Wine quality score prediction through reviews sentiment analysis and regression models

Maurizio Pinna ID: s259444
DAUIN
Politecnico di Torino
Italian Institute of Technology
s259444@studenti.polito.it

***Abstract*—In this work we analyze a sentiment regression problem. Summarizing the general sentiment of a review and combining it to the other categorical features to obtain a real-valued score. Sentiment analysis is a highly effective tool for a business to not only take a look at the overall brand perception, but also evaluate customer attitudes and emotions towards a specific product line or service. This data-driven approach can help the business better understand the customers and detect subtle shifts in their opinions in order to meet changing demand.**

I. Introduction

Sentiment classification is the problem of classifying the opinion or feeling of written text. It has many potential applications including systems for automatic product recommendation, “flame” detection in online forums, assigning ratings to written reviews, organizing written surveys by satisfaction level, email filtering, and organizing/summarizing reviews of products by feature.

II. Dataset Overview

The development dataset is composed by 120743 rows and 9 features. These last are characterized as follows. The country is where the wine has been produced. The province, region1 and region2 attributes are present. Province includes the regions, while the country includes the province and so the regions.
The designation is the name that the producer gives to the wine
The variety feature describes the type of grapes used, then there is the winery from which the wine is produced
Finally there is probably the most valuable attribute that is the description provided by the reviewer and the quality score, expressed in a range between 0 and 100 is the target one.
All the attributes are categorical and nominal, exception made for the quality one, that is numerical and ordinal too.

III. Attributes characterization

A. Null values

Region_2 has more than the half of null values, region_1 20000 records and designation 30000. There also some records (5) that have also the country value null. So as first solution, designation, region1 and region2 columns are dropped, and also the 5 rows with country value null.

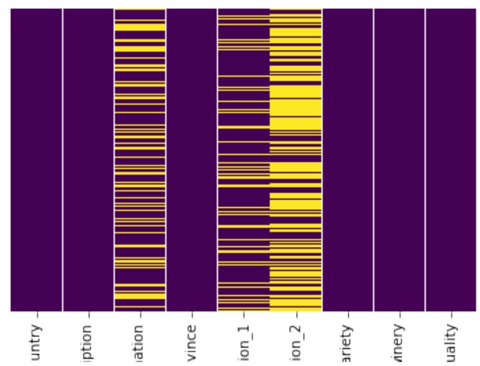

Fig. 1. Heatmap of the null values

B. Data distribution

The quality score value is the only numerical one. It has an average of 46.28 and a standard deviation of 11.92. There are:

- 48 distinct values for *Country*
- 85001 distinct values for *Description*
- 444 distinct values for *Province*
- 603 distinct values for *Variety*
- 14104 distinct values for *Winery*
- 86 distinct values for *Quality*

It’s interesting to notice that only half of the description values are unique. So maybe for the dataset has been used some oversampling technique. The distribution of the quality score is very close to a normal one, with 46,27 as mean as shown in Figure 2.
An interesting thing to notice is that there 15 scores assigned to 0. If an expert assigns 0 to a wine a range of 100, this must be like winegar, or maybe has been an error, so it has to be treated like outlier.

From Figure 3 is possible to see that the most represented country is US, followed by Italy.

While in Figure 4 is represented the average quality score divided by country that is pretty balanced.

