The dataset also contains a sensory score value that was graded by blind taste testers ranging from 0 (Poor) to 10 (excellent).

Data preparation is a crucial part of modelling as input quality can determine the effectiveness of the model. Since we want to predict both the quality of wine as well as the wine type, we concatenate the data of red wine and white wine. Then a new column called wine type is created, with 0 representing red wine and 1 representing white wine.

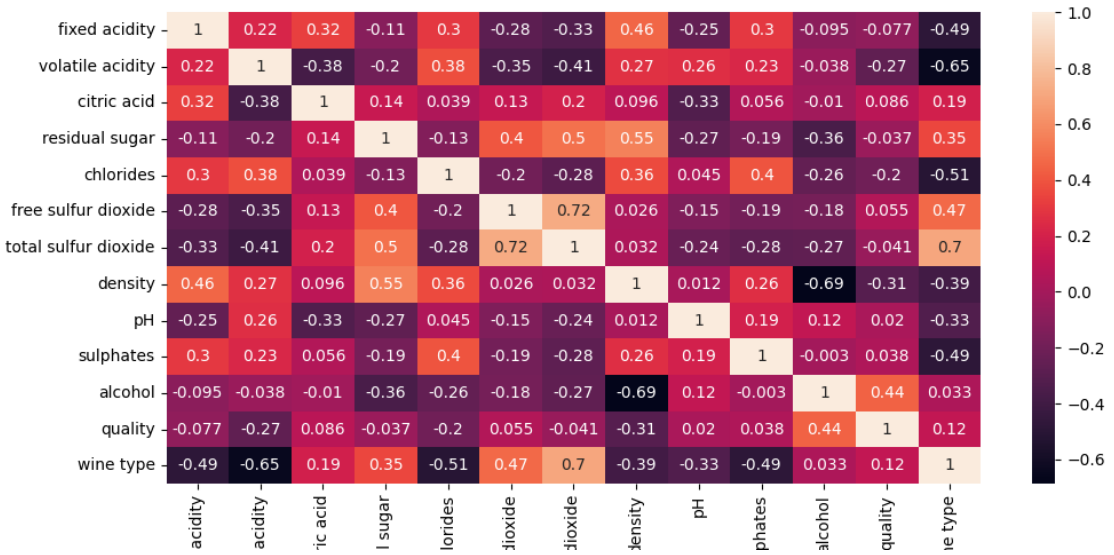

Figure 1. Correlation Matrix of Red and White Wine Dataset

To examine the linear correlation between features and target label, we plot the correlation matrix. If we regard an absolute value between 0.5 and 0.7 as moderate correlation, and an absolute value larger than 0.7 as strong correlation, then we can find many correlated features but very few of them are strongly correlated. The only Pearson coefficient larger than 0.7 is 0.72, which is between free sulphur dioxide and total sulphur dioxide, and this isn't of great help to our analysis.

We need to have a closer look at the line of quality. But none of the absolute values are larger than 0.5, which indicates there's no strong linear correlation between any single feature and quality. We'll need more sophisticated models to do the prediction.

2.1.1. Conversion to a Classification Problem

As we will be utilising classification models, the data format of all features should be changed from object into float. Then we categorize the quality score into three groups and make it a classification problem. If the score is below 4, then the wine falls in the bad class. If the score is between 5 and 6, then the wine is average. If the score is 7 or above, then the wine is good. The quality of the wines is classified into three classes, low-quality (Class 0), average-quality (Class 1) and high-quality (Class 2).

Quality score	Equal or less than 4	5 to 6	Equal or greater than 7
Class	Poor	Average	Good
Value	0	1	2

Table 1. Defining Classes

2.1.2. Data Preparation

Next, we check the distribution of all the feature values, and plot them out in a histogram to allow us to have a quick glimpse of the distributions.

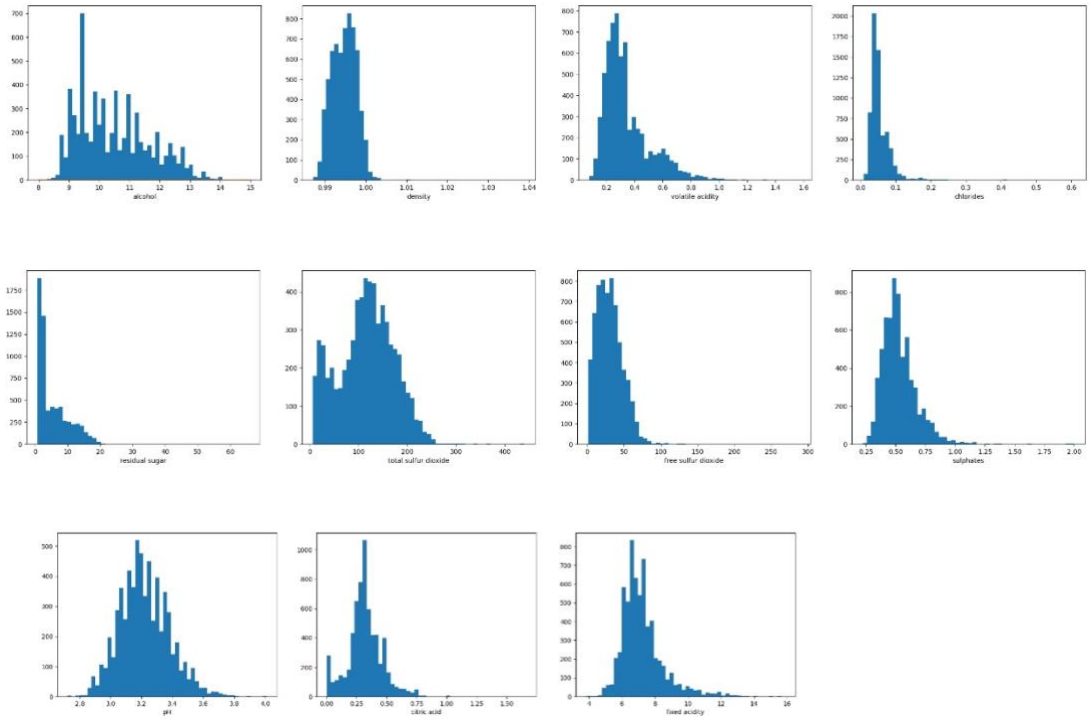

Figure 2. The distribution of all 11 features. From top to bottom, left to right: Alcohol, Density, Volatile acidity, Chlorides, Residual sugar, Total Sulphur dioxide, Free sulphur dioxide, sulphates, pH, Citric acid, Fixed acidity.

