APRENENTATGE AUTOMÀTIC 1

Wine type and quality determination

Aleix Torres i Camps, Luís Sierra Muntané, Àlex Batlle Casellas

Data Science and Engineering, UPC Spring 2020

Contents

1 Description and goals of the project.

Our goal with this project is that of obtaining a data-driven understanding of the relationships between the chemical characteristics of wine and features such as its colour or quality. The data in question is this data set featuring Portuguese wines of the *Vinho Verde* region and claims to have 4898 instances of wines, even though after looking at the dataset we saw there actually were 6497, and 12 attributes as physicochemical properties of the wine: `fixed acidity`, `volatile acidity`, `citric acid`, `residual sugar`, `chlorides`, `free sulfur dioxide`, `total sulfur dioxide`, `density`, `pH`, `sulphates`, `alcohol`, and `quality` (score between 0 and 10) which is the output variable based on sensory data. Furthermore, we consider whether the wine is red or white, given that there are two datasets, one for each colour of wine.

As such, the **main objective** will be that of classifying the wines into their corresponding colour class as white or red from their physicochemical properties, but also to determine their quality on a scale of 0 to 10 as subjectively decided by a panel of experts from their sensory perceptions of the wine. As such, we will want to see whether there are variables which can strongly predict the final quality of a wine or whether such variables are even relevant in determining such a metric. This is especially relevant since the variables, which are all real numbers, are usually associated with being important qualities for the wine and its final taste. All in all, the variable representing the colour of the wine will be predicted from the 11 explanatory variables and the variable representing the quality of the wine will be predicted using the 11 explanatory variables and the colour of the wine as well. It remains to be seen whether they will be good to predict the final factor variables of colour and quality, and to discern which variables yield better models for the prediction of the target variables.

2 Related previous work.

This data set has been previously used in various projects, as wine appears to be a frequent topic in examples of classification problems in ML. The first one we cite here is the one provided as a relevant paper, Modeling wine preferences by data mining from physicochemical properties. This paper, as it states in its abstract, proposes a data mining approach to predict human taste preferences on wine, through three different types of regression model: multiple regression, neural networks and support vector machines. This goal is similar to ours, but uses models we cannot comment on for the moment, as they have not been explained in class yet.

Apart from the data set webpage, we have looked for other projects that cite and use this data set. Searching through Kaggle we have found some different kernels operating on this data. Some of these use decision tree-based methods and support vector machines so we are not commenting on them either. Even though, they also use linear and generalized linear models to preliminary test variable significance.

The first kernel only uses white wine data. One thing to remark about most of the models used on this kernel is that they binarize the quality of the wines into "good" and "not good" and then use the previous stated models and logistic regression.

In the second kernel, it is remarkable the various ways in which the author tries to gain insight on the data. Initially they use k-Means to cluster data, and then based on the results of this method they visualize the data using a Radial Visualization (RadViz) plot, which we can map into our knowledge as being similar to a biplot. In this way they can observe if there exist any clear tendencies as to variable influence in wine quality. They also pay special attention to anomalies throughout the notebook.

The third kernel we are commenting on is an exploratory data analysis into the data. In this work it is remarkable the univariate distribution exploration, and especially the fact that this is done separately with white and red wine. This indicates that these two variants might need different treatment, which makes our second task (classification into red or white wine) pertinent in the combined data set.

We found some additional data exploration notebooks: namely, this one performs some comparative plots between variables and their correlations, and this other one also investigates univariate, bivariate and multivariate relationships between variables.

3 Data exploration.

First of all, the data set is given into two different tables, one for red and the other for white wine. Then we decided to perform the union and **obtain an unique data set**. After that, we extracted the two target vectors. The first one, *wine.type*, determines if the wine is red or white and the second one, *wine.quality*, is the quality value given by the oenologist.

After some exploration, we detected something that we did not expect: some rows have the exact same values in all of the variables, so we concluded that there are **duplicated rows** in the data set. So we went back and decided to delete the duplicated ones and leave just one of them. But a major problem was found, 2 rows were at the same time in the two original data sets. In this situation we decided, for now, to delete the rows from both tables, as we do not know whether they belong to red or to white wines, and we still have a lot of other rows.

