A Study on the Impact of Wine Quality Classification using Highly Imbalanced Data

Michael McAleer

Abstract There are several aspects of machine learning algorithms which may influence the performance achieved by classification models. One of aspects which may be considered a contributory factor is class imbalance, where one or more classes in a data set population is heavily represented in comparison to other minority classes. Data imbalance is a significant issue in the field of machine learning as the algorithms used assume that data is equally distributed. One method for dealing with class imbalance is re-sampling, where classes may be over- or under- sampled so there is a more even class distribution. Several methods of re-sampling were analysed against the highly imbalanced white wine data set to gauge performance of classification models on resampled data. The best combinations were found to be a random over-sampler and a combination of both over- and under-sampling called SMOTEENN which over-samples the data first using the SMOTE technique and cleans afterwards using the Edited Nearest Neighbour technique. Using a variety of pre-processing techniques, this re-sampled data was further processed, and accuracy impact analysed throughout until a favourable pre-processing sequence was found. The processed data was then fed into several classification models, where the best three models went through hyper-parameter optimisation to further fine-tune the prediction accuracy capabilities. It was found that a SMOTEENN re-sampled data set which had been standardised using a Robust Scaler provided the best processed data set to use with a k-Nearest Neighbour algorithm that performed best with 1 nearest neighbour, Euclidean distance calculations, and uniform weighting, to give an end prediction accuracy of 99.3-99.7% accuracy, an improvement from 45.51% prediction accuracy using k-Nearest Neighbour with no processing or optimisation applied.

1 Introduction

A range of Machine Learning (ML) techniques are widely used in the food industry to evaluate the quality of a given item [1] to enhance the production process and increase overall quality. One area where ML has been applied to perceive the overall quality of a product based on its constituent qualities is wine production [2]. Whilst based on empirical data, the quality of wine is not easy to

define as there are a lot of contributory characteristics which influence the perceived quality. Qualities such as alcohol volume, density, volatile acidity and chlorides can be determined by physiochemical tests, but the target classification, quality, can only be determined by sensory tests [3]. The analysis within should provide wine makes with a clearer idea as to which attributes influence the end quality of wine so that processes could be altered to improve the chances of producing wines with high quality values.

In this work, the UCI Machine Learning Repository data set 'Wine Quality Data Set' [4] has been used for all tests, the white wine data set as it is the larger of the two data sets available, consisting of 4898 samples which are ordered and not balanced. The two data sets are related to red and white variants of Portuguese wine and were made available by researchers from the University of Minho, Porto. The white wine data set Within the data only physicochemical and sensory variables are available, there is no data about grape types, brands, selling price etc. due to privacy and logistic issues [4]. Regarding the quality of each sample in the source data set, each sample was evaluated by a minimum of three sensory assessors using blind tests, these assessors then graded the wine in a scale that ranges from 0 (very bad) to 10 (excellent), the final sensory score is given by the median of these evaluations [2]. This quality variable will be the target classification value implemented in the prediction models. The 12 Physiochemical variables fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates and alcohol (a breakdown of each of these can be found in the supplied iPython Notebook Section *'Physiochemical Attribute Breakdown'*).

The 'Wine Quality' data set was chosen for this investigation into the impact of highly imbalanced data on classification tasks as it is very imbalanced in its original form (figure 1). Three of the classes in the quality target classes make up 92.58% of the overall amount of samples available in the data set, this results in models built on this data set over-fitting in favour of the large-majority classes 5, 6, and 7. Before running any tests or data exploration, it can be assumed that the target classification value 'quality' is determined not just by sensory tests but by several physiochemical attributes. Analysis of attribute correlation and feature selection also provide a more accurate classification whilst reducing the amount of computational power required.

Quality	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Sample Count	20	163	1457	2198	880	175	5
Sample %	0.4	3.3	29.75	44.86	17.97	3.57	0.1

Fig. 1 Target classification representation in the white wine data set

The programming language Python version 3.7 was used throughout the entirety of the work, with the machine library SciKit-Learn version 0.20.2 [5] used

for pre-processing and model building. For data balancing and sampling, the SciKit-Learn contributory library imbalanced-learn version 0.4.3 was used [6].