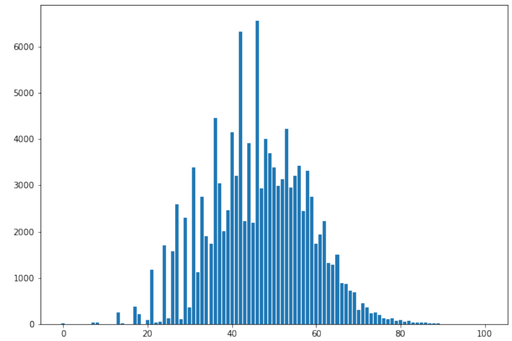

Fig. 2. Quality score values distribution

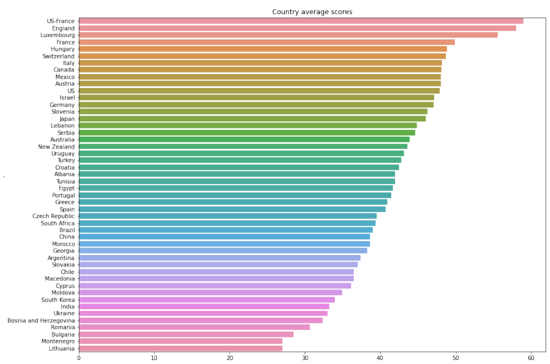

Fig. 3. Average score for each country

C. *Word clouds*

A Wordcloud is a visual representation that shows in which way the most frequent words are grouped together
In fig.5,6,7 respectively, can be seen the word cloud for the bad rated vines, the average wines and the great wines.
This categories were obtained dividing the scores in 5 equal ascending categories. For computing the division we use percentiles. Here are reported only three of them, so the two at the extreme and the average one, but in the notebook all are reported.

IV. Data preprocessing

The categorical attributes cannot be given to the machine learning models as they are. Because those models works with only numbers.
Regarding the description, first the puntuaction marks are removed and every letter is converted to the lower one.

A. *Lemmatization*

The next important step is to lemmatize words. From wikipedia, lemmatisation is the process of grouping together

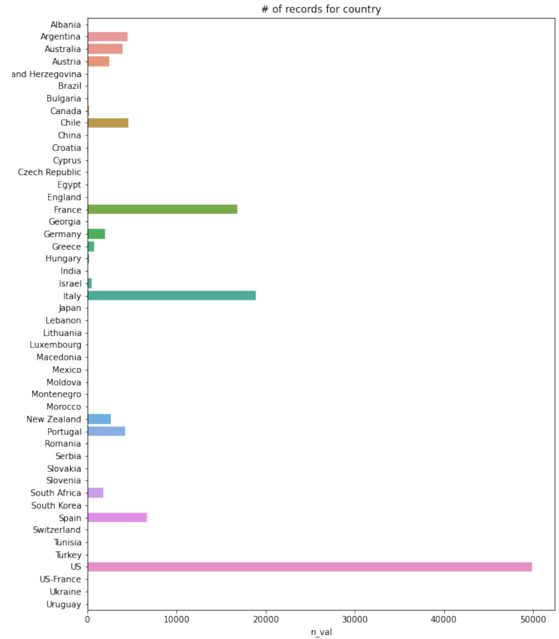

Fig. 4. Number of scores for each country

Fig. 5. Word cloud for bad rated wines

the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form. Unlike stemming, lemmatisation depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.
This is different from stemming that is the process of reducing inflected words to their word stem, base or root form—generally a written word form.

B. *Tfidf*

At the end of the steps described above, each description is a vector of word, and it is ready to be encoded. In this work Tfidf is used.
Tfidf is a numerical statistic that is intended to reflect how

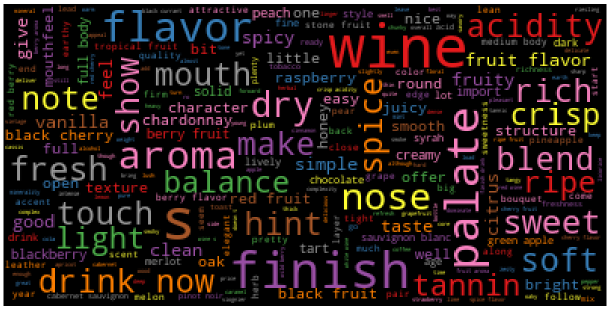

Fig. 6. Word cloud for average rated wines

Fig. 7. Word cloud for great rated wines

important a word is to a document in a collection or corpus. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general

$$tf - idf_{t,d} = \log(1 + f_{t,d}) \cdot \log \frac{N}{f_{t,D}} \quad (1)$$

N is the number of documents in the dataset
d is a given document from the dataset
D is the collection of all the documents
w is a given word in the document

One important thing to be aware is that the model with this encoding does not learn the ”meaning” of the words, for that there are other types of encoding based on different learning models.

Based on some simulation, stopwords are kept because they can bring additional value to the model, the difference with the validation score is not so high, with this type of encoding, but can be higher with the one mentioned in the conclusions.

C. One-hot encoding

For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.