The figure shows that the feature data roughly resembles normal distribution. So, we applied standardization. The outcome of standardization represents the number of standard deviations above or below the mean.

To determine the necessity of standardization, we tried random forest respectively on data that's been processed differently. The first is the standardized data, second is the normalized data which uses the min-max scaler, and the third is the original data without either transformation. And the accuracy score on the standardized data is the largest among the three, proving that standardization is a necessary step.

The last part of pre-processing is to perform a train-test split. The proportion of test data is set to 10%.

2.2. Modelling

This section will provide an overview of the different classification models that can be applied to the wine dataset. These include Decision Trees, Random Forests, AdaBoost or Adaptive Boosting, XGBoost or Extreme Gradient Boosting, and LightGBM or Light Gradient Boosting Model.

2.3. Decision Tree Classifier

Decision Tree is a supervised machine learning model that uses a set of rules to reach a decision or classification. Each node is a splitting rule by which the model aims to increase purity in a decision. A high purity means splitting the data such that the resulting records belong to the same class. Nodes can be followed by more nodes but when a node is followed by a decision it is a leaf. Decision Trees are made of nodes and leaves that when followed, helps determine the class of a record.

2.4. Random Forest Classifier

To explain the algorithm behind Random Forest, it is important to first define what ensemble learning is. Ensemble learning is a method that combines different models, also called weak learners, to create a master model. Random Forest creates multiple and different decision trees in parallel. For each record to predict, Random Forest takes the classification of all decision trees and uses the majority as the final classification. This means that all underlying tree models have equal weight in the final prediction.

2.5. AdaBoost

In contrast to Random Forest, AdaBoost uses a boosting method for its learning rather than an ensemble method. Boosting is a sequential method wherein each underlying model or weak learner is built on the output of a previous learner. This means that each model is dependent on another. The advantage of this is that each learner improves on the previous learner. AdaBoost algorithm starts with creating one tree with one root and two leaves, also called a stamp, which creates a prediction on each of the records. AdaBoost then creates a new stamp that prioritizes making an accurate prediction on the records misclassified by the

first stamp. This process repeats until a master model is achieved. The master model combines all stamps and assigns different weights to the stamps to optimize performance. The figure below shows the algorithm of AdaBoost.

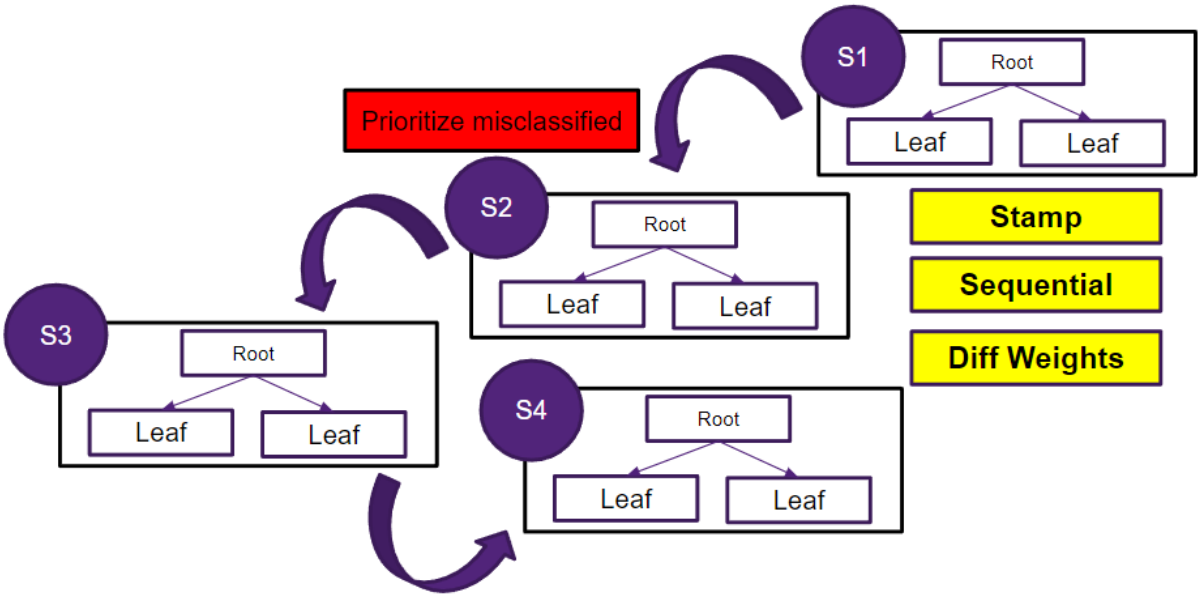

Figure 3. AdaBoost procedure

2.6. XGBoost

2.6.1. Gradient Boosting

XGBoost, or Extreme Gradient Boosting, is built on top of the Gradient Boosting Model. The algorithm behind Gradient Boosting starts with an initialization step (all automated and built-in within the model). The step involves creating an initial prediction for all the records. For regression problems, the initial prediction would be the average of all the output values. For classification, equal probabilities on the different classes are used. Based on the initial prediction, residuals (or the difference between the actual and predicted values) are computed. The model then creates a tree with varying sizes of eight to about 30 leaves. The tree is used along with the independent variables to provide prediction not on the target variable but on the residual. The step is reversed, and the target value is computed by adding the residual (multiplied by the learning rate, the speed of going through the gradient descent on errors and a parameter to be tuned) to the initial prediction. The target value replaces the initial prediction and the cycle restarts. Each cycle creates a new error model, and the

predicted residual is added to the computation of the target value. The figures below describe the algorithm behind Gradient Boosting.