1.1 Related Work

There have been several articles published pertaining to wine quality prediction and techniques dealing with imbalanced data. This section will briefly highlight those publications.

There has been an increase in the consumption of wine with 2016 seeing close to 800 million gallons of wine produced in the US alone compared to approx. 400 million gallons produced in 1996 [7]. With this sizable increase in wine consumption, several articles have been published which investigate wine quality in terms of physiochemical data. Many of these articles have been published using very small data sets, in [8] wine classification using pattern recognition used only 42 wine samples, [9] used 33 samples for principal component analysis on wine quality according to region, and [10] using 24 samples used in 2-stage classification to try to detect undesirable fermentation behaviour. More recent work such as that of Cortez et al. [2], used a much larger data set to predict human wine taste preferences using regression techniques using a procedure that involved variable and model selection. According to Cortez, their support vector machine (SVM) *"…achieved promising results, outperforming the multiple regression and neural networks methods […] to support the oenologist wine tasting evaluations and improve wine production"*. Hu et. al [11] improved upon Cortez's work by including data imbalance steps in the data preparation stage and incorporate a range of modelling algorithms, however their work was limited by only evaluating their models on a new target reclassified from the original 'quality' target class, improving the likelihood of successful prediction. This work will vary from that of both Cortez and Hu by investigating a range of data re-balancing, models and hyper-parameter optimization (HPO) techniques against the original 'quality' target class.

In the field of data imbalance in machine learning, Hu et. al [11] whilst implementing imbalance correction techniques for the wine quality data set do so only in the form of oversampling and testing with one method. Batista et. al [12] perform several methods for balancing machine learning training data, however the use of a wide range of data sets (13) detracts from the quality which may have been achieved if attention was applied to a smaller subset of data sets. This work will vary from both that of Hu and Batista by using a range of balancing techniques, taking the best and testing them with a range of models, and from the best of those models performing HPO to get the highest possible prediction accuracy.

2 Research

For this research article the impact of data balancing techniques on the overall prediction accuracy of machine learning algorithms will be investigated. Data imbalance is a significant issue in the field of ML as the algorithms used assume that data is equally distributed. When imbalanced data is used in ML, majority classes dominate over the minority classes, causing the ML classifier to be biased towards one or more majority classes. Even though said ML algorithms may have high accuracy rates, the frequency distribution (confusion matrix) tables often show that the high prediction rate is a result of predictions of the majority class and poor classification of minority classes or even complete misclassification of minority classes as majority classes. Looking at the white wine data set statistics in figure 1, if the ML algorithm only correctly predicted wines in majority classes 5, 6 or 7, the overall prediction rate would be approx. 93%.

As part of the data balancing research several techniques will be analysed, including over-sampling, under-sampling, and combined sampling consisting of both over and under sampling. Only re-sampling techniques will be investigated here, class weights will not be included in this research. Resampling is the process of reconstructing the source data set (population) with either statistical or non-statistical estimation. Non-statistical estimation is the process of randomly drawing samples from the population with the hope the data distribution has a similar distribution to the actual population. Statistical estimation involves estimating the parameters of the population and creating subsamples from those estimations. The aim of the statistical method is to extract data samples that carry information that closely as possible match that of the actual population.

2.1 Under-Sampling

When one or more classes of data are under-represented in the data sample, under-sampling techniques may be applied to the majority classes to remove samples to even the class imbalance. This technique effectively throws away data, so it is easier to learn characteristics of the minority classes.

2.1.1 Naïve Random Under-Sampling

With naïve random under-sampling, the majority class samples are chosen at random and removed until a similar number of samples as the minority classes is reached. A downside of random under-sampling is the possibility of losing characteristics which could be important to the classifier due to excessive under-sampling and removal of samples.

2.1.2 Tomek's Links

Tomek's links is a method of under-sampling whereby links are found between samples of different classes and the technique sampling strategy decides which of the two samples to remove. A Tomek's link exists if two observations of different classes are the nearest neighbour of each other, that is, sample classes x and y such that there is no example z for which $d(x,z)<d(x,y)$ or $d(y,z)<d(x,y)$ where $d( )$ is the distance between the two samples [16].

