# Data Science Lab: Process and methods

Daniele Giannuzzi
*Politecnico di Torino*
Student id: s290231
s290231@studenti.polito.it

***Abstract*—In this report we introduce a possible approach to the Wine Reviews Dataset regression problem. The project aim is to forecast the quality of a given wine. The proposed approach consists in using different encoding ways for the dataset categorical features. The proposed regression models outperform the baseline defined for the problem and obtain overall satisfactory results.**

## I. Problem overview

The proposed competition is a regression problem on the Wine Reviews Dataset, a collection of wine reviews. The proposed dataset consists of 150,930 entries, each of which refers to a review expressed by an expert on a specific wine. The dataset is divided into two parts:

• a *development set*, containing 120,744 records for which a quality value

• an *evaluation set*, comprised of 30,186 recordings, without the quality feature.

The goal of this competition is to correctly forecasting the wine quality of the evaluation dataset, using the development set to build a regression model.

Both the datasets contain wine reviews in tabular format. The numerical target variable of the regression problem is *quality*: a value, given by an expert, that identifies the wine quality on a scale of 0-100, as shown in Figure 1.

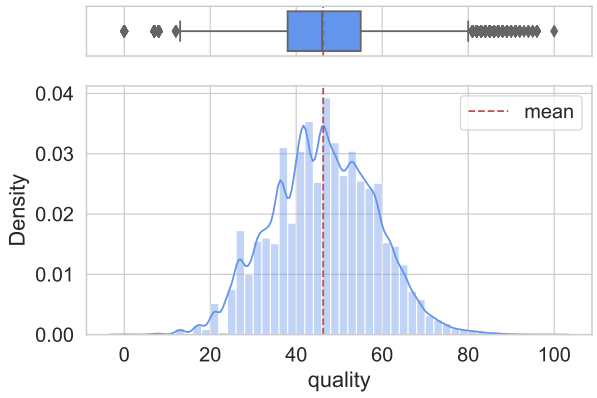


Figure 1: distribution of quality values

The distribution of target variable values looks like a normal distribution. Moreover, *quality* does not have missing values in development set.

Each review is also characterized by the following categorical attributes:

- *country*: the country that the wine is from;
- *description*: few sentences from an expert describing the wine flavour profile;
- *designation*: the vineyard where the wine grapes are sourced;
- *province*: the province or state that the wine is from;
- *region_1*: the wine growing area in a province or state;
- *region_2*: a more specific region in wine growing area;
- *variety*: the type of grapes used to make the wine;
- *winery*: the winery that made the wine.

Regarding development set, the count of the unique values for each attribute is summarized in Table I.

| Feature | Number of categories |
|---|---|
| country | 49 |
| description | 85005 |
| designation | 27801 |
| province | 445 |
| region_1 | 1207 |
| region_2 | 19 |
| variety | 603 |
| winery | 14105 |

TABLE I: amount of unique categories for each feature

We can make some considerations based on the table. First, as might be expected, *description* has the largest number of unique values. Second, number of different descriptions (85,005), is lower than number of development set entries (120,744). Therefore some null or replicate values must be there. To have a better understanding of this phenomenon, we focus our attention on the development set missing values.

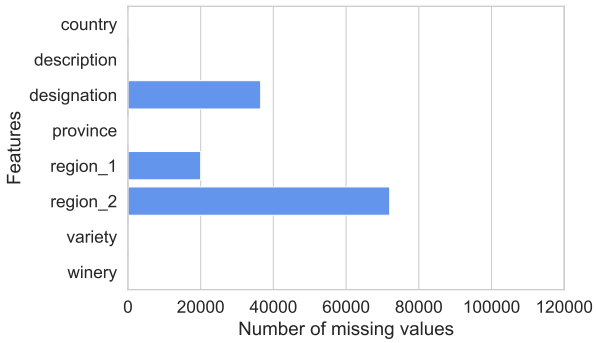


Figure 2: amount of missing values for each feature

| Feature | Missing values count |
|---|---|
| country | 5 |
| description | 0 |
| designation | 36518 |
| province | 5 |
| region_1 | 20008 |
| region_2 | 72008 |
| variety | 0 |
| winery | 0 |

TABLE II: amount of missing values for each feature

As we can notice from Table II, *description* does not have missing values. This means that development set has 35,739 duplicate descriptions. This also suggests that, most likely, there are duplicate entries within development set. Indeed there are 35,716 identical entries: the 29.6% of the entire development set. Intuitively, these duplicate objects are not different objects with same attributes values. In fact, although equal values of most of the attributes, such as *country, designation, province, region, variety, winery and quality*, can refer to two different reviews, however at least the wine description should be unique for each review. We conclude that exact duplicates are identical data objects, due to the used data collection method, probably merging data from heterogeneous sources.

Concerning geographical attributes: *country, province, region_1, region_2*, we observe they have increasing granularity. Moreover, as Figure 2 highlights, *region_2* has a lot of missing values. A thorough analysis shows that *region_2* values are specified for the US country only.

## II. Proposed approach

### A. Preprocessing

We have seen that development set contains exact duplicates. So we proceed in data cleaning by removing them. Now development set is comprised of 85,028 records.

There are now 24 duplicate descriptions. Analyzing entries sharing same description we notice they are strongly affected by noise, due to different quality values or even different wine type. We choose to remove these few data objects.

We can also reduce the noise due to capital letters and different accents. For instance, ”Château Bianca” winery does not differ from ”Chateau Bianca” or ”chateau bianca”. Therefore, accents removing and case normalization are applied on categorical features. In this way we avoid that categories are considered different from each other, only due to these two reasons.

