Prediction of a quality score associated with a wine review

Antonio Albanese
Politecnico di Torino
Student id: s282043
albanese.antonio@studenti.polito.it

Abstract—**In this report a possible approach to the Prediction of a quality score associated with a wine review problem is introduced.**

In particular the proposed approach consists in using a linear regression technique on mixed data composed by: textual data preprocessed as a Bag of Words using the Tf_idf weighting schema and some categorical ”like” data encoded under the On-Hot logic.

Then the proposed model has been evaluated in terms of R^2, giving good result.

I. Problem overview

Deducing quantitative evaluations from textual data is one of the most experienced subjects in the field of data-mining. This paper treats about a regression problem on a wine review dataset, a collection of reviews and quality scores assigned to different wines. The goal of the analysis is to correctly infer the quality score assigned to wines based on the review and other information contained in the dataset.

The dataset is divided in two parts:

- a *development* set, containing 120’744 entries with the quality score previously assigned
- a *evaluation* set, containing 30’186 entries where quality score has to be assigned

The information about *country, description, designation, province, region1, region2, variety, winery, quality* are provided for each wine in the development dataset, instead in the evaluation dataset quality is not given.

For some attributes in the dataset the value can be empty or Not a Number (NaN).

As it is possible to see in the Table I all entries in the dataset have *variety* and *winery* not empty, while *designation* and all information related to *production location* can contain NaN values.

Now would be very useful to establish if there is correlation between production location and wine quality. According to P. Lock, S. Mounter, E. Fleming and J. Moss *there are substantial benefits on winery ratings and subsequently wine quality from the maintenance of sound soil health* [1], but location impact on wine quality is not totally demonstrated. Nevertheless would be incorrect a priori dropping “location” information (explicated in country, province, region 1, region 2) in order to predict the quality of unknown wines. The impact of the inclusion of this data will be later evaluated in terms of performance and computational effort. In the columns *country* and *province* the number of missing values is negligible compared to the size of the dataset (Table II) and therefore we do not foresee great difficulties in their management or a great impact in terms of performance.

Column	Contains NaN
country	True
description	False
designation	True
province	True
region1	True
region2	True
variety	False
winery	False
quality	False

TABLE I: Test of the presence of NaN values in each column of the dataset

Variety information is also taken into account as it depends on the basis of many factors, such as the type of vineyard, production and aging techniques which certainly can influence the quality of the wine.

At the end, the *description* field contains the review associated with the quality score and of course it will be useful to this analysis.

II. Proposed approach

A. Preprocessing

1) ”Description” preprocessing: First of all, we consider appropriate to explore the space of the words used in the reviews through the utility, *WordCloud* of the *wordcloud* library, from which was also used the list of stop words *STOPWORDS*, a simple list containing the main stop words for the English language, useful also later on.

The Figure 1 gives an idea of the frequency of the words used in the reviews. As expected, the word ”wine” is the most frequent one, but it would not bring any real contribution to this analysis, whose goal is to explore a sort of sentimental meaning of the review about wine. For this reason, we add this word to the stop words list.

After removing the word *wine* there are still some words that are really frequent in the collection of reviews. In order to make the model focus on words that are less frequent but that can be more relevant in understanding reviews, we build

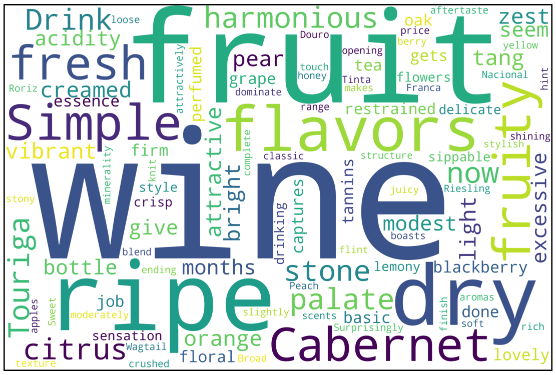

Fig. 1: Word cloud representation of reviews Bag of Word

a *bag of words* using the *Tf_Idf (Term frequency - Inverse documents frequency) weighting schema*. Thus, considering a word *w*, belonging to a review *d*, in the collection *D* containing *m* reviews, the weight associated with each individual word within a review is related to the frequency of the same word within the collection D, according to the relationship:

$$tf\_idf(w) = freq(w, d) * log\frac{m}{freq(t, D)}$$

Using this weighting schema avoids biasing the analysis by assigning too much weight to commonly used words within the topic covered in the reviews.

