POLITECNICO DI TORINO

MATHEMATICS IN MACHINE LEARNING

MAURIZIO VASSALLO, s276961

Contents

1 INTRODUCTION

This is the analysis of **Vinho Verde** wine dataset [1] that contains two different type of wine: red and white. The task is to build a model in order to predict the wine quality given its psychochemical characteristics. Each sample of wine is evaluated by some expert who gave a score to it, from 0 to 10, based on its flavour and the final quality score is the median of the different scores, at least 3. The task to predict wine quality is important since high quality wine translates into an increasing of sales and selling price for that wine. Automate this process is useful since tastes can be very different from person to person and giving to the wine a score is difficult since wine quality is not absolute.
Said that, having an automatic evaluator is important.

2 DATA EXPLORATION

First phase of the analysis is exploring data in order to obtain the highest possible number of information before to apply models.
This dataset contains the results of a chemical analysis of two types of wine grown in the regions of Portugal. It contains 12 features representing the quantities of 11 constituents found in each sample of the wine and 1 feature representing the score given by the experts. The two types of wine: red and white have 2 different flavour so their datasets can not be concatenate, and they must be processed separately.

The 11 features are:

Input variables (based on physicochemical tests):

- **Acidity**
  Acids are one of the fundamental properties of wine and contribute greatly to the taste of the wine. Wines with higher acidity feel lighter-bodied because they come across as lightly sparkling. Reducing acids significantly might lead to wines tasting flat. There are different types of acids:
  - **fixed acidity** (g/dm^3): Fixed acids include tartaric, malic, citric, and succinic acids which are found in grapes. These are fixed since they do not evaporate readily;
  - **volatile acidity** (g/dm^3): These acids are distilled out from the wine before completing the production process. It is primarily constituted of acetic acid though other acids like lactic, formic and

butyric acids might also be present. Excess of volatile acids are undesirable and lead to unpleasant flavor, vinegar taste;

- **citric acid** (g/dm^3): This is one of the fixed acids which gives a wine its freshness. Usually most of it is consumed during the fermentation process and sometimes it is added separately to give the wine more freshness;
- **pH**: Also known as the potential of hydrogen, this is a numeric scale to specify the acidity or basicity the wine. Fixed acidity contributes the most towards the pH of wines. Solutions with a pH less than 7 are acidic, while solutions with a pH greater than 7 are basic, with a pH of 7 the solution is neutral. Most wines have a pH between 2.9 and 3.9 and are therefore acidic.

- **Sweetness**
  Our human perception of sweet starts at the tip of our tongue. Often, the very first impression of a wine is its level of sweetness. When winemaking happens, yeast eats sugar and makes ethanol (alcohol) as a by-product, depending on the quantity of residual sugar after the fermentation the wine can be from dry to very sweet.

  - **residual sugar** (g/dm^3): This typically refers to the natural sugar from grapes which remains after the fermentation process stops, or is stopped. It's rare to find wines with less than 1 $gram/liter$ and wines with greater than $45 grams/liter$ are considered sweet;

- **Salty**
  Salty is not a common wine descriptor. Salinity is a concern in dry locations when frequent irrigation increases soil salinity, which increases wine salinity.

  - **chlorides** (g/dm^3): Chloride concentration in the wine is influenced by the soil and its highest levels are found in wines coming from countries where irrigation is carried out using salty water or in areas with slightly salty terrains. This is usually a major contributor to saltiness in wine;

- **Sulfites**
  Sulfites in wine are chemical compounds (sulphur dioxide, or SO_2) that occur naturally, to varying degree, in all types of wine. Sulfur Dioxide is naturally found in wines and is a product of fermentation, but most winemakers choose to add a little extra to prevent the growth of undesirable yeasts and microbes, as well as to protect against oxidation. Sulfur dioxide inhibits yeasts, preventing sweet wines from refermenting in

the bottle. It's an antioxidant, keeping the wine fresh and avoid the presence of oxygen.

  - **sulphates** (g/dm^3): These are mineral salts containing sulfur. They are a regular part of the wine-making process around the world and are considered essential. They are connected to the fermentation process and affects the wine aroma and flavor;
  - **free sulfur dioxide** (mg/dm^3): This is the part of the sulphur dioxide that when added to a wine is said to be free after the remaining part binds. Winemakers will always try to get the highest proportion of free sulphur to bind. They are also known as sulfites and too much of it is undesirable and gives a pungent scent.
  - **total sulfur dioxide** (mg/dm^3): This is the sum total of the bound and the free sulfur dioxide.This is mainly added to kill harmful bacteria and preserve quality and freshness. There are usually legal limits for sulfur levels in wines and excess of it can even kill good yeast and give out undesirable scent.

- **Alcohol**
  Alcohol is formed as a result of yeast converting sugar during the fermentation process. The percentage of alcohol can vary from wine to wine. We interpret alcohol using many different taste receptors which is why it can taste bitter, sweet, spicy, and oily all at once. Regardless, we can all sense alcohol towards the backs of our mouths in our throats as a warming sensation.

  - **alcohol**: It's measured in % vol or alcohol by volume.

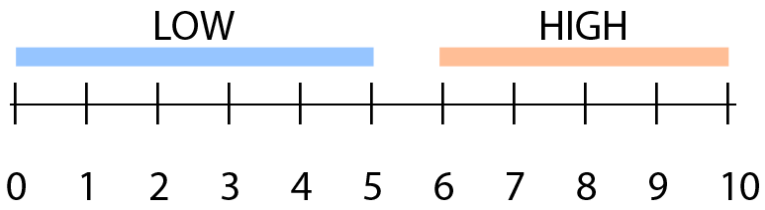
- **Body**
  Body in wine is the result of many factors – from wine variety, where it's from, vintage, alcohol level and how it's made. Body is a snapshot of the overall impression of a wine.

  - **density** (g/cm^3): This can be represented as a comparison of the weight of a specific volume of wine to an equivalent volume of water. It is generally used as a measure of the conversion of sugar to alcohol.

Output variable (based on sensory data):

  - **quality** (score between 0 and 10): as discussed before, wine experts graded the wine quality between 0 (very bad) and 10 (excellent);
  - **label** (derived from the quality feature): the samples are placed in groups: **low** and **high** quality labels according to the following rule:

wines with quality less or equal to 5 are labeled as low and wines with quality higher than 5 are labeled as high;



[https://winefolly.com/deep-dive/wine-characteristics/]

The two datasets have different sizes:

```
White:
(4898, 13)
--- --- ---
Red:
(1599, 13)
```

As we can see the white wine dataset is larger than the red one.

2.1 Null Values

The next step is perform a control whether there are *null/nan* values:

	Wine Type	
Name Feature	White	Red
fixed acidity	0	0
volatile acidity	0	0
citric acid	0	0
residual sugar	0	0
chlorides	0	0
free sulfur dioxide	0	0
total sulfur dioxide	0	0
density	0	0
pH	0	0
sulphates	0	0
alcohol	0	0
quality	0	0

There is no *null/nan* value for both the white and red wine. It is important to deal with null values since these can cause problems when apply a machine learning algorithm but also because they can distort the analysis. There are different ways to deal with null values:

- Remove them. This is usually done when the dataset is large,
- Replace these values with something meaningful like mean or median. This in the other case is performed when the dataset is small and so avoid removing samples.

2.2 Duplicates Values

Then a check on duplicates is performed. Duplicates values can cause problems like null values since the same sample can be both in the train set and in the validation set, making the evaluation not fair.
Here the results:

```
White:
937, ratio 0.1913
--- --- ---
Red:
240, ratio 0.1501
```

It is possible to see that there are a lot of duplicates. Since the ratios are large, I decided to perform a deeper analysis on the indices of these duplicates: I found out that some of them are one next to other (e.g. for white wine dataset rows: 3 and 4, 4879 and 4880 are equal) and one possible cause can be that the same sample was inserted two times accidentally, also because the features has 4 decimal digits and it is very unlikely that all the features have the same values. These duplicates are removed.
From this is possible to deduct that the dataset was not accurately build and some other problems may appear in the following steps.

