Instituto Superior Técnico

Project 2 Report
Dimensionality Reduction in Classification of Wine Quality

Mathematics for Machine Learning

2nd Semester 2021/2022

Group 11

Nº 50710	Filipe Martins	MECD
Nº 86694	André Roque	MMAC

1 Introduction

1.1 The Wine Dataset

The data were collected from May 2004 to February 2007 from the demarcated region of *vinho verde* (green wine) in Minho, in the north-west of Portugal [1], and comprises 11 physicochemical variables of numeric type, shown in Table 1, and one output/target variable, of categorical type. The target variable is the quality grade, based on sensory data and determined by the median of at least three blind evaluations made by wine experts. Each expert graded the wine quality between 0 (very bad) and 10 (very excellent). There are two datasets[1], red wine with 1599 samples and white wine with 4898 samples. A third dataset was considered, all wine, obtained by combining the red and white wine datasets with 6497 samples.

nb	**feature**	**nb**	**feature**	**nb**	**feature**
1	Fixed acidity (g/dm^3)	**2**	Volatile acidity (g/dm^3)	**3**	Citric acid (g/dm^3)
4	Residual sugar (g/dm^3)	**5**	Chlorides (g/dm^3)	**6**	Free $SO_2(mg/dm^3)$
7	Total $SO_2(mg/dm^3)$	**8**	Density (g/dm^3)	**9**	pH
10	Sulphates (g/dm^3)	**11**	Alcohol (vol %)		

Table 1: Physicochemical parameters of wine with respective numbering of features.

1.2 Exploratory Data Analysis

In order to have a better understanding of the datasets, we performed a preliminary statistics of the data. The summary statistics of the three datasets can be seen in Tables 9, 10 and 11, in the appendix. In sequence, the pairs panels of the features were generated, Figures 2, 3 and 4, in the appendix. By looking at the histograms, we suspect that most features deviate from a normal distribution, by looking at the scatter plots we see that the red wine data values are more spread in more variables than in the case of the white wine data variables values, as for the all wine data, we see less spread values in the variables than the other two datasets, as well as no visual evidence of clear separated clusters, showing (visually) that the values become more agglomerated when combined, where 75.4% corresponds to the white wine and 24.6% to the red wine. Concerning the correlation between variables we generated the respective matrices with a color scale for better visualization, they are presented in the appendix, Figures 5, 6 and 7. We found no evidence of high correlation between variables nonetheless the most relevant values of linear associations are: Fixed acidity and Citric acid (0.67), Fixed acidity and Density (0.67), Fixed acidity and pH (-0.68) and Free SO_2 and Total SO_2 (0.67) for red wine; Density and Alcohol (-0.78), Density and Residual sugar (0.84) and Free SO_2 and Total SO_2 (0.62) for white wine and, Density and Alcohol (-0.69) and free SO_2 and Total SO_2 (0.72) for all wine. In Figures 8 and 9, in the appendix, we observe normality in the histograms distribution of the values in the quality target variable and in the box-plots we observe that (quality) grades 3, 8 and 9 represent outliers, with respect to IQR criteria. Given the three datasets we decided for the white wine dataset, due to fact that it has more samples than the red wine and that we observed no experimental evidence of normality distribution of the white and red wine datasets, by performing QQ-plots on each variable, many of which shown no normality and by performing the Shapiro-Wilk test to assess multivariate normality, available in *mvnormtest* R package, which failed the test.

[1]The datasets are multivariate and there are no missing attribute values.

2 Methodology

To accomplish the requested tasks we defined four main stages: balance and split the dataset, in 2.1; apply the dimensionality reduction techniques, in 2.2; implement the two classifiers, in 2.3; and present the performance measures to assess the models performance, in 2.4.

2.1 Handling Imbalanced Data

The class distribution of the white wine dataset is shown in Table 2, where we clearly see the imbalance of data with respect to the classes. We have seven classes for quality, grades 3 to 9. Given the severe imbalance of grades 3, 4, 8 and 9, instead of using the seven classes we decided to create three new classes by aggregation of quality grades namely: class 0 (= grades 3 and 4) denoting "bad quality"; class 1 (= grades 5 and 6) denoting "medium quality"; and class 2 (= grades 7, 8 and 9) denoting "good quality", see Table 3. With this aggregation of classes we simplify the process of sampling the data and we make the classification more narrow.

grade	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**total**
N	20	163	1457	2198	880	175	5	4898
(%)	0.41	3.33	29.75	44.87	17.97	3.57	0.10	100

Table 2: Frequencies distribution of quality grades for white wine.

class	**0**	**1**	**2**	**total**
N	183	3655	1060	4898
(%)	3.74	74.62	21.64	100

Table 3: Frequencies distribution of new quality classes for white wine.

Classification tasks require the datasets to be split into two sub-datasets, one for training and the other for test. We used *stratified* from *splitstacksahpe* package ([6]) to perform a stratified splitting of the data, in order to keep the proportions of each class in the resulting two datasets and have considered 80% of the white wine set for training (3918 samples) and 20% for test (980 samples). The distribution of frequencies per class on both datasets is presented in Table 13 in the appendix. To achieve a more even distribution between the three classes for training of the classifiers, balancing by sampling was realized. We decided to create two balanced datasets, one where we perform undersampling first and then oversampling and the other, where we perform oversampling first and then undersampling. In this way, we can compare the performances of the considered models over two different orderings of sampling the data. We decided to equalize class 0 with class 2 and reduce the amount of data from class 1. The sampling methods used work for binary classes so, we aggregated 0 and 2 when performing undersampling of majority class 1, and aggregated 1 and 2 when performing oversampling of minority class 0. When undersampling, we applied two consecutive techniques, first, Tomek Link then Neighborhood Cleaning Rule, from *unbalanced* package ([7]) and when oversampling, we used Borderline SMOTE, from *smotefamily* package ([8]). The resulting class distributions after the data samplings are shown in Table 14 in the appendix.

2.2 Dimensionality Reduction

We performed Feature Selection based on Mutual Information (MIFS), in 2.2.1, on Principal Component Analysis (PCA), in 2.2.2, and on Sparse Principal Component Analysis (SPCA), in 2.2.3.