Depending on the population used, this technique may not achieve an evenly balanced data set across all classes, it will 'clean' the data by removing 'noisy' (data with a large amount of additional meaningless information) observations which should aid in providing an easier classification problem.

2.1.3 Neighbourhood Cleaning Rule

As an alternative to Tomek's links and other under-sampling techniques using nearest-neighbour techniques, the neighbourhood cleaning rule (NCL) uses Wilson's edited nearest neighbour rule (ENN) to select majority class samples to remove from the population. But unlike the ENN technique, NCL will clean samples rather than condensing them [17]. With NCL, for each instance a in the dataset its three nearest neighbours are computed. If a is a majority class instance and is misclassified by its three nearest neighbours, then a is removed from the data set. Alternatively, if a is a minority class instance and is misclassified by its three nearest neighbours, then the majority class instances among a's neighbours are removed [18].

2.2 Over-Sampling

When one class of data is under-represented in the data sample, over-sampling techniques may be applied to duplicate those results to provide a more balanced amount of positive results in training. Over sampling is often used when the amount of data collected is insufficient. The white wine data set is a prime example, although it has 4898 samples in total, the top classes 8 and 9, and the lowest classes 3 and 4 are very under represented. Over-sampling this data set would see the minority classes have additional samples added so they are on an even standing with the majority classes.

2.2.1 Naïve Random Over-Sampling

The most naïve form of over-sampling is to randomly select the minority class and replicate the sampled observations. With random over-sampling it is important to remember that this technique will reduce the variance in the population.

2.2.2 Synthetic Minority Oversampling Technique (SMOTE)

The synthetic minority over-sampling technique (SMOTE) applies a k-NN approach where it selects k nearest neighbours, joins them and creates new samples in the links between the original samples. The algorithm generates new samples by interpolating between samples in the population.

For a given observation *Xi*, a new sample is generated by interpolating between one of the k-nearest neighbours, *Xzi*. $Xnew=Xi+\lambda(Xzi-Xi)$ where λ is a random number in the range [0,1]. This interpolation will create a sample on the line between *Xi* and *Xzi*. Figure 2 illustrates this addition of samples to the population.

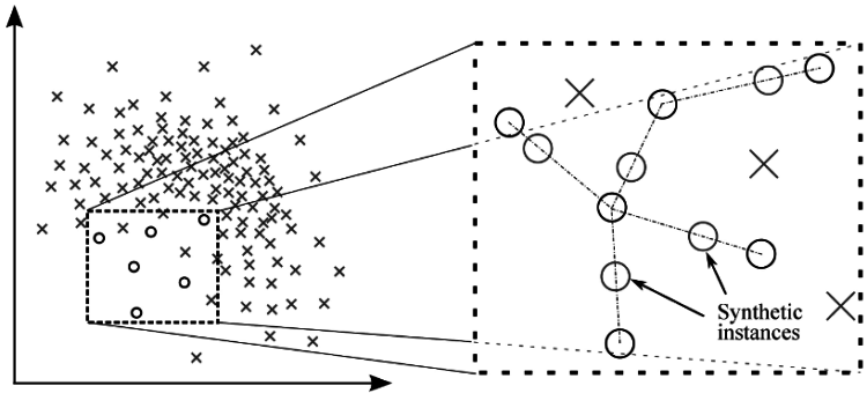

Fig. 2 Graphical illustration of SMOTE over-sampling technique [19]

2.2.3 Adaptive Synthetic (ADASYN)

Adaptive synthetic (ADASYN) over-sampling works in a similar manner as SMOTE, the difference being for ADASYN the number of samples generated for a given sample is proportional to the number of nearby samples which do not belong to the same class. This technique of over-sampling has a focus on density distribution whereas SMOTE generates the same number of samples for each minority class sample.

2.3 Combination of Over and Under Sampling

Both over and under sampling have their advantages and disadvantages, with techniques of both having favourable characteristics which would work well when combined with other techniques.

2.2.1 SMOTETomek

Tomek links can be used as either an under-sampling or cleaning method, in the case of SMOTETomek, Tomek links to the over-sampled training set as a data cleaning method. Instead of removing only the majority class exampled from the population that from Tomek links, example from both classes are removed. First the data set is over sampled using SMOTE then Tomek links are identified and removed producing a balanced data set with well-defined class clusters [12]. This method first appeared in research into improving the classification of proteins in Bioinformatics [20].