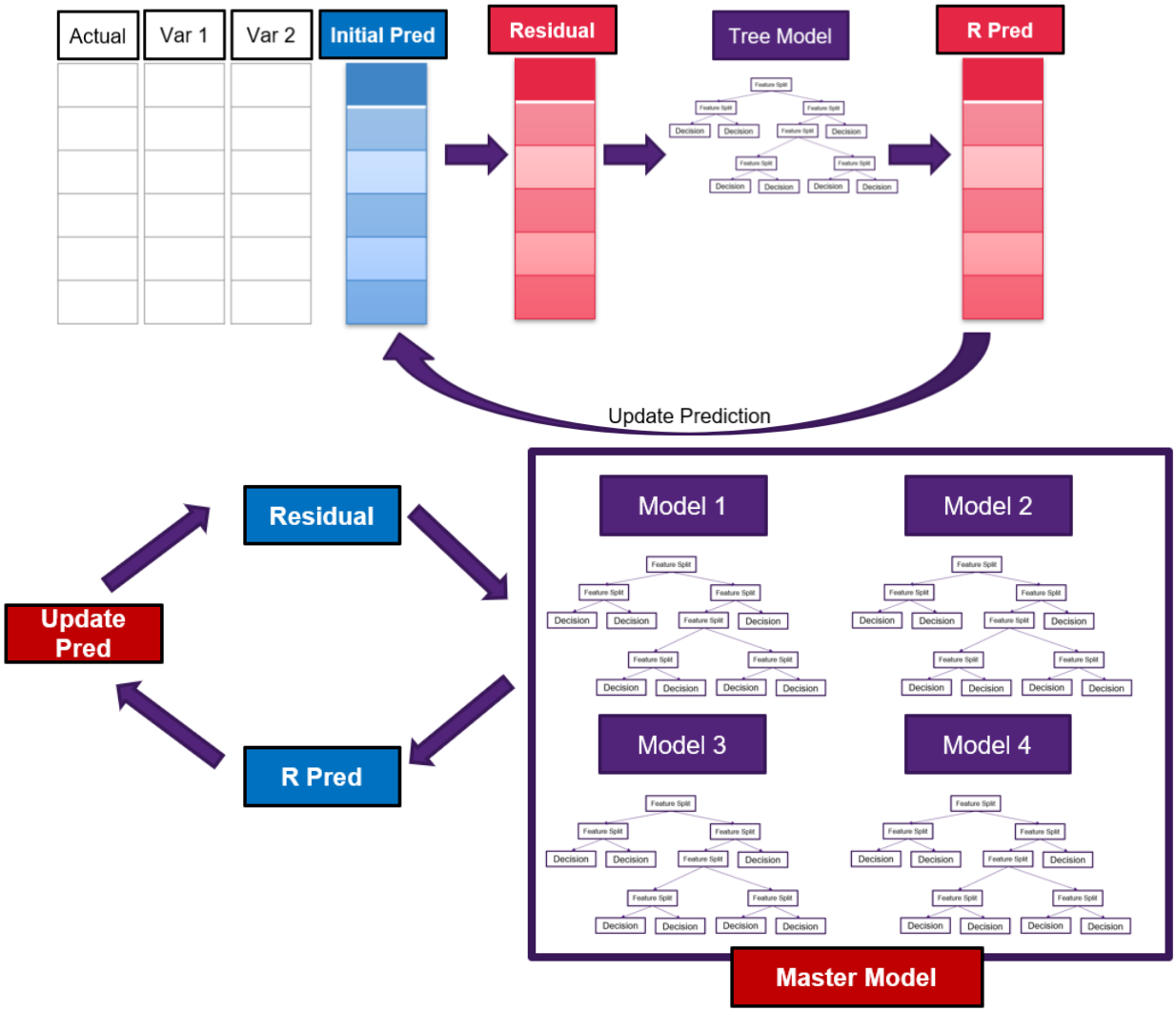

Figure 4. Gradient boost procedure

2.6.2. Extreme Gradient Boosting

XGBoost is an improvement on Gradient Boosting. One improvement is increased speed and memory which this paper will not focus on. The other improvements are involved with the algorithm. First is that XGBoost applies regularization on its learners such that the learners would not overfit the data. Second is its Auto Pruning feature such that the tree wouldn't grow too large which again would result in overfitting. The last improvement is that XGBoost is capable of handling missing data.

2.7. LightGBM

LightGBM is another Gradient Boosting model that features increased speed and better performance. Again, this paper will not focus on the improvement of speed but rather, focus on the algorithm. It has two main differences from Gradient Boosting. The first one is called Exclusive Feature Bundling. This means that multiple variables that have exclusive values for each are combined and turn into one variable. The figure below shows how this works. Var 1, Var 2, and Var 3 are three variables that can only have 1 as a value for one of the variables. For example, the first record is valid as only Var 1 is 1 and the rest are zeroes. The same goes for the second and third records. But the fourth record is not valid as two of the variables have 1 as the value. If this record is valid, it means that Var 1 and Var 2 are not exclusive to each other. LightGBM combines the values of the three variables and turns them into 1 as shown for records 1, 2, and 3.

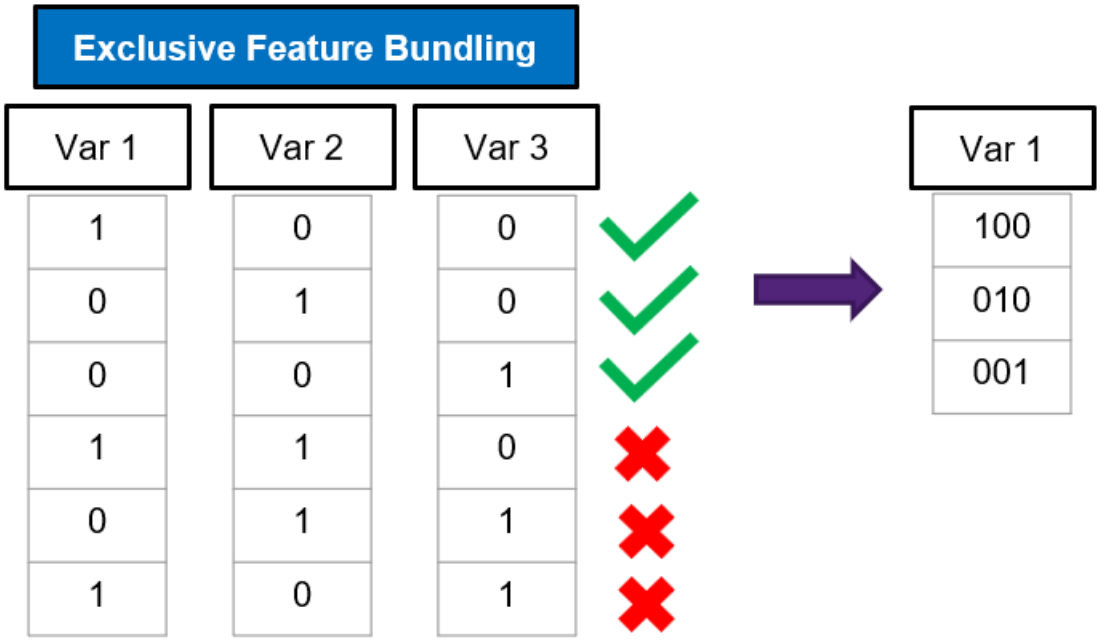

Figure 5. LightGBM bundling procedure

Another feature of the LightGBM is the Gradient Based One Side Sampling. This sampling's objective is to improve the performance of the learners by focusing on the highest errors. In the first iteration of learning, the model computes for the gradient descent of the errors on the target values. In a sample size of 100, there would be 100 measures of gradient errors. These gradient measures are sorted from highest to lowest. The higher the gradient error is,

