Wine-Quality Regression Problem

Daniele Falcetta
Politecnico di Torino
Student Id: s289319
daniele.falcetta@studenti.polito.it

***Abstract*—In this report we introduce a possible approach to the *Wine-Review* Regression problem. In particular, we try to build up a regression model capable to deduce wines quality based on some given attributes. The proposed approach consists in the extraction of some numerical features from the given categorical ones and use this new dataset as an input for some regression models.**

I. Problem overview

The proposed competition consists in building a regression model on a dataset containing a collection of wine reviews from different states and regions in the world. The goal of the competition is to correctly predict the quality of each non-labelled wine. The dataset is divided into two parts:

- a *Development test*, containing exactly 120744 reviews with the respective *quality* label,an integer value between 0 and 100 .
- an *Evaluation set*, comprised of 30186 review.

We will use the development set to build a regression model to correctly label the records in the evaluation set, assigning to each of them the corresponding quality value.
We can make some considerations about both the development and the evaluation set.
Without considering the attribute *quality*, both sets are made up of 8 non-numerical features: one is the textual description of each wine and the other 7 are categorical attributes describing other characteristics, such as the geographical origin or the signature of each bottle.

country	count	min	max	mean
US	49909	0.0	100.0	47.86
Italy	18848	8.0	93.0	48.15
France	16835	8.0	96.0	49.86
Spain	6657	0.0	91.0	40.69
Chile	4647	7.0	85.0	36.5
Argentina	4519	0.0	80.0	37.41
Portugal	4228	7.0	94.0	41.48
Australia	3949	7.0	92.0	43.87
New Zealand	2646	17.0	71.0	43.57
Austria	2447	21.0	95.0	48.02

Fig. 1. Price distribution of top 10 countries

By analyzing the price distribution of the top 10 countries, we notice that some wines have the minimum value of quality equal to 0: this means that some outliers exist in the dataset. Thus, our next step is to remove these reviews.

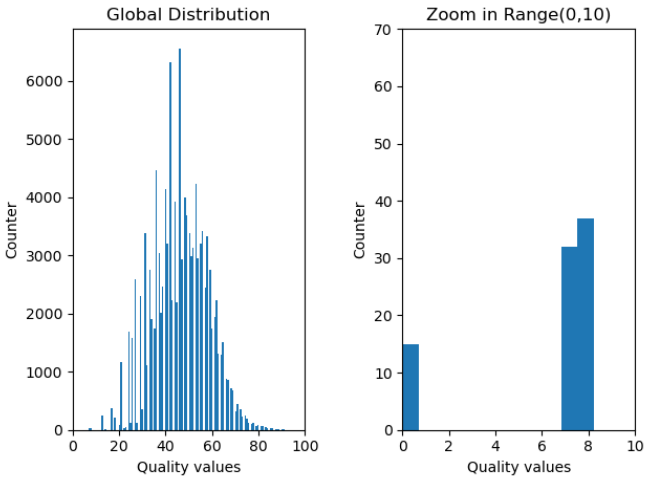

Fig. 2. Distribution of quality values in Dev dataset and analysis of ouliers

II. Proposed approach

A. Preprocessing

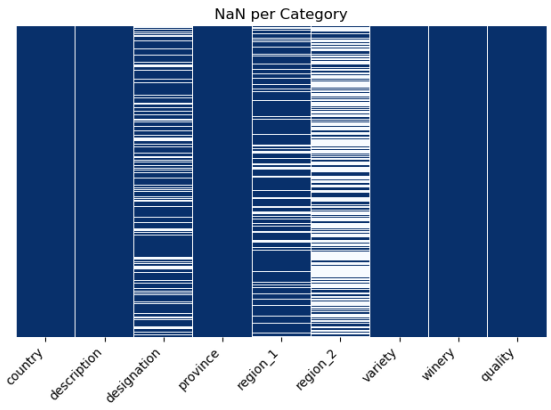

Fig. 3. Distribution of NaN per category in Dev. dataset

In order to apply one of our models we have to analyze our development dataset and extract features from row data, transforming them into suitable format for the machine

learning algorithms. Before starting, we decide to remove the *description* feature and to focus just on the other attributes.
First of all, analyzing the quantity of null value for each column of the dataset, we can notice that the attribute *region 2* contains more than half *NaN* value: so we decide to eliminate the whole field.
Secondly, we need to encode our categorical features: after trying One-Hot Encoding, we decide to avoid this technique due to his high cost for both storage and computation.As a matter of fact, using One-Hot, each distinct feature corresponds to a different category. As an alternative, to reduce the total number of features we decide to use the Binary Encoder, a method that works really well when there are a high number of categories, like in our case. In the Figures I,II,III we can notice the difference between the two methods with a toy-example: just with few distinct categories we can appreciate the strengthness of Binary Encoding.

