Classifying White and Red Wine Quality based on Physicochemical Features

Final Report

Hien Le, Mahdiazhiari Austian, Vera Grosskop

Supervisor: Dr. Gosia Migut

Amsterdam University College

December 20th, 2017

***Note**: All of the codes used for this report can be accessed at https://github.com/baohien97/wine-stuff/tree/master/Project*

CONTENTS

1. Background and Problem

This report outlines and specifies the process of classifying the quality of two types of wine - white and red - based on 11 of their physicochemical features. The datasets were obtained from the paper “Modeling wine preferences by data mining from physicochemical properties” (2009) by Cortez et al. In that paper, three regression techniques were applied: Support Vector Machine (SVM, or SVR), Multiple Regression, and Neural Networks. These were all used on 6 or 7 classes, which are scores from 3 to 8/9, based on the number of instances available for each class. Later in this report, it will be shown that we did not follow the same process, and instead only did classification on up to 4 classes. Another difference between Cortez et al.’s paper and this project is that while the authors used regression algorithms for a big part of their project, here 3 classification algorithms are applied throughout, namely Logistic Regression, SVM and Decision Tree/Random Forest.

Just like what Cortez et al. aimed to do in their own paper, this project would like to find out whether physicochemical features make up a good predictor of “good” wine, and how well classification algorithms can predict the quality of wine, as well as which one does it the best. The implications of this are mostly seen in marketing in the wine industry, where manufacturers and companies could actually make use of the readily available data, which are objective features extracted from their wine products, in order to find out which feature values would result in most profit.

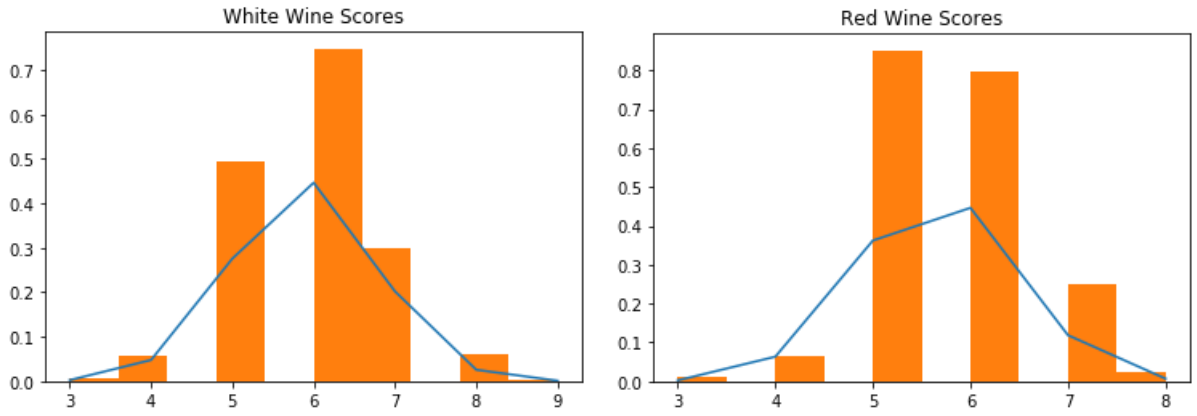
What Cortez et al. found was that the SVM model provided the best results on these 6/7 classes, with the accuracy score being 0.624 for red wine and 0.646 for white wine. Later the number of classes were reduced to 2, and their accuracies significantly improved, with 0.89 for red wine and 0.868 for white wine. These points will also be discussed in more details later in this report.

2. Analysis of datasets

The datasets consist of 4898 white wine and 1599 red wine instances, all of which are rated on a scale of 0 to 10, with 10 being the best quality. Each instance is scored on 11 features, namely fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol. Potentially, this can be a 10-class classification problem, however, during the preprocessing of data, it

was noticed that there was a lack of instances for a number of classes. This can be seen from the distribution of classes on the following 2 plots.