the poorer the performance of the model. The model then puts the top gradient errors into 1 bin and the rest into another bin. For each sample taken by the learner to fit on, the top gradient errors are included and only a random portion of the rest are included. This allows the model to focus on the worst performers. The prediction of the weak learner is then passed on to the next and the gradient errors are recalculated. Again, the process is repeated and the cycle restarts. The figure below shows how the Gradient Based One Side Sampling works.

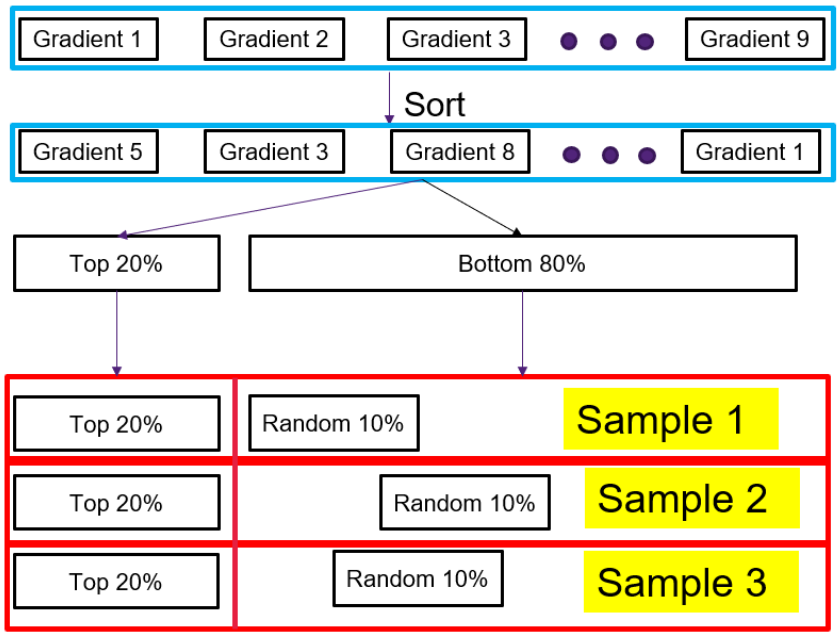

Figure 6. LightGBM sorting procedure

2.8. Python Application

This section explains how each of the models above is implemented in Python. To get access to these models' algorithm, the following packages must be installed.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
import lightgbm as lgb
from xgboost import XGBClassifier
```

Figure 7. Python Implementation I

The previous sections have discussed how the data are cleaned and transformed. The next step is to create the train and test splits which sklearn conveniently provides a method for. This method can be accessed using the train_test_split from sklearn.model_selection. The method provides data for X and Y training sets as well as X and Y test sets. The training sets will be used by the models to learn from, and the test sets will be used to evaluate model performance. To start building the models, there are certain parameters that must be set, and the following enumerates these for each of the models.

1. Decision Tree Classifier
    a. Criterions
    b. Splitters
    c. Maximum Depth
    d. Minimum number of samples per split
    e. Minimum number of samples in leaf
2. Random Forest Classifier
    a. Criterions, Maximum Depth, Minimum number of samples per split, Minimum number of samples in leaf
    b. Number of estimators
    c. Bootstrap
3. Adaboost
    a. Maximum Depth, Number of Estimators
    b. Learning Rate
4. XGBoost
    a. Maximum Depth, Number of Estimators
    b. Gamma
    c. Lambda
    d. Alpha
    e. Minimum Child Weight
    f. colsample_bytree
5. LightGBM
    a. Maximum Depth, Minimum number of samples in leaf, Learning Rate
    b. Number of Leaves

For each of these parameters, different values have been selected and formatted in a grid as shown in an example for a Decision Tree below. The criteria to be used for splitting can be “Gini” or “Entropy”. The splitters can also either be “Best” or “random”. The maximum depth has been set to different integer values between 3 and 15. Several values have also been set for the minimum number of samples per split and the minimum number of samples in leaf. All these parameter settings have been organized in a dictionary called the random_grid.

```
criterions = ['gini', 'entropy']
splitters = ['best','random']
depths = range(3,15)
min_samples = [2,3,5,10,15,20,30,40,50,60]
min_samples_leaf = [1, 2, 4]