In creating the Bag of Word, also bigram (combinations of two adjacent words in the same review) have been considered, with the objective to better capture possible negative sentiments.

This method produces as result a huge matrix containing the above descripted frequencies. To handle this kind of output, we use the data type *Sparse* (supported by the SciPy library).

2) "Winery" and "variety" preprocessing: In this case we explore two solutions: the first, a *binary encoding*, with the aim of obtaining a result with a small dimensionality, made by the support of the *category_encoders* library [2]. Even if we obtain a reduced dimensionality, with this method the model does not produce the desired results. Hence, another way to proceed is considered.

The second technique adopted is the *One-Hot Encoding*, that creates vectors that agree with the general intuition of nominal categories: orthogonal and equidistant [3]. In these bit vectors, the legal combinations of values are only those with a single high (1) bit and all the others low (0).

The technique is used by thinking of this information as categorical data. Therefore, at this step the output has a column for each category, that means, one for each winery and for each variety. In these columns there is 1 if the wine is produced by the specific winery and belong to the specific variety, 0 otherwise.

For high-cardinality categories such in this analysis, One-Hot Encoding leads to feature vectors of high dimensionality. Since classical features reduction techniques would not have produced useful results, precisely because of the way One-Hot Encoding works, the problem of huge dimensionality has been avoided again by using the *Sparse* data type.

3) Production location related data preprocessing: As mentioned before (Table II), some of these columns, such as *country* and *provinces* contain NaN values, but in very limited number, so these values are replaced with the string *missing*. Also in this case we decide to use the technique of One-Hot Encoding as for *winery* and *variety*. Although the size of the output is much smaller than seen so far, for simplicity we consider appropriate to continue using the *Sparse* data type. Moreover, some models are tested with and without this data, deciding whether or not to use it.

Column	Number of NaN values
country	**5**
description	0
designation	36518
province	**5**
region_1	20008
region_2	72008
variety	0
winery	0
quality	0

TABLE II: Number of NaN values for each column in the dataset

B. Model selection

In this step several models are theoretically considered. But first of all it is needed to keep in mind that we are working with a big number of features so complex models are excluded because they are judged computationally unfeasible for this analysis also because of limited equipment performance. Simple linear models are instead further explored.

Firstly, we compare the performance in term of *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) and R^2 of *LinearRegression* end *Ridge* with all default parameters as served by the library *SciKitLearn*, obtaing the results shown in Table III. This comparison is made firstly considering only review, winery and variety data.

Model	MAE	MSE	R^2
LinearRegression	4.819	48.187	0.661
Ridge	5.102	46.353	0.634

TABLE III: First comparison of LinearRegression and Ridge models

The two models have very similar performance but, looking at the R^2 value, *Ridge* model seems to perform a little better than *LinearRegression* (Table III).

Once the Ridge model is chosen, we still need to evaluate the inclusion of the data related to the place of production. To take this decision we proceed testing the chosen model (Ridge with default parameters) using:

1) Only reviews, winery and variety (A)
2) (A) + the *country* column
3) (A) + the *province* column
4) (A) + both *country* and *province* columns

The results obtained are in Table IV that shows how the inclusion of production place data of *country* and *province* improves the performances of the model.

Data included	MAE	MSE	R^2
1	5.102	46.353	0.674
2	5.044	45.502	0.680
3	4.975	44.269	0.688
4	4.972	44.235	0.689

TABLE IV: Deciding whether include or not production place data

In conclusion we decide to use *Ridge* regression model with *reviews, variety, winery, country* and *province* attributes.

