Wine Reviews Dataset Project Report

Maria Margherita Lovera
Politecnico di Torino
s278425
Data Science Lab: Process and methods
Winter First Call, A.Y. 2020/2021

***Abstract*—In this report we introduce a possible approach to the wine reviews dataset regression problem. In particular, the proposed approach consists in retrieving all available information from each wine review and analyze the dataset in order to use just the most informative part of it. Then, by using it as input for regression modeling, we propose a model capable of predicting the quality score of a wine while obtaining overall satisfactory results.**

I. Problem overview

The proposed competition is a regression problem on a dataset containing wine reviews in tabular format. It counts 150.930 entries, each of which refers to a review expressed by an expert on a specific wine with further information on location, designation, variety and winery. The target of the competition is to correctly predict the quality score associated with a wine review. The dataset is divided into two parts:

- a *development set* containing 120.744 reviews, for which the quality score is given
- an *evaluation set* containing the remaining 30.186 reviews.

We will need to use the development set to build a regression model that correctly assign a quality score for each wine review in the evaluation set.

We can make some considerations based on the development set. First, there are 89.191 reviews with missing values. In particular, as it is shown in Figure 1, most of them are located in the fields related to the designation and regions of each wine. Specifically, we have 36.518 missing values for designations, 72.008 for regions of group two, 20.008 for regions of group one and 5 for both countries and provinces. Second, Figure 2 shows that, on a total of 48 countries, there are three countries (i.e. US, Italy and France) with more than 15.000 wine reviews, while all the others have fair less than 10.000. Regarding the 444 distinct provinces, the most frequent ones are those situated in the most frequent countries. Concerning the regions (1206 regions of group one and 18 regions of group two respectively), we decide for now to leave the analysis for later on in order to find more information on the relation between the two groups. Indeed, the great amount of missing values for regions of group two (see Figure 1) makes us think that just one of the two regions will be enough to retrieve information about the location of wines.

Third, by observing the occurrences of the 27800 designations along reviews, we find some of them to be the same ones but reported in different languages (see Figure 3: Reserve, Reserva, Riserva, Réserve): we suppose that it will be probably necessary to differentiate designations with respect to countries or provinces. Finally, we count 603 varieties of wine and 14.105 wineries. By doing some further research on wineries we also observe that some wineries are situated in different countries and, in particular, provinces. This information is useful since now we know that wineries will probably not give enough information in order to determine uniquely the location of a specific wine.

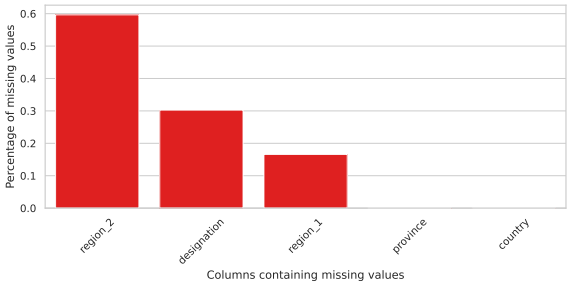

Fig. 1. Missing values in development set

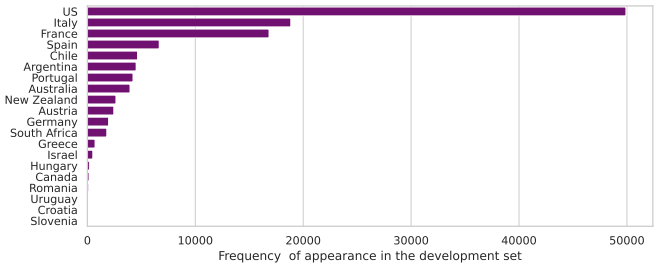

Fig. 2. First 20 most frequent countries in development set

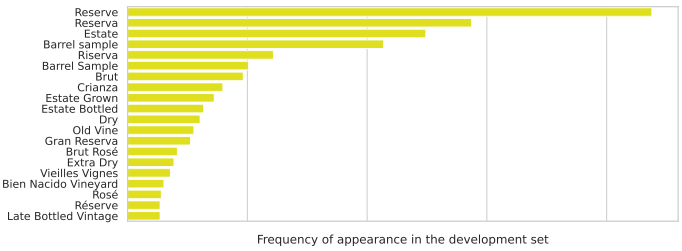

Fig. 3. First 20 most frequent designations in development set

Another important feature to analyze and discuss is the description, that is given for each wine review. This will probably represent the core feature in our model since it expresses the overall feeling of the reviewer about the wine. The Word Cloud below shows the most frequent words in all the descriptions gathered together.