random_grid = {'criterion': criterions,
               'splitter': splitters,
               'max_depth': depths,
               'min_samples_split': min_samples,
               'min_samples_leaf':min_samples_leaf,}
```

Figure 8. Python Implmentation II

To find the best value for each of the parameters, this paper used the method called RandomizedSearchCV inside the sklearn.model_selection package. To call the method, the estimator or the model must be selected. For this case, it is the specific classification model. The dictionary of parameter values is also used as input. Number of iterations is set (100 is used for this study) and the number of times the cross-validation is applied should also be set (3 for this study). The method returns the best parameters for the model and provides a predict function to evaluate on the test set. This method of parameter tuning has been applied to all the classification models.

2.9. Classification Model Assessments

There are various performance metrics that can be used on classification models to determine their effectiveness.

2.9.1. Accuracy, Precision and Recall

Accuracy is the amount of the correct prediction to the total number of predictions made as a percentage. This is a simple metric that is useful only if the dataset has an equal distribution of classes. If there is a class imbalance it is better to use the F1 score metrics to assess model effectiveness (Korstanje, 2021).

$$Accuracy = \frac{\#\ of\ Correct\ Predictions}{\#\ of\ Total\ Predictions}$$

Equation 1. Accuracy Formula

Precision is a metric that looks at the total number of things that was predicted to be positive and counts the percentage that is correct (Korstanje, 2021).

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

Equation 2. Precision Formula

Another metric is recall which looks at everything that was positive and counts the number of positives that was positive given as a percentage (Korstanje, 2021).

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

Formula 3. Recall formula

It would be ideal if the model can identify all the positive cases and only the positive cases. However, this cannot practically be done as there is a trade-off between precision and recall. Hence, a combination of the two metrics can be used on imbalanced data.

2.9.2. F1 Score

The F1 score is a metric defined as a harmonic mean of precision and recall, combining the two metrics into a single measure. As the F1 score is the average of precision and recall, it gives equal weights to both metrics. Therefore, it is the desired metric score for imbalanced data (Korstanje 2021).

$$F1\ score\ =\ 2\ \times \frac{Precision\ \times\ Recall}{Precision\ +\ Recall}$$

Formula 4. F1 score formula.

2.9.3. ROC and AUC

The Receiver Operating Characteristic (ROC) curve is a graph that shows the performance of a classification model. The plot is made of two parameters: True Positive Rate (TPR) and False Positive Rate (FPR). The curve plots the two parameters at different classification thresholds. The area under the ROC curve (AUC) provides an aggregate measure of the model performance across the classification thresholds. AUC provides a probability that a random positive sample is ranked highly than a random negative sample (Google, 2022).

2.10. Feature selection

There are various features that can be used to predict wine quality however some maybe better than others to allow for a better prediction. Each of the models has a function that provides values on how important the features and on the target value or prediction. These were used and averaged to provide a ranking of all the features. Based on the rank, different sets of features have been tested on each of the classification models. For example, on the first iteration, only the most important features have been used by the models. On the second iteration, the two most important features have been used, and so on. The different measures of performance such as accuracy, precision, recall, and F1 were all recorded. The next section provides a discussion on the results.

3. Results and Discussion

3.1. Quality Prediction

We need to measure the predictive capabilities of different machine learning models with metrics like accuracy, precision, recall and F1 score. The quality of the wines was classified into three classes, low-quality (Class 0), average-quality (Class 1) and high-quality (Class 2). The following section presented is the result of classifiers on the training set.