Now, we were interested in the **dimensions of the data set**. So we observed there are 5318 rows and 11 input variables. From these rows, 1357 are red wines and 3959 white. And the table of wine quality with respect to the numbers of rows with that value is:

Table 1: Number of wines of each quality

wine quality	3	4	5	6	7	8	9
num. of wines	30	206	1751	2323	855	148	5

Then, from these information we may extract some conclusions. The first one is that not all possible marks ([0,10]) were given. And more important, **the data set is unbalanced** with respect to wine type and wine quality. There are three times white wines as red ones and most of the marks are 6's or 5's, followed by 7's, while the other ones have a very small amount.

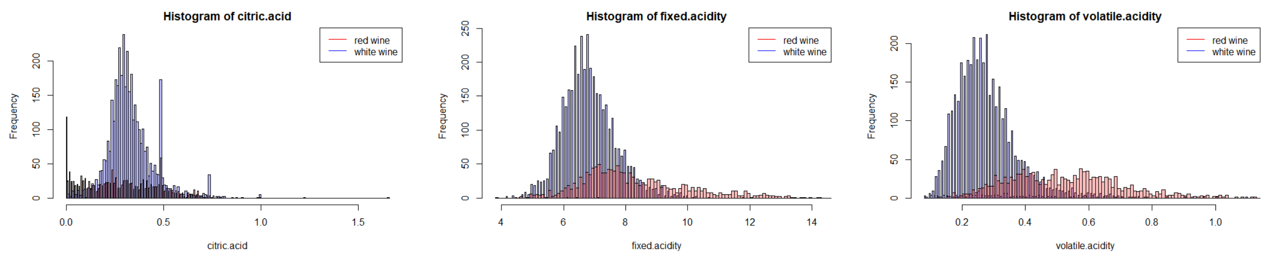
The next thing we did is caring about **missing values**. After some exploration using *summary* and some *histograms*, the conclusion is that there are no missing values. Although a variable was suspicious, *citric.acid*, because it has a very large amount of 0 values, most of them coming from the white wines. We investigated a little bit on the topic and found that it is a supplement used as a natural preservative or a flavour potential, so we may assume that some wines have no citric acid at all.

In the review of the data, some **outliers** were found. Such as some wine with 1.66 in *citric.acid*, while the 3rd quartile of this variable is 0.4, or another with *residual.sugar* 65.8, while its 3rd quartile is 7.5. For the moment we decided to keep them but we may delete them in the future.

So then, we perform a **particular analysis of each variable**. We saw that all the variables of the new data set are numerical, therefore we can perform computations over them. But two variables seem to be discrete (*free.sulfur.dioxide* and *total.sulfur.dioxide*). One of the target variables is also discrete and the other is a factor of two elements.

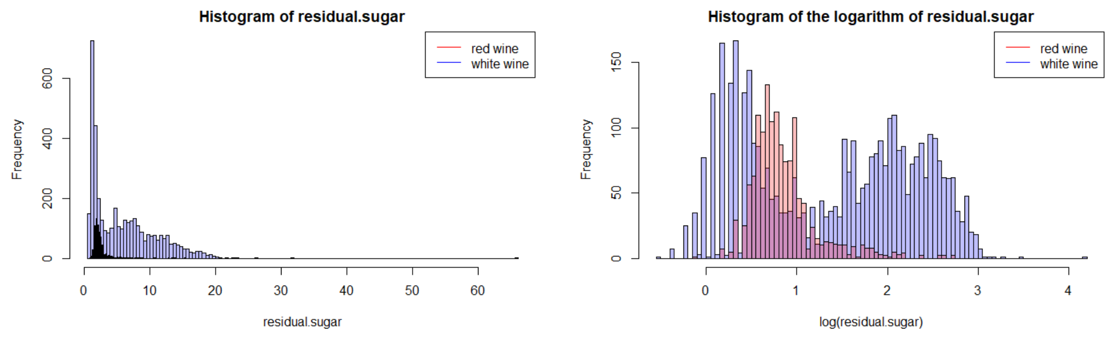
We start by plotting the **histograms**, with different colors for each wine type. We detect that for *fixed.acidity*, *volatile.acidity* and *citric.acid*, the two wine types have different behaviours: the white type is more like a normal distribution, while the red is flatter and in some cases looks more like a uniform distribution. These variables have in common that all of them are related to acidity so maybe in the future we might join them and create a unique variable.

