Wine Quality Regression Problem

Politecnico di Torino
Student id: sXXXXXX
sXXXXXX@studenti.polito.it

***Abstract*—In this paper is analyzed a solution approach to a regression problem. Due to the elevated cardinality of all the categorical variables, different embedding techniques have been used to transform the data such as the Poincarè embedding for the geographical variables and FastText for the natural language processing. Different models have been evaluated for the regression, managing to get an out-of-train R2 score higher than 0.8.**

I. Problem overview

In this paper is presented a solution to a regression problem that aims to predict the proper quality of a wine having different features of the wine itself.
More in detail, the problem is based on a training set that contains 120744 different wines. Each one of them is described by 8 variables:

- *country* : The country where the wine is made
- *province* : The province where the wine is made
- *region_1* : The region where the wine is made (a region belongs to a province)
- *region_2* : The sub-region where the wine is made(a sub-region belongs to a region)
- *winery* : The winery where the wine is made. Generally speaking, a winery is located in only one sub-region, but it could also be spread among two.
- *description* : A description of the wine. It is always in English.
- *designation* : The designation of the wine.
- *variety* : The variety of the wine.

The target variable is *quality*: it is a integer variable that has a range between 0 and 100.

The final results are computed on an evaluation set which contains 30186 different wines.

It is important to notice that all the variables (except the target) are not numerical. In particular, *description* contains only natural language text, while all the other ones are categorical variables.
The elevated cardinality of these variables could represent a problem if not properly managed. Following, a table with all the cardinalities.

TABLE I
Variables cardinality

Variable	Cardinality
Country	49
Province	444
Region_1	1206
Region_2	18
Winery	14105
Designation	27800
Variety	603

Due to the elevated numbers, a 'one-hot encoding' approach is not possible.
Another problem with the data is related to the missing values. Below is reported the number of missing values per variable.

TABLE II
Variables missing values

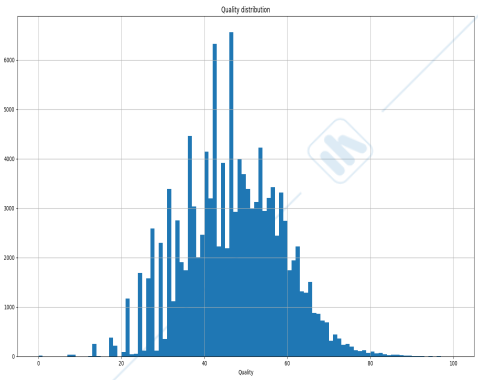
Variable	# Na
Country	5
Province	5
Region_1	20008
Region_2	72008
Winery	0
Designation	36518
Variety	603
Description	0

The large number of missing values related to *region_1* and *region_2* is strictly related to the country where the wine is made. For example, only the wine produced in the US has the data relative to *region_2*. There are other countries where the data cover only the country and the province of production.
The geographical variables follow a hierarchy: each *province* is in a *country*, each *region_1* is in a *province*, each *region_2* is in a *region_1* and each *winery* is in a *region_2* [1].

Regarding the target variable, *quality* behaves like a normal distribution with a skewness equals to 0.035 and a kurtosis equals to -0.123. The normal behavior can be observable also in a graphical representation of the variable reported in figure I.

[1]There exists a portion of wineries that are located in different countries/province/region_1/region_2. This portion is quite small, so the hierarchical structure of the geographical variables holds.

Fig. 1. Quality distribution

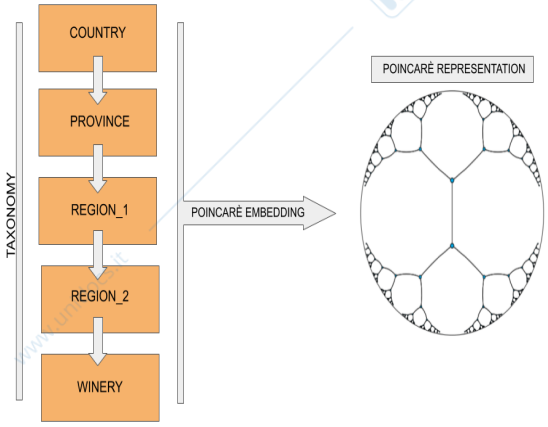


II. Proposed approach

A. Preprocessing

Due to the hierarchical behavior the five geographical variables have, all of them are transformed with the *Poincarè embedding* [1]. This technology is tailored to manage taxonomies by creating a vector representation for all the elements in the taxonomy and preserving the 'distances' among them. In this context, the geographical variables can be seen as a taxonomy [2] [2].

Fig. 2. Poincarè embeddings



With this technique every country, province, region_1, region_2 and winery is associated with a vector created by the Poincarè embedding. The idea is that if a winery a belongs to a province b (and their vector representations are v_a and v_b respectively), the distance between v_a and v_b will be small while the distance between v_c and v_b will be big if v_c is the vector representation of a winery who is not located in the province b. Moreover, for every wine, the five geographical variables (country, province, region_1, region_2 and winery) are substituted with a single vector in $\mathbb{R}^{50}$ which is the vector representation of the most detailed geographical information related to the wine. For instance, if a particular wine has information only relative to country, province and region_1, its geographical representation will be the vector relative to region_1 since it's the most detailed information available. Thanks to this method, all the missing values relative to the geographical variables are handled, except for five wines that do not have any geographical information (the five missing values for the variable *Country*). These five wines are dropped.