Figure 2.1. Distributions of White and Red Wine Scores

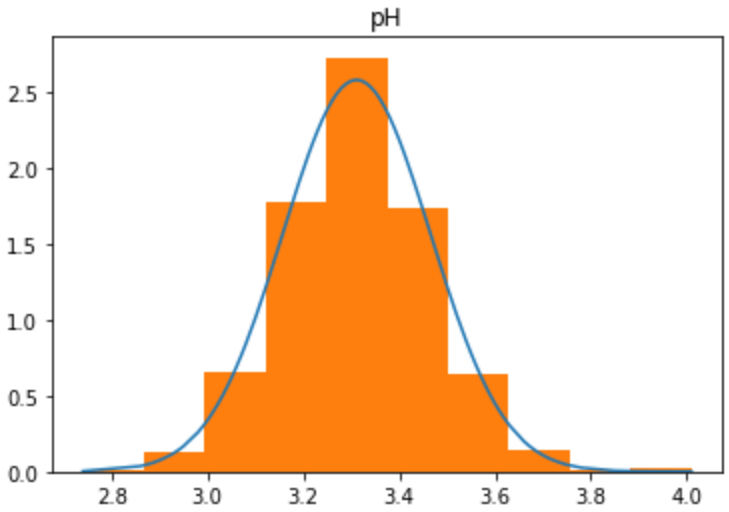


What can be deduced from these two plots is that both distributions are very skewed: classes 0, 1, 2 and 10 are completely absent from both datasets, while class 5 and 6 apparently prevail. This means that a 10-class classification will not be able to predict most of the classes, and hence will perform poorly on the test set.

Due to this lack of information, we decided to relabel the datasets in three different ways. First, the classes were split into 2 labels: 0 and 1, with 0 signifying wines that score under 6 and 1 denoting the rest, in other words, 0 denotes “bad wine” and 1 “good wine”. Second, the data’s labels were transformed into 3 classes: “good” (for those with scores > 6), “medium” (score 5 or 6), and “bad” (4 or lower). And last but not least, 4 classes were considered: “good” (> 6), “medium high” (= 6), “medium low” (= 5) and “bad” (< 5). Classifications were then applied on both datasets on all 3 types of label transformation.

With regards to features, plots from both datasets of white and red wine demonstrated that almost all features are normally distributed, especially pH, total sulfur dioxide, fixed acidity, density and volatile acidity. Figure 2.2 displays the distribution of pH in red wine as an example.

Figure 2.2. Distribution of pH in red wine dataset



Since the number of features is high, what could be done is feature selection. For this, we tried out the *sklearn*'s feature_selection module. There are various ways to make use of this module, one of which is to remove features that have low variances. This reduced the number of features in both datasets to 5. However, it was found that fitting the classifiers on this new training set did not create any improvement.

3. Applied Algorithms/Methodology

The current project fit 3 classification algorithms to the two datasets. The first one was Logistic Regression, due to it being the most primary algorithm in classification and one that can be applied to almost all datasets (data do not have to be normally distributed). Another advantage of Logistic Regression is that its parameters, such as iterations and regularisation parameter lambda, can be easily tuned.

The second one was SVM, as this was the option that generated the best results in the paper by Cortez et al. Applying SVM in this project would produce a nice comparison with the results found in the paper. Moreover, as a large-margin classification approach, SVM is often expected to be more accurate and hence perform better than Logistic Regression.

The last algorithm that we used for this project was Decision Tree and Random Forest. Due to a large amount of continuous feature values in the datasets, it would be interesting to see how well the *sklearn*'s Decision Tree/Random Forest modules classify when having to set multiple thresholds and then arrive at the right label. It was nevertheless expected that Random Forest would perform well, given how the algorithm

works: growing different trees with different attributes on each subset of the datasets at training time, thus avoiding overfitting at testing.

4. Results

4.1. Logistic Regression

For logistic regressions the most valuable parameter to tune was the regularization parameter (or it's inverse more precisely), because it remarkably improved the accuracy of the classifier. In addition to the regularization parameter the solver for linear regression was tuned. This proved efficient, since logistic regression performs differently on multiclass labels compared to binary labels. Because we decided not to scale our data 'sag' and 'saga' were not given as a solver to tune. Surprisingly the tuning resulted in 'liblinear' as the solver for the 4-class logistic regression for red wine, although it specifically says in the *sklearn* documentation that for multiclass problems only the solvers 'lbfgs' and 'newton-cg' should be used. The tolerance parameter was also tested, but only slightly different accuracy scores were received. However, the tuning took increasingly more time and therefore it was decided to not tune on the tolerance.