Figure 1: Histograms of 3 variables



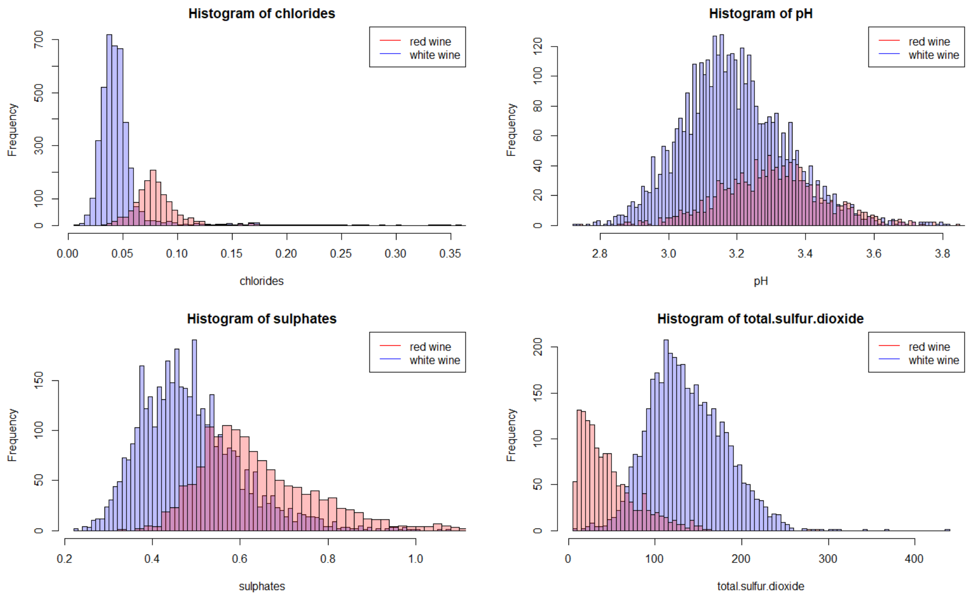
The second thing we detected is that the variable *residual.sugar* has a lot of small entries, so we considered applying a logarithm and seeing the result. There is a clear improvement, and something interesting was discovered. The red wine can be modeled as a unique normal distribution while the white wine as two of them, as there are two density regions with a separation between them. So we can treat the logarithm of the *residual.sugar* as a useful **feature**.

Figure 2: Histograms of *residual.sugar* and its logarithm



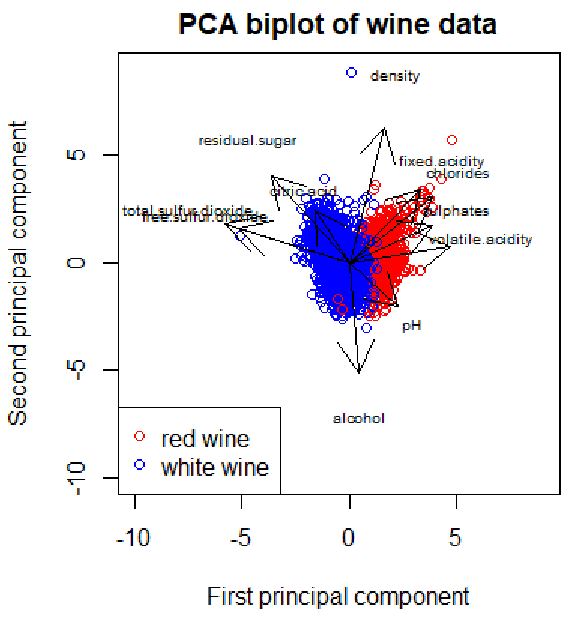
Finally, the other variables *histograms* have a **similar shape** in both wine types, but usually with **different means**. This fact could be useful in order to perform *classification*. And it will be relevant as well for the *regression model* to differentiate the *wine.type* as factors. Here are some of the other *histograms*:

Figure 3: Histograms of 4 variables



The next thing we considered was performing a **PCA**, because it can help us visualize the data and start seeing which variables are more relevant to our goal. We made the following *biplot*, where the first two components explain 50% of the variability.