2.2.1 SMOTEENN

SMOTEENN performs resampling of data with the same idea behind it as SMOTETomek does. ENN tends to remove more samples from the population that Tomek links does, so it is expected that ENN will perform better at cleaning the data. SMOTE is applied to the data set first for over-sampling then ENN is applied after for data cleaning, removing both the noisy examples and borderline examples, providing a smoother decision surface [12].

3 Methodology

The source white wine data set consists of 4898 samples from Portugal. The classes are ordered and non-balanced, there are much more normal wines than there are excellent or poor wines. Before any processing of the data was carried out to prepare it for the modelling stage data exploration was carried out using standard Pandas functions (section 2.x in the related iPython notebook). A breakdown of all attributes and explanations for each can be found in section 2.2 '*Physiochemical Attribute Breakdown*'.

3.1 Data Exploration

The data set consists of 11 physiochemical features and 1 target class called quality. From looking at the data description in section 2.3 of the iPython notebook '*Describing the Data*', the values for these features range in size, from values in the 0-1 range up to 400+. This large variance in data will need to be addresses in the processing stage so all feature values are on a similar scale for determining accurate distance between samples.

A correlation grid was created using all attributes in the data set including the target class – section 2.4 in the iPython notebook '*Correlations Between Columns in the Raw Data*'. Looking at all the attributes, strong correlations can be found between the density of the wine and the amount of residual sugar (0.84), the total sulphur dioxide and the free sulphur dioxide (0.62), and the quality and the alcohol content (0.44). When looking at the correlation between the target class 'quality' and other attributes, the top correlations are found with alcohol (0.44), pH (0.1) and sulphates (0.05).

Before beginning any pre-processing or resampling of the data set a baseline accuracy was taken of the data set and the prediction accuracy so any future predictions would have a point of comparison (section 2.5 of the iPython notebook '*Prediction Accuracy Before Processing*'). The data set was split into random train and test subsets with an 80/20 split in favour of the training data set, and the `random_state` set to 0 so all results could be replicated time after time (this is repeated through all tests). After training the Random Forest Classifier model and the training data predicted, an accuracy score of 63.163% was achieved. To get a more in-depth look at the results, a confusion matrix was output to see the prediction frequency distribution (figure 3). The results are varied, the classes see most of predictions occur correctly but there is still a sizeable amount of incorrect predictions and the '9' wine class is not predicted, this is likely due to the significant under-representation of samples of wine in the data set with a quality of '9'.

```
Confusion matrix:
Predicted   4    5    6    7   8  All
Actual
3           1    5    3    0   0    9
4           7   23   20    1   0   51
5           5  194   91    5   0  295
6           1   64  312   29   3  409
7           0   16   66   98   3  183
8           0    5   11    9   8   33
All        14  307  503  142  14  980
```

Fig. 3. Base line accuracy of Random Forest Classifier confusion matrix

3.2 Data Pre-Processing

The white wine data set contained no null or missing values, so no action was required to resample or replace samples with null or missing values. All values for each attribute are all continuous values so no encoding of categorical data or dealing with nominal/ordinal values is required (sections 3.1 & 3.2 of iPython notebook).

As discussed in length, the data set is very imbalanced in favour of standard quality wines – those with a quality of 5, 6 or 7. To get the best possible solution to the imbalance problem, a combination of under- and over-sampling techniques were applied, along with techniques consisting of both together (section 3.3 in iPython notebook '*Class Imbalance – An In-depth Look*').

A number of outliers were detected in the data set (section 3.4 in iPython notebook '*Scaling Data & Dealing with Outliers*'). Before removing these the data needed to be either normalised or standardised. A robust scaler was used over the min/max or standard scaler as the data set contains features with many outliers. Outliers can often influence the sample mean/variance in a negative way, in such cases the media and interquartile range often give better results [21]. To deal with outliers, the clustering technique DBScan was used to perform multivariate outlier detection.