2.2.1 Feature Selection based on Mutual Information

To perform MIFS[2] we considered three methods: Mutual Information Maximization (MIM), formula (1), Maximum Relevance Minimum Redundancy (mRMR), formula (2), and Joint Mutual Information method (JMI), formula (3). In (1), (2), (3), F and S correspond to the sets of unselected and selected features, respectively and C, X_i and X_s correspond to the class, candidate and selected variables, respectively.

$$\text{MIM}: \quad \arg\max_{X_i \in F} \{\text{MI}(C, X_i)\} \tag{1}$$

$$\text{mRMR}: \quad \arg\max_{X_i \in F} \left\{ \text{MI}(C, X_i) - \frac{1}{|S|} \sum_{X_s \in S} \text{MI}(X_i, X_s) \right\} \tag{2}$$

$$\text{JMI}: \quad \arg\max_{X_i \in F} \left\{ \text{MI}(C, X_i) - \frac{1}{|S|} \sum_{X_s \in S} (\text{MI}(X_i, X_s) - \text{MI}(X_i, X_s|C)) \right\} \tag{3}$$

The terms in each maximization in (1), (2) and (3) are given by (4), (5) and (6). Based on the known formulas from Information Theory we derived the formulas below[3].

$$\text{MI}(C, X_i) = H(C) + h(X_i) - h(C, X_i), \tag{4}$$

$$\text{MI}(X_i, X_s) = H(X_i) + h(X_s) - h(X_i, X_s), \tag{5}$$

$$\text{MI}(X_i, X_s|C) = h(X_s|C) - h(X_s|X_i, C) = h(C, X_i) + h(C, X_s) - H(C) - h(C, X_i, X_s) \tag{6}$$

We know from Information Theory that $H(X^{\Delta}) + \ln(\Delta) \xrightarrow[\Delta \to 0]{} h(X)$, with X, a continuous random variable and X^{Δ} its discretization thus, computationally, the previous formulas have to discretized as well as each continuous variable and as their distributions are unkwnon, the probabilities in the discretized formulas have to be estimated. We considered an histogram based estimation available by function *entropy* in *infotheo* R package [9]. Specifically, we considered two estimators, empirical and Miller-Madow. The discretized versions of (4)-(6) are given by (7)-(9), respectively.

$$\text{MI}(C, X_i) \approx H(C) + H(X_i^{\Delta_1}) + \ln(\Delta_1) - H(C, X_i^{\Delta_2}) - \ln(\Delta_2) \tag{7}$$

$$\text{MI}(X_i, X_s) \approx H(X_i^{\Delta_3}) + \ln(\Delta_3) + H(X_s^{\Delta_4}) + \ln(\Delta_4) - H(X_i^{\Delta_5}, X_s^{\Delta_6}) - \ln(\Delta_5) - \ln(\Delta_6) \tag{8}$$

$$\text{MI}(X_i, X_s|C) \approx H(C, X_i^{\Delta_7}) + H(C, X_s^{\Delta_8}) - H(C) - H(C, X_i^{\Delta_9}, X_s^{\Delta_{10}}) + \sum_{k=1,2} (\ln(\Delta_{6+k}) - \ln(\Delta_{8+k})) \tag{9}$$

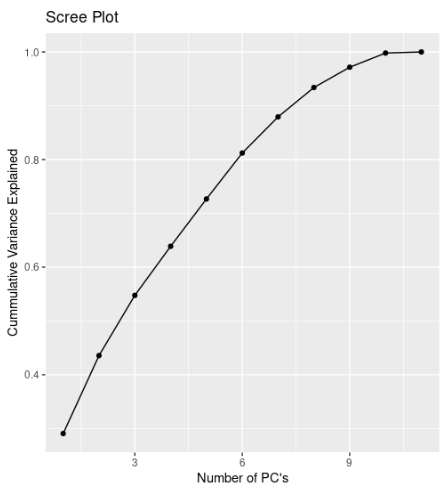
The number of bins for the discretization of each variable is based on the formulas in [4], $\sqrt{n}$, where n is the number of observations in the dataset. For the discretizations we used *discretize* from *infotheo*. We have also considered the estimations with the correction terms, $\ln(\Delta)$, and without. For the correction terms not to cancel we had to select different numbers of bins, small variations in the root order. We remark that we have considered four different number of bins, where $\Delta_k, k = 3, 4, 9, 10$ have the same number of bins, these terms add, and where $\Delta_k, k = 5, 6, 7, 8$ have the same number of bins, these terms subtract. The correction terms for these discrete steps do not cancel in (8) and (9), because, the variables have different ranges. To determine the ordering of features for the three datasets, the procedure followed is the same for each of them, except by the formulas that define each method.

[2]Mutual Information is an information-theoretic metric able to capture both linear and non-linear dependencies among random variables.

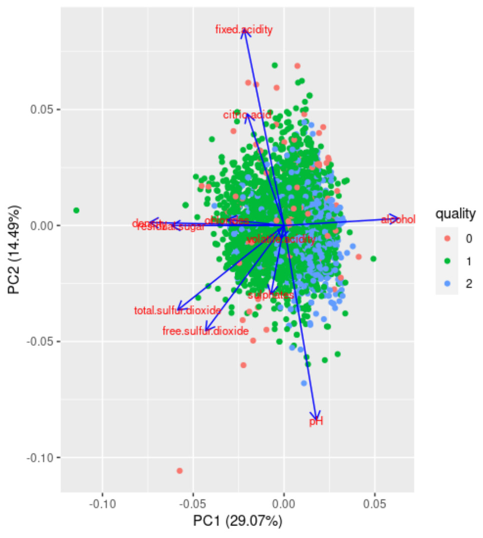
[3]Given that the physicochemical parameters are understood as continuous variables, $h(.)$, stands for the differential entropy, and $H(.)$, for the entropy (for discrete variables which is the case of class variable C).

2.2.2 Principal Component Analysis

PCA is a dimension reduction technique based on projection methods. PCA constructs new features (Principal Components) that are made from a linear combination of the initial features. It does so by finding the directions in a p-dimensional space that approximate the data as well as possible in the least squares perspective. Statistically, each PC corresponds to a eigenvector of the covariance metric. PC's are then sorted by the amount of data variability each one explains. For the PC's computation, it was used the function *prcomp* from the R package *stats* [12]. This function was applied to an unlabeled standardized version of each dataset. The standardization of the datasets is justified by the fact that the features free sulfur dioxide and total sulfur dioxide showed considerably greater variance than the remaining variables. Being so, not normalizing the data would very likely lead to biased results. As it is shown in Figure 1(b), alcohol seems to play an important role when it comes to differentiate a good wine from an average wine. From each result, we decide to retain the first 6 PC's, since they were enough to explain more than 80% of each dataset variability, Figure 1(a). As it can be seen in Table 4, every feature is represented with significance in at least one of the Principal Components. We also found interesting to see how our classifiers behaved if more PC's were retained, and to accomplished that, for each dataset (the original and the one were oversampling and unsersampling was applied) two new datasets were created by applying the rotations given be the PC's loadings, one with six and the other with ten. With those same loadings, four new test datasets were created, in order to apply our classifiers on data with the same rotation that where used to created the datasets on which they were trained.



(a) Scree Plot for the Cumulative Variance Explained.



(b) Projection of data on the first 2 PC's and respective eigenvectors.

Figure 1: Visualization of the PCA analysis performed on the original dataset.

PC1	(+) density , (+) alcohol, (-) residual sugar, (-) total sulfur dioxide
PC2	(+) fixed acidiy , (-) pH
PC3	(+) volatile acidity, (-) citric acid, (-) sulphates
PC4	(+) chlorides, (-) free sulfur dioxide, (+) sulphates
PC5	(-) sulphates, (+) chlorides, (-) volatile acidity, (-) fixed acidity
PC6	(+) free sulfur dioxide, (+) total sulfur dioxide, (+) volatile acidity, (-) density

Table 4: Main coefficients on each Principal Component. Every feature is represented in at least one of the PC's.