Table 4.1.1. Results of parameter tuning for Logistic Regression

	White Wine		**Red Wine**	
# Classes	**C**	**solver**	**C**	**solver**
2	1.00	liblinear	0.384	lbfgs
3	0.810	newton-cg	0.289	newton-cg
4	0.953	newton-cg	0.811	liblinear

The above parameters were used to receive the following results for the classification. For the 3-class classification for both the red and white wine the code gives an undefined metric warning, which means that one of the labels has not been predicted, resulting in a division by 0. The labels in the classification have been distributed in such a way that all labels exist in the dataset, so this means that all instances in a certain class have been wrongly classified. The C parameter was tuned in different ranges, but still resulted in the same issue. Due to computing power constraints other additional parameters were not tested.

To illustrate the wrongly classified classes the confusion matrix of all datasets were computed. As can be seen in the table below for both the three and four class classification there is a class with 0 correctly predicted labels, which is strange because only 3-class classification results in the error message.

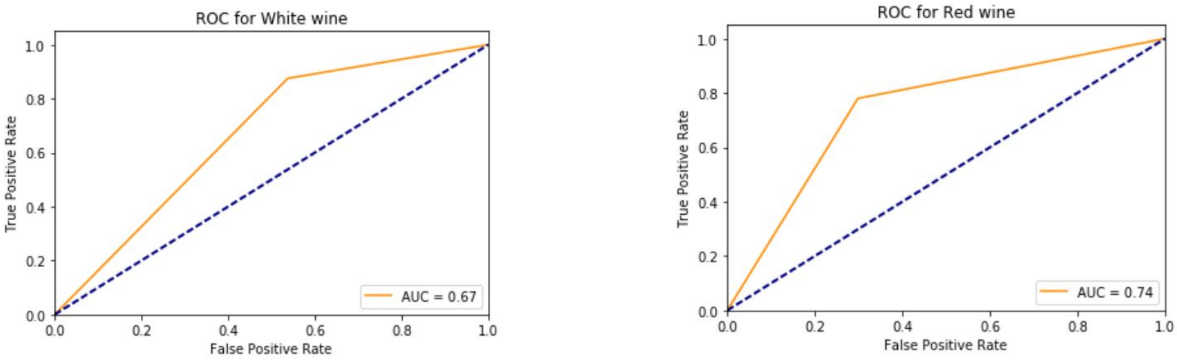
Table 4.1.2. Confusion Matrix for Logistic Regression

	White Wine	**Red Wine**
2 classes	**168** 232 85 **740**	**140** 51 57 **152**
3 classes	**0** 3 38 0 **68** 186 0 42 **888**	**0** 0 16 0 **10** 43 0 3 **328**
4 classes	**0** 3 15 27 0 **106** 189 7 0 73 **388** 84 0 5 163 **165**	**0** 0 7 11 0 **19** 39 2 0 8 **106** 52 0 1 55 **100**

Table 4.1.3. Results of classification with Logistic Regression

	White Wine			**Red Wine**		
classes	**accuracy**	**precision**	**recall**	**accuracy**	**precision**	**recall**
2	0.740	0.73	0.74	0.710	0.74	0.74
3	0.781	0.73	0.78	0.800	0.78	0.84
4	0.545	0.54	0.55	0.593	0.53	0.56

Figure 4.1.1. ROC curves for Logistic Regression



4.2. SVM

One of the initial considerations in an SVM classifier is the choice of kernel. It is found that the RBF (Radial Basis Function) kernel worked the best for this problem. There are a many parameters for an SVM classifier. However, for the RBF kernel specifically, we chose C and gamma as the parameters to tune. One of the reasons for this was computing power constraints. It took more than 40 minutes to tune both C and kernel parameters with GridSearchCV and this process bugged out our computer. Another reason is that many of the advanced parameters were made for other kernels (such as coef0 and degree only for poly kernels).