An additional target class was added to the data set in the form of a 'review' category. Wine samples were assigned a review classification from either 'poor', 'standard', or 'excellent', depending on their quality rating. The rationale for this was to determine if the classification models used could better predict a more narrower range of classes where there were more samples in each than those classed by quality. Outlier detection tests were also run against this new class to analyse the impact of the outlier removal on the new class.

The last stage in pre-processing of the data set is feature selection and dimensionality reduction (section 3.5 in iPython notebook '*Feature Selection/Dimensionality Reduction*'). Calculations were run on all the features to determine which have the most impact on the probability of calculating the correct target classification. The impact should be a reduction in the amount of processing required to predict the classification of the samples in the test data set whilst not affecting the overall prediction accuracy. A range of selection techniques were analysed including select percentile, select kBest, random forest, greedy selection, and principal component analysis.

To gauge the impact of the pre-processing steps, the best options from each of the pre-processing steps were combined to provide a new accuracy prediction that could be used for comparison with the prediction accuracy once the processed data had been run through various models and then HPO performed.

3.3 Modelling Algorithms

A wide variety of modelling algorithms were tested on the processed data which was possible due to the ease of use of the SciKit-Learn library and how the various models are implemented. A basic method to check model accuracy was built which only required for the data, target, and model to be passed in and the accuracy returned along with confusion matrix output (section 4 in the iPython notebook '*Modelling Algorithms*'). Models tested included but not exclusive to; Ada Boost, Decision Tree, Gaussian Naïve Bayes, k-Neighbours, Linear SVC, Logistic Regression, and Stochastic Gradient Decent Classifier. From the 10 models tested the top 3 were selected for HPO.

3.4 Hyper-Parameter Optimisation (HPO)

After the top three models were selected from the model testing phase, several parameters for each were selected for hyper-parameter optimisation. These parameters were not randomly chosen, SciKit-Learn documentation provides details of the most relevant parameters which may be chosen for alteration and fine-tuning [22] [23].

For each of the chosen models, three parameters were chosen for optimisation (figure 4). Like the approach taken for the model tests, a method was built so the model and parameters were passed in to it along with the grid or random search specification, the accuracy was then returned, along with the best combination of parameters and confusion matrix for a detailed look at the distribution of results.

	k-Neighbours	*Random Forest*	*Decision Tree*
Parameter 1	`n_neighbours`	`criterion`	`criterion`
Parameter 2	`weights`	`n_estimators`	`splitter`
Parameter 3	`p`	`max_features`	`max_features`

Fig. 4. Classifier parameters chose for optimisation

4 Evaluation

With a base line accuracy of 45.510% from the white wine data set using a k-nearest neighbour classifier with default parameters it was possible to achieve several improvements throughout all stages of the pre-processing, modelling, and hyper-parameter tuning phases

During the research into data imbalance and resampling techniques several findings were made, and the overall quality of the data set improved. Both over- and under-sampling had techniques which could improve the overall prediction

accuracy of a given model (kNN in this case). Using just over- and under-techniques, improvements were seen from the base of 45.5% to a range of 49% to 81.58% (figure 5). The most significant improvements were found in the combination of over- and under-sampling techniques applied together. SMOTEENN, which applies SMOTE over-sampling first then cleaning afterwards with ENN seen an accuracy of 94.294%, an improvement of 48.784% over the base accuracy. The most interesting results however were from both the random over- and under-samplers. They out-performed their counterparts for both over- and over-sampling, even though they are random in nature whereas their counterparts have specific over- or under- sampling processes in sample selection.

	Prediction Accuracy %	*Improvement %*
Base	45.510	-
Random Under-Sampler	63.216	17.706
Tomek's Links	49.880	4.37
NCL	69.3	23.7
Random Over-Sampler	81.579	36.069
SMOTE	78.135	32.625
ADASYN	77.3	31.79
SMOTEENN	94.294	48.784
SMOTETomek	79.746	34.236

Fig. 5 Results of resampling techniques applied to resolve class imbalance

Standardising the data had a positive impact on the prediction accuracy (section 3.4 in iPython notebook '*Scaling Data & Dealing with Outliers*'). Accuracy after scaling was 53.061%. The data set had a large volume of outliers when all features were analysed against the quality target class. However, when applying outlier filtering techniques on the data, the minority classes 3, 4 and 9 were removed from the training data (figure 6), so misclassification or just non-representation of classes was found in the prediction accuracy tests to gauge outlier filtration impact. Although the prediction accuracy improved after filtration to 70.753%, looking at the related confusion matrix it was evident that the improvement was because of predicting mainly majority classes, the minority classes were the outliers here even though they were not outliers as such, just under-represented when compared to the majority classes. For this reason, outlier filtration techniques were not applied to the data in the processing stage in preparation for model selection and hyper-parameter tuning.