For the variables *Designation* and *Variety* several approaches can be tested. A simple labelling encoding could be a suitable solution that generates fair results (see section 3 for details). An alternative solution for encoding the variables could be the *mean encoding*. In this technique, the two categorical variables are replaced with two continuous ones where the values are the mean of the target (*quality*) inside every category [3]. This solution, of course, suffers from over-fitting. In order to overcome this problem, a sampling approach is adopted: for every category, a sample with a dimension equal to the 70% of the category cardinality is used to calculate the mean.

The last variable that has to be pre-processed is *Description*. This variable is not a categorical one, but contains plain text in natural language. All the descriptions are in English. The approach used for this variable exploits the *word embedding* technology, more in details the library *fastText* [3]. The idea behind *fastText* and all the word embedding methods is, given a text corpus, to assign a vector to each word in a way that, if two words are similar, the two respective vector representations will be close to each other.
FastText is also able to perform classification tasks assigning a predicted label to a text. Exploiting this function and considering the target variable *quality* as a categorical variable with 100 categories, *fastText* is capable of assigning a prediction of *quality* given the description of a wine. With that, the algorithm returns also the confidence (between 0 and 1) of the prediction. Both prediction and confidence will be used as input for the regression model. More in detail, a cross-validation approach has been used: taking only the description and the target variables, the data have been split in 5 folds. For each one of them, a fasttext model has been created with the other four as train data[4]. The selected fold has been used for the prediction.

Before the technical details about the models, in figure 3 is presented a general overview of the preprocessing phase.

[2]Also the geographical values only contained in the evaluation set have been taken into account in order to create a taxonomy that covers all the geographical domain of the problem.

[3]For the categories only present in the evaluation set, a simple global mean is used as value.

[4]The parameter of these models have been retrieved by the automatic tuning function already implemented in fasttext working an all the data

Fig. 3. Preprocessing overview

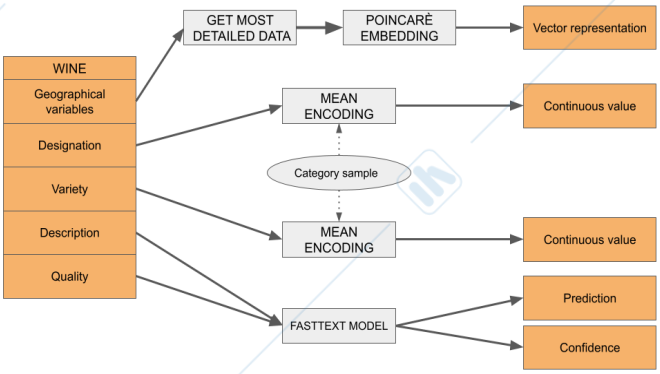
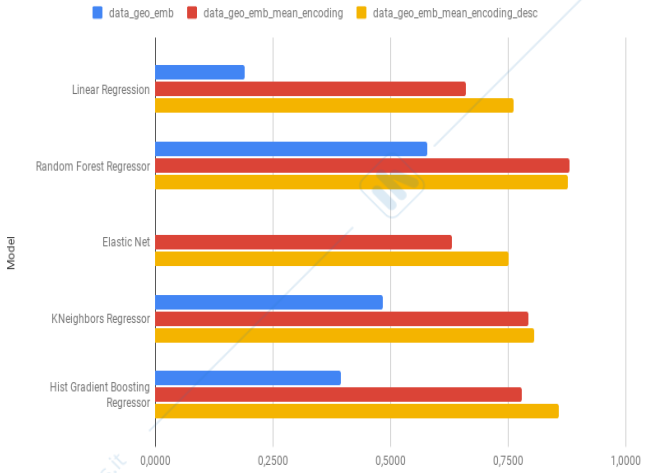


Fig. 4. Model selection plot



B. Model selection

Different models have been evaluated for this project:

- *Linear Regression*: The most standard and simple regressor. Useful to get a naive result.
- *Random Forest Regressor*: An ensemble of decision trees trained with different subsets of variables. A very powerful model but very expensive in terms of training time.
- *Elastic Net*: A more robust version of the linear regression which implements a combination of Lasso and Ridge normalization.
- *KNeighbors Regressor*: Classical KNeighbors model where the predicted label is the mean of the k nearest train data's labels.
- *Hist Gradient Boosting Regr.*: A sequence of decision trees where each one of them tries to "improve" the prediction of the previous one.

To get an idea of the possible performances, all the models are trained, without a proper fine tuning, with 75% of the data and the score is evaluated on an "out-of-train" dataset (25% of the data) . Figure 4 and table III show the results.