2.3 Quality Distribution

The datasets quality feature is labelled with an integer number from 0 to 10:

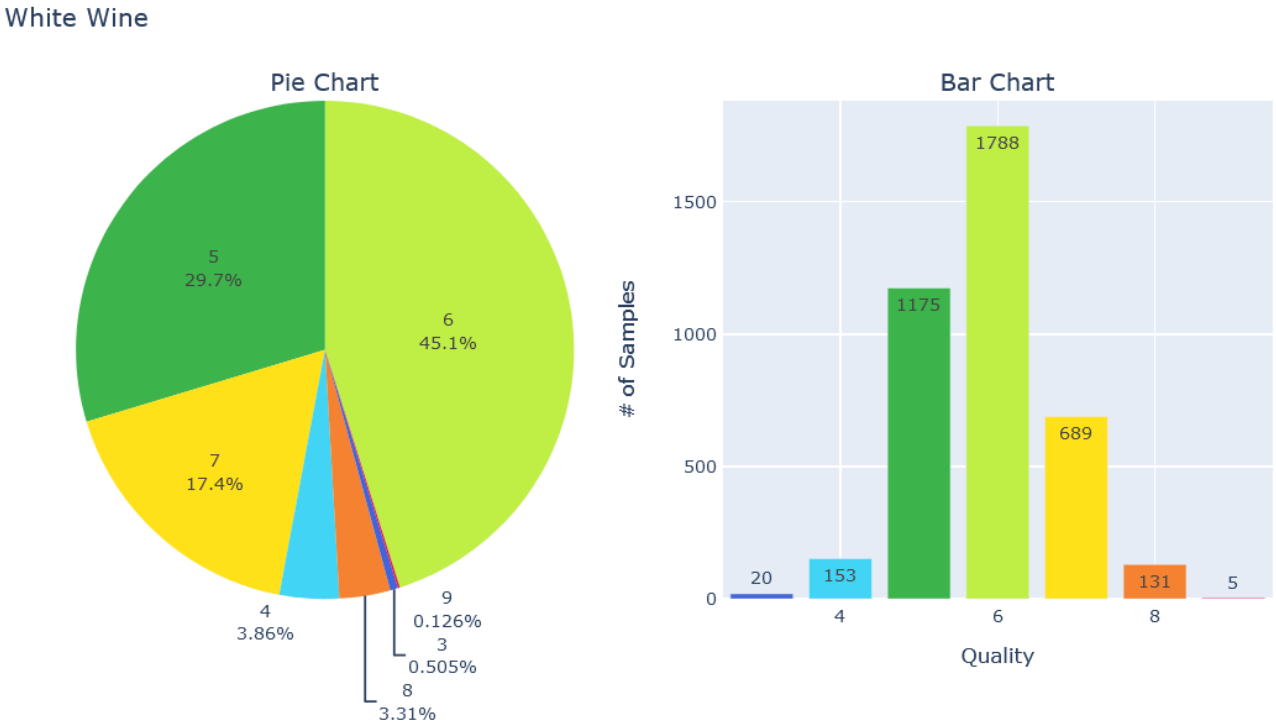


Figure 1: White wine quality distribution

Most of the figure are made with ***Plotly*** *so they can be opened on a web page following the appropriate link (usually under the figure itself).*

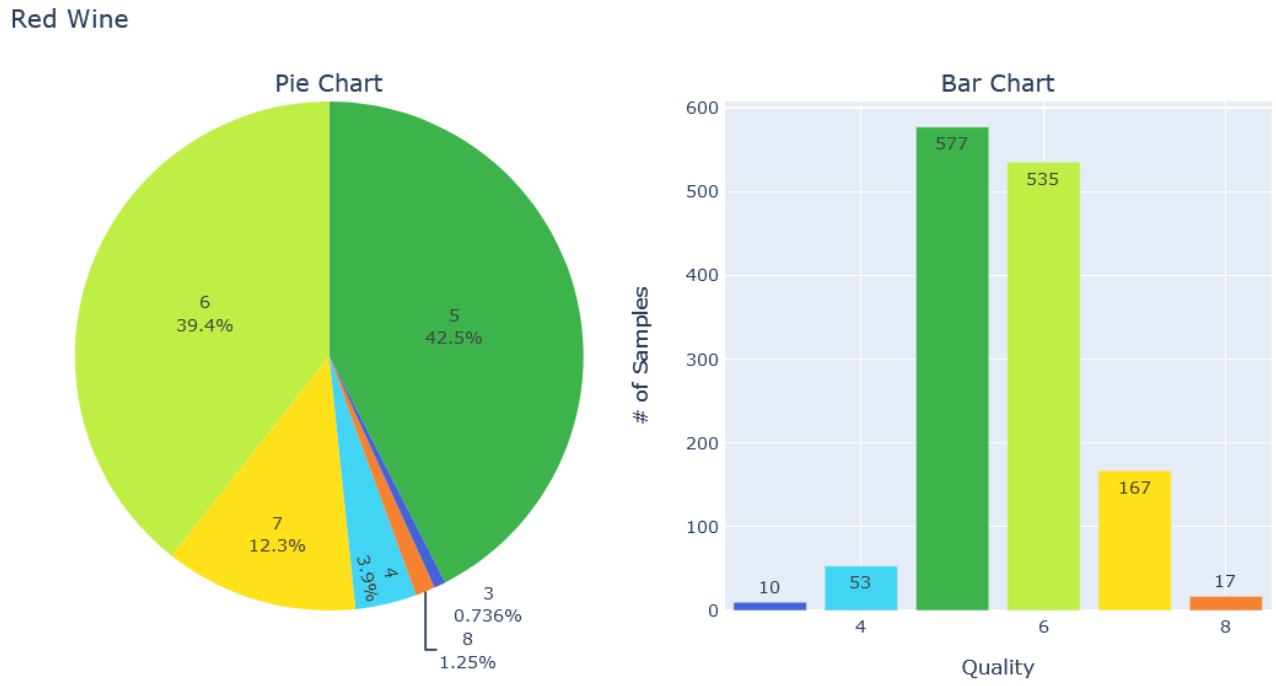


Figure 2: Red wine quality distribution

These are the graphs for the class distribution of the two datasets after removing the duplicates. From the graphs above it possible to conclude that the

classes are unbalanced: most of the samples wine have a medium quality (between 5,6,7) and the extremes (low and high quality) are low, samples with quality equal to 0,1,2 and 10 are not present.