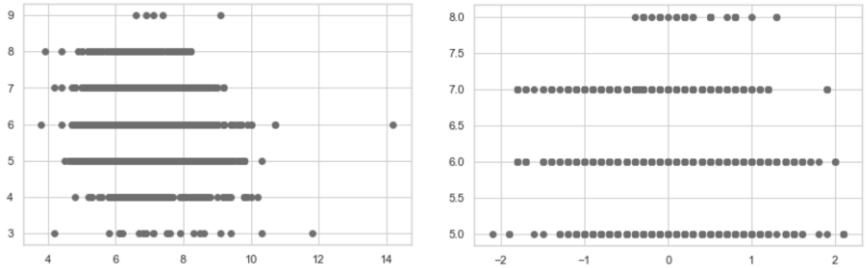

Fig. 6 Scatter plot showing fixed acidity (x) by quality (y) before and after standardisation and outlier filtration applied

A side test was run to reclassify the wines into categories 'poor', 'standard', and 'excellent' based on their wine quality rating (section 3.4.1 in iPython notebook '*Another Option*'). This technique consolidated the minority classes resulting in better predictions for both the lowest and highest rated wines. Using this new classification, prediction rates for broader wine category ratings rose to 90.510%, and with outlier filtration applied the accuracy rose again to 93.196%. This reclassification of wine quality into broader categories is a viable option for class imbalance without altering the data to a point where important feature characteristic are lost through resampling or outlier filtration algorithms.

Feature selection and dimensionality reduction techniques on the data yielded some promising results initially when run against the source data (section 3.5.4 in iPython notebook '*Feature Selection Tests with Accuracy*') with kBest and Percentile features getting the optimal features down to only 2, and Random Forest Classifier (RFC) achieving optimal tree count of 17. However, after running the RFC algorithm with the processed data a reduction in accuracy was seen. Couple this reduction with the amount of time taken to run the RFC algorithm and it is evident that feature reduction is not best suited to the processing techniques applied. For this reason, feature selection was not implemented in the modelling or hyper-parameter tuning phases.

In total ten models were tested for their ability to accurately classify the samples in the processed data set. The results were mixed across all the models, but several models stood out as excellent classifiers (figures 7 & 8). Three models stood out over multiple runs of the models and provided accuracies in the 90%+ range; Decision Tree, k-Neighbours, and Random Forest classifiers.

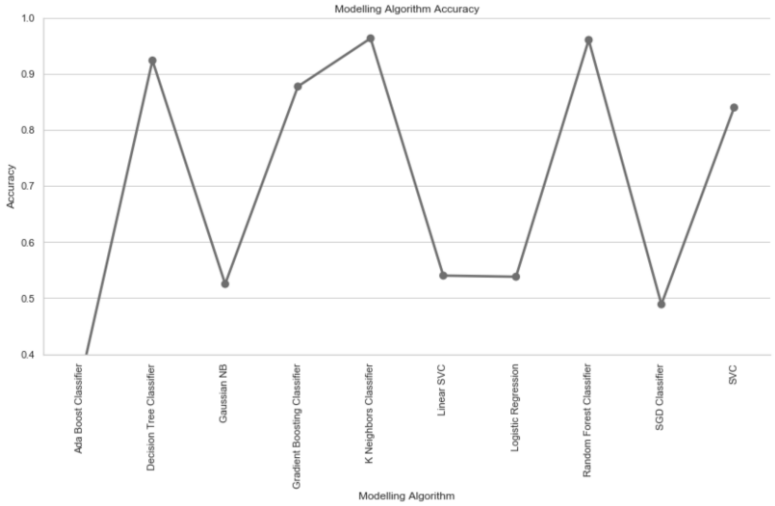

Fig. 7 Line chart depicting model prediciton accuracy using processed wine data

Modelling Algorithm Classifier	*Accuracy*
ADA Boost	0.3469
Decision Tree	0.9241
Gaussian Naïve Bayes	0.5250
Gradient Boosting	0.8777
k-Neighbours	0.9637
Linear SVC	0.5401
Logistic Regression	0.5381
Random Forest	0.9608
SGD	0.4888
SVC	0.8402