	KNN	KNN bagging	Decision tree	Random Forest	Adaboost	XGBoost	Light GBM	ANN
Accuracy	0.80	0.81	0.79	0.85	0.83	0.85	0.85	0.89
Precision (macro)	0.65	0.64	0.59	0.79	0.73	0.73	0.73	-
Recall (macro)	0.50	0.51	0.46	0.54	0.57	0.56	0.56	-

Table 2. 10-folds CV on training set

All models achieved an accuracy of more than 79%. Notably, boosting and bagging models have a higher accuracy than single model.

	KNN	KNN Bagging	Decision tree	Random Forest	Adaboost	XGBoost	Light GBM	ANN
Accuracy	0.78	0.80	0.77	0.84	0.85	0.79	0.85	0.82
Poor (Class 0) F1-score	0.11	0.00	0.07	0.07	0.27	0.00	0.20	0.32
Average (Class 1) F1-score	0.87	0.88	0.86	0.90	0.91	0.88	0.90	0.89
Good (Class 2) F1-score	0.54	0.54	0.40	0.65	0.66	0.42	0.65	0.64

Table 3. Evaluation on test set

The results on test set are like those on the training set, indicating the models can generalize well from the training data to other data from the problem domain. For single classifier, it is much more likely to be affected by outliers and thus have a relatively lower accuracy. For bagging models like RF, they outdo a single strong learner. RF averages the probabilistic prediction from all the trees in the forest for the final prediction instead of taking the actual

prediction votes and then averaging it. Due to the randomness of feature and bootstrap sampling, bias increases, and overall variance decreases, giving us a robust ensemble model, which generalizes well. For boosting classifiers, they have a better performance than bagging. Although the increase in accuracy is not obvious, they have higher F1-score for minority class 0 (low-quality wine). However, XGBoost is not as accurate as the other boosting models, because model performance also closely depends on dataset and its properties. The size of the wine dataset is relatively small. ANN does not outperform the LightGBM or RF, but it has the highest F1-score for minority class.

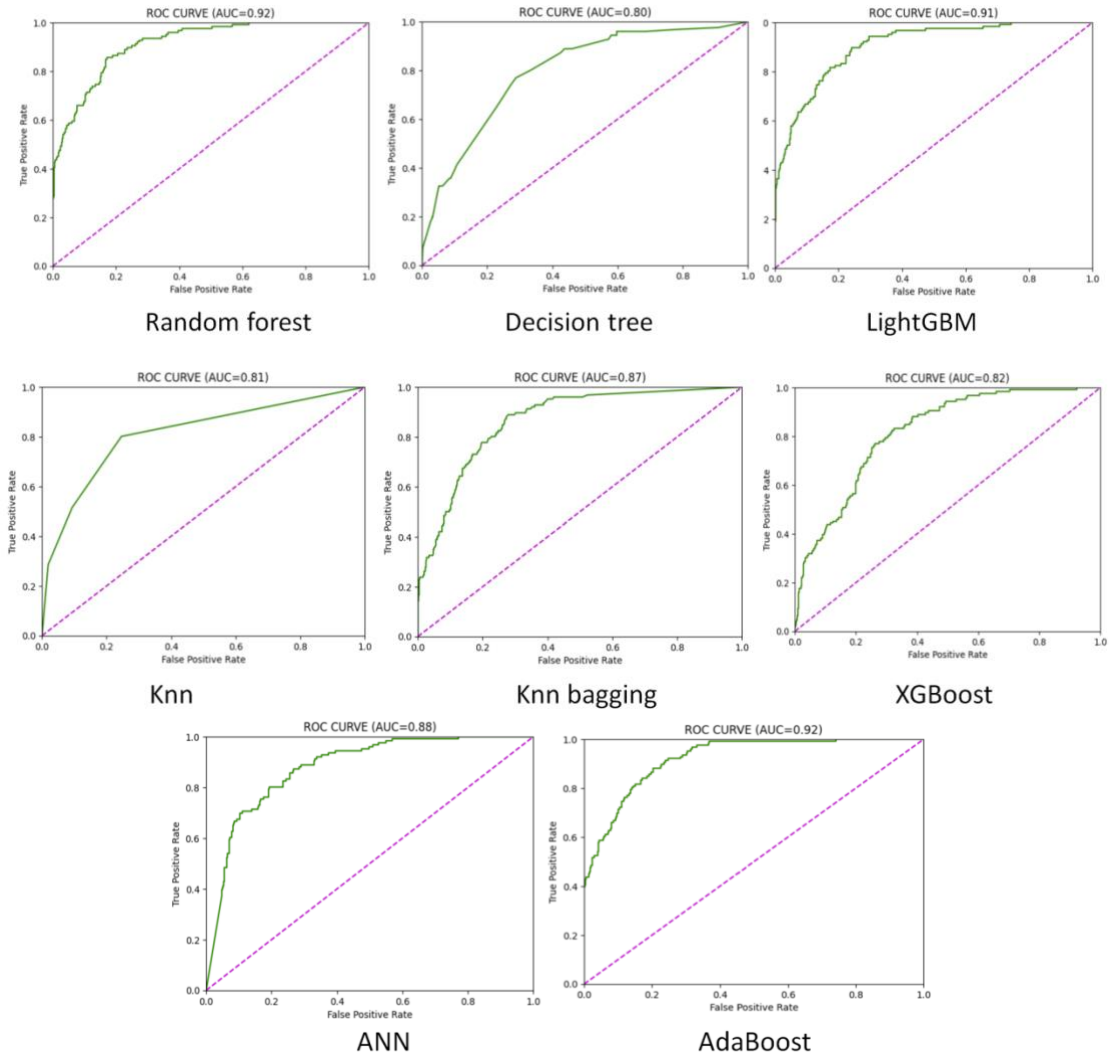

Figure 9. ROC curve and AUC

Then we focus on the high-quality wine (Class 2) and solve the problem of binary classification based on these models. With ROC curve, it can visualize how samples are separated and the area under the curve can be a very good metric to measure the performance of a binary classification algorithm. From the graph, Adaboost, Random Forest, LightGBM and ANN are good classifiers based on AUC.

Furthermore, we can make stacked models by the combination of single models. The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification task and make predictions that have better performance than any single model.

Classifier	Meta classifier	Accuracy (Test set)
KNN + Decision Tree	Logistic Regression	0.81
KNN + XGBoost	Logistic Regression	0.83
XGboost + SVM + KNN	Logistic Regression	0.86

why only these few combinations

Table 4. Stacking model

3.2. Type Prediction

A wine can either be a 'red' or a 'white' wine, which is also a binary classification task. The predictive models in this chapter can predict the type of wine according to independent variables. All the algorithms give great results, and the best seems to be the random forest.

	Accuracy	F1-score
Logistic Regression	0.9936	0.9957
Random Forest	0.9950	0.9967
SVM	0.9925	0.9950
KNN	0.9878	0.9919

Table 5. Wine type prediction

3.3. Feature Importance

In assessing model interpretability, it is important to determine which features play an important role in the model predictions. We will try to interpret why the model predicted a class label and which features are influential in its decision.

Firstly, we built a Linear Discriminant Analysis (LDA) model for the whole dataset and obtain the coefficients. Discriminant function coefficients are useful for describing group differences and identifying variables that distinguish between groups.

Features	Class 0	Class 1	Class 2
Fixed acidity	0.1092	-0.1211	0.4507
Volatile acidity	**1.4152**	0.0071	-0.3026
Citric acid	-0.0317	-0.0004	0.0077
Residual sugar	-0.9121	-0.1700	**0.8379**
Chlorides	0.0907	0.0128	-0.0676
Free sulfur dioxide	-0.3050	-0.0267	0.1629
Total sulfur dioxide	-0.3633	0.0568	-0.1514
Density	0.8716	**0.2330**	-1.0755
pH	0.0751	-0.0773	0.2868
Sulphates	-0.1831	-0.0657	0.2915
Alcohol	-0.2662	-0.1207	0.5209

Table 6. Discriminant function coefficients

Then we use LIME, a model-agnostic method to interpret the output (Hulstaert, 2018). LIME enables us to explain the nature of models using visualization and explains the model-based local values. Because RF and LightGBM have high accuracy in both the test and training sets, we can create explainers on them. Firstly, we focus on the sample that classified as low-quality (Class 0) by RF.