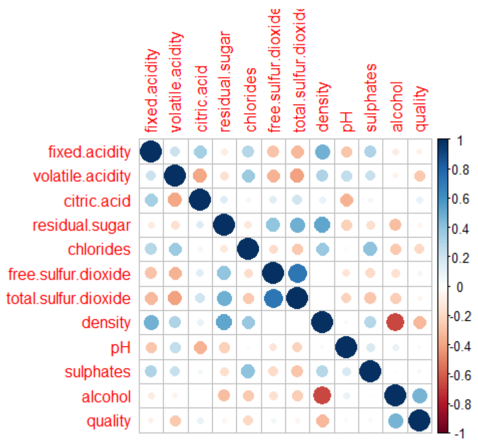
Figure 4: PCA biplot



As the two groups seem to be separated in the horizontal edge, it seems that the **density and alcohol variables explain very little about the difference between the groups**. However, variables such as *total.sulfur.dioxide* or *volatile.acidity* do. In the PCA we can also see that **some variables are correlated**, for example, *total.sulfur.dioxide* and *free.sulfur.dioxide* in positive or *density* and *alcohol* in negative. There is a last thing to observe here, the **outliers**. There are points that are not only far from the others on one variable, but they are so in multiple of them. We will consider whether to remove them or not.

The next thing we can do is a **bivariate analysis**. We have already seen that some variables may have correlation among them, thus, it is interesting to do the *correlation matrix*. The following graphic is a representation of it using colored circles.

Figure 5: Correlation matrix



We have added the *quality* variable, as it can help us have an intuition about which input variables could be useful to predict it. As it can be seen, only few of them have a linear correlation with the target variable. Other variables present mutual correlation, as we suspected.

4 Resampling protocol & Modeling methods considered.

We have separated our data in the following way: we have a **training set** which has $\frac{2}{3}$ of randomly chosen samples from the whole data set. We will use this training set for model selection. The remaining $\frac{1}{3}$ of the data goes to the **testing set**, which we will not use until we want to assess our selected models' performance on new data.

As we have a large dataset, we can allow ourselves to be computer-friendly and use **K-fold Cross-Validation** to perform model selection on the training set (we will use $K = 10$ throughout all of the work). So, as usual, this involves separating the data into K different subsamples and perform K experiments with each model, fitting it to $K - 1$ folds and calculating its **validation error** when predicting for the held-out fold. Then, these K errors will be averaged and we will have a pretty robust estimation for the error of the model, the **cross-validation (CV) error**. Out of all the models we train, we will select the ones with less CV-error, and assess them based on their prediction error on new data.

As well as we have two problems, we will consider two separate approaches:

- To predict the type (red or white), we first considered to apply discriminant methods such as **LDA**, **QDA** and **RDA**, as this is a binary classification problem. Also, we considered to apply **kNN** (with original, scaled and LD-projected data), and **Logistic Regression**. We also consider using a **MLP** and finally also a **Random Forest** model.
- To predict the quality (0 to 10), we have two different approaches:
  - The first one, to perform multi-class classification. This can be done via **LDA/QDA/RDA** as in the type classification, or using **kNN**.
  - The second one would be to perform regression and then round the results to obtain a prediction. This first approach could be done with **Linear Regression** (regularized with both the LASSO L^1 and the Ridge L^2 methods), **MLP** and **RBF-NN** (with just one neuron in the output layer), and **Random Forest** with regression trees.

