# *Wine Quality Dataset* Regression Problem

Simona Maria Borrello
*Politecnico di Torino*
Student id: s277789
s277789@studenti.polito.it

***Abstract*—This report takes wine quality evaluation as the research object and explores the relationship between different kind of factors and wine quality. Standard Data Science skills are used to clean, visualize and interpret data. Our analysis shows a correlation between wine quality and the winery and identifies which words are the most significant in wine reviews.**

## I. Problem overview

The aim of this work is to analyze a dataset which contains wine reviews, the rating of the wine and other relevant information.

The goal is, therefore, to identify which features are the most indicative of a good wine quality and build a regression model capable of detecting the rating, measured as an integer between 0 and 100 (see figure 1).
The dataset is divided into two parts:

- a development set, containing 120441 observations for which the score of the wine is provided;
- an evaluation set, comprised of 30186 observations.

The development and the evaluation sets only differ for one column: *quality*, our target column.
If we take a look at the development set, we can notice there are more than 30k duplicate data; particularly, if we decide to remove them from our analysis, the number of development records will drop to 84810. In our case, the presence of duplicate records sounds quite unusual because this assumes the presence of perfectly identical reviews. This fact could, for example, be caused by merging datasets from different resources. We will deal with this issue later on and we will decide whether to remove these observations or not.

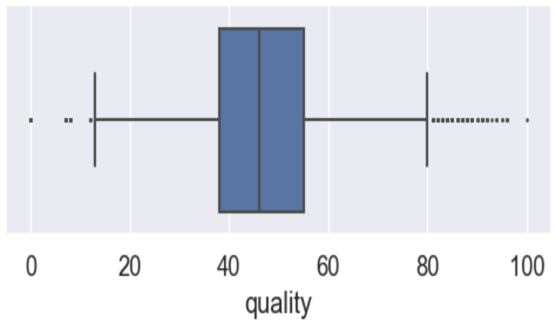

Fig. 1: Boxplot of Quality (Dev. set)