2.2.3 Sparse Principal Component Analysis

SPCA is a variant of PCA, used to reduce the dimensionality of a dataset by means of introducing sparsity into the PCA loadings. Unlike in the PCA, where it is assumed that each instance of the features can be rebuilt using the same components, SPCA uses a limited number of components without the limitation of having a dense projection of our data. Comparatively to PCA, it has the advantage of improving the interpretability by setting to 0 the less important coefficients of each PCA loading. It does so by writing the PCA as a regression optimization problem by using the Lasso variable selection technique. In this work, the SPCA was computed by using the *spca* from the package *elasticnet* [11]. By visual inspection of the PC's, it was decided that the number of non-negative components on each PC should be 6, since this number was enough for keeping the most relevant coefficients in each PC. Unlike in the PCA, we decided to use only the first 6 sparse principal components to create new datasets. The procedure to create the new train and test datasets was the same that was described for the PCA, producing a total of 3 rotated datasets.

2.3 Classifiers

2.3.1 KNN Classifier

K-Nearest Neighbors (KNN) is maybe the most intuitive supervised learning method, since it relies on the assumption that similar observations are closer to each other according to some metric, that is, similar observations constitute neighbors. Given a new observation, KNN considers a set of k nearest neighbors and classifies ithem according to the most represented class in the set. KNN belongs to a group of classifiers known as The Bayes classifiers. The Bayes classifiers are a family of classification algorithms based on the Bayes' Theorem[4]. Given an observation $\boldsymbol{x} = (x_1, ..., x_p)$ from a random vector $\boldsymbol{X} = (X_1, ..., X_p)$, the classifier tries to find the class ω that maximizes the *a posteriori* probability (10):

$$P(Y = \omega|\boldsymbol{X} = \boldsymbol{x}) = \frac{p(\boldsymbol{X} = \boldsymbol{x}|Y = \omega)P(Y = \omega)}{p(\boldsymbol{X} = \boldsymbol{x})} \tag{10}$$

where p is the conditional probability function of the random variable $\boldsymbol{X}|Y = \omega$ (continuous or discrete). KNN has the disadvantage of not performing well in high dimensional data - a phenomenon known as *The Peaking Phenomenon* [10] - thus, it is expected that the dimensionality reduction methods make a difference by improving our classification results. We have used the method "knn" available in the *caret* package [13], [14], that implements the KNN method.

2.3.2 SVM Classifier

Support Vector Machines (SVM) aims to maximize the distinction between the different classes, such that new observations are mapped into that same space and predicted observations belong to a category based on the gap they have regarding each class. To accomplish that, the SVM algorithm finds hyperplanes that maximize the margin between the classes. It looks for points - known as support vectors - that are located near the closest dividing line. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with the support vector in that space is constant, where such a set of (support) vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. Then, the algorithm calculates the maximum gap between the support vectors and the dividing plane. This constitutes the SVM with Linear kernel, $K(x, y) = x^\top y$. Given the possibility of the data points to be non-linearly separable, given the respective classes, we have also considered a different kernel, the Radial Basis Function kernel (RBF),

[4]Given two events A and B, the conditional probability of A given B is $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, $P(B) \neq 0$

$K(x,y) = exp(-\gamma\|x-y\|^2)$, $\gamma > 0$, which enables to transform the data into a higher-dimensional feature space where SVM is able to produce a linear separation (using the hyperplabes), this is known as the *kernel trick*. For our project, we used "svmLinear" and "svmRadial" that implement the Linear and RBF SVM methods, respectively, available in the *caret* package. Concerning the RBF SVM model we considered a grid search space for parameters γ and C, C is the cost to penalize misclassification while training, with a total of 36 combinations.

2.4 Performance Measures

The Performance metrics used were the Accuracy, the Macro Precision, the Macro Recall and the Macro F1-Measure. For a given class i, the definition of the the metrics is as it is shown below:

$$\text{Accuracy} = \frac{\text{N}^{\underline{\text{o}}} \text{ of observations correctly predicted}}{\text{N}^{\underline{\text{o}}} \text{ of observations}} \tag{11}$$

$$\text{Recall(i)} = \frac{\text{N}^{\underline{\text{o}}} \text{ of observations of the class i assigned do class i}}{\text{N}^{\underline{\text{o}}} \text{ of observations that belong to class i}} \tag{12}$$

$$\text{Precision(i)} = \frac{\text{N}^{\underline{\text{o}}} \text{ of observations assigned to class i that belong to to class i}}{\text{N}^{\underline{\text{o}}} \text{ of observations assigned to class i}} \tag{13}$$

$$\text{F1-measure(i)} = 2 \times \frac{\text{Precision(i)} \times \text{Recall(i)}}{\text{Precision(i)} + \text{Recall(i)}} \tag{14}$$

The Macro Precision, the Macro Recall and the Macro F1-Measure are simply the arithmetic means of the Precision, the Recall and the F1-Measure for each class, respectively. Since our dataset is heavily unbalanced for the class 0, the Macro Recall (in particular, the Recall(0)) plays an important role when it comes to access the quality of our classifications. Attending to the fact that the 3 classes have the same importance in the context of our problem, F1 measures comes at hand since penalizes the choice of the majority class by the classifier.

3 Results and Discussion

In Table 12, we present the features selected by the MIFS methods in decreasing order of importance. In MIM method it just changes the order of the last three features, with respect to the probability estimator, for the imbalanced dataset, for under-over sampling dataset the the selections are the same and for the over-under sampling dataset the the selections are also the same, but between the three datasets the orderings are different. In mRMR method, it just changes the order of the first three features, with respect to the probability estimator, for the imbalanced dataset, for the under-over the orderings are the same and for the over-under the ordering of the first five is different, with respect to the probability estimator, but between the three datasets the orderings are different. In JMI method, the orderings are the same for the imbalanced one, for the under-over the orderings are the same and for over-under the orderings are also the same, but different between the three datasets. We observed no difference between the orderings obtained with and without the correction terms, with respect to each method and dataset and less variation of the selection of features with respect to its importance in the mRMR and JMI than in the MIM, as well as in the JMI than in the mRMR.

For the classification of the three original datasets and the ones obtained from the MIFS methods we considered the evaluation of the training phase, for KNN and SVM, with two different summary functions namely, the Accuracy and the Macro F1-Measure. It was not clear if Macro F1 gives better results, in same cases it did in others it did not, but the differences are not very significant. In some cases, few, the results

KNN (k = 3)	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
JMI imbal SumFun: acc	0.7714286	0.6140774	0.5482334	0.5679564
unov SumFun: acc	0.7265306	0.5351951	0.6051632	0.5583146
MIM ovun SumFun: MacroF1	0.7265306	0.5482290	0.6170652	0.5717695
RBF SVM	**Accuracy**	**Macro Precision**	**Macro Recall**	**Macro F1-Measure**
imbal SumFun: MacroF1	0.8316327	0.7453155	0.5677005	0.6116604
JMI unov mm SumFun: acc = MacroF1	0.7775510	0.6048920	0.6264691	0.6148956
JMI ovun mm SumFun: MacroF1	0.7816327	0.6263321	0.6357297	0.630651

Table 5: Best results for KNN and RBF SVM with respect to all generated datasets, except the ones from PCA and SPCA.[5]

were the same, with which we were somewhat surprised.