Fig. 8 Table showing model accuracy results using processed wine data

Gradient Boosting and SVC classifiers provided good accuracy results but against 90%+ rated models they were not deemed good enough to take into the HPO stage.

During the HPO phase three parameters were chosen for each of the three models to undergo optimisation (section 5 in iPython notebook '*Hyper-Parameter Optimisation*'). Both Grid Search and Random Search optimisation techniques were employed but results were the same across both as the optimal parameters were found to the same using both.

Across all models and optimisations, the results returned were very impressive across prediction accuracies, confusion matrices, and classification reports. The two forest classifiers random and decision whilst seeing improvements of up to 1% did not compare to the improvement seen by the k-Neighbours classifier

which seen a 3% accuracy increase to bring the overall prediction accuracy to 99.47% (figure 9).

Modelling Algorithm	*Accuracy*	*Optimised Accuracy*	*Improvement*
k-Neighbours	0.963718	0.994700	3.098247
Random Forest	0.960864	0.969833	0.896861
Decision Tree	0.924174	0.931512	0.733795

Fig. 9 Results of models before and after hyper-parameter optimisation

The precision, recall, and f1-score for the best performing model kNN seen results of mostly 0.99-1.0, the exception being for the wine classification quality '6'. This can be attributed to the sampling technique applied to the data set, the wine quality 6 was reduced significantly to rebalance the data set, in doing so it can be assumed that the features pertaining to this quality were adversely affected and thus affected predictions of this class type.

5 Conclusion

I have presented an analysis of white wine quality using a data set with 4898 samples using both the intended target class 'quality' and an additional target class 'review'. Impressive results were seen across both these tests for various reasons. The 'quality' target classification benefited most from the performance of the resampling techniques applied by SMOTEENN, the 'review' class benefited most from consolidation of minority classes, so classification accuracy was increased thanks to more samples in each class. Another very interesting finding from the research into resampling data is the performance of the random over- and under-samplers. These performed on par with the algorithms used specifically built for resampling, even though they are random in nature. This should be taken into consideration for future work as the time taken to randomly over- or under-sample data could be a lot less than that taken to run complex resampling algorithms on the same data for equal or marginally better results.

The additional pre-processing techniques gave mixed results both in terms of impact on prediction accuracy and computational efficiency. Whilst outlier detection may work on certain data sets, with the white wine data set it only proves to remove the minority classes and adversely affect the classes included for prediction. Feature selection both increased the amount of time involved in predictions and decreased the overall accuracy.

The models investigated highlighted the difference in ability of certain algorithms against certain data sets or classification tasks. Not all performed well but some excelled in their ability to correctly predict the target class. Taking these models into the HPO stage the accuracy was further improved to provide accuracy

results in the 99%+ range. This is even more significant as it not only beats any accuracy seen by related work, it does so without resorting to reclassification of the target class.

Although a very high prediction rate has been found for the white wine data set, it is only for one wine colour from one region. The models used in this research would benefit from testing across multiple geo-locations, wine varieties, and a better data set to start from. Pre-processing is very useful when the data set needs to be altered, but ideally it would be best to start with a data set that is as close to ready for modelling as possible. In this case, the main recommendation from the author would be to increase significantly the representation of samples in the minority classes. Increased representation of the minority classes, along with a much larger data set, would allow for more extensive work in the HPO phase to further increase the prediction rate.