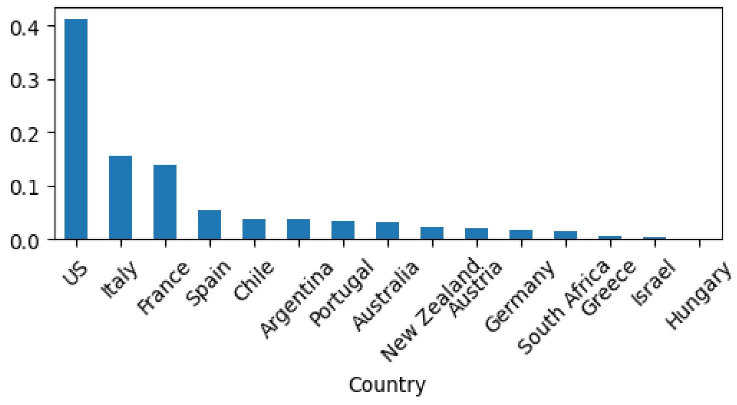

Fig. 2: Most 15 frequent countries

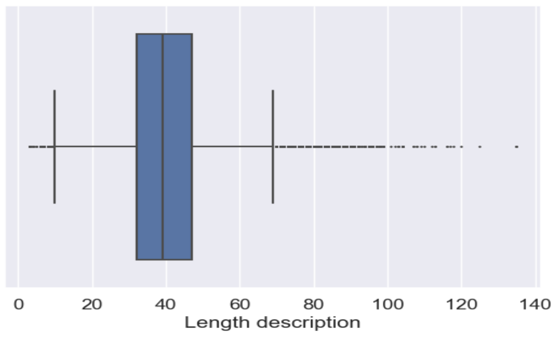

Fig. 3: Boxplot of number of words of reviews

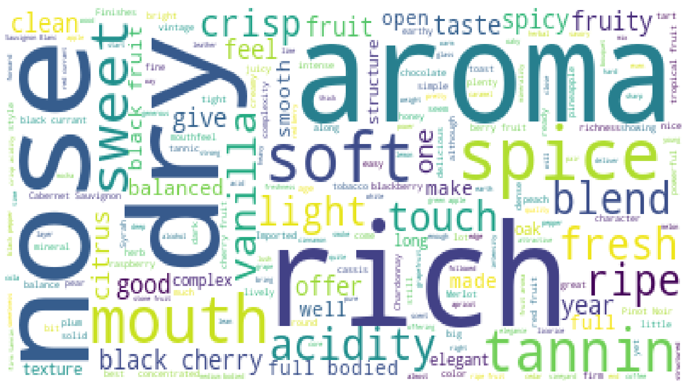

Fig. 4: WordCloud of reviews

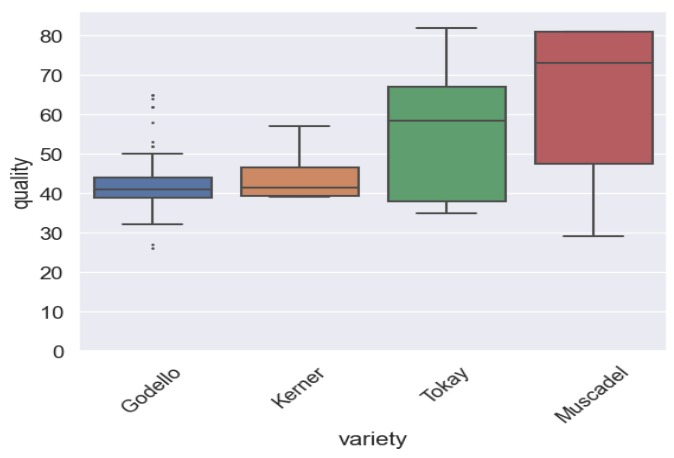

Fig. 5: Boxplot Variety-Quality (Dev. set)

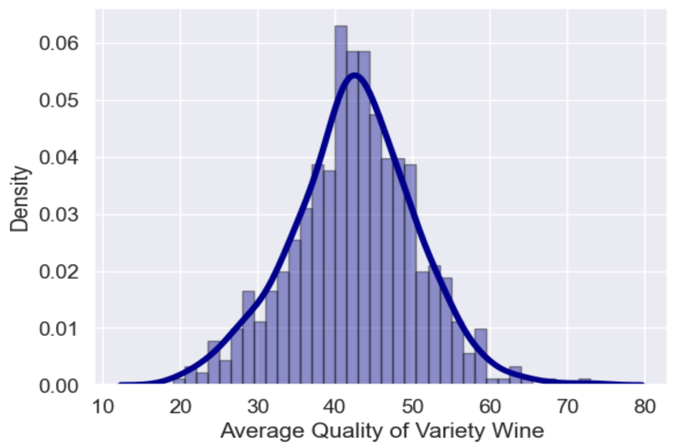

Fig. 6: Distribution of Average Quality of the Variety (Dev. set)

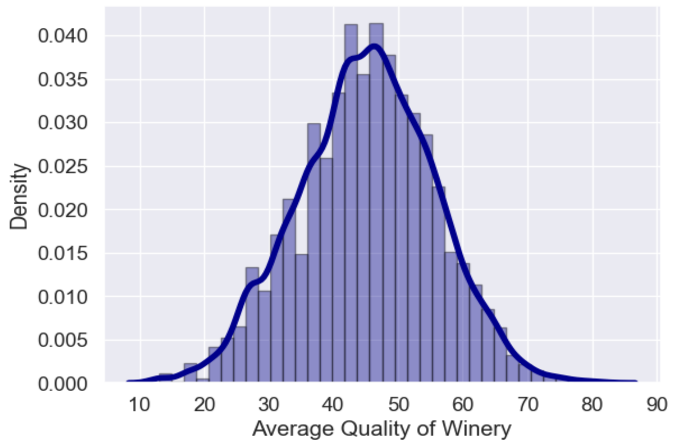

Fig. 7: Distribution of Average Quality of the Winery (Dev. set)

## II. Proposed approach

### A. Preprocessing

Data preprocessing is an important step in the data mining process.
First, we looked for outliers by means of z-scores. The higher the score, the higher the probability that one observation is an outlier. We considered as outliers observations whose z-score (with respect to the variable 'quality') lies outside the interval [-3,3] (for further details, see [1]). In this way, we dropped 303 observations from development set.
Now, we can take a look at the columns that comprise our data and make some considerations based on it:

- ***Country***. There are 48 different countries in the dataset, but, as we can see from figure 2, there are much more frequent countries than others. It was reasonable to rename the least frequent categories by "other", while we considered the most frequent 14 countries separately ( and converted them into dummy variables).
- ***Description***. This column contains the wine review. From figure 3, we can see that the reviews have different lengths; the shortest ones are 3 word long ( e.g. '*'Sweet and fruity'*), while the longest one has 135 words.
  We created a numeric feature, that keeps information about the length of the description. We also processed the textual data using *sklearn's TfidfVectorizer*, which splitted each review into tokens and removed stopwords. We refined the stopwords set to include common but not descriptive wine review words like *"wine"* or *"palate"*.
  In detail, we tried 2 different weighting schemes ( *tf-df* and *tf-idf*) and several combinations of TfidfVectorizer parameters ( *min_df* $\in \{0.1, 0.01, 0.001\}$, *max_df* $\in \{0.7, 0.8, 0.9\}$). We also changed N, the maximum number of words to extract, and let it vary in the range (150-1500). However, we noticed that all these changes did not affect much the performance, so we preferred the most effective method in terms of memory saving and computation time ( *min_df*= 0.8 , *max_df*= 0.01 and $N = 150$). Finally, the wordcloud shown in figure 4 gives us a graphic idea of the words' distribution.
- ***Designation***. Since this column has more than 30k distinct values and even the most frequent value is present in less than 0.02 % of the data, we decided to discard this feature. Before doing this, we introduced a new binary feature, called *'designation_presence'*, that takes into account whether or not, for a given observation, we have info about designation.
- ***Province***. First, we dropped observations in which the province contains missing values. This did not affect our analysis, because the number of these observations is negligible (only 5). Similarly to what was done for the variable *'Country'*, we decided to consider separately only the 10 most frequent provinces, while we discarded provinces that occur with a lower frequency than 0.02 %.
- ***Region1; Region2***. Since these variables assume a large number of distinct values, we considered separately only the 10 most frequent values. Also, we introduced 3 new binary features: two of these take into account if *region1/region2* are missing values, while the last one keeps the information about the equality, for a given

record, between *region1* and *region2*.

- ***Variety***. Variety is a categorical feature with 632 distinct values. In order to reduce the number of levels of this feature, we took a different approach with respect to the previous variables.
  Figure 5 shows the quality of wine for 4 different varieties in the development set. The first two boxes overlap so we may think of combining the two varieties in a single category. This graphic method could not be used because of the high number of categories, but we created, according to the average quality of the variety, several macro-categories. For this scope, we performed a quantile-based discretization ( binning into equal areas the density in figure 6). We used the function *'pandas.qcut'* and set the number of quantiles $q = 20$. For the 29 varieties that are in the evaluation set, but not in the development set, we provided a default class.
- ***Winery***. For this feature, which assumes 14809 distinct values, we used the same approach we followed for *Variety* and reduced the number of categories to $q = 50$. Figure 7 shows the average quality distribution of the wineries in the development set.

Before starting preprocessing, the dataset had only 8 features. At the end of this step, the number of features rose to 403.

## *B. Model selection*

We tested the following algorithms:

- ***Random Forest.*** Since random forests are based on decision trees and since decision trees work on one feature at a time, there is no need to normalize the dataset just yet.

The following models assume that the data they work with is in a standard range. So the normalization of feature vectors is very important.

- ***SVM.*** Since the dataset is very large, the algorithm provided by *sklearn.svm.SVC* took too much time for training. Therefore, we used *sklearn.svm.LinearSVC* that, according to the *sklearn* documentation, should scale better to large numbers of samples.
- ***Linear Regression.*** Linear Regression is a very simple algorithm. There is no hyperparameter to tune. We performed it in 2 ways: by changing the boolean parameter *fit_intercept* that allows us to specify if we want to include an intercept to the model.
- ***Ridge Regression.*** It is useful to reduce the complexity of the model and prevent overfitting. We tuned $\alpha$, which represents the regularization strength. Larger values specify stronger regularization.
- ***Lasso Regression.*** Like Ridge, it is a regularization technique but it also performs feature selection by setting some coefficients to zero.