Table 4.2.1. Tuning parameters for SVM

	White Wine		**Red Wine**	
# Classes	**gamma**	**C**	**gamma**	**C**
2	0.1	1	0.001	100
3	0.1	1	0.1	0.5
4	0.1	5	0.0015	100

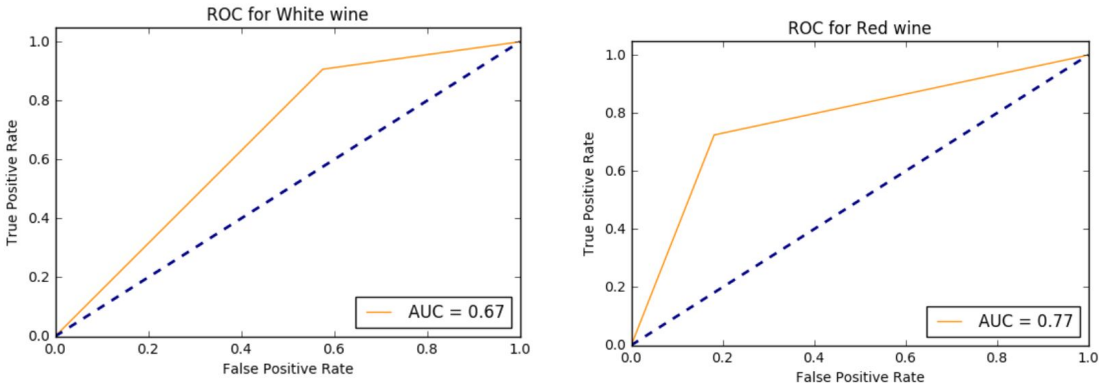
Initially, we tried to tune the parameters with a set of exponentially growing sequence of C and gamma. However, the computer was left for hours but the process still was not finished. Therefore, we aborted this idea and decided to go with a smaller set of C and gamma. Borrowing from values After customising our SVM with these parameters, we tested their performance on our test set and recorded our results in the following table.

Table 4.2.2. Results of Classification for SVM

	White Wine			Red Wine		
# Classes	**accuracy**	**precision**	**recall**	**accuracy**	**precision**	**recall**
2	0.745	0.74	0.75	0.77	0.77	0.77
3	0.789	0.78	0.79	0.847	0.84	0.85
4	0.619	0.62	0.62	0.58	0.55	0.58

Similar to Logistic Regression, for our SVM classifier, we encountered UndefinedMetricWarning, but only for the 3-class red wine and 4-class red wine. The ROC curves for red and white wines are as follows:

Figure 4.2.1 ROC Curves for SVM



4.3. Decision Tree and Random Forest Classifier

With a set of datasets that consisted of many continuous feature values, the Decision Tree algorithm was applied in the hope of it being able to find the appropriate thresholds for each of the 11 attributes. Since Decision Tree and Random Forest are essentially the same algorithm, they are customised in the same parameters, namely “max_features”, which, in this project, is tuned with 4 options ‘sqrt’, ‘log2’, ‘auto’ and None; and “criterion”, which specifies the function to measure the split quality and has 2 options ‘gini’ and ‘entropy’. The best parameters found using the GridSearchCV module of *sklearn* are as followed:

Table 4.3.1. Results of parameter tuning for Decision Tree and Random Forest

		2 classes		3 classes		4 classes	
		max_features	**criterion**	**max_features**	**criterion**	**max_features**	**criterion**
Decision Tree	White	log2	entropy	none	entropy	none	gini
	Red	sqrt	entropy	auto	gini	none	gini
Random Forest	White	auto	entropy	auto	entropy	sqrt	gini
	Red	auto	entropy	sqrt	gini	log2	gini

Both classifiers were then customised with these tuned values and fit to the training sets. Their performances were then demonstrated on the test sets and recorded on a classification report. The following table gives an interpretation of the report.