In Table 5 we present the best results obtained from KNN (k = 3) and RBF SVM with respect to the three original datasets and the ones obtained from the MIFS methods. In Tables 15, 16 and 17 we present all the results obtained with KNN and in Tables 18, 19 and 20, we present all the results obtained with RBF SVM, these tables are included in the appendix. The best overall results were observed for the JMI applied to the over-under sampling dataset, which justifies the use of balancing techniques and feature selection methods, with experimental evidence. The imbalanced dataset showed the best accuracy and Macro Precision results but worst Macro Recall and Macro F1-measure, given the pour results in the Macro Recall metric, which was expected.

In Tables 6 and 7 we present the results with respect to KNN model applied to the PCA and SPCA methods, with k = 3 and with the best value of k that minimizes the total probability of misclassification in each train dataset, the previous obtained via *tuneLength* parameter in *train*. From observing the results, the model behaves well for the selected first six PC's. For these datasets, the choice of k did have an impact, with a greater k value providing for every dataset a best or equal value of accuracy, and in the case of the imbalanced dataset provided a great increase in the Macro Precision. The improvement may be more due to the loadings application than the removal of features, since the use of less features decreased the quality of the results. Attending to our results, we confirmed that the use of the Euclidean distance over data on which it was applied the transformations provided by the PCA (and consecutively by the SPCA) was a good choice. Nonetheless, other metrics should also be tried, such as Minkowsky distance and cosine similarity distance.

KNN (k = 3)	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
PCA imbal - 6 features	0.75714	0.53732	0.49424	0.50623
PCA imbal - 10 features	0.70301	0.45237	0.40719	0.41488
PCA ovun - 6 features	0.57143	0.40119	0.40527	0.38221
PCA ovun - 10 features	0.70306	0.50894	0.58016	0.52678
SPCA imbal - 6 features	0.68776	0.47029	0.49001	0.47784
SPCA ovun - 6 features	0.71224	0.52879	0.59096	0.55013

Table 6: Results for KNN (with k = 3) for PCA and SPCA feature selection.

[5]imbal = imbalanced, unov = under+over sampling, ovun = over+under sampling, mm = Miller-Madow estimator, SumFun = SummaryFunction, used in the training evaluation.

KNN - Best k value	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
PCA imbalanced - 6 features	0.77142	0.68314	0.47682	0.49176
PCA imbal - 10 features	0.74387	0.41347	0.37707	0.37228
PCA ovun - 6 features	0.56225	0.41524	0.41195	0.38667
PCA ovun - 10 features	0.70306	0.51575	0.60508	0.53719
SPCA imbal - 6 features	0.66531	0.45243	0.47216	0.45948
SPCA ovun - 6 features	0.70816	0.52694	0.59881	0.55054

Table 7: Results for KNN (With the best value of k) for PCA and SPCA feature selection.

For the Linear SVM classifer the use of PCA and SPCA resulted in a negligible difference in the results obtained using the original datasets. We think that this may be due to the use of a Linear Kernel. Since the PCA and the SPCA rotations don't change the distances between each observation (that is, they correspond to isometries), the distances that are used by the Linear SVM for creating the hyperplanes are the same in the original datasets and on the rotated ones. Hence, the classifier will produce the respective rotation of these hyperplanes originated by applying the algorithm to the original data. As result, the classifications produced will be equal.

Linear SVM	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
PCA imbal - 6 features	0.74592	0.24864	0.33333	0.28482
PCA imbal - 10 features	0.74592	0.24864	0.33333	0.28482
PCA ovun - 6 features	0.60918	0.39885	0.40540	0.38409
PCA ovun - 10 features	0.67551	0.49191	0.53287	0.48240
SPCA imbal - 6 features	0.74592	0.24864	0.33333	0.28482
SPCA ovun - 6 features	0.71122	0.50374	0.54103	0.51585

Table 8: Results for Linear SVM for PCA and SPCA feature selection.

4 Conclusions and Future Work

Generally, the use of balancing techniques improved greatly the Precision of the classifiers at the cost of some accuracy, a result that can be important if, for example, our concern was to identify bad wines, since those represent the minority class. As it was seen in the results, by reducing the amount of features the results can be unpredictable, since in some case the performance measures were significantly worse, but in others there was a great improvement, which can explained by the removal of redundancy and correlation between features. Nonetheless, the cases where the results were slightly worse should not be disconsered, since for some applications the trade-off between the amount of information used and the decrease of quality in the results might be acceptable, if we were concerned in saving processing power. Although several feature selection methods were applied, it was not possible to cleary identify patterns on the features that could be decisive on quality of the wine, since our experiment with the multiple selection methods resulted in a little conflicting information. A more in-depth analysis of the number of bins, discretization steps for the estimation of probabilities and experimental analysis of convergence of the Mutual Information formulas should be realized. One of the problems regarding the dataset was the low amount of representatives of grades 3, 4, 8 abd 9.. In order to improve the results, other classifiers should be tried, and other metrics experimented in the case of the KNN and SVM classifiers, and a more in-depth search of available functions in packages and possible customization of function to be used for the trainControl and train of the models, as well as, considering a wider and a finer search space for the RBF SVM parameters.

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis (2009). *Modeling wine preferences by data mining from physicochemical properties.* In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

[2] C. Pascoal, M. R. Oliveira, A. Pacheco and R. Valadas (2017). *Theoretical Evaluation of Feature Selection Methods Based on Mutual Information.* Neurocomputing Vol. 226, N^{o} 1, pp. 168 - 181

[3] M. R. Oliveira. InfoTheory_FS2.pdf. Class Notes

[4] M. R. Oliveira. Information Theory_Intro_v2.pdf. Class Slides

[5] M. Kuhn. Building Predictive Models in R Using the caret Package. Journal of Statistical Software, 28(5), 1–26, 2008. `http://dx.doi.org/10.18637/jss.v028.i05`.