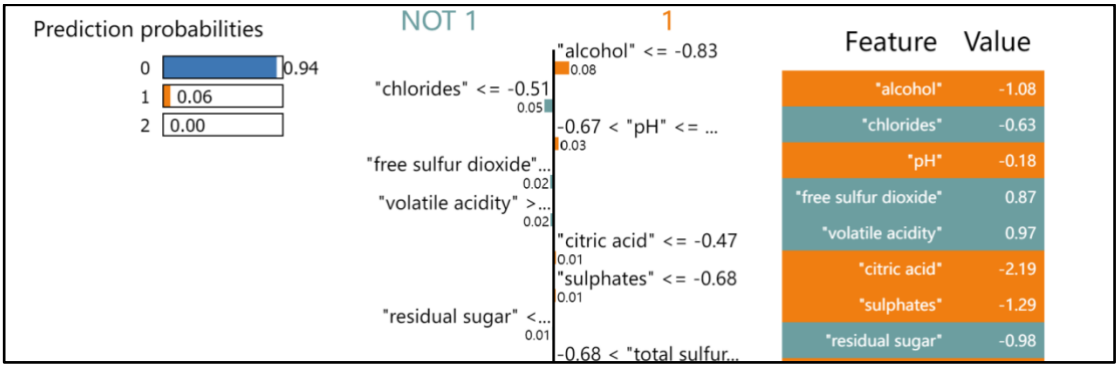

Figure 10. LIME analysis for sample 1

The model is 94% confident this is a low-quality wine. The values of free sulphur dioxide, volatile acid increase wine's chance to be classified. The citric acid and free sulphates are the factors that decrease it. Then we look at the good quality wine example. Here's the corresponding interpretation:

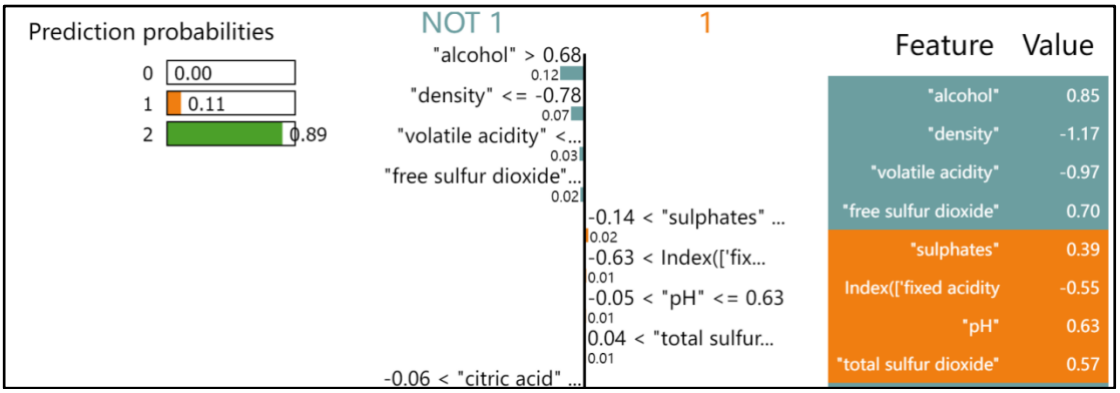

Figure 11. LIME analysis for sample 44

The model is 89% confident this is a high-quality wine. The values of free sulphur dioxide, alcohol, and pH increase wine's chance to be classified. The density and volatile acidity are the factors that decrease it.

From the above two samples, we can find that the order of importance of variables that distinguish the three categories is basically consistent with the results of LDA. Then we focus on the sample that classified as low quality (Class 0) by LightGBM.

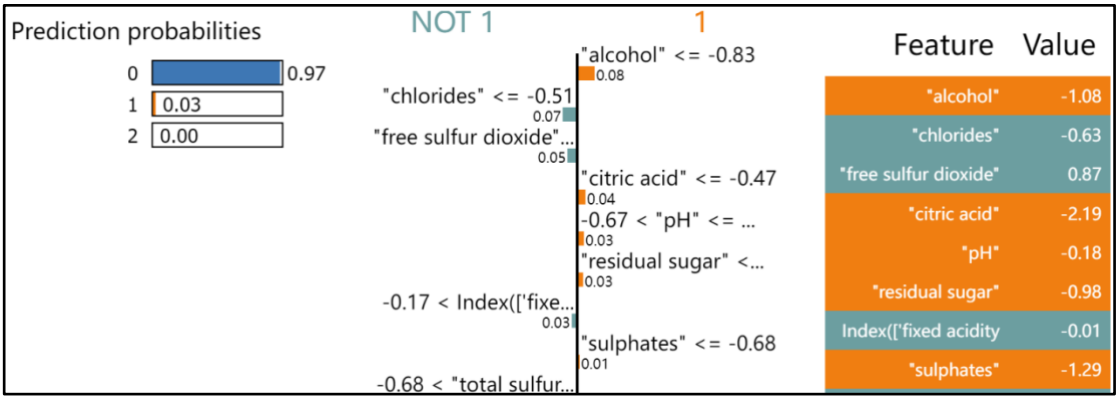

Figure 12. LIME analysis for sample 1

For sample 1, LightGBM also has a high level of confidence (97%) to classify it as low quality. Most of the variables have the same importance value and order. The main difference is that volatile acidity is not the main feature which partly explain the 0.17 gap of F1-score.

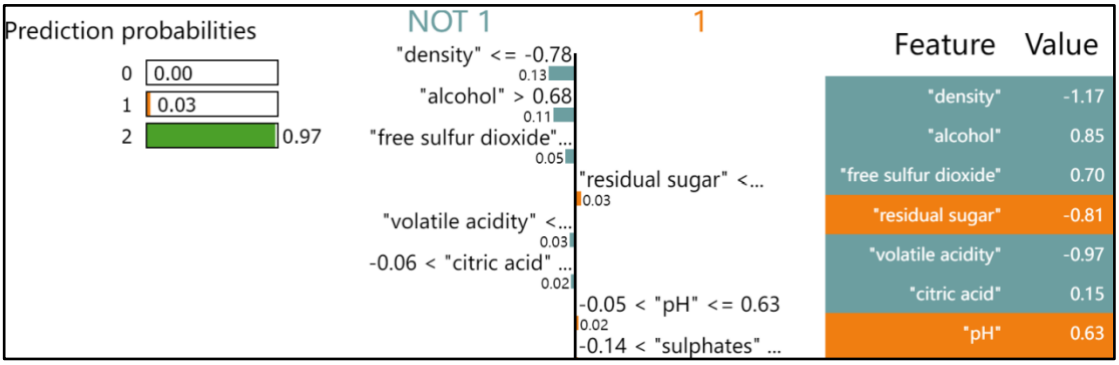

Figure 13. LIME analysis for sample 44

For sample 44, LightGBM also has a high level of confidence (97%) to classify it as high quality. All the features and values are the same as RF which explain two models have same F1-score for class 2.

Finally, we used SHAP (or Shapley Additive Explanations), another interpretability method, for the entire model (Zlaoui, 2021). SHAP is a bit different from LIME. It bases the explanations on shapely values — measures of contributions each feature has in the model.

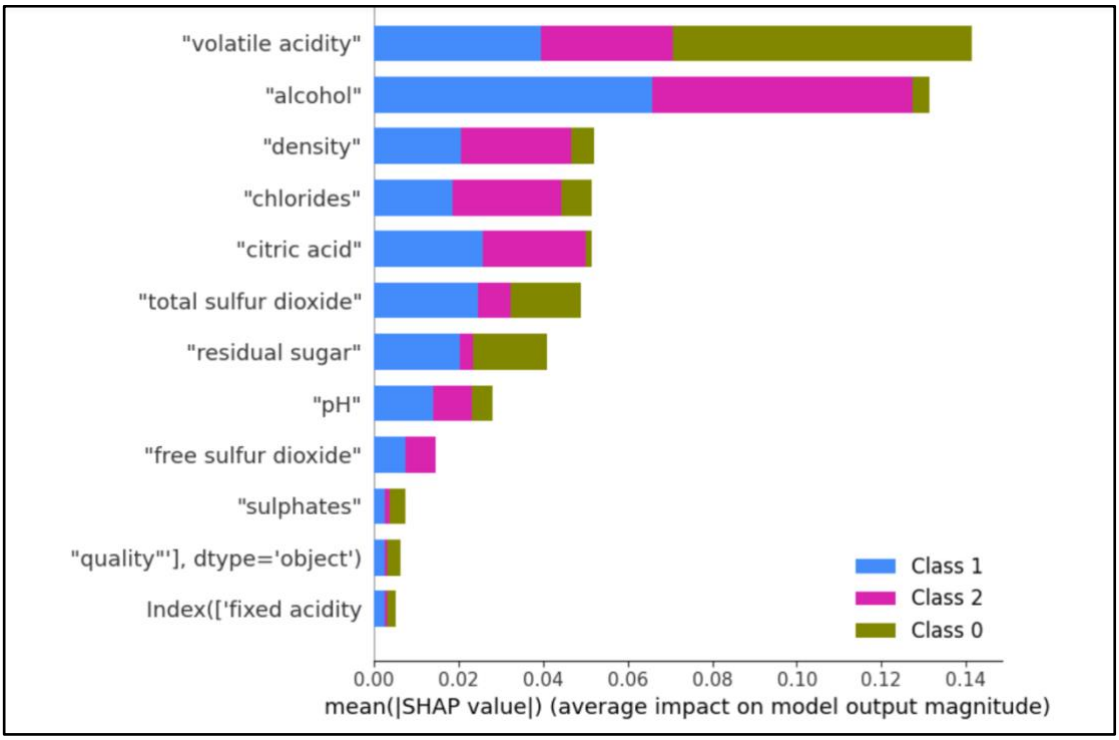

Figure 14. SHAP analysis

As shown in Figure 14, alcohol contents, volatile acidity, density, critic acid and chloride contents emerge as the 5th most important factor. Alcohol contents and critic acid improve quality of wine. This is expected considering people usually tend to like the taste of alcohol in wine. On the other hand, considering sulphur dioxide is usually found as by-product of wine, it could be interpreted that we usually want lower level of sulphur dioxide as possible. Thus, increase of level of sulphur dioxide tends to lower wine quality.

3.4. Limitations

One issue, due to the classification criteria, the three categories we obtained are not balanced. Tree based classifiers create biased trees if some classes are dominant. It is therefore it is necessary to balance the dataset prior to fitting with the decision tree. It is necessary to restore the balance on the training set by under sampling the large class or by oversampling the small class, to prevent bias from arising in the first place. The use of sampling method can generally improve the generalization ability of the model, but there is a risk of overfitting is likely, so the regularization model should be applied together. In this case, combination of oversampling (or SMOTE) method and classifier with hard constraint (like XGBoost) may enable better performance. Alternatively, we can adjust the weights of different classes when training the model with the training set.

With the application of weights, none of the models achieved accuracy more than 90%. In addition to addressing class imbalance, we can choose more models such as extra tree and CatBoost. Because we didn't reduce dimension to obtain more information, application of Extra trees is likely to obtain a higher performance than RF in the presence of noisy features. For CatBoost, categorical feature processing allows for quicker training and an overfitting detector will automatically stop model training when required.

Finally, wine classification systems are very complex, and we don't have a complete and comprehensive understanding of wine classification due to the size of datasets. We can do further analysis with other data sets. For example, vintage is an important indicator of wine quality. The year shows the climatic conditions such as light, temperature, precipitation, humidity, and wind, which further affect the growth and physicochemical indexes of grape.