## *C. Hyperparameters tuning*

Because of the size of the dataset, using cross-validation was too time consuming and we preferred the hold-out method in order to choose the optimal hyperparameters. We divided the development set in training (50%), validation (25%) and test set (15%). We used the first one to train model configurations, the second one to to verify which model configuration is the best one. Thus we trained the selected configuration on the union of the training and validation sets and we finally used the test set to compare the performance between our predictive models. Table (I) reports a summary of this tuning phase. In order to decrease computation time, in case of Random Forest, we did not try all combinations of hyperparameters, but we performed a randozimed search.

| **Model** | **Parameter** | **Values** |
|---|---|---|
| **Random Forest** | *n_estimators* | {50, 100} |
| | *max_depth* | {5, 50, 150, 200} |
| | *max_features* | {auto, log2, sqrt} |
| **SVM (Linear)** | $C$ | {0.001, 0.01, 0.1, 1} |
| **Linear Regression** | *fit_intercept* | {True, False} |
| **Ridge Regression** | $\alpha$ | {0.05, 0.1, 1} |
| **Lasso Regression** | $\alpha$ | {0.1, 0.001} |

TABLE I: Summary of Hyperparameters tuning

| N_estimators | Max_depth | Max_features | MSE | $R^2$ |
|---|---|---|---|---|
| 50 | 5 | *auto* | 92.143 | 0.344 |
| 50 | 50 | *auto* | 29.893 | 0.787 |
| 50 | 150 | *log2* | 30.873 | 0.780 |
| 50 | 150 | *auto* | 29.076 | 0.793 |
| 50 | 200 | *sqrt* | 38.321 | 0.727 |
| 100 | 5 | *log2* | 111.474 | 0.206 |
| 100 | 50 | *auto* | 32.244 | 0.770 |
| 100 | 50 | *sqrt* | 30.695 | 0.781 |
| 100 | 150 | *sqrt* | 27.961 | 0.801 |

TABLE II: Hyperparameters tuning for Random Forest

| C | MSE | $R^2$ |
|---|---|---|
| 1 | 91.205 | 0.351 |
| 0.1 | 61.086 | 0.565 |
| 0.01 | 51.221 | 0.635 |
| 0.001 | 50.668 | 0.639 |

TABLE III: Hyperparameters tuning for SVM

| Model | Configuration | MSE | $R^2$ |
|---|---|---|---|
| Linear Regression | *fit_intercept= True* | 37.627 | 0.732 |
| Linear Regression | *fit_intercept= False* | 37.622 | 0.733 |
| Ridge Regression | $\alpha$= 1 | 100.838 | 0.282 |
| Ridge Regression | $\alpha$= 0.1 | 38.321 | 0.727 |
| Ridge Regression | $\alpha$=0.05 | 37.846 | 0.732 |
| Lasso Regression | $\alpha$=0.1 | 143.153 | 0.001 |
| Lasso Regression | $\alpha$=0.001 | 37.819 | 0.730 |

TABLE IV: Hyperparameters tuning for Linear/Ridge/Lasso Regression

## III. Results

Table (II) shows that the best configuration for random forest is { $N\_estimators = 100$, $Max\_depth = 150$, $Max\_features = 'sqrt'$ }. Also, an interesting thing is that, for example, the improvement in $R^2$ is remarkable when we increased *Max_depth*= 5 to 50, while it is negligible when

we set *Max_depth* from 50 to 150.
As shown in table III, smaller values of $C$ correspond to better performances.
As we can see from table IV, including the intercept or not is not relevant in terms of the result. Moreover, it is clear that regularization techniques do not work well with our data. Indeed, high alpha values correspond to a drop in performance. As we could expect, when alpha tends to 0, the regularization techniques have similar results to those of linear regression.
Given the results, we decided to continue the analysis with only two models and we trained the best performing random forest and linear regression on the union of training and validation sets. So, we obtained,respectively, $R^2_{rf} = 0.828$ and $R^2_{lr} = 0.734$ on the test set. It is clear that random forest is the best model in terms of $R^2$, while linear regression outperforms all the other ones in terms of training time. At the end, we trained the best performing random forest on all available development data. The public score obtained is 0.826.