5 Obtained results.

We have used the original data set without any extracted variables, unless explicitly said so in some sections. We essentially separated our models problem-wise, meaning we have some models for wine type classification and some others for wine quality, for both regression and classification. First of all, we will explain the results for type classification:

5.1 Wine type prediction

This problem is modeled as a binary classification. In order to classify the observations, we have used the following methods:

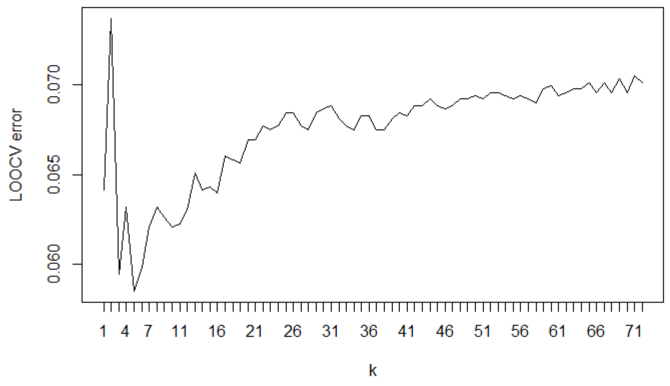
Discriminant Analysis: We first of all consider Regularized Discriminant Analysis, as it is the most general method and may help discard LDA or QDA. Applying it, we reached the conclusion that we should directly use LDA, because RDA regularization parameters are $\lambda = 1$ and γ very close to zero. Now on to LDA. Using K-fold CV, with 13 misclassified observations, we reach a cross-validation error of around 0.37%.

Table 2: Confusion matrix of LDA

	Pred as White	Pred as Red
Real White	889	6
Real Red	7	2642

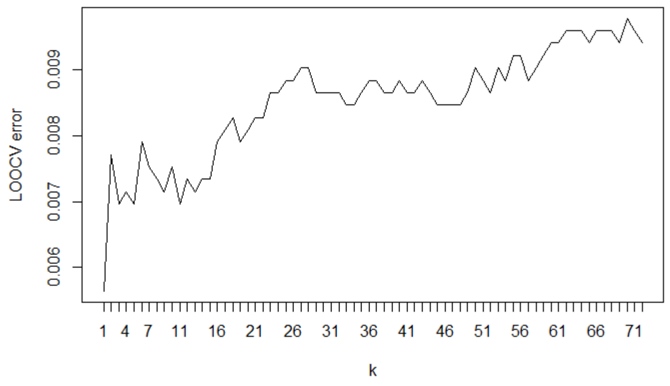
k **Nearest Neighbors:** Now, on to kNN; as we know that kNN is very sensitive to the scale of the data, we considered three ways to apply this method. The first one, to use the training set with its original scale. The second one, to use scale dataset. And the third one, to use data projected on to LD space. To see if these changes in the dataset affect the predictions, we considered appropriate to plot the LOOCV error with respect to the first few values of *k*.

Figure 6: kNN LOOCV error



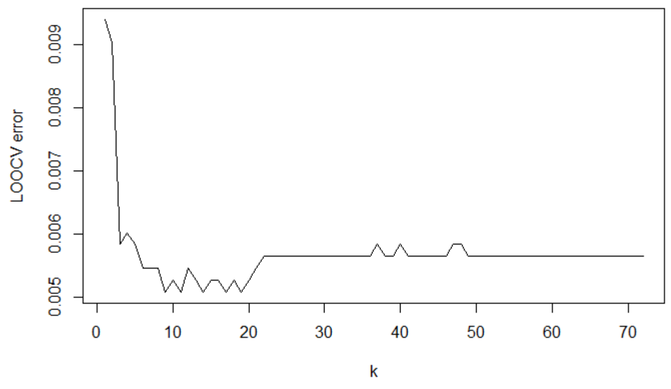
In this first approach, we see that the LOOCV decrease quickly and then increases slowly. The minimum error is at $k = 5$, with a LOOCV error of roughly 5.8%. If we scale the data, we can see a clear improvement:

Figure 7: kNN LOOCV error using scaled data



Now the plot presents a minimum at $k = 3$ with a LOOCV error of around 0.59%, so a major improvement have been made. We went further and used kNN with data projected onto LDA space, and the LOOCV error results were the following:

Figure 8: kNN LOOCV error using LDA projected data



Now the plot is more constant to small values, in particular, the minimum is at $k = 8$, with a LOOCV error of 0.34%. So, for the minimum LOOCV error up until now, we would use this method. Now, we proceed to calculate the cross-validation error from 1 to 25 neighbors. There are several minimums, but the first of them is at $k = 5$. As we want to have the simplest model, we will use these value instead of any of the larger ones. This error evaluates to around 0.34% (12 misclassified). The confusion matrix is the following:

Table 3: Confusion matrix of kNN

	Pred as White	Pred as Red
Real White	890	5
Real Red	7	2642

Logistic Regression: Now we considered a logistic regression model, so a GLM with binomial distribution and logit link. This model gives probabilities of each class, so we had to add a decision boundary for the predicted probability (0.5). In this case the cross-validation error is 0.42% (15 misclassified), and the confusion matrix is the following:

Table 4: Confusion matrix of Logistic Regression

	Pred as White	Pred as Red
Real White	889	6
Real Red	9	2640

Multi-Layer Perceptron: In order to adjust a reasonable MLP, we have proceeded by fixing a large number of units in a single hidden layer and using weight decay to regularized the model. We considered a MLP with one hidden layer consisting of 20 fixed neurons and variable weight decay parameter. The best decay parameter was selected by cross-validation, and then we calculated the cross validation error of the MLP using the chosen decay parameter. So the error evaluated to 1.3% (46 misclassified) and the confusion matrix is:

Table 5: Confusion matrix of MLP

	Pred as White	Pred as Red
Real White	870	25
Real Red	21	2628

Random Forest: Now, we consider a Random Forest model. To adjust the number of decision trees we adjusted models with different values for this parameter and then we selected the one with the least out of bag error. We first iterated through the first ten powers of two, which resulted in 256 trees for the minimum oob error. Then around this value we searched for a better number considering the multiples of ten. The final number of trees is 290. Finally, the cross validation error is 0.45% (16 misclassified) and the confusion matrix is:

Table 6: Confusion matrix of MLP

	Pred as White	Pred as Red
Real White	882	13
Real Red	3	2646

5.2 Wine quality prediction

5.2.1 Classification

In this part, we consider the problem of the determining the quality as a multi-class classification problem. We will see that these methods have pretty bad results: this may be due to the classes (which here are the wine marks) being unbalanced. For this reason this section is quite brief.

Discriminant Analysis: First of all, we have to note that due to some classes having less observations than variables, we cannot perform QDA. So, we directly jump to RDA, whose regularization parameters point towards using only LDA. Then, we adjust a LDA model with the training set, and its cross-validation error amounts to around 45.91% (1627 misclassified). The confusion matrix is the following:

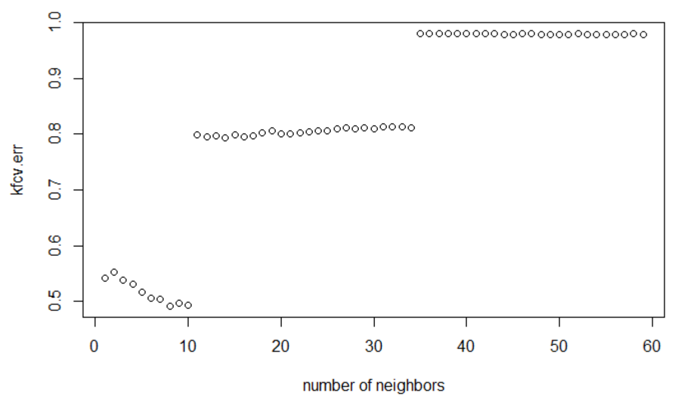
Table 7: Confusion matrix of LDA

	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
Real 3	2	0	9	9	1	0	0
Real 4	3	1	76	46	3	0	0
Real 5	2	2	674	467	10	0	1
Real 6	1	1	354	1086	136	0	1
Real 7	0	0	21	378	154	0	0
Real 8	0	0	3	63	36	0	0
Real 9	0	0	0	1	3	0	0

We can see that this is a pretty bad model, maybe due to the classes being unbalanced as we have said before.

kNN: So, now on to kNN again. This time, when we check the LOOCV errors to see what is the kNN method (original data, scaled data and LD-projected data) that gives better results, as we did before with the type classification, we see that it is again the one using LD-projected data. So, again our 10-fold cross-validation error function will project the data on to LD space and then perform the kNN classification on this data. Also, we have chosen the k as before, but the minimum is now around $k = 55$. To choose the appropriate k for the model, we loop over the interval $[1, 59]$ and choose the k with the least cross-validation error; we bound the interval with 59 as it is approximately the square root of the number of rows in the training set, and this is a well-known boundary for the number of neighbors. We plotted the results in the graphic below.