We can use the climate data of different years and the physical and chemical values of wine to analyse the influence of specific climate conditions on wine quality and which features will be greatly affected.

4. Conclusion

For this project, our primary aim was to predict wine quality using machine learning techniques based on physicochemical properties of wine. There are various literatures that have studied classification models including some ensemble and boosting methods. We explored some of the models outlined in literature but also explored and assessed other ensemble and boosting models for the prediction of wine quality. We have found AdaBoost, Random Forest and LightGBM to be the most effective models for the prediction of wine quality. Limitations include limited sample size and imbalanced classification which resulted in varying performances in predicting the different classes of wine quality with a greater number of average quality wine in the dataset. Therefore, the models showed higher F1 scores in determining average quality wines. The effectiveness of predicting good quality wine was comparatively lower while predicting poor quality wine was significantly low. Of the wine quality features examined, it was determined that volatile acidity, alcohol, and density had the most significant influence on the models. In general, our results reflected the results presented in past literature and furthermore, we have found that ensemble and boosting methods are the most effective in predicting wine quality. As mentioned by the limitations of this study, for future studies, more compilation of poor and good quality wine is needed to help resolve the imbalanced classifications as well as exploring various other features that may affect the quality of wine to provide a more comprehensive model for the prediction of wine quality.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modelling wine preferences by data mining from physicochemical properties. *Decision support systems*, *47*(4), 547-553.

Mor, N. S. (2022). Wine Quality and Type Prediction from Physicochemical Properties Using Neural Networks for Machine Learning: A Free Software for Winemakers and Customers

Hoseini, C., Badar, M. A., Shahhosseini, A. M., & Kluse, C. J. (2021). A review of machine learning methods applicable to quality issues. In *Proceedings of the International Conference on Industrial Engineering and Operations Management* (pp. 1225-1240).

Kumar, S., Agrawal, K., & Mandan, N. (2020). Red wine quality prediction using machine learning techniques. In *2020 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-6). IEEE.

Dahal, K. R., Dahal, J. N., Banjade, H., & Gaire, S. (2021). Prediction of wine quality using machine learning algorithms. *Open Journal of Statistics*, *11*(2), 278-289.

Bhardwaj, P., Tiwari, P., Olejar Jr, K., Parr, W., & Kulasiri, D. (2022). A machine learning application in wine quality prediction. *Machine Learning with Applications*, *8*, 100261.

Kothawade, R. D. (2021). Wine quality prediction model using machine learning techniques.

Korstanje, J. (2021). *The F1 score*. Medium. Accessed 5 Nov 2022. https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

Kumar, Aman.[Unfold Data Science] (2022, Feb. 4) *LightGBM algorithm explained | Lightgbm vs xgboost | lightGBM regression| LightGBM model* https://www.youtube.com/watch?v=9uxWzeLglr0&t=7s

Kumar, Aman.[Unfold Data Science] (2020, Feb. 18) *Gradient Boost Machine Learning|How Gradient boost work in Machine Learning* https://www.youtube.com/watch?v=j034-r3O2Cg

Kumar, Aman.[Unfold Data Science] (2020, Feb. 11) *Boosting Explained-AdaBoost|Bagging vs Boosting|How Boosting and AdaBoost works* https://www.youtube.com/watch?v=HRBMlBiOo7Q

Kumar, Aman.[Unfold Data Science] (2020, Sep. 2) *XGBoost Overview| Overview Of XGBoost algorithm in ensemble learning* https://www.youtube.com/watch?v=FakVn1RgDms

Google. (2019). *Classification: ROC Curve and AUC | Machine Learning Crash Course*. Google Developers. Accessed 5 Nov 2022. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Lars Hulstaert. (2018). *Understanding model predictions with LIME*. Medium; Towards Data Science. Accessed 5 Nov 2022. https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b

Zlaoui, K. (2022). *Interpretable Machine Learning using SHAP — theory and applications*. Medium. Accessed 5 Nov 2022. https://towardsdatascience.com/interpretable-machine-learning-using-shap-theory-and-applications-26c12f7a7f1a