II. Proposed approach

A. Data preprocessing

- *Filling missing values*
  We have seen that there are more than 89.000 reviews with missing values, mostly located in the fields related to designation and regions of each wine. For this reason, our first goal is to find a solution to the missing data problem in order to retrieve all the available information from wine reviews. After checking that the 5 missing values related to both countries and provinces were located in the same rows, we decide to remove them in favor of focusing on strategies aimed at solving designations an regions missing values problem. Our attempt will consist in filling rows with the most frequent designation and regions with respect to other informative attributes, where possible.
  - *Designation missing values*
    As we already observed before, there are same designations reported in different languages: we consider then the hypothesis that, in consideration of filling the missing values with the most frequent designation, we should differentiate with respect to locations. Since it is known that the designation of a wine depends upon the type of grapes and the place where it is produced, we decide to fill missing designations with the most frequent one with respect to each distinct province and variety of wine respectively. With this strategy we remain with 1052 missing values over the initial 36.518 found. In order to still retrieve some information, we implement the same filling strategy but this time with respect to each distinct country and variety of wine so to make it more general. By the end, we end up with 291 missing values on designations. By making some further considerations about these remaining rows, we count how many of them contain missing values also for both regions: 179 over 291 wine reviews do not have information about both designation and regions. So we decide to remove those reviews knowing that we already retrieved all the informative part with respect to designations.

Before starting to solve the regions missing data problem, we want to be sure to work just with the relevant information. Knowing that we have 19.824 missing values on regions of group one and 71.754 on regions of group two, we count the frequency of missing values in each row and, on a total of 120.448 rows, we have the situation shown in Table I: the last line tells us that every time a region in group one is missing, also the correspondent region in group two is missing (19.824 is exactly the number of missing values on regions of group one). Furthermore, by constructing a dictionary with keys equal to all distinct regions of group one (we are not considering rows with region_1 missing) and values the number of distinct regions of group two occurring, we notice that the first group of regions determines the second group quite uniquely (if region_2 is not missing). Indeed, we can see from Figure 4 that for 987 regions of the first group region_2 is missing, for 219 region_2 is uniquely determined and it occurs just one time that we have a region of group one with two distinct values for region_2. So we decide not to consider the column concerning the second group of regions since we do not believe it to be informative in view of the construction of the regression model.

TABLE I
Number of rows & Number of missing values

Number of rows with 0 missing values	48.694
Number of rows with 1 missing values	51.930
Number of rows with 2 missing values	19.824

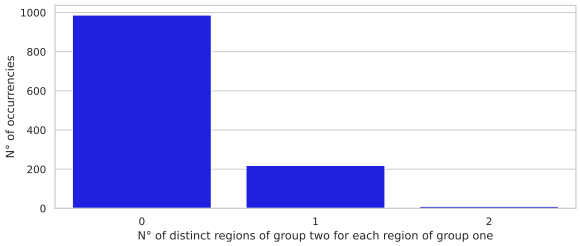

Fig. 4. Relation between NaNs of region_1 and region_2

- *Regions missing values*
  Now, we deal with regions missing values (notice that from now on we will refer to regions of group one as regions, since we removed region_2 from our analysis). Similarly to what we did with designations, we ask ourselves which of the other features could help us giving a significant interpretation of the missing regions. In order to do that, as a first thing we exclude wineries from the features used for regions filling since we already observed that wineries are situated in different countries and provinces, so