[6] `https://cran.r-project.org/web/packages/splitstackshape/splitstackshape.pdf`

[7] `https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf`

[8] `https://cran.r-project.org/web/packages/smotefamily/smotefamily.pdf`

[9] `https://cran.r-project.org/web/packages/infotheo/infotheo.pdf`

[10] `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.418.6517&rep=rep1&type=pdf`

[11] `https://cran.r-project.org/web/packages/elasticnet/elasticnet.pdf`

[12] `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp`

[13] `https://cran.r-project.org/web/packages/caret/caret.pdf`

[14] `https://topepo.github.io/caret/`

Appendix

	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Fixed acidity	8.32	1.74	7.9	8.15	1.48	4.6	15.9	11.3	0.98	1.12	0.044
Volatile acidity	0.53	0.18	0.52	0.52	0.18	0.12	1.58	1.46	0.67	1.21	0.004
Citric acid	0.27	0.19	0.26	0.26	0.25	0	1	1	0.32	-0.79	0.005
Residual sugar	2.54	1.41	2.2	2.26	0.44	0.9	15.5	14.6	4.53	28.49	0.035
Chlorides	0.087	0.047	0.079	0.080	0.015	0.012	0.61	0.60	5.67	41.53	0.001
Free SO2	15.87	10.46	14	14.58	10.38	1	72	71	1.25	2.01	0.262
Total SO2	46.47	32.90	38	41.84	26.69	6	289	283	1.51	3.79	0.823
Density	0.997	0.002	0.997	0.99	0.002	0.99	1.004	0.01	0.07	0.92	0.00005
pH	3.31	0.15	3.31	3.31	0.15	2.74	4.01	1.27	0.19	0.80	0.004
Sulphates	0.66	0.17	0.62	0.64	0.12	0.33	2	1.67	2.42	11.66	0.004
Alcohol	10.42	1.07	10.2	10.31	1.04	8.4	14.9	6.5	0.86	0.19	0.027

Table 9: Descriptive statistics for red wine.

	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Fixed acidity	6.85	0.84	6.8	6.82	0.74	3.8	14.2	10.4	0.65	2.17	0.012
Volatile acidity	0.28	0.10	0.26	0.27	0.09	0.08	1.1	1.02	1.58	5.08	0.001
Citric acid	0.33	0.12	0.32	0.33	0.09	0	1.66	1.66	1.28	6.16	0.002
Residual sugar	6.39	5.07	5.2	5.80	5.34	0.6	65.8	65.2	1.08	3.46	0.072
Chlorides	0.046	0.022	0.043	0.043	0.010	0.009	0.35	0.34	5.02	37.51	0.0003
Free SO2	35.31	17.01	34	34.36	16.31	2	289	287	1.41	11.45	0.243
Total SO2	138.36	42.50	134	136.96	43.00	9	440	431	0.39	0.57	0.607
Density	0.99	0.003	0.99	0.99	0.003	0.99	1.04	0.05	0.98	9.78	0.00004
pH	3.19	0.15	3.18	3.18	0.14	2.72	3.82	1.1	0.46	0.53	0.002
Sulphates	0.49	0.11	0.47	0.48	0.10	0.22	1.08	0.86	0.98	1.59	0.002
Alcohol	10.51	1.23	10.4	10.43	1.48	8	14.2	6.2	0.49	-0.70	0.018

Table 10: Descriptive statistics for white wine.

	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Fixed acidity	7.22	1.30	7	7.06	0.89	3.8	15.9	12.1	1.72	5.05	0.016
Volatile acidity	0.34	0.16	0.29	0.32	0.12	0.08	1.58	1.5	1.49	2.82	0.002
Citric acid	0.32	0.15	0.31	0.32	0.10	0	1.66	1.66	0.47	2.39	0.002
Residual sugar	5.44	4.76	3	4.70	2.52	0.6	65.8	65.2	1.43	4.35	0.059
Chlorides	0.05	0.04	0.05	0.05	0.02	0.009	0.61	0.60	5.40	50.84	0.0004
Free SO2	30.53	17.75	29	29.32	17.79	1	289	288	1.22	7.90	0.220
Total SO2	115.74	56.52	118	115.92	57.82	6	440	434	-0.001	-0.37	0.701
Density	0.99	0.003	0.99	0.99	0.003	0.99	1.04	0.05	0.50	6.60	0.00004
pH	3.22	0.16	3.21	3.21	0.16	2.72	4.01	1.29	0.39	0.37	0.002
Sulphates	0.53	0.15	0.51	0.52	0.12	0.22	2	1.78	1.80	8.64	0.002
Alcohol	10.49	1.19	10.3	10.40	1.33	8	14.9	6.9	0.57	-0.53	0.015

Table 11: Descriptive statistics for all wine.

MIM imbal emp no-corr	**mRMR imbal emp no-corr**	**JMI imbal emp no-corr**
11 8 10 6 7 5 4 3 2 9 1	1 6 11 10 3 4 5 2 7 9 8	1 11 10 9 7 2 3 8 6 4 5
MIM imbal emp corr	**mRMR imbal emp corr**	**JMI imbal emp corr**
11 8 10 6 7 5 4 3 2 9 1	1 6 11 10 3 4 5 2 7 9 8	1 11 10 9 7 2 3 8 6 4 5
MIM imbal mm no-corr	**mRMR imbal mm no-corr**	**JMI imbal mm no-corr**
11 8 10 6 5 7 4 3 2 1 9	1 11 6 10 3 4 2 5 7 9 8	1 11 10 9 7 2 3 8 6 4 5
MIM imbal mm corr	**mRMR imbal mm no-corr**	**JMI imbal mm no-corr**
11 8 10 6 5 7 4 3 2 1 9	1 11 6 10 3 4 2 5 7 9 8	1 11 10 9 7 2 3 8 6 4 5
MIM unov emp no-corr	**mRMR unov emp no-corr**	**JMI unov emp no-corr**
11 6 7 2 8 4 5 3 10 9 1	1 6 11 2 5 4 10 7 3 8 9	1 11 10 7 2 9 6 3 5 4 8
MIM unov emp corr	**mRMR unov emp corr**	**JMI unov emp corr**
11 6 7 2 8 4 5 3 10 9 1	1 6 11 2 5 4 10 7 3 8 9	1 11 10 7 2 9 6 3 5 4 8
MIM unov mm no-corr	**mRMR unov mm no-corr**	**JMI unov mm no-corr**
11 6 7 2 8 4 5 3 10 9 1	1 6 11 2 5 4 10 7 3 9 8	1 11 10 7 2 6 9 3 5 4 8
MIM unov mm corr	**mRMR unov mm no-corr**	**JMI unov mm no-corr**
11 6 7 2 8 4 5 3 10 9 1	1 6 11 2 5 4 10 7 3 9 8	1 11 10 7 2 6 9 3 5 4 8
MIM ovun emp no-corr	**mRMR ovun emp no-corr**	**JMI ovun emp no-corr**
6 11 7 2 8 4 5 3 10 1 9	1 6 11 2 4 5 10 7 3 8 9	1 11 10 7 2 9 6 3 8 4 5
MIM ovun emp corr	**mRMR ovun emp corr**	**JMI ovun emp corr**
6 11 7 2 8 4 5 3 10 1 9	1 6 11 2 4 5 10 7 3 8 9	1 11 10 7 2 9 6 3 8 4 5
MIM ovun mm no-corr	**mRMR ovun mm no-corr**	**JMI ovun mm no-corr**
6 11 7 8 2 4 5 3 10 1 9	1 6 11 2 4 10 5 7 3 9 8	1 11 10 7 2 6 9 3 8 4 5
MIM ovun mm corr	**mRMR ovun mm no-corr**	**JMI ovun mm no-corr**
6 11 7 8 2 4 5 3 10 1 9	1 6 11 2 4 10 5 7 3 9 8	1 11 10 7 2 6 9 3 8 4 5