And for the quality labels:

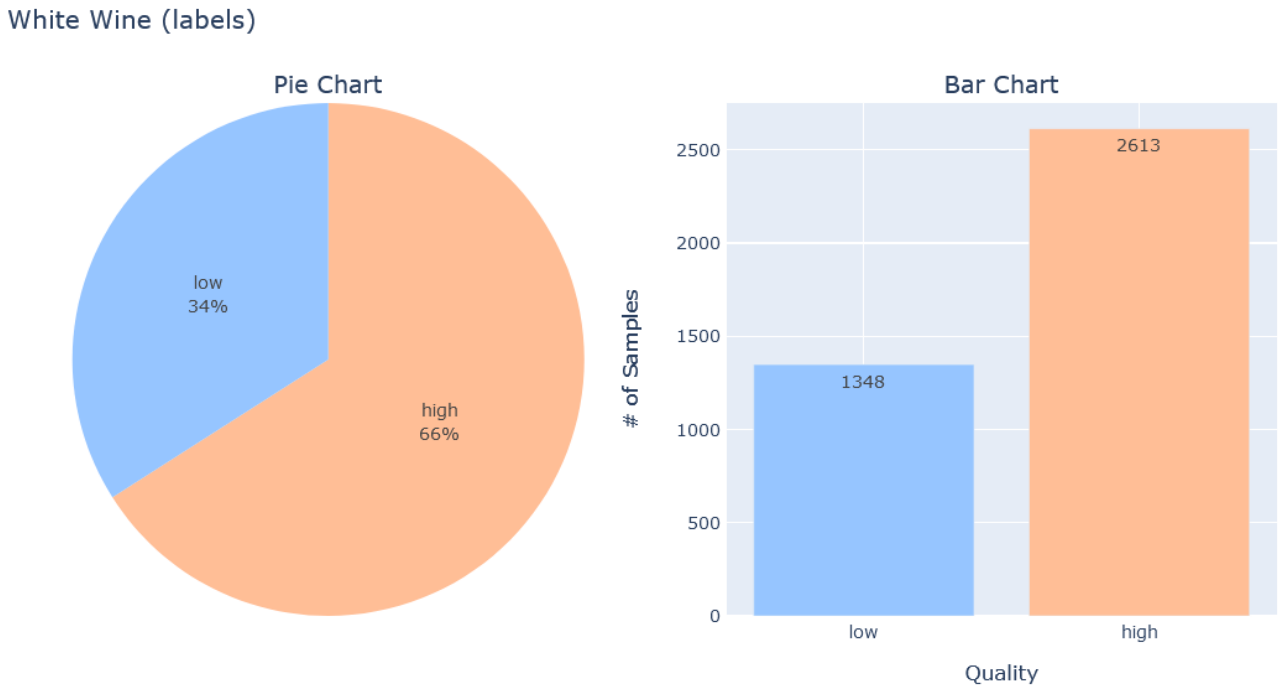


Figure 3: White wine quality label distribution

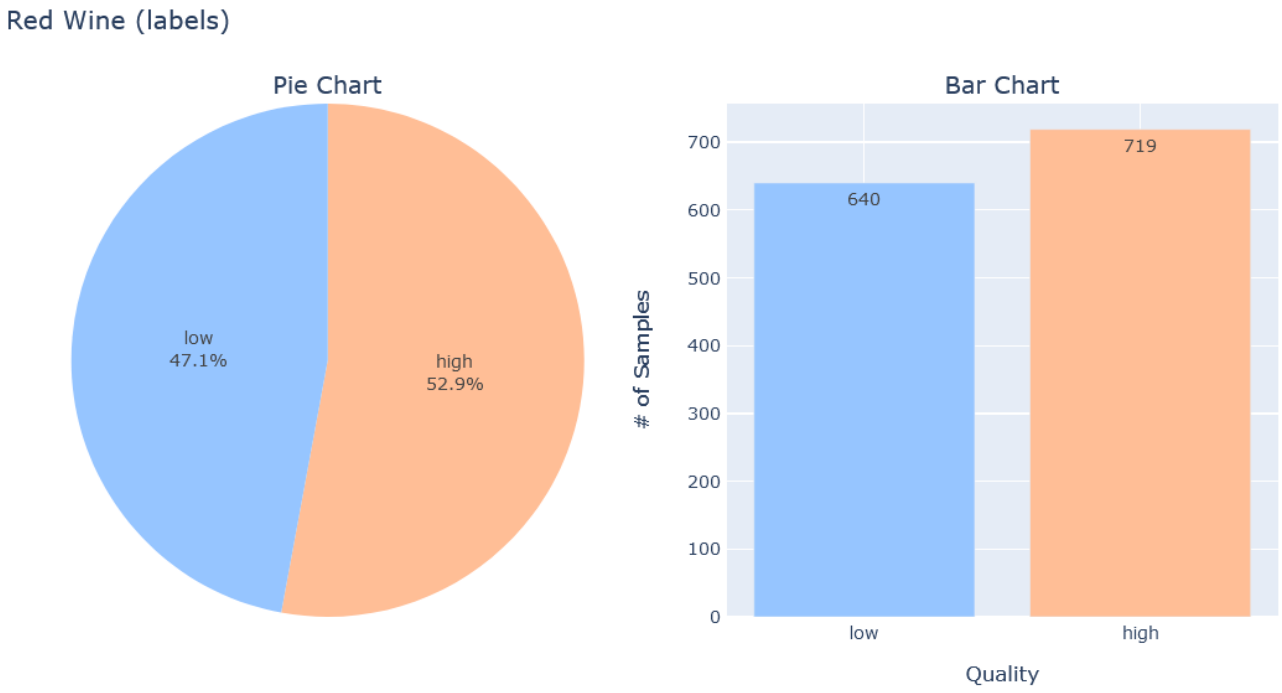


Figure 4: Red wine quality label distribution

It is possible to see that creating a label from the existing quality features make the classes more balanced, therefore a classifier should perform better.
This is useful to compensate the problem of subjectivity in evaluating a sample of wine and also understand better the differences between a low quality wine and an high quality one.

2.4 Univariate Analisys

The next step is to plot the features one by one in order to compute a univariate analysis. These graphs allow to have a clue of the general distribution of each feature.
The plots are omitted for avoiding using too much space. They can be found at this link: Univariate Distribution White and Red.
It is possible to see from the bar plots that almost all the features are symmetric with respect to the mean value (e.g. fixed acidity), only some ones are not (e.g. volatile acidity) because are more skewed. It is possible to note that the tails are long due to the presence of some outliers.

Data is also plotted in boxplots.
The box plot is a graphical representation describing the features of a distribution exploiting quartiles. A boxplot is a standardized way of displaying the dataset based on a five-number summary: the minimum, the maximum, the sample median, and the first and third quartiles.

- **First quartile** (Q1, 25th Percentile) : also known as the lower quartile $q_n(0.25)$, is the median of the lower half of the dataset;
- **Median** (Q2, 50th Percentile) : the middle value of the dataset;
- **Third quartile** (Q3, 75th Percentile) : also known as the upper quartile $q_n(0.75)$, is the median of the upper half of the dataset.
- **Interquartile Range (IQR)**. It is the distance between the upper and lower quartile:
$$IQR = QR3 - QR1 = q_n(0.75) - q_n(0.25)$$
- **Minimum**: the minimum acceptable value:
$$Min = Q1 - 1.5 * IQR$$
- **Maximum**: the maximum acceptable value:
$$Max = Q3 + 1.5 * IQR$$

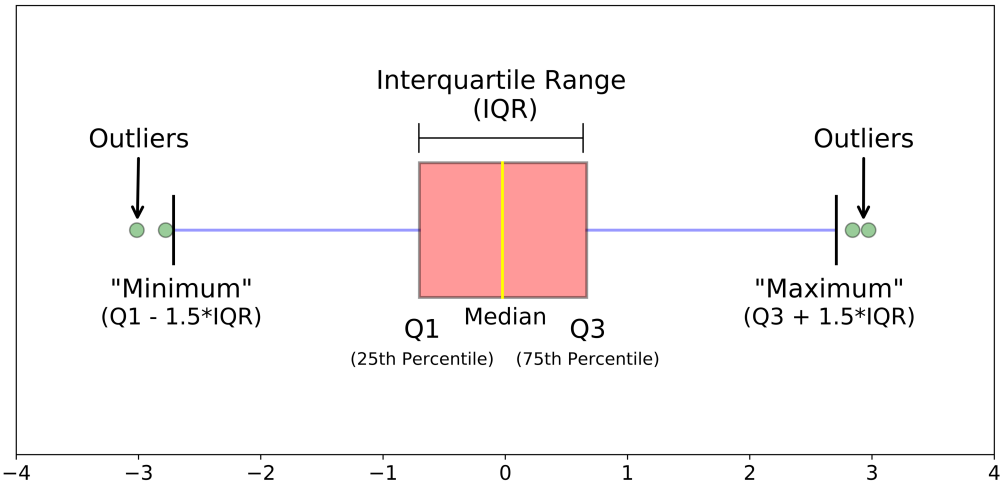


Figure of a BoxPlot: it is possible to see the different important values: minimum, Q1, median, Interquantile Range, Q3, maximum and outliers.