we do not believe them to be informative in this context. So we rely on common knowledge: it is known that the region of a specific wine is mainly influenced by the variety of the produced wine (since the variety produced is strictly related to the region of production) and, of course, the province in which it is situated. So we try a first filling by selecting the most frequent region with respect to both province and variety of wine. After this first filling, we are left with 19.028 missing values over the initial 19.824. To generalize the idea so as to achieve a more satisfactory fill and retrieve all the possible information from regions, we decide to fill the remaining missing values with the most frequent region with respect to just the province: this led us to reduce the number of missing values to 18.575. Since we believe that regions are in general more informative and specific than the other locations features (indeed, each region is in a specific province and each province is in a specific country as shown in Figure 5), we decide to conclude the filling by inserting the respective province when no other information is given (i.e. in those situations in which a province does not have any information on regions). We do that so to retrieve all the possible information about regions.

Notice that, by doing this filling, we decide not to include provinces in the construction of the model to avoid possible correlation between features and, since with this filling we decided not to lose any information about regions, we will also not include countries for the model implementation: indeed, they are far less specific than regions (see again Figure 5).

- *Feature extraction*
  We now focus on the feature extraction phase. We underline once more that the features that we will use in order to build our model are those regarding the description, designation, region, winery and variety of each wine. Furthermore, none of the considered attributes has numerical values, so we need to extract information by means of vectorization techniques.
  It is important to specify that, in order to evaluate the performance of our model, we will split the development set into a training set and a test set. Therefore, concerning the feature extraction steps, each vectorizer will be fit on the training part and then both the training and test sets will be transformed with respect to the previous fit on the training set.
  - *Tf-idf on descriptions*
    Regarding descriptions, we choose to adopt a term frequency-inverse document frequency approach in order to give more importance to those words frequent in a single description but not so frequent along all descriptions. So we implement the TfidfVectorizer from scikit-learn with the following parameter setting, the best one in terms of results:

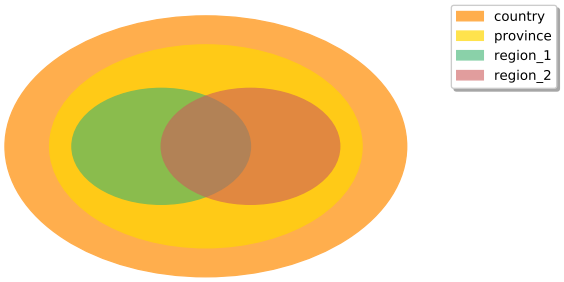

Fig. 5. Hierarchy of locations features

    * extraction of both single words (unigrams) and contiguous sequences of two words (bigrams);
    * removal of stopwords by the English stopwords list from nltk library;
    * conversion of all characters to lowercase;
    * tokenization (the default one).
  - *Count Vectorizer on the remaining features*
    For each of the remaining features (i.e. designation, region_1, winery and variety) we implement the CountVectorizer from scikit-learn, but with a binary setting so to assign value 1 to the words appearing in the 'document' (in this case just a designation, a region, a winery or a variety of wine) and 0 to the others: indeed, we are simply mapping each value of the columns to its integer counterpart. As before, we extract both unigrams and bigrams, but this time we decide not to convert characters to lowercase and we also do not require the removal of English stopwords (we suppose that we have just names regarding designations, locations, wineries and varieties so we do not want to lose information).

Observe that scaling techniques have not been considered since the features space is stored in a sparse format: this should not be a problem since the vectorizers already make a sort of scaling of features. Furthermore, each description is normalized by default with respect to the L_2 norm after being transformed into the Tf-Idf space: this means that all documents will have the same importance, regardless of their length.

B. Model selection

We consider and analyze three different regressors, the linear regression, the SVR and the linear SVR, so to understand if our data have either a possible linear behavior or not.

We remember once again that, in order to evaluate the performance of our model, we will split the development set into a training set and a test set.

- LinearRegression, a very simple and easily interpretable regressor that fits a linear model to minimize the residual sum of squares between the observed targets in the dataset and the targets predicted by the linear approximation. It is reasonably fast but not optimal for large amount of data.