Table 4.3.2. Results of Decision Tree and Random Forest Classification

		2 classes			3 classes			4 classes		
		a	**p**	**r**	**a**	**p**	**r**	**a**	**p**	**r**
Decision Tree	White	0.786	0.79	0.79	0.779	0.77	0.78	0.609	0.60	0.61
	Red	0.763	0.76	0.76	0.795	0.81	0.80	0.605	0.61	0.60
Random Forest	White	0.812	0.81	0.81	0.805	0.80	0.80	0.602	0.60	0.60
	Red	0.780	0.78	0.78	0.858	0.85	0.86	0.695	0.68	0.69

a: accuracy, p: precision rate, r: recall rate

Figure 4.3.1 demonstrates the performances of both classifiers when the labels are binarily through the ROC curves and their AUC values.

Figure 4.3.1. ROC curves (for 2-class classification) of both classifiers

Figure 4.3.1.1. Decision Tree Classifier

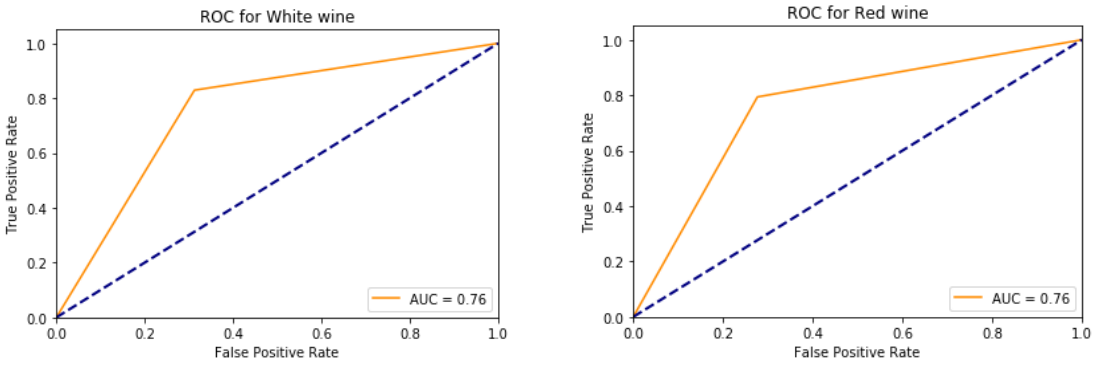
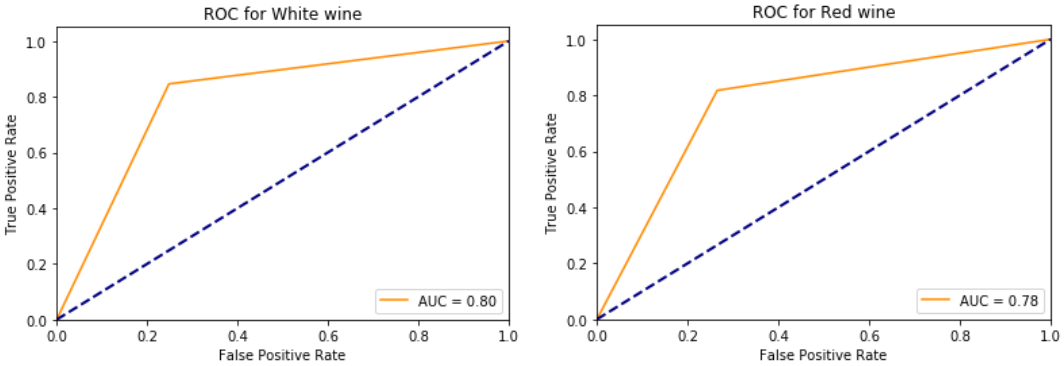


Figure 4.3.1.2. Random Forest Classifier



Looking at these results, one can see that in general, Random Forest performs better, and the best results are obtained when the labels are split into 2 and 3 classes. While the maximum accuracy score that could ever be achieved with the Decision Tree classifier was around 0.79 for white wine and when the targets were split into 2 classes “good” and “bad” (or in the code, 1 and 0). The best that Random Forest could get was a little above 0.80 for white wine, also when the targets were binarily labelled. These findings are understandable and reasonable, due to a number of reasons.

First of all, it makes sense that Random Forest would give a better performance, due to the fact that the algorithm, as mentioned above, can very effectively avoid overfitting. Meanwhile, it is also not surprising that the 4-class labelling does not give good results, as again the number of instances were not enough for each class. However, the fact that 3-class labelling observed some high scores indicates that the number of instances was sufficient for at most 3 classes.