TABLE I
INITIAL DATASET

CITY	NY	LA	Rome	Paris	NaN	Madrid	Tokyo

TABLE II
ONE HOT ENCODING

CITY	NY	LA	Rome	Paris	NaN	Madrid	Tokyo
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	1.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	1.0

TABLE III
BINARY ENCODING

CITY	$City_0$	$City_1$	$City_2$	$City_3$
0	0	0	0	1
1	0	0	1	0
2	0	0	1	1
3	0	1	0	0
4	0	1	0	1
5	0	1	1	0
6	0	1	1	1

We have also another issue: not all the possible categories are present in the development dataset. So, in order to encode properly our features we have to concatenate both datasets, fit and transform with our Binary Encoder and then split them again.
At this point our datasets are ready to be processed and so we can proceed with the choice of our model.

B. Model selection

Before selecting the proper model, we perform two other passages.
Firstly, we take a random sample of our development dataset in order to be faster and save time for further steps: in this case, we chose to take a 10% sample.
Secondly, we split it into Train Set and Test Set for the purpose to evaluate the R2-Score of our model. Even if taking a sample reduce our R2-score, in the end we must train our model on all Development dataset and use it to label the evaluation dataset. For the choice of our model, our approach consist in trying the all the most common regression model without any (hyper)parameters, in their default setting, and then select the best one according to our preprocessing-phase and proceed with it to our next step: the hyperparameters tuning. In the Table IV, we can see the results of our analysis.

TABLE IV
TESTED MODELS

LinearRegressor	SVR	MLPRegressor	RandomForestRegressor
0.198	0.298	0.384	0.498

As a consequence, we select RandomForestRegressor as our chosen model.

C. Hyperparameters tuning

At this point we can run a grid search on the Random Forest Regressor model, based on the hyperparameters defined in Table V.

TABLE V
TESTED MODELS

Model	Parameter	Values
RandomForestRegressor	max_depth	{None,2,10,30,50}
	n_estimators	{100,500,900}
	criterion	{mae,mse}

We decide to select the best configuration based on the result of R2-Score

III. RESULTS

The best configuration for random forest was found for {'max_depth':None, 'n_estimators':900, 'criterion':'mae'} and our best score found in the test dataset was R2_Score $\approx$ 0.51.
We trained the best performing on all available development data.Then the model has been used to label the evaluation set. Nevertheless this configuration proved to be too time consuming due to the single parameter {'criterion':'mae'}: so we decided to avoid using it and using a less time consuming configuration even that this means losing a little bit in score. The public score obtained is 0.824.

IV. DISCUSSION

The proposed approach obtains result that far outperform the defined naive baseline. The following are some aspects that might be worth considering to further improve the obtained results:

- We could not remove the *description* and carry out a complete sentiment analysis by analyzing the words distribution with, for example, the tf-idf technique. This

approach would allow us to surely achieve a better result. However, we decide to not take this path because of its high time and disk-space consuming.

- A better way to handle NaN values: instead of handle them implicitly inside the encoder, we may decided to follow another approach, for example by filling each of them with the most likely value within each feature keeping in mind the values of the other features and avoiding inconsistency problem.
- Other feature extraction techniques or encoding types may be considered. For example, the One Hot Encoding followed by the PCA (Principal Component Analysis) method to reduce the total number of features. We may also decide to select other algorithms such as the MLPRegressor,a regression algorithm based on a specific type of feedforward neural network (FFNN).
- Run a more thorough grid search process both for random forest and for other additional regression model. Use a more powerful tool or computer to find an even better configuration: for instance, it would just be enough to use the best configuration found in this report (with {'criterion':'mae'}) to reach a tiny but significant improvement in score.

The result obtained, however, are already promising even if there is a margin for improvement.

REFERENCES