TABLE III
MODEL SELECTION RESULTS

Name	**R^2 in different datasets**
	data + geo_emb
Linear Regression	0.1906
Random Forest Regressor	0.5787
Elastic Net	-0.0001
KNeighbors Regressor	0.4842
Hist Gradient Boosting Regr.	0.3955
	data + geo_emb + mean_encoding
Linear Regression	0.6604
Random Forest Regressor	0.8807
Elastic Net	0.6306
KNeighbors Regressor	0.7941
Hist Gradient Boosting Regr.	0.7798
	data + geo_emb + mean_enc. + descr
Linear Regression	0.7616
Random Forest Regressor	0.8780
Elastic Net	0.7507
KNeighbors Regressor	0.8065
Hist Gradient Boosting Regr.	0.8585

It is quite clear that the information relative to the mean encoding of the variables *designation* and *variety* added with the information relative to the prediction of *fastText* based on the *description* works very well for every model.
The best models are the Random Forest Regressor and the Hist Gradient Boosting Regressor with a R^2 score greater than 0.85 without a proper fine tuning.

C. Hyperparameters tuning

In this section, all the previous models are fine tuned and here their grid-search is reported.
Since, from the model selection, the best dataset turned out to be *data+geo_emb+mean_enc+descr*, from now on, all the models and the relative results have been obtained working on this dataset.
The fine-tuning process uses a cross-validation approach with five folds.

TABLE IV
GRID SEARCH

Model	**Parameter configuration**
Random Forest Regressor	max_features = ['sqrt', None]
	n_estimators = [100, 200, 300]
Elastic Net	alpha = [1, 0.5, 0.2]
	l1_ratio = [1, 0.5, 0.2]
	normalize = [True, False]
KNeighbors Regressor	metric = ['manhattan', 'chebyshev', 'euclidean']
	n_neighbors = [5, 10, 20]
	weights = ['uniform', 'distance']
Hist Gradient Boosting Regressor	l2_regularization = [1, 0.5, 0.2]
	learning_rate = [0.1, 0.01]
	loss = ['least_square', 'poisson']

III. RESULTS

In this section are reported the main results of the analysis.
The R^2 score reported in table V, is the mean of the five R^2 scores obtained by each model (with the best configuration

TABLE V
FINE TUNING

Model	Parameter config.	R^2 (avg folds)
Random Forest Regressor	max_features: 'sqrt' n_features: 300	0.8842
Elastic Net	alpha: 0.2 l1_ratio: 0.5 normalize: False	0.7526
KNeighbors Regressor	metric: 'manhattan' n_neighbors: 20 weights: 'distance'	0.8464
Hist Gradient Boosting Regr.	l2_regularization: 0.5 learning_rate: 0.1 loss: 'poisson'	0.8733

among the ones reported in table IV) during the fine tuning process (one score for each fold used in the cross-validation). The Random Forest regressor and the Hist Gradient Boosting regressor outperform all the others model with a R^2 score greater than 0.85.

All the models, with their best configurations, have been also evaluated in a test-set which contains new data not used for the training nor for the fine tuning process. The R^2 scores obtained in this evaluation are very close with the ones reported in table V.

Regarding the *public score* performance, since the Random Forest regressor is the best model, its predictions have been submitted to the Leaderboard achieving an R^2 score equals to 0.840.
Also the other models predictions have been submitted achieving, as expected, lower scores.

Comparing the Random Forest regressor's score with the leaderboard baseline, the quality of the prediction is improved from an R^2 score equals to 0.436 to an R^2 score equals to 0.840.
Also, the difference between this solution and the head of the leaderboard is around 0.05.

IV. DISCUSSION

It is quite interesting the fact that even the linear regression is able to achieve an R^2 score equals to 0.76 if evaluated with the (data + geoemb + meanencoding + descr) dataset. This probably means that the preprocessing applied to the original data, thanks to the Poincarè embedding, and FastText, is very effective.

The difference between the score obtained in the leaderboard and the one reported in table V is quite big. This is probably because not all the preprocessing methods are unbiased.
For example, the mean encoding is a very effective technique but it suffers from overfitting, and, even working with only a sample for each category is not enough to avoid the problem.

Possible improvements can be probably found in a better fine tuning of the models. In this project, very few combinations of parameters have been tested for each model (due to a too long computational training time).
Also in the pre-processing phase there are some parameters that should be fine tuned in order to improve the results, for example, the sampling dimension in the mean encoding. Also the FastText model should be better tuned: in this project, the automatic tune process implemented by FastText has been used but only few parameters configurations have been texted due to the too long train time.

Regarding the description, probably a proper pre-processing (for example tf-idf) could have been useful for better improving the predictions of the fasttext model.

REFERENCES

[1] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems*, pp. 6338–6347, 2017.
[2] F. Dassereto, L. Di Rocco, G. Guerrini, and M. Bertolotto, "Evaluating the effectiveness of embeddings in representing the structure of geospatial ontologies," in *International Conference on Geographic Information Science*, pp. 41–57, Springer, 2019.
[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.