5. Discussion and Evaluation

5.1. Evaluation of Classifiers

The results of the three classification algorithms used in this project convey that their performances are comparable across the 2 datasets of white and red wine, as well as across the three types of label transformation (2-class, 3-class and 4-class).

The best accuracy score that Logistic Regression observed was 0.8 for red wine when the labels are split into 3 classes. For SVM, it was also in the same dataset and with the same label split that it got its highest score 0.847. Unsurprisingly, the pattern was repeated in both Decision Tree and Random Forest, where they got 0.795 and 0.858, respectively. This concludes that the red wine dataset is best suited with a 3-class classification problem. The same statement can also be made for the white wine dataset, however, Decision Tree and Random Forest actually performed best when this dataset was binarily labelled.

Looking at these best accuracy scores, one can also claim that Random Forest performed the best. However, the difference between the highest score of SVM and that of Random Forest is minute, while Logistic Regression's is not too far behind, either.

What is worth noting is the fact that even though the red wine dataset has over about 3000 fewer training instances than that of white wine, most of the time, the higher scores were observed in this dataset. This suggests that even though the amount of data for the other dataset was abundant, its lack of training examples for a number of classes was rather influential. Nevertheless, the ROC curve for the Random Forest classifier in the white wine dataset indicate a rather impressive AUC (0.80), suggesting its compatibility with the dataset as well as its robustness even when having to deal with a large number of continuous figures.

5.2. Comparisons with the paper's findings

Cortez et al. (2009) found in their project that SVM performed the best out of the three regression techniques that they applied (Multiple Regression, SVM and Neural Network), while this algorithm only came second in this project. One explanation is that while Cortez et al. treated this problem as more or less a regression problem (with initially 6 or 7 "classes"), the present project looked on it as an entirely classification

problem. Moreover, it also uses two other algorithms that are different from those in the paper.

However, when reducing the the problem to a classification problem with only 2 classes, Cortez et al.'s SVM achieved significantly higher accuracy scores for both wine datasets: 0.868 for white and 0.89 for red wine. There are a number of factors behind this difference, one of which lies in the way the authors transformed the wines' score labels. They used the tolerance values *T* in order to round the regression response into the nearest classes, instead of deterministically grouping certain scores into one class like what was done in this project. Another explanation for the superior performance of the authors' SVM could potentially be the fact that they used a different environment that is more suited for data analysis, in this case R - a very robust environment for machine learning algorithms. This is not to say that the IPython Jupyter Notebook environment and its *sklearn* library are not suitable, yet during the implementation process, we did come across a number of technical problems (as outlined in the section 4 of this report), a large number of which are in the tuning procedure (using *sklearn*'s GridSearchCV) which could take a substantial amount of time and still may not provide the optimal parameters, with 5 or 10-fold cross validation.

5.3. Problem of *UndefinedMetricWarning*

This warning is a result of lack of training data. As we noted above, there is a lack of instances for a number of classes. During the training phase, the dataset is split multiple times, if there are too few training examples, then the resulting split might not contain any examples representing these classes. Initially, we thought that this problem resulted from parameter values that are not good. However, after further research, the only way to handle this warning is to increase the number of training samples. As *UndefinedMetric* is only a warning, training will still be successful, however, the resulting model's predictions might be weak on the classes where we lack training data.

6. Future Implications

Based on the results obtained from the current project, there are a number of lessons and remarks to be made so as to improve the implementations in the future.

If this project were to be replicated, it would be wise to try out more algorithms, especially regression algorithms, next to classification ones. However, to save time and guarantee efficiency, a step to be taken before this is to gather more training instances

for each score value ranging from 0 to 10, as at the moment most of the instances have the score of 5, 6 or 7. Alternatively, better methods to transform the data labels could be applied, such as using the tolerance value *T* to round the targets to the nearest class.

Furthermore, since this project encountered a number of technical problems with the *sklearn* library as well as the IPython Jupyter Notebook, in terms of runtime during tuning and metric errors, it may be more feasible to work with a different environment, such as the one used in the original paper - R.

References

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