Figure 9: kNN CV error using LD-projected data



The minimum here is attained at $k = 8$ neighbors, where the CV error is around 49.18%. Again, this model is not able to predict accurately the quality of the wine. Given the distribution of the values in the data set, the algorithm tends to predict mediocre scores for the wines. In other words, given that there are very few outlier wines in the original data set (only five wines had a score of 9) the algorithm cannot predict such a score for wines in the data set. As such, this method is very poor at finding outliers and so it is only partially effective at predicting the score of a wine.

5.2.2 Regression

Instead of trying to solve the quality problem using classification methods, another possibility is that of trying to do so using some kind of regression algorithm. The reason for this is that of trying to make use of the ordinal structure of the classes, in the sense that two "good" wines valued with a score of 7 and 8 could be more similar amongst themselves than to a third wine with a lower score. That being said, due to the different units and scales among the attributes, the data was first standardised in order to perform the regression analysis without the aforestated concerns.

LASSO: Firstly, considering a LASSO regression using all of the predictor variables, the model with the least mean squared error after the cross validation had a value of the penalizing parameter λ being 0.9. Moreover, the MSE of such model was 0.71 and an R-squared value of 0.21, which points towards the fact that the variables are necessary but not enough to explain the variability in the dataset, but more variables are needed in order to fully explain the information contained in the sample.

Ridge Regression: In a similar fashion to the LASSO regression, a ridge regression was used to fit the data and the model that produced the lowest error was selected after the round of cross-validation. This minimum error produced was found with a penalising parameter of $\lambda = 28.159$, and the error amounted to approximately 0.797.

MLP: A multi-layered-perceptron model was trained by considering a regularised network similar to the one used for classification, where a number of hidden units were fixed and the model was regularised using weight decay. The model with the lowest final mean squared error was of size 5 (a single hidden layer with 5 perceptrons) and had a MSE of 0.85 and an R-squared of 0.31.

Random Forest: When considering a random forest for regression, the results over the training data were found by measuring the least out-of-bag error over the models while iterating over the number of variables to split at each node, comparing the mean squared error of each model. The number of splitting variables yielding the minimum error was 2 with a mean square error of 0.61, where splitting over more variables failed to improve on such values, possibly due to lower generalisation. The R-squared value of the preferred model was 0.40.

RBF: Now, to use a Radial Basis Function model on the data we considered a number of prototypical examples selected using a non-supervised clustering algorithm and then implemented the RBF using those starting points. The number of prototypes was determined via the cross-validation method, as usual, and the model selected was the one with the smallest error. The final values were attained after around 4000 iterations and using 354 prototypes, with a final error of 0.96.

Therefore, we can consider the LASSO regression model to be the best since it has the lowest overall MSE out of the models considered.

6 Final Model & Estimated generalization error.

6.1 Wine type best model

Regarding all the models we have considered, the decision of which we have to select is quite straight-forward as there is a model with the smallest error. We have to note that all the models have a pretty small cross-validation error, being the MLP the greatest. It is very likely that this last model overfits the data. This is a summary of all the models we have considered alongside with their respective errors:

Table 8: Summary of the models and errors

Model	**Num. Misclassified**	**% Misclassified**
LDA	13	0.37 %
kNN	12	0.34 %
Logistic Regression	15	0.42 %
MLP	46	1.3 %
Random Forest	16	0.45 %