Box plot are excellent tools not only to understand the distribution of a variable but also for detecting outliers: points that are below the minimum value or above the maximum value.

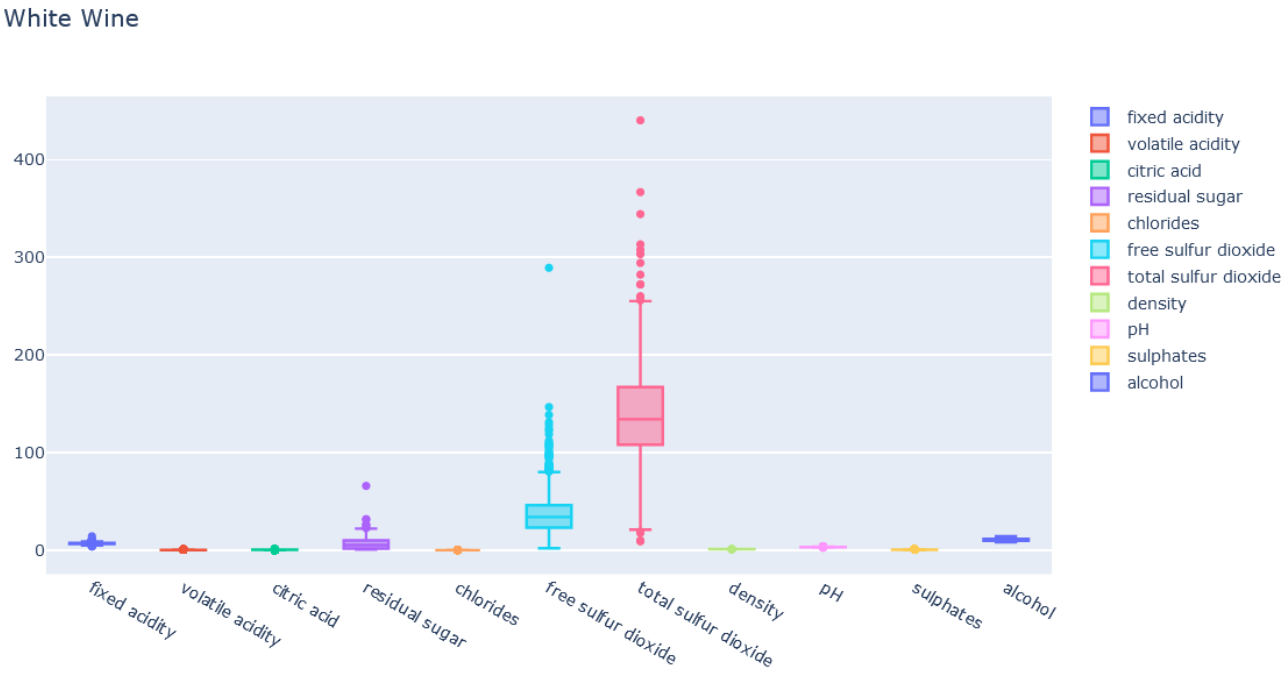


Figure 5: White wine boxplot

It possible to see that the scale of 'total sulfur dioxide' is very large with respect to the other features, so it is difficult to get some information.

The solution is to normalize the data using the z-scaler for visualization purposes, using the following formula:

$$z = \frac{x - min(x)}{max(x) - min(x)}$$

where x is our data and z in the new scaled data, with $0 \leq z \leq 1$

These are the results:

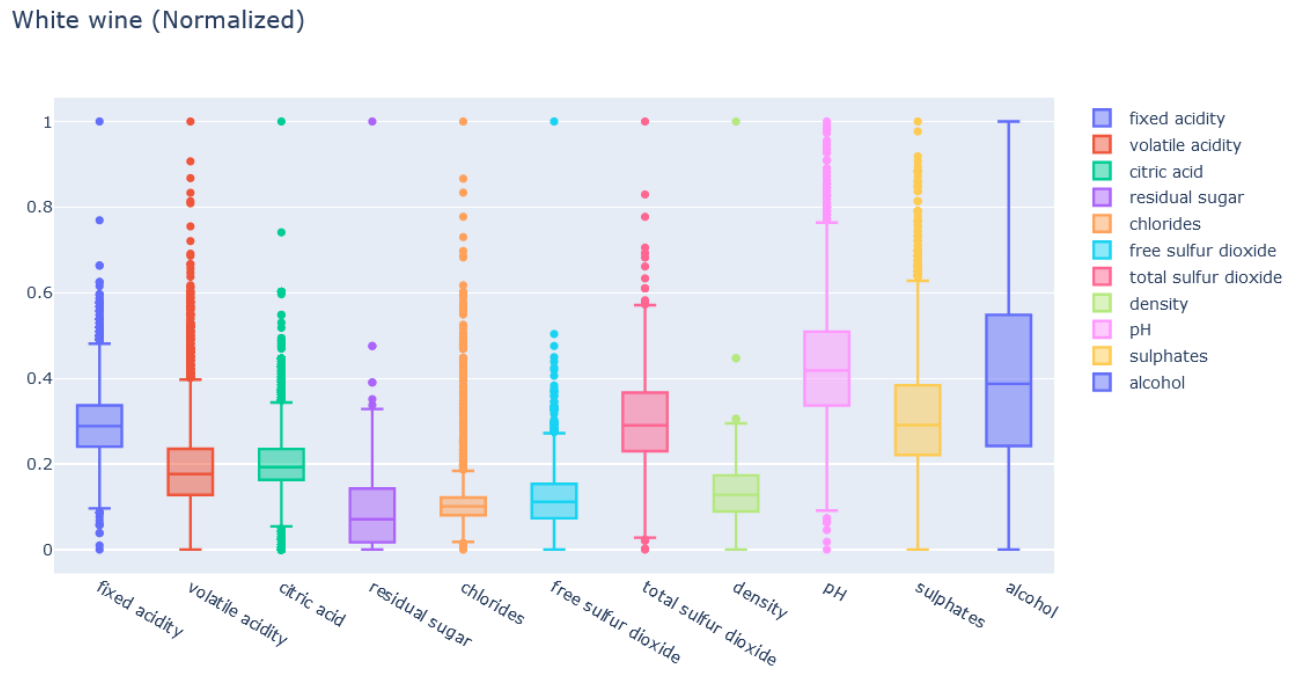


Figure 6: White wine boxplot

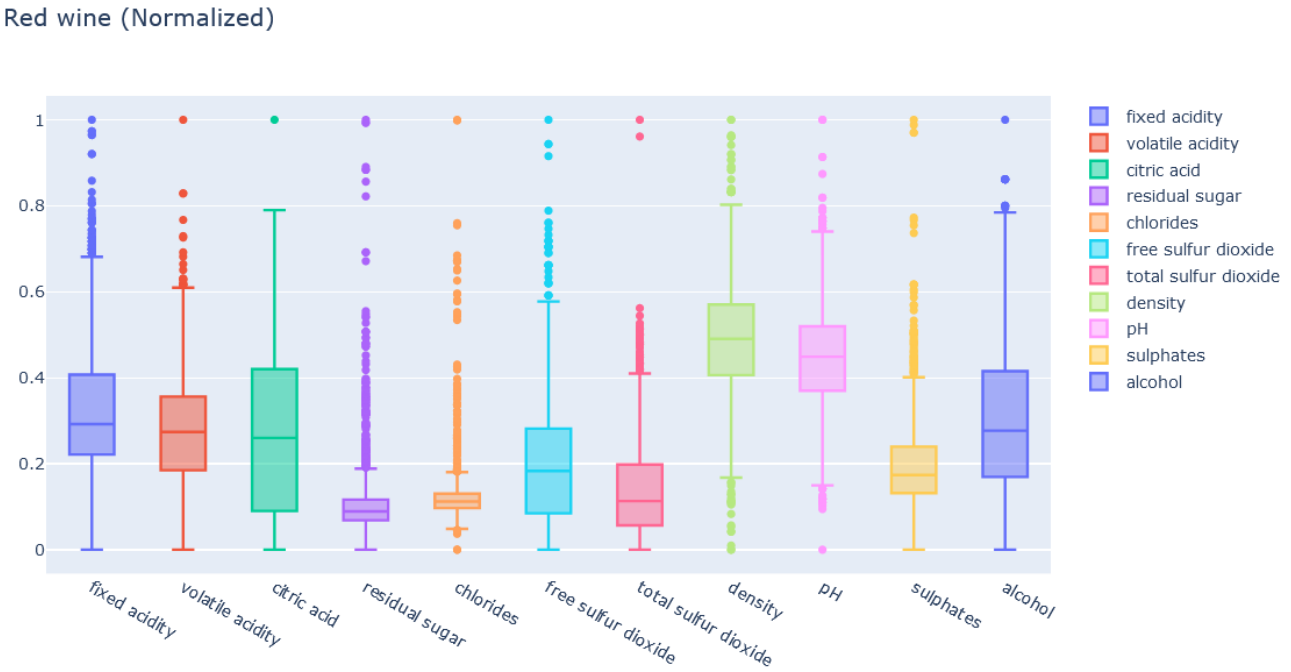


Figure 7: Red wine boxplot

From the boxplots with normalized data it is possible to see the distribution of the different features and the presence of some outliers.
These outliers are removed. I also tried to substitute their value with the median but the score dropped.

2.5 Bivariate Analisys

The next step is to scatter plot the features two by two in order to compute a bivariate analysis. The plots are omitted for avoid using too much space. They can be found at this link: Bivariate Analysis White and Red.
From the graphs it is possible to see that most of the features are not correlated, while some of them are, like:

- *density and residual sugar*: are highly correlated as expected, since higher residual sugar would increase the presence of sugar in the wine and so an increasing in density;
- *density and alcohol*: are highly negative correlated. This means that an higher value of density means less alcohol. This case is just the opposite of the previous one (*density and residual sugar*), since alcohol is produced "consuming" (thanks to fermentation) sugar;
- *alcohol and residual sugar*: are weakly negative correlated.

For the red wine there are the same correlation as the white wine dataset, but there are also:

- *citric acid and fixed acidity*: are highly correlated;
- *citric acid and volatile acidity*: are highly negative correlated;
- *pH and fixed acidity*: are weakly negative correlated;
- *pH and citric acid*: are weakly negative correlated.

Also from the scatter plots is possible to see that there is no a single couple of features that separate the classes well enough, this is important to know since it demonstrates that the wine qualities are not very well separated and some models could not obtain high score for example algorithm which requires linear separable data such as SVM with linear kernel.

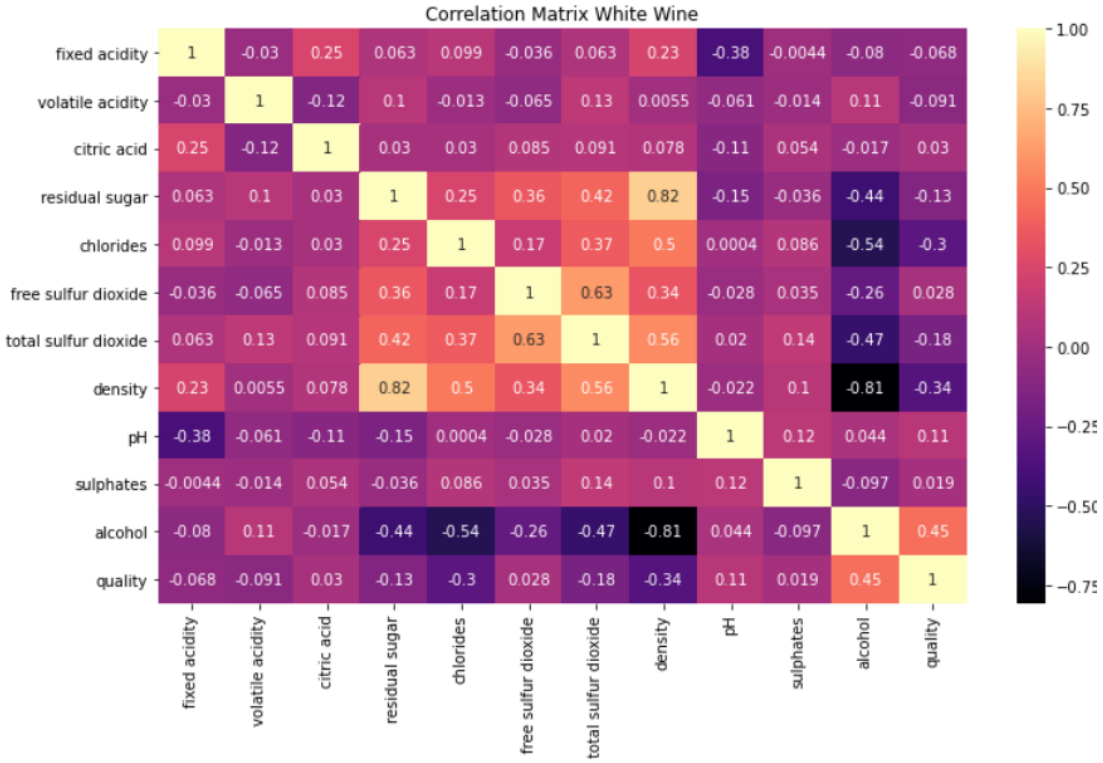
A correlation matrix is plotted in order to evaluate the correlation between the features.
The correlation matrix is a square matrix that contains the Pearson correlation coefficients which measure the linear dependence between pairs of features. The

features are listed in both x and y axis, and for each couple of features we have a number between -1 and 1 which represents the degree of positive or negative correlation between them. This value is calculated taking the Covariance of the two features and divide it by the product of their standard deviations, in formula:

$$Corr(X, Y) = \frac{Cov(X, Y)}{\sigma_x \cdot \sigma_y} \tag{1}$$

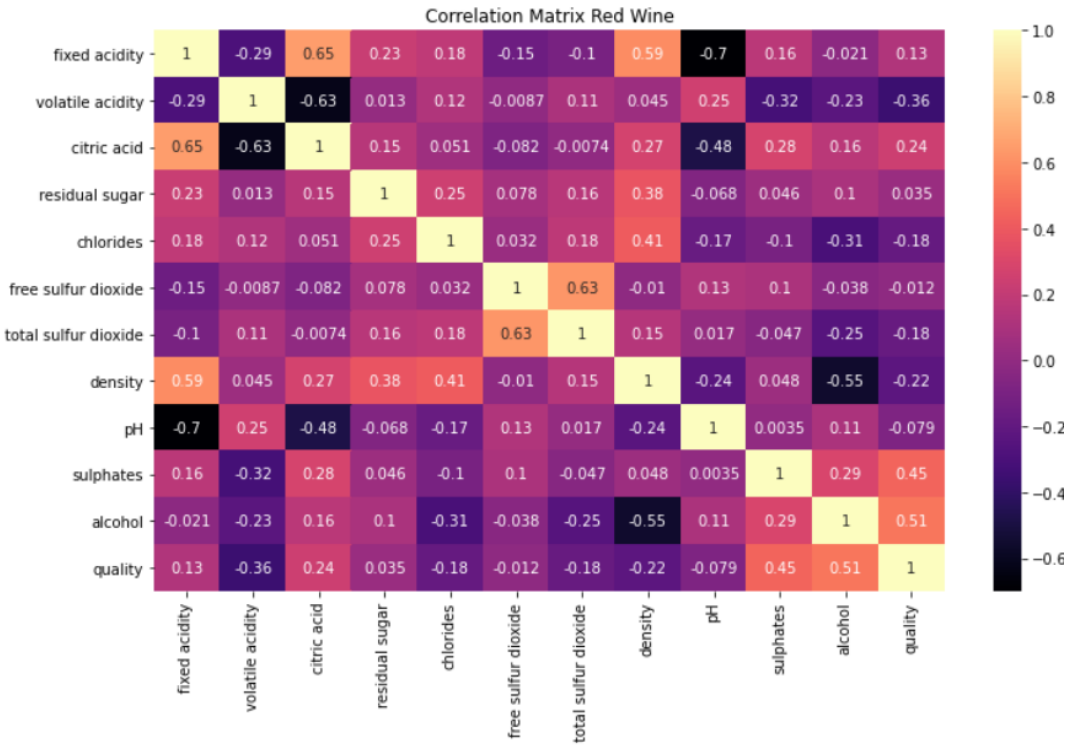
Where $Cov(X, Y) = \mathbb{E}((X - \mu_x) \cdot (Y - \mu_y))$.

These are the 2 scatter matrices:



Given these scatter matrix it is easier to understand the correlation between the features.
For the white wine, it is possible to see that:

- density and residual sugar are highly correlated. Their correlation coefficient is 0.82, very near to 1;
- total sulfur dioxide and free sulfur dioxide are highly correlated, with a correlation value of 0.63;
- alcohol and density: are highly negative correlated with a correlation value of -0.81, near to -1.



For the red wine, instead:

- citric acid and fixed acidity are highly correlated. Their correlation coefficient is 0.65;
- citric acid and volatile acidity are highly negative correlated. Their correlation coefficient is -0.63;
- total sulfur dioxide and free sulfur dioxide are highly correlated, with a correlation value of 0.63;
- density and fixed acidity are positive correlated;
- pH and fixed acidity are highly negative correlated.

3 PREPROCESSING

In the preprocessing part, data is processed in order to make it suitable for the different machine learning algorithms that I will use. In particular: the outliers will be removed, data will be standardized, a features reduction method will be applied and data will be splitted in training, validation and test sets.

3.1 Standardization

Standardization refers to shifting the distribution of each attribute to have a mean of zero and a standard deviation of one. This is obtained subtracting the mean and then dividing by the standard deviation:

$$z = \frac{x - \mu_x}{\sigma_x}$$

where x is our data, μ_x is the mean of x, σ_x is the standard deviation of x and z in the new scaled data.
It is important to standardize data to prevent features with wider ranges from dominating the distance metric and also to have better performances with different machine learning models. In particular, starndadization should be applied before:

- **PCA**:
  In Principal Component Analysis, features with high variances/wide ranges, get more weight than those with low variance, and they end up dominating the First Principal Components (Components with maximum variance).
  Standardization can prevent this, by giving same weight to all features.

- **SVM**:
  Since Support Vector Machine tries to maximize the distance between the separating plane and the support vectors, if one feature has very large values, it will dominate over other features when calculating the distance. So Standardization gives all features the same influence on the distance metric.

- **KNN**:
  k-nearest neighbors is a distance based classifier that classifies new observations based on similarity measures (e.g., distance metrics) with labeled observations of the training set. Standardization makes all variables to contribute equally to the similarity measures .

3.2 PCA

From the data the data best representative features are extracted using PCA method.
PCA (**P**rincipal **C**omponent **A**nalysis) is a statistics technique used for dimentionality reduction. PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system, in a lower dimensional space, such that the greatest variance of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.
Given a matrix $\mathbf{X}$, our data, of size $n \times p$, where n is the number of rows and p the number of columns. $\mathbf{X}$ has mean 0 and standard deviation 1, this is done to avoid that a feature with higher variance may be more important than a feature with lower variance.
Then multiply the transpose of this new normalized matrix by itself in order to obtain the covariance matrix, $\mathbf{X}^\top\mathbf{X}$.
Then decompose this $\mathbf{X}^\top\mathbf{X}$ matrix into $\mathbf{PDP}^{-1}$, where $\mathbf{P}$ is the matrix of eigenvector and $\mathbf{D}$ is a diagonal matrix with eigenvalues on the diagonal and values of zero everywhere else. the first element of $\mathbf{D}$ is λ_1 and the corresponding eigenvector is the first column of $\mathbf{P}$.
The eigenvectors are also called **principal components**. Choosing a subset (say m principal components, with $m < p$) of the principal components, I applied dimensionality reduction.
There are different ways to choose the number of principal components:

- Choose a fixed number. This is useful when you have constrains on the number of feature you need, say you have to plot the data in a 2D graph, then you choose the number of principal components equal to 2;
- Choose a fixed number of explained variance you need; say you need 90% of the total explained variance, then you will choose a number of feature such that they explain at least 90% of the total variance.
- Plot in a graph the number of principal components used and the cumulative explained variance. Then you look for an "elbow" in the graph.

I will use the 3^{rd} method, since I do not have any constraints on number of features or explained variance.

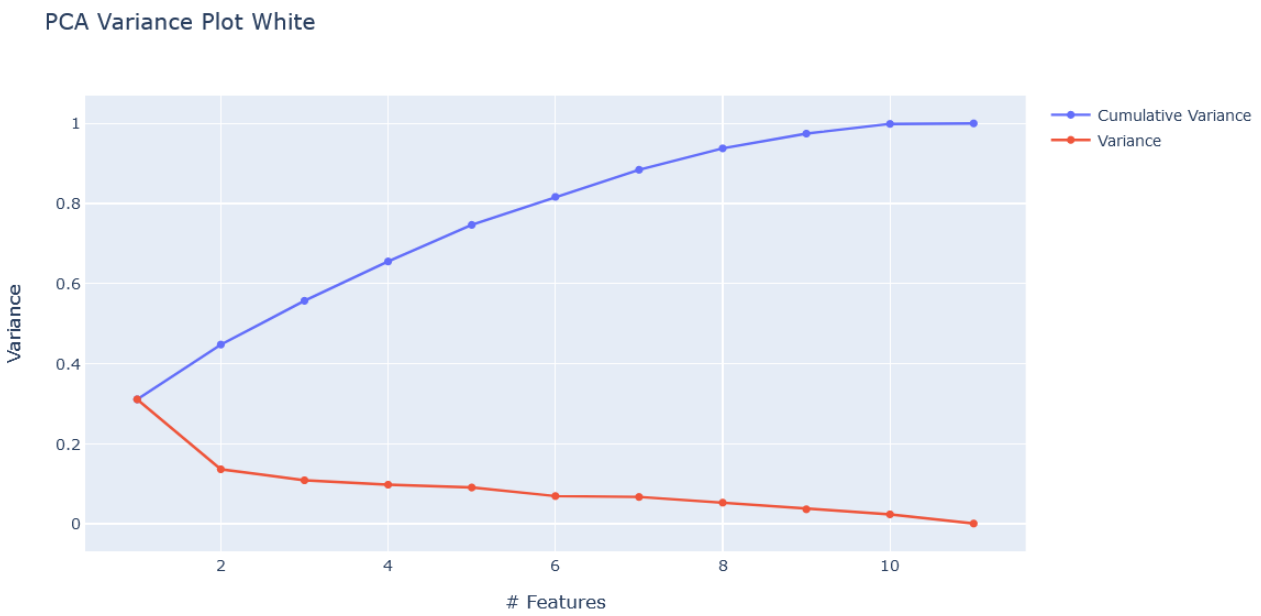


Figure 8: White Wine PCA Explained Variance: White

The graph for the red wine is omitted for sake of space, since it is similar to the white one. It can be found at this link: Red Wine PCA Explained Variance: Red.