Table 12: Selected features for each dataset per MIFS method, probability estimator and correction terms.[6]

[6]imbal = imbalanced, unov = under+over sampling, ovun = over+under sampling, emp = emprical estimator, mm = Miller-Madow estimator, no-corr = without correction terms and corr = with correction terms.

	train (80%)				**test (20%)**			
class	**0**	**1**	**2**	**total**	**0**	**1**	**2**	**total**
N	146	2924	848	3918	37	731	212	980
(%)	3.73	74.63	21.64	100	3.78	74.59	21.63	100

Table 13: Frequencies distribution of classes in train and test sets.

	unders + overs				**overs + unders**			
	1st undersamplig				**oversampling**			
class	**0**	**1**	**2**	**total**	**0**	**1**	**2**	**total**
N	146	2520	848	3541	854	2924	848	4626
(%)	4.16	71.71	24.13	100	18.46	63.21	18.33	100
	2nd undersampling				**1st undersampling**			
class	**0**	**1**	**2**	**total**	**0**	**1**	**2**	**total**
N	146	2024	848	3018	854	2533	848	4235
(%)	4.48	67.06	28.10	100	20.17	58.81	20.02	100
	oversampling				**2nd undersampling**			
class	**0**	**1**	**2**	**total**	**0**	**1**	**2**	**total**
N	842	2024	848	3714	854	2026	848	3728
(%)	22.67	54.50	22.83	100	22.91	54.34	22.75	100

Table 14: Frequencies distribution of classes after each sampling.

KNN	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
imbal SumFun: acc	0.7795918	0.5858410	0.5206406	0.5388798
MIM imbal SumFun: acc	0.7693878	0.5068808	0.4859563	0.4933847
mRMR imbal SumFun: acc	0.7367347	0.5074839	0.4784235	0.4873914
JMI imbal SumFun: acc	0.7714286	0.6140774	0.5482334	0.5679564
imbal SumFun: MacroF1	0.7795918	0.6249565	0.5280773	0.5522435
MIM imbal SumFun: MacroF1	0.7785714	0.5861334	0.5157193	0.5360812
mRMR imbal SumFun: MacroF1	0.7387755	0.4897508	0.4730151	0.4784634
JMI imbal SumFun: MacroF1	0.7693878	0.5931426	0.5428560	0.5603692

Table 15: Results for KNN (k = 3) for the imbalanced dataset and the ones obtained from MIFS methods applied to the imbalanced dataset.[7]

KNN	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
unov SumFun: acc	0.7265306	0.5351951	0.6051632	0.5583146
MIM unov SumFun: acc	0.7142857	0.5263978	0.5911382	0.5476852
mRMR unov SumFun: acc	0.6887755	0.4951275	0.5544403	0.5133573
JMI unov emp SumFun: acc	0.7030612	0.5078372	0.5719876	0.5272947
JMI unov mm SumFun: acc	0.7183673	0.5220031	0.5728683	0.5407255
unov SumFun: MacroF1	0.7224490	0.5236237	0.5840005	0.5437951
MIM unov SumFun: MacroF1	0.7163265	0.5295013	0.5898176	0.5502901
mRMR unov umFun: MacroF1	0.6897959	0.4975714	0.5571290	0.5159412
JMI unov emp SumFun: MacroF1	0.7081633	0.5082186	0.5720349	0.5265072
JMI unov mm SumFun: MacroF1	0.7204082	0.5243659	0.5715476	0.542306

Table 16: Results for KNN (k = 3) for the under-over dataset and the ones obtained from MIFS methods applied to the under-over dataset.[8]

KNN	Accuracy	Macro Precision	Macro Recall	Macro F1-Measure
ovun SumFun: acc	0.7142857	0.5218360	0.5866729	0.5431294
MIM ovun SumFun: acc	0.7204082	0.5437392	0.6120966	0.5669905
mRMR ovun emp SumFun: acc	0.7132653	0.5380534	0.5891883	0.5563054
mRMR ovun mm SumFun: acc	0.7204082	0.5289531	0.5726639	0.5459186
JMI ovun emp SumFun: acc	0.7122449	0.5279674	0.6028669	0.5518135
JMI ovun mm SumFun: acc	0.7244898	0.5255615	0.5789532	0.5447524
ovun SumFun: MacroF1	0.7102041	0.5149251	0.5740632	0.5344147
MIM ovun SumFun: MacroF1	0.7265306	0.5482290	0.6170652	0.5717695
mRMR ovun emp SumFun: MacroF1	0.7102041	0.5224861	0.5707143	0.5393988
mRMR ovun mm SumFun: MacroF1	0.7224490	0.5324863	0.5724596	0.5484254
JMI ovun emp SumFun: MacroF1	0.7112245	0.5233589	0.5949743	0.5459446
JMI ovun mm SumFun: MacroF1	0.7193878	0.5241300	0.5807609	0.5441711

Table 17: Results for KNN (k = 3) for the over-under dataset and the ones obtained from MIFS methods applied to the over-under dataset.[9]

[7] imbal = imbalanced, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

[8] unov = under+over sampling, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

[9] ovun = over+under sampling, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

RBF SVM	**Accuracy**	**Macro Precision**	**Macro Recall**	**Macro F1-Measure**
imbal SumFun: acc	0.8295918	0.8550898	0.5310659	0.5920970
MIM imbal SumFun: acc	0.7979592	0.6490238	0.5143363	0.5473777
mRMR imbal SumFun: acc	0.8061224	0.7754409	0.5005006	0.5392404
JMI imbal SumFun: acc	0.8122449	0.8009481	0.5541936	0.6051948
imbal SumFun: MacroF1	0.8316327	0.7453155	0.5677005	0.6116604
MIM imbal SumFun: MacroF1	0.7979592	0.6490238	0.5143363	0.5473777
mRMR imbal umFun: MacroF1	0.8102041	0.6862728	0.5521653	0.5920768
JMI imbal SumFun: MacroF1	0.8122449	0.8009481	0.5541936	0.6051948

Table 18: Results for RBF SVM for the imbalanced dataset and the ones obtained from MIFS methods applied to the imbalanced dataset.[10]

RBF SVM	**Accuracy**	**Macro Precision**	**Macro Recall**	**Macro F1-Measure**
unov SumFun: acc	0.7857143	0.6034500	0.6245519	0.6117755
MIM unov SumFun: acc	0.7673469	0.5765646	0.6152277	0.5919380
mRMR unov SumFun: acc	0.7581633	0.5724443	0.5973501	0.5836145
JMI unov emp SumFun: acc	0.7806122	0.5887246	0.5880433	0.5883546
JMI unov mm SumFun: acc	0.7775510	0.6048920	0.6264691	0.6148956
unov SumFun: MacroF1	0.7857143	0.6034500	0.6245519	0.6117755
MIM unov SumFun: MacroF1	0.7673469	0.5765646	0.6152277	0.5919380
mRMR unov umFun: MacroF1	0.7581633	0.5724443	0.5973501	0.5836145
JMI unov emp SumFun: MacroF1	0.7775510	0.5865109	0.5989550	0.5918126
JMI unov mm SumFun: MacroF1	0.7775510	0.6048920	0.6264691	0.6148956