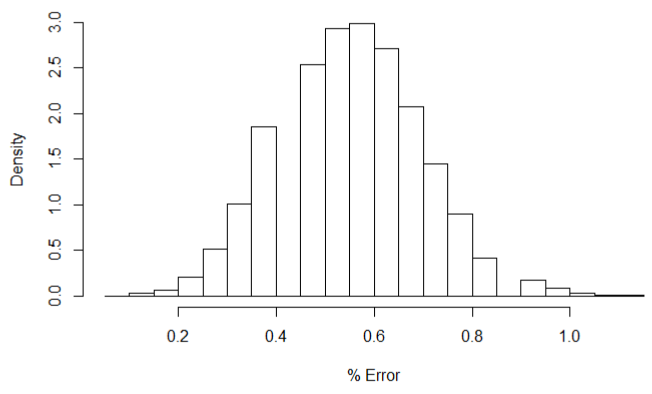
Clearly, the least error belongs to the kNN model using the LD projected data and $k = 5$. Finally, the testing error is 0.96% (17 misclassified) and the confusion matrix is:

Table 9: Confusion matrix of kNN

	Pred as White	Pred as Red
Real White	454	8
Real Red	9	1301

As can be seen, test error is quite larger than the training error. Although this is expected behavior, it may imply that the model overfits a bit the data. To verify it, we proceeded to replicate this procedure 100000 times. In particular, we randomly split the dataset into 2/3 for training and 1/3 for testing. Then, using the training set, predict the testing set as we did before to obtain the error of each iteration. Finally, with all the errors, we plot the following histogram:

Figure 10: Histogram of the % Error of testing 10000 times



It appears that the % testing errors adjust to a normal distribution centered at 0.56%. Indeed, the most common error is this which corresponds to 10 misclassified observations. Therefore, the test error that we first obtained was greater than the mean, so the model we chose did not overfit at all.

6.2 Wine quality best model

Therefore, we can consider the Random Forest model to be the best since it has the lowest overall MSE out of the models considered. The error from the test data when using this model was 0.78, and so even though this is higher than the validation error, we can assume that the overfit to the training data was not too significant but noticeable. As such, the generalisation to unseen data sufferes from overfitting the data.

7 Conclusions.

7.1 Scientific and personal conclusions.

We have attained pretty good results in the type classification; surely there are different explanations regarding this, but we think it is mostly due to the notable chemical differences between both classes of wine. We observed these differences back in the data exploration section, and they allow quite a good separation as we have seen. Also, our final classification model performed quite well on the testing data set, with an estimated generalization error of less than 1%. We consider it to be an interesting model, as it is a combination of two steps, a dimensionality reduction step and the classification step in itself.

In the quality prediction, things have not been as good for the classification approach as they have been for the type classification. We think that this is mainly due to the fact that the multiple ”classes” (from 1 to 10) were very unbalanced, and hindered our models very much. In the kNN model in particular, we tried the three approaches (as can be seen in the code), and only the scaled data approach worked better than LD-projected data, by just 1% of the CV error.

The regression approach,, irrespecitve of the final results obtained, had some serious shortcomings. This is because the quality of a wine is a relatively subjective metric that is ranked on a scale of 1 to 10, and so the bounds on the quality of a wine make regression unsuitable from a theoretical point of view given that theoretically a regression would be able to produce results outside of the aforementioned bounds.

7.2 Possible extensions and known limitations.

We could have separated the rows where the classifiers most frequently failed, to deduce from them some kind of pattern. This would have most certainly allowed us to predict better their class than we have done.

We could have used also other models for both problems. But, we think that the amount of models that we have adjusted is enough to see the possibilities of our data set.

8 References.

1. Web page where the data set can be found:
   http://archive.ics.uci.edu/ml/datasets/Wine+Quality
2. Relevant paper (a previous approach):
   https://www.sciencedirect.com/science/article/pii/S0167923609001377?via%3Dihub
3. Kernels that study this data set on Kaggle (kaggle.com):
   First kernel: https://www.kaggle.com/indra90/predicting-white-wine-quality,
   Second kernel: https://www.kaggle.com/conradws/how-good-is-this-wine-m-l-for-quality-control,
   Third kernel: https://www.kaggle.com/danielpanizzo/red-and-white-wine-quality.
4. Exploratory Data Analysis notebooks:
   https://rstudio-pubs-static.s3.amazonaws.com/57835_c4ace81da9dc45438ad0c286bcbb4224.html,
   https://rstudio-pubs-static.s3.amazonaws.com/219996_9cc8cf9f2e7e41fe8912454c3ad2685a.html.