For comparison, we implemented 2 naive models: the former uses only features extracted from reviews as predictors (we just increased their number from 150 to 1000) and the latter,conversely, uses all the other variables except from *description*. Their public scores are, respectively, 0.678. and 0.808. The last result is surprising, because it is not very far from the best performance. Probably, this suggests that the information in reviews could be exploited more effectively, for example, by means of *sentiment analysis*.

## IV. Discussion

Random forest provides global feature importance, i.e. an estimate of which features are important in the prediction (see figure 8). Surprisingly, the first feature by importance is the length of the review, while in tenth place we find the fact that the wine has the designation or not. It is noteworthy that the most relevant features concern the winery, so we may think that there is a correlation between the quality of wine and the winery that produces it.

In figure 9, we can see the words that are selected by random forest as predictive variables. The size of the word in the figure is proportional to his importance in the random forest. If we analyze the content of the word cloud, we can group the words in macro-topics, such us colors, fruits, etc.

Also, we may repeat all the previous analysis without duplicates. For the sake of simplicity, we just ran the optimal configuration chosen on the dataset without duplicates. As we could expect, the performance ($R^2$) of the algorithms decreases (by only 0.03), probably because the training size decreases and the risk of data overfitting increases.
A further consideration is that, looking closely at the data, we notice that the quality value is an integer, so we modified the output of the predictive model and rounded it to the nearest integer. However, this change did not impact the performance. The results obtained, however, are quite satisfactory.

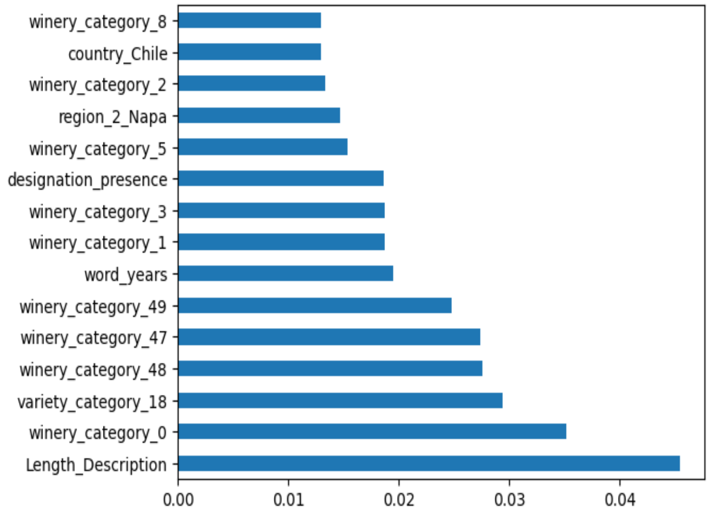

Fig. 8: 15 most important features selected by Random Forest

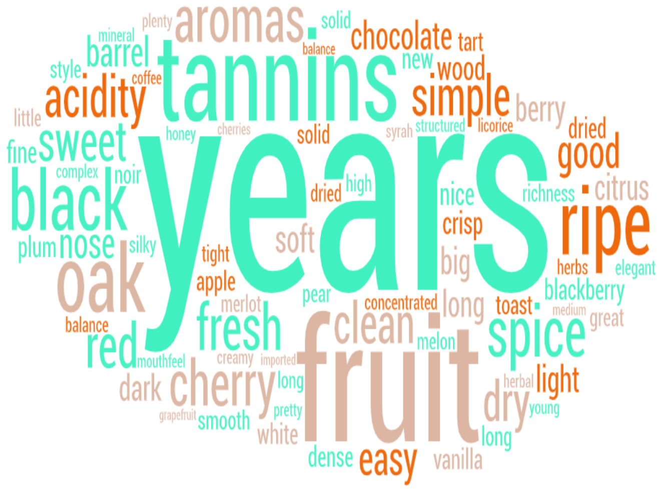

Fig. 9: Words selected by Random Forest

## References

[1] D. Cousineau and S. Chartier, "Outliers detection and treatment: A review," *International Journal of Psychological Research*, vol. 3, 06 2010.