Wine Quality Regression Problem

Gabriele Gioetto
Politecnico di Torino
s285501
s285501@studenti.polito.it

***Abstract*—This report presents a method to obtain a quality score of different wines produced by different wineries, analyzing a data set of wine reviews. To achieve that, we will do some preprocessing on our data, trying to manage missing data and to understand what features are useful for our predictions. Lastly we will build a regression model in order to predict the quality score of wines based on their reviews**

I. PROBLEM OVERVIEW

The *Winery Reviews Dataset* is a dataset which includes information about several wines and a quality score assigned by a reviewer to each wine. In particular, the features of each record specify information about the wine's:

- production location
- description
- variety
- production winery

The dataset is divided into two parts:

- A development set, which contains 120744 records. Each record is labelled with a quality score
- An evaluation set, which contains 30186 records. This portion of the data doesn't include the information about the quality score

We will build a model using only data from the development set and we will evaluate our results on the evaluation set, to be sure not to build an overfitting model.

The entirety of the features are categorical, apart from the description. So we will need to encode them with the aim of building a regression model. Analyzing the data we can see that *region_1* and *region_2* and *designation* are the only features with many missing data, meanwhile *country* and *province* have only 5 null values. The rest of the features are always not-null (Table 1).

Moreover in the dataset 35716 duplicates are present, so we will have to decide how to manage them. Without considering the duplicates we can notice that in 4192 wines the *region_1* and *region_2* features have the same value

II. PROPOSED APPROACH

A. A. Data Preprocessing

Calculating the quantiles of the *quality*, we can see that the 98% of the data have a quality in the range [21,74], whereas the median is 46. There are 15 records that have quality equal to zero. Assuming they are errors, we remove them from the dataset (Fig. 1).

Column	Non-Null count	Dtype
country	120739 non-null	object
description	120744 non-null	object
designation	84226 non-null	object
province	120739 non-null	object
region_1	100736 non-null	object
region_2	48736 non-null	object
variety	120744 non-null	object
winery	120744 non-null	object
quality	120744 non-null	float64

Table. 1: values in the Wine Review Dataset

Regarding the duplicates, we decide to remove them, since they can create biased estimator's coefficients towards the duplicated data. This wouldn't be a problem if the duplication of data had some inner meaning (For instance in a transactional database we shouldn't remove duplicated records), but this isn't the case.

Focusing on the *designation*, from the bar plot (Fig. 2) we can notice that that the same designation have different values. This is caused by different languages (”Reserve”, ”Reserva”, ”Riserva”) or by lower-case characters (”Barrel sample” and ”barrel sample”). To solve this problem we convert every *designation* to lower-case and we translate each designation with more languages to English.

With the purpose of reducing variability, we unite *region_1”* and *region_2* into an unique value, keeping only one value if the two are equal and replacing *NaN* values with empty strings. If both the regions are null, we try to see if there are other data with the same *province* which have regions values not null. If this is the case, we pick the most frequent region in the province as our region value.

We generate a new column called *description_len*, that contains the number of words in the *description*. Wine reviews with more words tend to have an higher quality score (Fig. 2)

To encode the categorical values we use the One-hot Encoding method, excluding designation and winery. We exclude them because their domain is too large to being entirely encoded with a *One Hot Encoder*, so we encode only the N most frequent values.

Lastly, to discretize the *description* attribute, we pick the M most frequent words and we consider only if the word is present or not in the sentence with a binary value. We decide to use a *tf-df* binary approach in view of the fact that the content of the description features is homogeneous

After implementing all the preprocessing step for the development set, we utilize the same columns for the evaluation set, since only information present in the evaluation set is usable to predict a quality score. The null values are all filled with the value *None*, so that a column will be created to identify null values for each feature.