Table 19: Results for RBF SVM for the under-over dataset and the ones obtained from MIFS methods applied to the under-over dataset.[11]

RBF SVM	**Accuracy**	**Macro Precision**	**Macro Recall**	**Macro F1-Measure**
ovun SumFun: acc	0.7897959	0.6240217	0.6263759	0.6214769
MIM ovun SumFun: acc	0.7683673	0.5986517	0.6350224	0.6123292
mRMR ovun emp SumFun: acc	0.7632653	0.5756825	0.5933097	0.5836515
mRMR ovun mm SumFun: acc	0.7744898	0.6055289	0.6306827	0.6167256
JMI ovun emp SumFun: acc	0.7775510	0.5948640	0.6223814	0.6072963
JMI ovun mm SumFun: acc	0.7877551	0.6318606	0.6146617	0.6225439
ovun SumFun: MacroF1	0.7897959	0.6240217	0.6263759	0.6214769
MIM ovun SumFun: MacroF1	0.7683673	0.5986517	0.6350224	0.6123292
mRMR ovun emp umFun: MacroF1	0.7632653	0.5756825	0.5933097	0.5836515
mRMR ovun mm umFun: MacroF1	0.7744898	0.6055289	0.6306827	0.6167256
JMI ovun emp SumFun: MacroF1	0.7775510	0.5948640	0.6223814	0.6072963
JMI ovun mm SumFun: MacroF1	0.7816327	0.6263321	0.6357297	0.6306510

Table 20: Results for RBF SVM for the over-under dataset and the ones obtained from MIFS methods applied to the over-under dataset.[12]

[10] imbal = imbalanced, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

[11] unov = under+over sampling, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

[12] ovun = over+under sampling, SumFun = SummaryFunction, used in the train evaluation, acc = Accuracy, MacF1 = Macro F1-Measure.

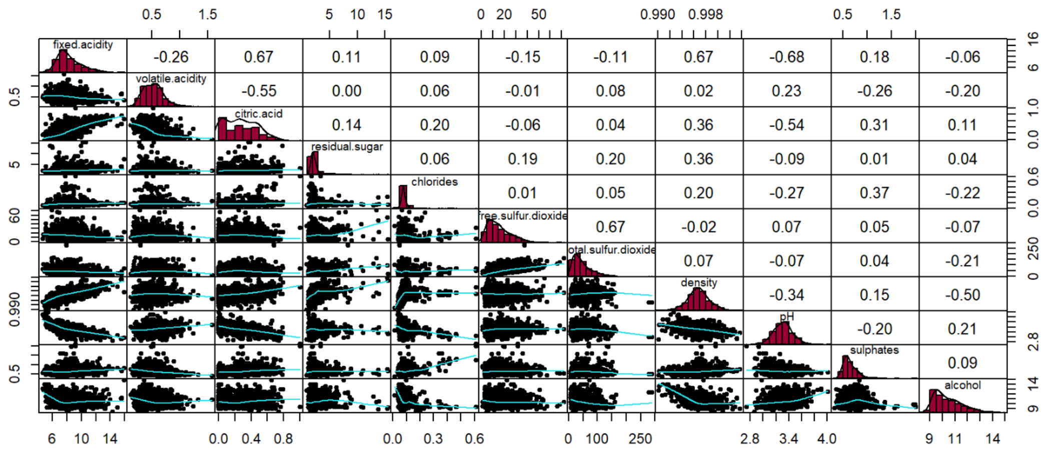


Figure 2: Pairs panels for the red wine dataset.[13]

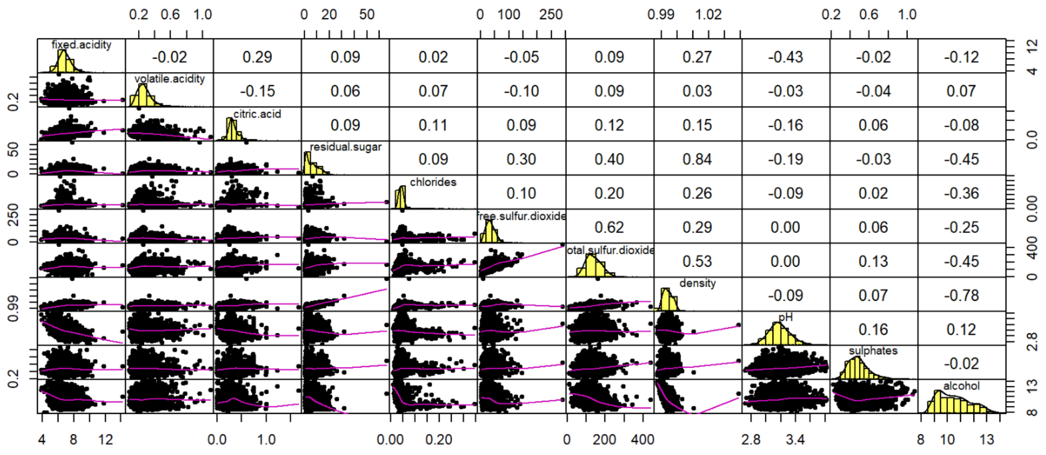


Figure 3: Pairs panels for the white wine dataset.

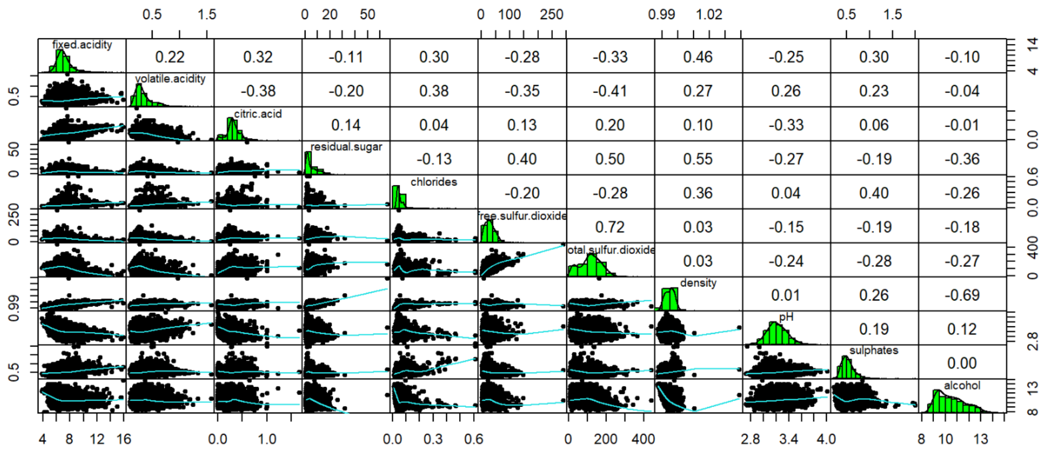


Figure 4: Pairs panels for the all wine dataset.