C. Hyperparameters tuning

There are two main sets of hyperparameters to be tuned:

- *max_features* for the preprocessing
- *Ridge* hyperparameters

First of all, it is necessary to address the tuning of the *max_features* parameter in the *TfidfVectorizer* function used to derive the Bag of Words of the reviews.

The purpose of this hyperparameter is to limit the size of the dictionary to the desired top frequent words. The default value is *None*, which then returns a dictionary with all possible combinations.

As said in the previous section, we include in the Bag of Words also digrams. The dictionary obtained at first contains 539130 n-grams (unigrams and digrams). We test the performances of the model limiting the dimension of the vocabulary to a smaller number of n-grams. The search starts with values [150, 1'000, 10'0000, 100'000, 200'000, 300'000, 400'000, None] and the best result is obteined for $max\_features = 400'000$. Moreover, the model is tested again with values between 350'000 and 450'000, with step=10'000 to find more precisely the best value.

Regarding the *Ridge* regression model, the following hyperparameters are adjusted:

- alpha: it is the regularization strength; regularization improves the conditioning of the problem and reduces the variance of the estimates. [4]
- fit_intercept: whether to fit the intercept for the model. [4]
- normalization: if True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. [4]
- tol: precision of the solution [4]; the algorithm stops once the tolerance achieved is equal to this value.

To find the optimal values for these hyperparameters we use the *GridSearchCV* class of the SciKitLearn library, based on the values in Table V.

Model	Hyperparameter	Values
Preprocessing	max_features	[150, 1000, 10000, ...]
Ridge	alpha	[**1.0**, 5.0, 10.0, 100.0]
	fit_intercept	[**True**, False]
	normalize	[True, **False**]
	tol	[0.0001, **0.001**, 0.1]

TABLE V: Hyperparameters values explored; (bold are default)

III. Results

The best result for *max_features* hyperparameter is obtained with the dictionary dimension of 380'000 which gave $MAE = 4.972$, $MSE = 44.187$ and $R^2 = 0.699$. Meanwhile the Table VI reports the values obteined with *GridSearchCV* for the hyperparameters of *Ridge*.

Hyperparameter	value
alpha	1.0
fit_intercept	True
normalize	False
tol	**0.0001**

TABLE VI: Hyperparameters configuration obtained with GridSearchCV (In bold values different from default)

At the end using *max_features=380'000* and hyperparameters found with *GridSearchCV*, a final $R^2 = 0.762$ is obtained.

IV. Discussion

The proposed approach for inferring the quality score assigned to wines achieve good results. The model gains the best performance considering together *country, province, text review, winery and variety* and using the default values for *Ridge* hyperparameters except for *tol* which is set to a lower value.

Also we could expect to take the default value *False* for the hyperparameter *normalization*, in fact great part of our data has been encoded using One Hot Encoder, and normalizing on that data would be not only useless but also counterproductive.

Focusing on *max_features* tuning we have shown that the model performs better with a large dictionary, that more likely includes digrams. This result is theoretically consistent with the assumption that digrams can help the model understanding sentiment of a review.

It is also suggested to deep diving into the text preprocessing trying to select N top n-grams separately, so N top unigrams, N top digrams and maybe even N top trigrams and look how the model performs.

The model can be surely improved, but the outcome obtained is satisfying, even considering the practical limitations faced up.

References

[1] P. Lock, S. Mounter, E. Fleming, and J. Moss, *Wineries and wine quality: The influence of location and archetype in the Hunter Valley region in Australia*, vol. Wine Economics and Policy 8. ScienceDirect, 2019.

[2] W. McGinnis, C. Siu, A. S, and H. Huang, “Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data,” *The Journal of Open Source Software*, vol. 3, p. 501, 01 2018.

[3] J. Cohen, P. Cohen, S. West, and L. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences.* Routledge, 2013.

[4] SciKitLearn, “Documentation: sklearn.linear_model.ridge.”