Wine quality regression problem

Valerio Zingarelli
Politecnico di Torino
Student id: s281586
s281586@studenti.polito.it

***Abstract*—In this report we introduce a possible approach to a regression problem based on wine quality prediction. The proposed solution consists in a concatenation of one hot encoding for categorical features and Term frequency–inverse document frequency for dealing with wines' descriptions. The proposed pipeline provides good results and allows to outperform the provided baseline.**

I. Problem overview

The proposed competition is a regression problem on a wine quality dataset which is composed of 150.930 different entries. It's divided into:

- a development set: 120.744 istances with a quality label
- an evaluation set: composed of 30.186 entries

Each istance of the dataset is characterized by 9 columns:

- country: wine's country of production
- description: a brief description of the wine
- designation: the name of the wine given to the it by the producer
- province: wine's province of production
- region_1 and region_2: wine's regions of production
- variety: wine's variety
- winery: company which has produced the wine
- quality: quality of the wine. In evaluation set, quality column is not present

For what concerns features, they are all categorical, except for $description$ which is a text. The goal of the competition is to build a regression pipeline in order to correctly predict wines' quality. Analyzing development test is clear that our dataset is quite unbalanced: as it's possible to see in Fig.1, majority of wines has a quality score between 25 and 75. Something important to notice is the massive presence of NaN values in some columns of our dataset: as shown in Fig.2, $region\_1$, $region\_2$ and $designation$ have many null values, whereas $country$ and $province$ columns have 5 NaN values.

Analyzing development and evaluation datesets, it's possible to notice that 35716 duplicated are present in development set and 2595 in evaluation set. Fig 3. shows that most of the wines come from US, France and Italy, so we can wonder if average quality has same distribution.About that, Fig.4 answers to our question: we can see that highest mean quality scores in wines are in England, Luxemburg and France. For istance, a country of dataframe, named "US-France", has been considered as an outlier and substituted with "US".

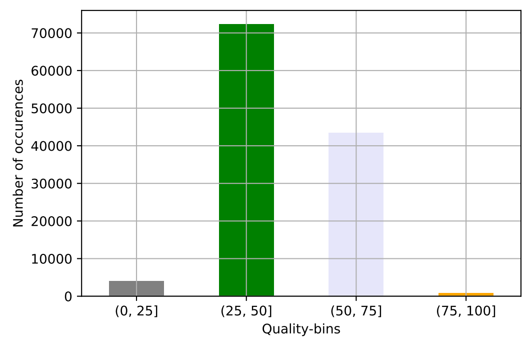

Fig. 1. Quality distribution

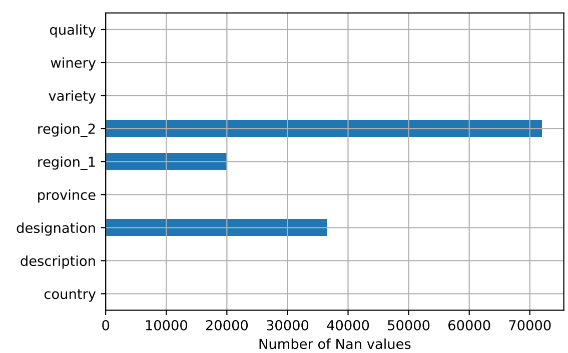

Fig. 2. Nan values distribution for each column

II. Proposed approach

A. Preprocessing

First of all, we have to deal with NaN values. All of them have been filled with a string composed of "$Unknown$" added to column name (e.g. for $country$ column, NaN $\rightarrow$ "$Unknown\_country$"). After that, we have to encode categorical features. Chosen encoder is OneHotEncoding by sklearn because it allows us to not introduce a hierarchical order in our data. In order to encode features, we merge evaluation set and development one to avoid dimension mismatch problems. Regarding $description$ column, we have to deal with a text and Term Frequency inverse document frequency (from now on Tf-idf) algorithm is the best choice to transform natural language into something our regressor can use.
Tf-idf of term t in document d of collection D (consisting of m documents) is:

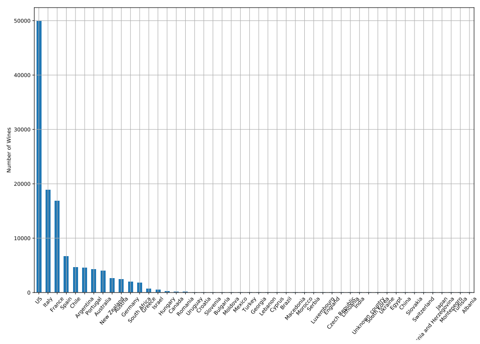

Fig. 3. Number of wines distribution for each country

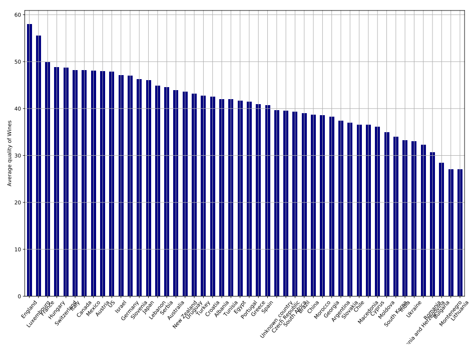

Fig. 4. Average quality distribution for each country

$$\text{Tf-idf(t)} = freq(t,d) * \log(\frac{m}{freq(t,D)})$$

Terms occurring frequently in a single document, but rarely in the whole collection, are preferred. In order to create word chunks to use in Tf-idf algorithm, NLTK has been used. [1]NLTK stands for natural language tool kit and it is one of the best tools for this purpose. It includes functions to tokenize sentences, lemmatize words (group together inflected forms of a word and analyze them as a single item), etc. In the end, we have to select the stopwords to use in our algorithm. Stopwords are words considered not to be meaningful and so we can ignore them. NLTK standard stopwords for english language have been used and many others, such as " ' " or " 've ", have been added to them.

After encoding and Tf-idf, we have a dataset too large to deal with and so, for space and speed reasons, it has been converted into a sparse matrix using scipy's function $hstack$ that stacks encoding and Tf-idf results in a single sparse matrix.

B. Model selection

Once we have prepared our data, we have to select a model and tune and validate it. The following regressors have been used:

- LinearRegression: ordinary least squares linear regression.
- Ridge: it's a regressor that tries to assign values closer to zero to the coefficients assigned to features that are not useful for the regression

C. Hyperparameters tuning

A 80/20 Train-Test split has been performed and a grid search has been runned in order to get the best parameters for our regressors. In Table 1 it is possible to see parameters' values and the relative best r2 score obtained on Public Leaderboard and during local tests.

Regressor	Parameters	Values	Local	Public-score
Ridge	alpha	[0.1,0.01,0.2]	0.869	0.885
	tol	[1e-5,1e-8]		
Linear Regression	fit_intercept	[True,False]	0.867	0.883
	normalize	[True,False]		

TABLE I

CONSIDERED HYPERPARAMETERS

III. RESULTS

Both regressors managed to obtain good results in local and on public leaderboard. Proposed pipeline leads to a general model: in fact, local scores, as it's possible to see from Table1, are even lower then the ones provided by public leaderboard. This means our model generalizes well and does not lead to overfitting issues. Linear Regression's best score has been obtained with {fit_intercept=True, normalize=False} whereas, on the other side, best performance with Ridge regressor has been achieved with {alpha=0.01,tol=1e-5}. Furthermore, we can notice there are not significative differences between regressors because they lead to very similar results. OneHotEncoding of categorical features, for istance, has lead to a significant improvement: a model consisting of just Tf-idf has tested and obtained a public score of 0.716 that was, anyway, enough to overcame the naive baseline.

IV. DISCUSSION

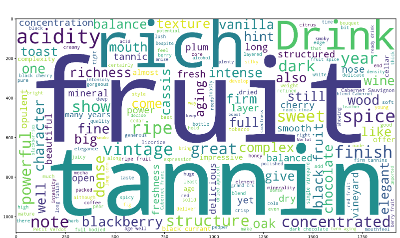

Fig. 5. Word cloud from best wines' descriptions

Proposed regression pipeline manages to overcome by far provided naive baseline encoding categorical features and using Tf-Idf on textual description. Fig.5 and Fig.6 are

Fig. 6. Word cloud from worst wines' descriptions

wordclouds obtained from the descriptions associated to least qualitative wines and most qualitative ones. Word clouds are a visual representation of textual data where the bigger is the word the more frequent it is. Stopwords used for generating them are the same used in Tf-idf and some other words that were present in all the descriptions, such as "wine" or "aroma",have been added to them. It's possible to see how lemmas like "rich" or "tannin' are associated with very high quality wines, whereas, on the opposite, terms like "nose" or "sweet" are correlated with low quality wines. From word clouds it's possible, as well, to infer that most of the wines in dataset are fruity because of the massive presence of "fruit" in descriptions.

Even if obtained results are quite satisfactory, in order to improve this model's performances, other regressors,like Deep Neural Networks, could be used and,in addition to this, a grid search with more parameters could be run.

REFERENCES

[1] "Nltk." https://www.nltk.org/api/nltk.html.