It is possible to see from the graph that it does not present an important "elbow", so I will pick 7 of the 11 features that corresponds to around 0.9 of total variance.

3.3 Split Dataset

The dataset, given that the it is unbalanced, will be splitted using a stratified method. This method allows to divide a set in training and test set preserving the percentages, $p \in (0, 1)$, of samples of each class. The split is performed using the *train_test_split* method from the sklearn library. This process is repeated two time to divide the dataset in training, validation and test set. The overall proportions are: 6-2-2.
The different models will be trained on the train set, the will be evaluated on the validation set to choose the best hyperparameters and finally the model will train with both train and validation sets and tested on the test set to obtain the final score.

4 MODEL APPLICATION

4.1 SVM

Support-**V**ector **M**achines (**SVM**) are supervised learning models used for classification and regression analysis. The algorithm tries to find the best hyperplane that separates two classes; in particular, given a training dataset of n points x_i, where x_i is a n-dimensional vector and y_i, which indicate the class to which the point x_i belongs, we want to find the "maximum-margin hyperplane" that divides the groups of points x_i that belong to one class from the points x_i that belongs to the other class. The hyperplane has equation:

$$w^T x + b = 0,$$

where w is a vector that represents the hyperplane's normal.

If the points are linearly separable, we can find an hyperplane such that the margin, the minimum distance from the hyperplane and each points, is the maximum possible. Since the margin can be written as $\frac{2}{||w||}$ and we want to maximize this, which is equivalent to:

$$\min \frac{||w||^2}{2}$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1, \forall i$$

An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those x_i that lie nearest to it. These x_i are called **support vectors**.
But not always the points are linearly separable, in these case we can still apply SVM but with soft-margin. The idea is to allow some margin of error (misclassification) To extend SVM to this case we have to introduce slack variables the Hinge Loss function:

- The slack variables are positive variables ($\xi_i \geq 0$) that relaxes the "hard" condition of linear separability. The new objective function is:

$$\min \frac{||w||^2}{2}$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \boldsymbol{\xi_i}, \forall i$$
$$\boldsymbol{\xi_i} \geq \mathbf{0}, \forall i$$

- To calculate the value of these slack variables we use the Hinge Loss function:

$$max(0, 1 - y_i(w^T x_i + b))$$

  We want the values of the slack variables to be the minimum possible. This function is zero when the point is in the correct part of the hyperplane. For data on the wrong side the function's values is proportional to the distance from the margin.

Given these concept the overall SVM objective function is:

$$\min\{\frac{||w||^2}{2} + \boldsymbol{C}\sum_{\boldsymbol{i}} \boldsymbol{\xi_i}\}$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0, \forall i$$

where the parameter C determines the trade-off between increasing the margin size and ensuring that the x_i lie on the correct side of the margin. Thus, for sufficiently large values of C, the second term in the loss function will become huge, hence, it will behave similar to the hard-margin SVM.

Even with soft margin SVM it may be not possible to find an appropriate hyperplane to separate the data. In these cases it may be helpful to use the *Kernel Trick*. This is a "trick" to make computation faster. In formula:

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

Where $K(x, x')$ is the kernel function, x and x' are n-dimension inputs, $\Phi(\cdot)$ is the mapping function that maps the input data from the input space (n-dimensional space) to the feature space (m-dimensional space) and $\langle \cdot, \cdot \rangle$ is the scalar product.

Normally we would calculate first $\Phi(\cdot)$ and then the dot product, but this is expensive and not always possible, so we use the kernel function to avoid this step.

There are some hyperparameter to tune: C and Gamma.

- C is the degree of importance that is given to miss-classifications. Higher C higher means penalize the cost of miss-classification a lot.
  Tested values for C are: [0.001, 0.01, 0.1, 1, 10, 100]
- Gamma decides that how much curvature we want in a decision boundary (only specific kernels). The higher the Gamma value it tries to exactly fit the training data set.
  Tested values for Gamma are: [0.001, 0.01, 0.1, 1, 10, 100]

SVM can be used also for regression. Since the prediction would be a real number we would not be able to calculate the accuracy, but using a variation of the method used in the paper [2], the prediction is converted to a integer number (a number between 0 and 10) using the following formula:

$$p_i : \begin{cases} y_i & if \quad |y_i - \hat{y}_i| < T, \\ y_i' & if \quad otherwise \end{cases}$$

Where $\hat{y}_i$ is the model prediction, p_i is the modified prediction, y_i is the truth label, T is a real value number and y_i' is the approximation of $\hat{y}_i$ to the nearest integer.
This formula is applied on this and the next regression algorithms.

4.2 KNN

K-**N**earest **N**eighbors algorithm (**KNN**) is a supervised algorithm used for classification and regression.
KNN algorithm uses "feature similarity" to predict the class of new data points this means that the new data point will be assigned a value based on how closely it matches the points in the training set. So after the training phase, for a new point we calculate the distance between the test point and all the other points, we pick the k points with lowest distance and assign a class to the new point based on most frequent class of the k points.
The distance can be calculated using different methods like: Euclidean, Manhattan or Hamming distance. The most commonly used one is Euclidean.

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Distance between point x and y, two point in Euclidian n-dimentional space.

In case of a regression task the predicted value is the mean of the k nearest values:

$$p_x = \frac{1}{k}\sum_{i}^{k} c(x'_i)$$

Where p_x is the prediction of the model given as input the data point x, k is the number of nearest point to consider, x'_k are the k-nearest points to our point x and $c(x'_i)$ is the actual class of the k-nearest points.

The value k is an hyper parameter and the best choice of k depends on the data; generally, larger values of k reduces effect of the noise on the classification but make boundaries between classes less distinct. A good k value can be selected plotting the accuracy got with the different values of k and select the best performing one.
In this case I tried different values of k, between 2 and 150, plotted the accuracy obtained at with the validation set and then select the best performing k value.