[13]Above the main diagonal of each panel we have the Pearson's bi-variate correlation values, in the main diagonal we have the histograms for each feature and below the main diagonal we have the bi-variate scatter plots of the features.

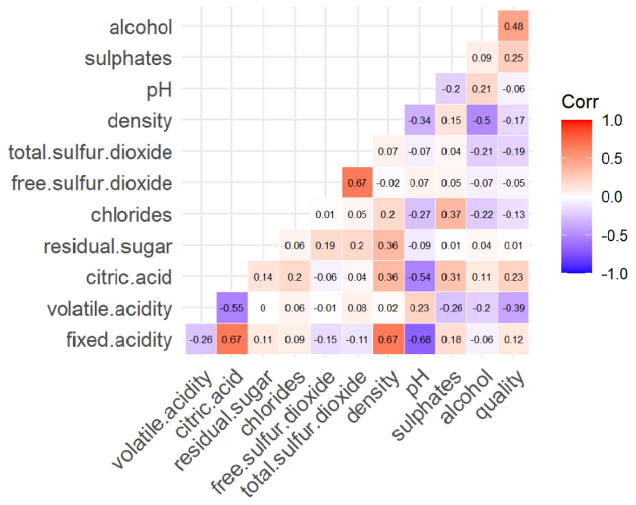


Figure 5: Correlation matrix for the red wine dataset.

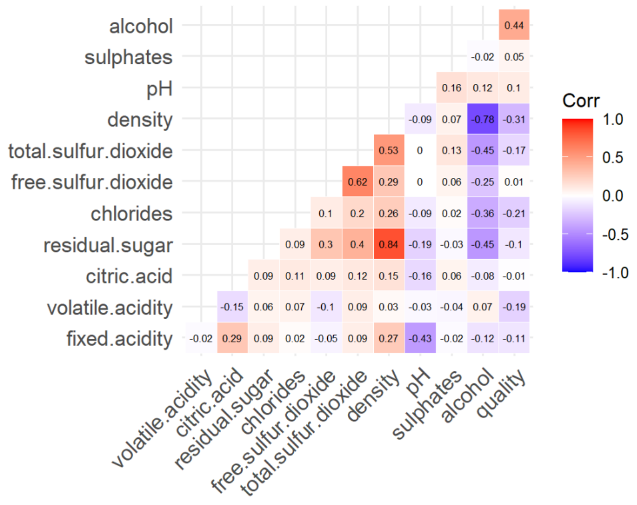


Figure 6: Correlation matrix for the white wine dataset.

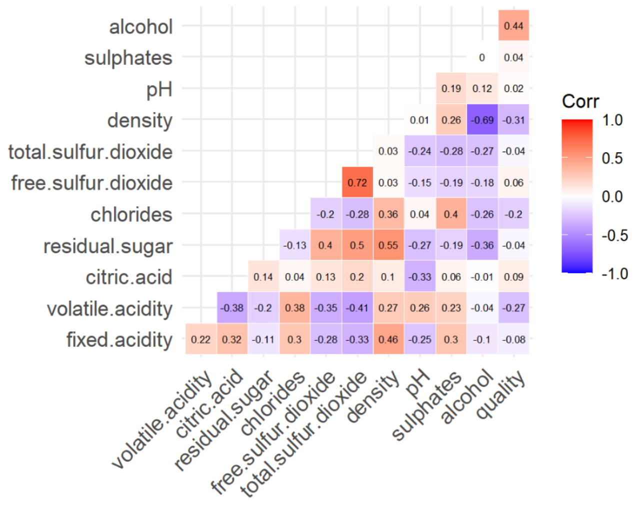


Figure 7: Correlation matrix for the all wine dataset.

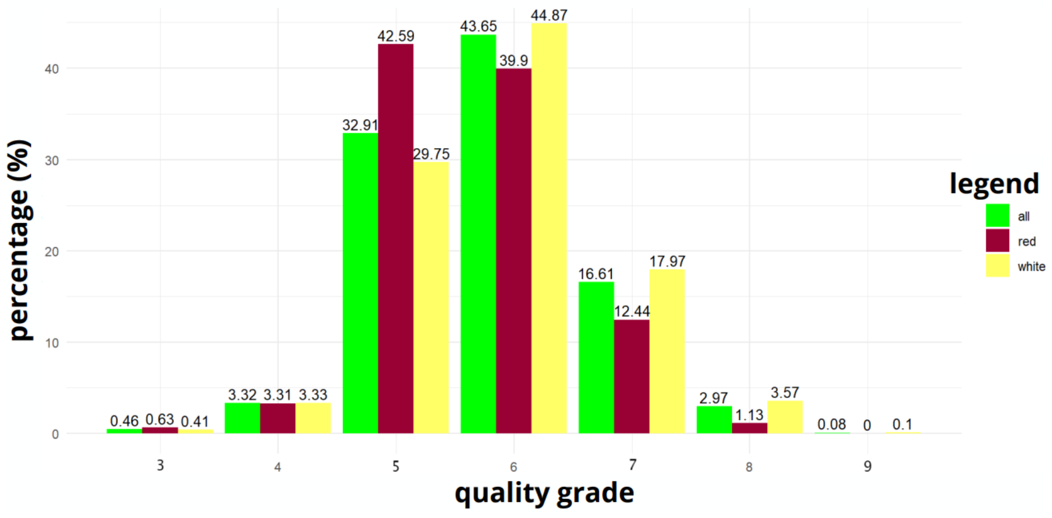


Figure 8: Histogram distribution for quality per wine type.

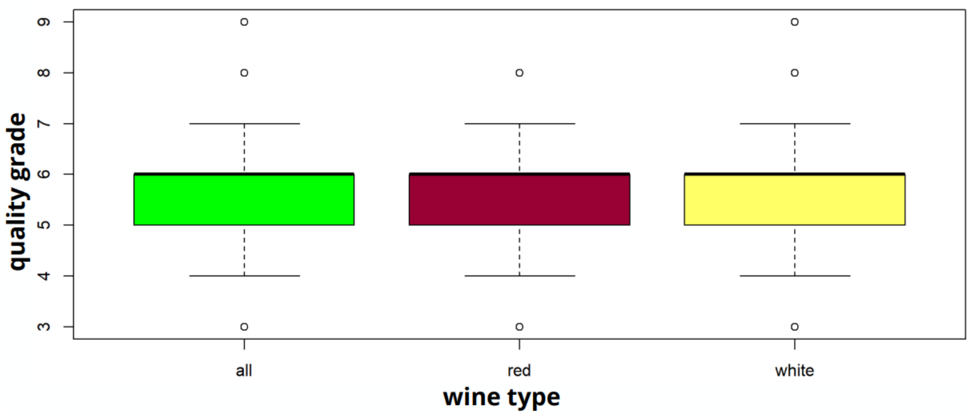


Figure 9: Box-plots for quality grade per wine type.