References

1. Brosnan, T., Da-Wen, S. January 2006. Learning techniques used in computer vision for food quality evaluation: a review, Journal of Food Engineering, Volume 72, Issue 1, pp. 39-55
2. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J. 2009, Modelling wine preferences by data mining from physiochemical properties, Elsevier, 47(4):547-553
3. Gupta, Y. January 2018, Selection of important features and predicting wine quality using machine learning techniques, Procedia Computer Science 125:305-312
4. UCI Machine Learning Repository, 2019, Wine quality data set [online], Available at: https://archive.ics.uci.edu/ml/datasets/wine+quality [Accessed 1st January 2019]
5. SciKit-Learn Library, 2019 [online] Available at: https://scikit-learn.org/stable/ [Accessed 1st January 2019]
6. Imbalanced-Learn, 2019 [online] Available at: https://github.com/scikit-learn-contrib/imbalanced-learn [Accessed 1st January 2019]
7. Silicon Valley Bank Wine Division (SVB), 2018, US Wine Consumption, State of the Wine Industry 2018, Available at: https://www.svb.com/globalassets/library/uploadedfiles/content/trends_and_insights/reports/wine_report/svb-2018-wine-report.pdf [Accessed 1st January 2019]
8. Latorre, MJ., Garcia-Jares, C., Medina, B., Herrero, C., 1994, Pattern recognition analysis applies to classification of wines from Galicia (North-western Spain) with certified brand of origin, Journal of Agricultural and Food Chemistry, 42(7):1451–1455
9. Kallithraka, S., Arvanitoyannis, IS., Kefalas, P., El-Zajouli, A., Soufleros, E., Psarra, E., 2001, Instrumental and sensory analysis of greek wines; implementation of principal component analysis (pca) for classification according to geographical origin, Food Chemistry, 73(4):501–514.
10. Urtubia, A., Ricardo, J., Soto, A., Pszcz´olkowski, P., 2007, Using data mining techniques to predict industrial wine problem fermentations, Food Control, 18(12):1512–1517.
11. Hu, G., Xi, T., Miao, H, Mohammed, F., March 2016, Classification of Wine Quality with Imbalanced Data, Department of Computer Science, Central Michigan University, USA, School of Computer Engineering and Science, Shanghai University, China.
12. Batista, G., Prati, RC., Carolina Monard. M., 2004, A Study of the Behaviour of Several Methods for Balancing Machine Learning Training Data, Instituto de Ciências Matemáticas e de Computação, Caixa Postal 668, 13560970, Sao Carlos, Brazil.

13. Xinjian, G., Yilong, Y., Cailing, D., Gongping, Y., Guangtong, Z., 2008, On the Class Imbalance Problem, School of Computer Science and Technology, Shandong University, Jinan, 250101, China.
14. Kotsiantis, D. Kanellopoulos and P. Pintelas, 2006, Handling imbalanced datasets: A review, GESTS International Transactions on Computer Science and Engineering 30 (1), pp. 25-36.
15. Visa, Ralescu, A., 2005, Issues in Mining Imbalanced Data Sets-A Review Paper, in Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, MAICS-2005, Dayton, pp. 67-73.
16. Imbalanced-Learn: Tomek's Links Documentation, 2019 [online] Available at: https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html#tomek-s-links [Accessed 1st January 2019]
17. Imbalanced-Learn: Neighbourhood Cleaning Rule Documentation, 2019 [online] Available at: https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html#condensed-nearest-neighbors-and-derived-algorithms [Accessed 1st January 2019].
18. Laurikkala, J, April 2001, Improving Identification of Difficult Small Classes by Balancing Class Distribution, Department of Computer and Information Sciences, University of Tampere, Series of Publicaitons A, A-2001-2.
19. Data Science Central, April 2017, Handling imbalanced dataset in supervised learning using family of SMOTE algorithm [online], Available at: https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family [Accessed 1st January 2019].
20. Batista, G. E. A. P. A., Bazan, A. L., Monard, M. C., 2003, Balancing Training Data for Automated Annotation of Keywords: a Case Study. In WOB (2003), pp. 35–43.
21. SciKit-Learn: Robust Scaler Documentation, 2019 [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html [Accessed 1st January 2019].
22. SciKit-Learn: Neighbours User Guide, 2019 [online] Available at: https://scikit-learn.org/stable/modules/neighbors.html#neighbors [Accessed 1st January 2019].
23. SciKit-Learn: Grid Search User Guide, 2019 [online] Available at: https://scikit-learn.org/stable/modules/neighbors.html#neighbors [Accessed 1st January 2019].