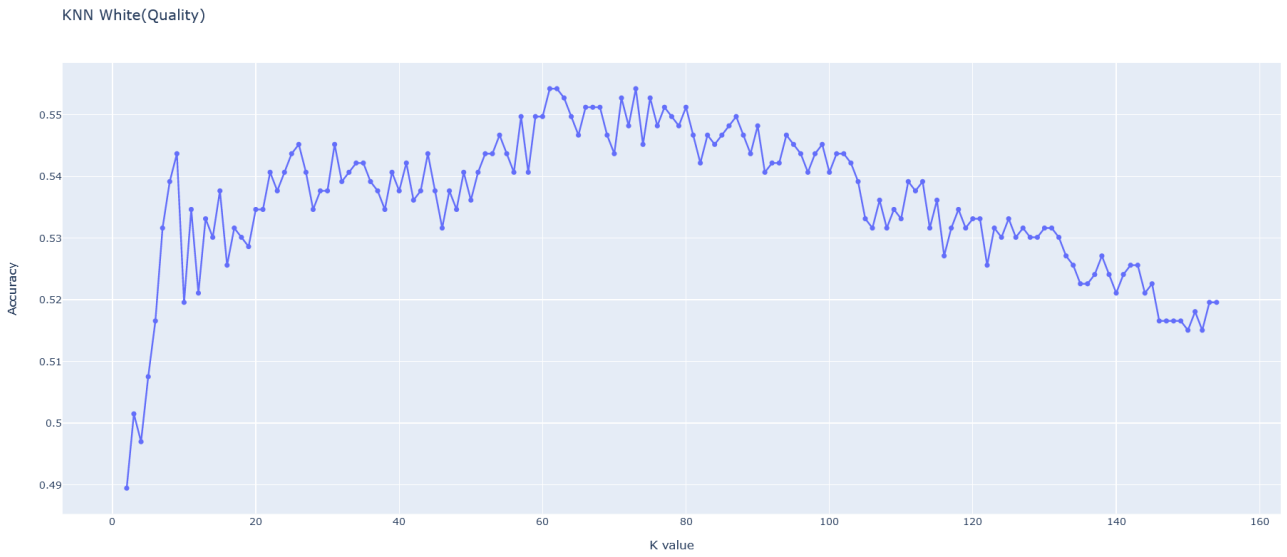


Figure 9: KNN plot: White Wine

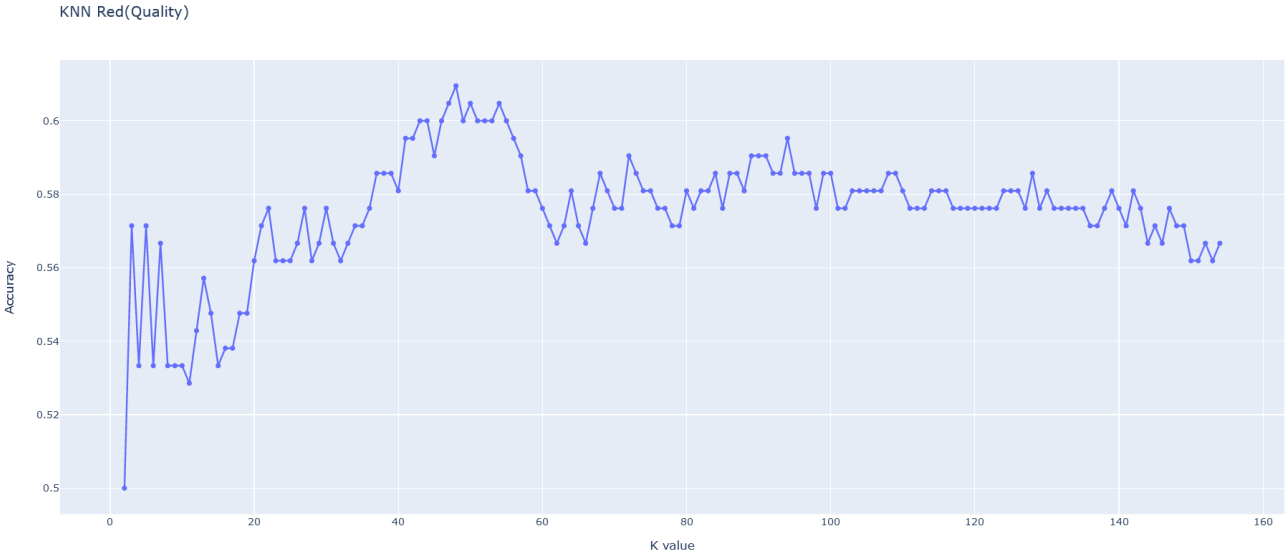


Figure 10: KNN plot: Red Wine

4.3 Random Forest

Random forest is a supervised learning algorithm used for classification and regression tasks.
The term "forest" refers to the construction of different decision trees, usually trained with the “bagging” method. Every tree is trained on a subset on samples randomly picked from the original dataset. These samples are chosen with replacement, this means that the same sample can be used multiple time to train different trees. After picking this sample points, we choose randomly some features on which the trees will be trained, so the split that the tree will learn. This method allows trees to be decorrelated and also prevent that a strong predictor to be the one that’s always included in the top split (since the split are computed

with a greedy method) and therefore avoid all trees looking very similar. A typical number for m (subset number of the p features) is $\sqrt{p}$, but its value depends on the problem's features number and their importances.
After the training phase, for the classification case, each tree predicts a class and the final output is the majority all these tree votes, this allows to have a more accurate and stable prediction.

For the regression case the final result is an average of the different trees output:

$$p_i = \frac{1}{n} \sum_{i}^{n} \hat{y}_i$$

Where p_i is the final prediction, n is the number of trees and $\hat{y}_i$ is the prediction of the i-tree.

The hyperparameter to optimize are:

- The number of trees, so how many prediction we will have.
  The values for number of trees are: 20 and 1000 with first step of 10 between 20 and 100 and then a step of 100.

- The max depth of each tree: so how deep the tree will be expanded.
  For the max depth values, I tried [10, 20, 50, 70].

5 Results

These are the accuracy results for the different algorithms:

REGRESSION				
	White Wine		Red Wine	
	T=0.5	T=0.75	T=0.5	T=0.75
SVM	0.538	0.692	**0.623**	**0.842**
KNN	0.519	0.742	0.571	0.828
Random Forest	**0.554**	**0.762**	0.590	0.766

CLASSIFICATION		
	White Wine	Red Wine
SVM	**0.775**	0.715
KNN	0.765	0.714
Random Forest	0.676	**0.771**

It is possible to see that the models performs well on regression task and better on the classification task, since in this case the classification task is an easier task.
In general Random Forest performs better than the other methods in the regression task for the white dataset and for the classification task for the red dataset, SVM performs well for the regression task for the red dataset and for the classification task for the white dataset and KNN performs worse than the other models in both cases of regression and classification.

I calculated also the Absolute Mean Error (MAE) for the regression task and the confusion matrix for both classification and regression task.
This is the confusion matrix obtained with the the best hyperparameters of the SVM model with the white wine dataset with T=0:
(Best parameters: Best C=1 and best Gamma=0.1 with validation accuracy=0.561747. Test accuracy=0.53614)

	3	**4**	**5**	**6**	**7**	**8**	**9**
3	0	0	0	2	0	0	0
4	0	0	9	6	2	0	0
5	0	0	90	92	1	0	0
7	0	0	45	244	22	0	0
6	0	0	10	72	45	0	0
8	0	0	1	16	6	0	0
9	0	0	0	1	0	0	0

True values on the vertical axis and predicted value on the horizontal axis.

Looking at the confusion matrix, it is possible to see the most well predicted values are the qualities with more samples. It is possible to see that the precision and recall for quality 4 and 8 are 0.
I tried to balance the dataset, first with SMOTE and then with ADASYN. These are the results, using SVM model as before:

SMOTE

	3	**4**	**5**	**6**	**7**	**8**	**9**
3	0	0	1	1	0	0	0
4	0	1	6	9	1	0	0
5	0	5	79	91	8	0	0
6	0	2	55	216	34	3	1
7	0	1	10	72	41	2	1
8	0	0	0	12	9	2	0
9	0	0	0	1	0	0	0

ADASYN

	3	**4**	**5**	**6**	**7**	**8**	**9**
3	0	0	1	1	0	0	0
4	0	1	8	8	0	0	0
5	0	5	69	99	9	1	0
6	0	3	65	190	50	3	0
7	0	1	9	73	41	3	0
8	0	0	2	14	7	0	0
9	0	0	0	1	0	0	0

SMOTE:(Best parameters: Best C=10 and best Gamma=1 with validation accuracy=0.883978. Test accuracy=0.45331)
ADASYN:(Best parameters: Best C=1 and best Gamma=1 with validation accuracy=0.882974. Test accuracy=0.47590)

For both methods I obtained a slightly decreasing in the accuracy of the model but the predicted values where able to get the qualities value that without the over-sampling where not correctly predicted, in particular the recall and precision of the quality 4 and 8 are slightly greater than 0 (at least for the SMOTE case).

6 References

1 https://archive.ics.uci.edu/ml/datasets/wine+quality

2 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