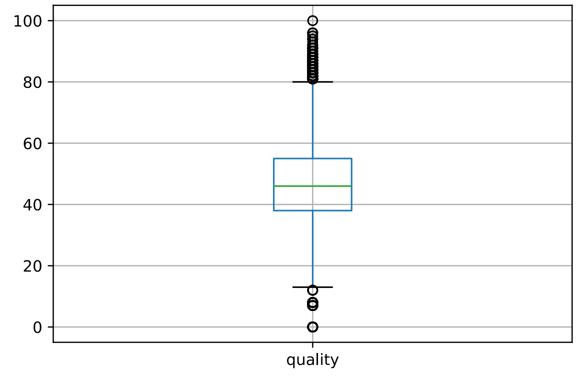

Fig. 1: Boxplot of quality

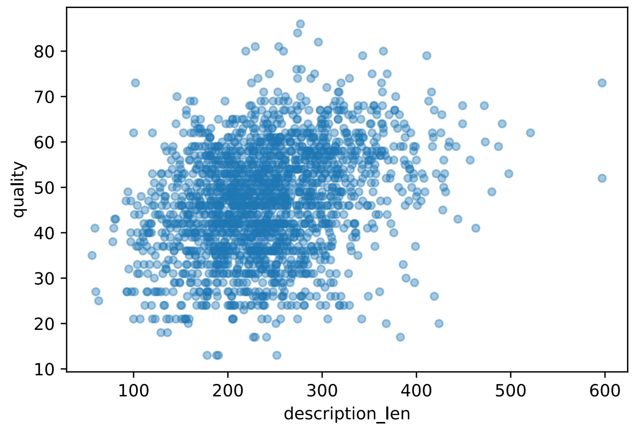

Fig. 2: Quality and description length with a sample of 2000 reviews

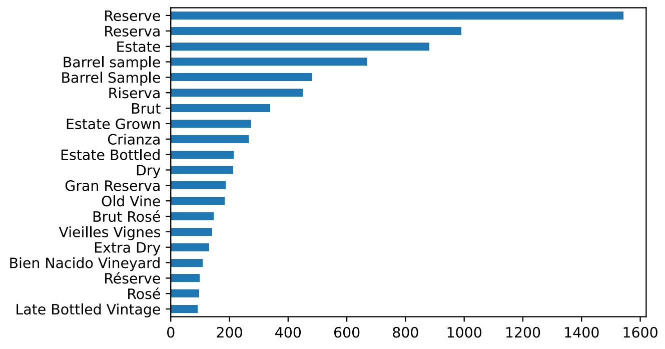

Fig. 3: 20 Most frequent designations in the dataset

B. Model Selection

The regression models that have been tested are:

Model	Parameter	Values

Preprocessing	Winery threshold
	N designations
	N description
		{15, 25, 40, 60}
		{100, 500, 700, 1000}
		{300, 500, 700, 100}

Random Forest	number of estimators
	min samples leaf
	min samples split
	max features
		{50, 100}
		{1, 2, 4}
		{2, 4, 5}
		{sqrt, log2}

Lasso	alpha
	fit_intercept
		{0.1, 0.2, ... , 1}
		{True, False}

Ridge	alpha
	fit_intercept
		{0.1, 0.2, ... , 1}
		{True, False}

Linear Regression	fit_intercept	{True, False}

Table. 2: Hyperparameters tuning

- *Linear regression*: it is the simplest method amongst all, and it is the fastest one. On the negative side, it can be influenced heavily by outliers
- *Random Forest*: it is a technique capable of performing regression task using multiple decision trees. It is robust towards outliers and it works well with sparse data. However it is biased to features with high cardinality
- *Ridge*: this model uses an L2 regularization technique, driving down the overall size of the weight values during optimization and it reduces overfitting
- *Lasso*: this model uses an L1 regularization technique, so it sets some features weight to zero, thus eliminating those features from our model

C. Hyperparameters tuning

We have two type of hyperparameters: those related to the preprocessing step and those related to the model.

In particular the hyperparameters used in the preprocessing step are:

- Winery threshold: we use one hot encoding only for wineries that are more frequent than this parameter. The other wineries are not considered
- N designation: we pick only the most frequent designations based on this parameter
- N description: we use a binary encoding only for the most frequent words in the entire set of description of the development set

Firstly, we choose the hyperparameter related to the preprocessing step running a Grid Search with a Random Forest with the default parameters. We obtain different r2 scores for each possible combination of parameter.

Then, utilizing the preprocessing hyperparameters which return the best result, every model present in the Table 2 is tested, trying every combination of hyperparameters related to the respective model. (Table 2)

III. Result

The best results for the first tuning step are reached setting Winery threshold, N designation and N description respectively to 60, 500 and 700.

Instead for the second tuning step, the Random Forest, Ridge and Lasso algorithm have similar r2 scores on the private dataset (about 0.53). However, Random Forest have the best r2 score on the public dataset. The best configuration for Random Forest is reached with {*number of estimators* = 100, *min samples leaf* = 1, *min samples split* = 4, *max features* = sqrt }

Considering that the result on the public set is way higher than the private set, we can state that there is no over-fitting on the development set. The higher result on the public set could be caused by records on the evaluation set that are equal to records on the development set. This could create perfect or almost to perfect predictions for those records. However this is only a supposition, and the real cause of the higher values on the public set could be others. On the other side, the public result is one of the lowest in the competition, hence there is surely room for improvement

IV. Discussion

The final model has plenty of features (2319), so to avoid curse of dimensionality we should try to reduce them. Using PCA we witness worsening results, therefore we could try different approaches. To get better results, one solution could be to test different methods to manage missing data, or to try different model, such as:

- MLPregressor (Multi-layer Perceptron regressor): it is a supervised learning algorithm based on a feedforward neural network. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers [1]
- SVM (support-vector machines): It aims to create a decision boundary between two classes that enables the prediction of labels from one or more feature vectors. This decision boundary, known as the hyperplane, is orientated in such a way that it is as far as possible from the closest data points from each of the classes. These closest points are called support vectors [2]

References

[1] *Scikit-learn library: https://scikit-learn.org/stable/modules/neural_networks_supervised.html.*

[2] *Open Access Applications of Support Vector Machine (SVM) Learning in Cancer Genomics SHUJUN HUANG, NIANGUANG CAI, PEDRO PENZUTI PACHECO, SHAVIRA NARRANDES, YANG WANG and WAYNE XU Cancer Genomics Proteomics January 2018, 15 (1) 41-51;.*