In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

V. Learning Algorithms

A. Multiple Linear Regression - Ridge

The linear regression algorithm attempts to learn a function f that maps input vectors to scores. It represents f by a linear combination of the input features.

$$f(\vec{x}) = w_0 + \sum_i w_i x_i \quad (2)$$

Rdige is a regression method that perform both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.
When training, Ridge minimizes the term

$$MSE + a(\sum_i w_i^2) \quad (3)$$

B. Multi Layer Perceptron Regressor

This is a module supplied by scikit learn. It’s based on neural network. So with respect to the previous model we lose the ”interpretability” of the model but we gain in performance. In fact it’s known that NN are most of the time the best in perfomances. What Scikit-Learn does, is to provide like a wrapper to construct the multi layer perceprton regressor, with just a line of code, and this is an advantage. The disadvantage is that with scikit learn is no possible to use GPUs. In fact GPUs perform way better than CPUs for this kind of matrix computations.
Otherwise, the solution would have been implementing the MLPRegressor from scratch with Pytorch, or there are also some libraries that wraps Pytorch with scikit-learn. Combining the advantages of the two.
For this reason no cross validation was applied for this model. But, different setting of some of the hyperparameters were explored in a differnt notebook, for avoiding a kernel restart.

Two **hidden layers** has been used. The activation function used is the **Rectified Linear Unit (ReLu)**. This function is defined like:

$$R(x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ if } x <= 0 \end{cases} \quad (4)$$

The biggest advantage of Relu function is that there is no saturation of the gradients. This speed up a lot the stochastic gradient discent, with respect to the tanh and sigmoid fucntions.

The **Learning rate** is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope. Using a low learning rate might be a good idea, in terms of making sure we do not miss any local

minimum, but it can also mean that it will take a lot of time to converge, expecially working with CPUs and not GPUs, in particular if we get a stuck in sort of plateau region. There is a lot of research in this field how to initialize the weights and how to adjust the learning rate, also because sometimes is difficult to know how big is the "wall" that gets the model stuck in a local minima.
In this project the lr is set to the default value 0.001. Other values has been tried (e.g. 0.01) with the adaptive mode (i.e. the lr is divided by 2 if the loss does not improve for the epochs in n_iter_no_change), the classifier gets stuck in a sort of local minimum, so it finds a sub optimal solution, and the perfomances are worst than setting the intial lr to 0.001 and not adaptive.

The **number of epochs** i.e. max_iteration is set to 180, because after that there no significant improvement on the loss.

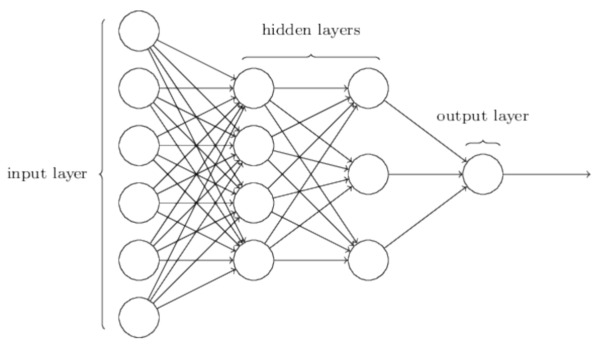

Fig. 8. Multilayer Perceptron

VI. Results

	Validation Set	Evaluation Set
Ridge	0,734	0,744
MLPRegr 128-64	0,794	0,827
MLPRegr 256-128	0,813	0,844

TABLE I
R2 SCORES OF THE MODELS IN DIFFERENT SETS

In Table 1 there are the results of the different models used. In particular Ridge with alpha=0.5, and the Multilayer Linear Regressor with respectively two hidden layers of 128-64 neurons, and 256-64 neurons.

VII. Conclusions

The resuts shows that the MLPRegressor performs better than Ridge Learning algorithm in both the configurations.
As mentioned in the MLPRegressor algorithm descritpion the hyperparameter tuning rapresents an importante aspect that contributes to the score of the model. Since in this work no GPUs were used the finetuning was quite limited for the long training time. But the results achieved compared to the baseline were good.
An important thing to say is that for models that involves neural networks like that, in general, it's not possible to analytically calculate the number of layers or the number of nodes to use per layer in an artificial neural network to address a specific real-world predictive modeling problem.
Apart from this another possible limit is given not by the learning algorithm yet by the way the description is encoded. In fact with Tfidf, the machine does not "learn" the "meaning" of the word. For example, it does not knows that, referred to a wine, bad is more similar to dusty, than good.
For this other type of representation like the Word Embeddings are used, and example can be Word2Vec.

References

B.Caputo - Machine Learning and Deep Learning Course 2019-2020
Scikit-learn MLPRegressor
Towordsdatascience.com