- SVR, Support Vector Machine for Regression. It is definitely more complex than the Linear Regression, but it usually hold best accuracy measures. It is not an interpretable regressor and, above all, it suffers from curse of dimensionality.
- LinearSVR, similar to SVR but with parameter kernel set to 'linear'. This regressor should scale better to large amount of data (i.e. it is faster than SVR in general) without losing in accuracy.

By performing a general analysis with all the regressors trained and tested on the development set, we can assert that:

- as expected, in our setting the SVR approach is not implementable due to important hardware limitations and large dimensions of the provided dataset. Indeed, even if we could reach some good performance measures, it would require a very large amount of time. For this reason, we decide to analyze the LinearSVR instead since it is less general than SVR, but faster and, also, still a very good model;
- LinearRegression is comparable to LinearSVR in terms of accuracy: indeed, we obtain a better R2 score with the LinearSVR, but the difference is less than 0.4%. The real improvement linked to the LinearSVR is in terms of computational time: we reduce the time elapsed from more than 10 minutes to less than 30 seconds.

In the end, by considering all the previous observations, we decide to select the LinearSVR as our final regressor.

C. Hyperparameters tuning

By fitting the simple Linear Regression model on our training set without setting any hyperparameter (since this regressor has not significant hyperparameters to set) and evaluating it on the test set, we obtain an R2 score of 85,73%. So we are confident that, since our data seem to follow a quite linear trend, by using the LinearSVR instead of the general one we will not lose too much generality.

The LinearSVR instead has been trained on the training set with different configurations. Tuning has been performed with a cross-validated search grid on the three main hyperparameters C, Epsilon and max_iter, with assigned values shown in Table II below.

TABLE II
HYPERPARAMETERS GRID SEARCH

Parameter	Values
C	0.1, 1(default), 10, 20, 100
Epsilon	0(default), 0.1, 0.5, 0.8, 1
max_iter	1000(default), 2000, 3000

The best configuration is found for C = 10, Epsilon = 0.8 and max_iter = 3000.

III. Results

Before training the best performing LinearSVR regressor on all available development data and label the wine reviews present in the evaluation set, we will first make some observations about the performance of the best performing regressor by fitting it on the training development set and evaluating it on the test development set, in order to understand something more about each feature's contribution.

TABLE III
FEATURES IN THE MODEL

N° of features	Features present in the model
1	description
2	description, winery
3	description, winery, designation
4	description, winery, designation, region_1
5	description, winery, designation, region_1, variety

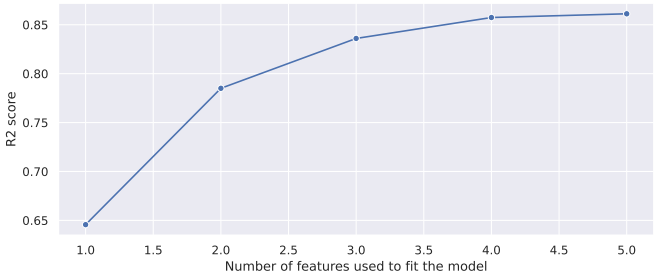

Fig. 6. R2 scores for LinearSVR with different number of features

By looking at both Table III and Figure 6 we can see that only with the Tf-Idf on descriptions we already reach an R2 score of 64,56% and with all 5 features we reach the 86,12%.

Than by training the model on all available development data we obtain a public score equal to 86,5%.

IV. Discussion

By looking at our analysis, we can state that the most important part of the work is surely the data preprocessing. Indeed, although the final result is satisfying, there are some aspects that might be worth considering in order to further improve the obtained results:

- analyze all steps of data preprocessing and, where possible, improve it as much as possible
- build a model based on a non-linear regressor, such as the SVR, since we could expected it to give different (probably better) results in terms of performance. In general, consider more regression models
- run a more thorough grid search process over the set of hyperparameters to achieve a better configuration

Even if maybe we could have improved the performance of our model by attempting one of the above mentioned options, the results obtained are already very promising given the very large amount of data in our possess.

References

[1] LinearSVR, available at the following webpage