Regarding handling missing values, after data cleaning missing value distribution is still similar to the one shown in Figure 2. Dealing with categorical data makes missing values estimate very tricky. We could use the description feature to predict missing values. However the description is usually centered on the wine flavor profile, rarely mentioning designation or wine growing area. When it does, values are not missing.

As we have many high-cardinality categorical attributes to train our model, as Table I explains, we have to be careful during the encoding phase. Following encoding ways are possible.

• *Target encoding*: it is a categorical encoding using target variable statistics, based on Empirical Bayesian statistical method [1]. In ordinary target mean encoding, each attribute value is replaced with the mean target value for that category. But, relying on an average value could cause overfitting when the category frequency is low. To mitigate the effect of low frequency categories, *targetEncoder* from *category_encoders* uses a weighted blend of priori probability of the target over all the training data and posterior probability of the target given a specific categorical value.

• *One-Hot encoding*: it works quite well on categorical data but it causes problems when number of distinct categories in any features becomes too large:it would produce a huge number of columns. Therefore, before one-hot encoding, feature selection is required.

Furthermore since categories with low frequency can affect negatively the robustness of the model, assigning a general category like ”Other” to less frequent values helps to keep the robustness of the model. Using a larger or smaller frequency, for each attribute treated in this way, affects the performance of the model.

To understand how important each feature is first of all we remove *description* column. We split development set in 80/20 train/test and we Target encode remaining categorical features. A naive *RandomForestRegressor()* can help us in feature selection, determining how useful each feature is for the model.

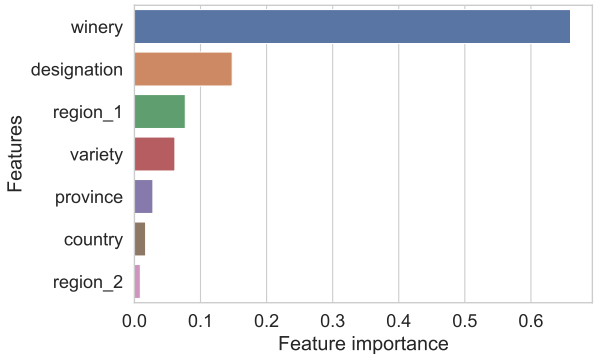


Figure 3: feature importance using Target encoding and Random Forest Regressor

As we can see from Figure 3, as expected, the model assigns a very high feature importance to the *winery*. Despite the missing values, also the *designation* and *region_1* seem to be useful for the prediction. Much lower importance is assigned to the most and least granular geographical information: probably because *country* is an attribute that generalizes too much and *region_2* has too many missing values.

Concerning *description*, tf-idf (term frequency–inverse document frequency) led to different results depending on the

encoding method used. We can however affirm that the description does not have much weight in the prediction. It is very likely due to the fact that the description is more focused on the wine fruity flavor rather than its goodness. To extract additional knowledge from this feature, we can also imagine that the expert who writes the review could write more words for a great wine and less for a poor one. Analyzing the correlation between *quality* and description length in terms of number of words, as Figure 4 highlights, there is no strong correlation that can improve model performance. In fact, testing models with *Description_Length* feature, performance does not improve or degrade. So we can avoid adding this attribute.

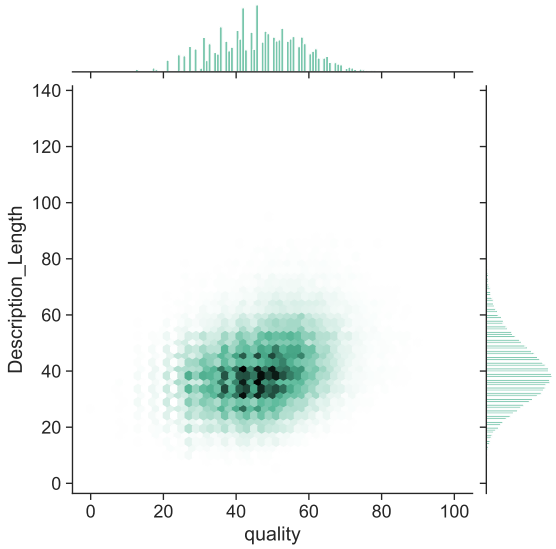


Figure 4: hexbin of quality and description length

### B. Model selection

For *Target encoding* approach, the following models have been selected due to the higher performance:

• *Random Forest Regressor*: it is characterized by many decision trees. It reduces decision tree overfitting problem, thanks to bootstrap aggregation and feature bagging. The idea is to combine multiple decision trees in determining the final output rather than relying on individual decision tree. The final output is the mean of all the decision trees outputs. The performance of a random forest scales with the number of estimators, up to a certain point.

• *K-Nearest Neighbors Regressor (KNN Regressor)*: it is a very simple algorithm. Given an evaluation test data point, the closest K data points of development set are selected, based on the chosen distance metric. The average of these data points is the final prediction for the new point. Despite its simplicity, it has proven to be incredibly effective at certain tasks.

For *One-Hot encoding* approach, the following models showed the best results:

• *Multi-layer Perceptron Regressor (MLP Regressor)*: it is a class of feedforward artificial neural network that utilizes a supervised learning technique called backpropagation for training. It consists of three layers of nodes: input, hidden and output layers. It is the one with the highest performance among the proven regressors.

• *Linear SVR*: Support Vector Machine can also be used as a regression method: it tries to minimize error, identifying the hyperplane which maximizes the margin.

For all these regressors, the best-working configuration of hyperparameters has been identified through a grid search, as explained in the following section.

### C. Hyperparameters tuning

There are two main sets of hyperparameters to be tuned:

• feature selection and feature frequency threshold for the preprocessing

• models parameters

For hyperparameters tuning we can use an 80/20 train/test split on the development set.

For *Target encoding* approach, description is not helpful in achieving better results so it is ignored.

| Model | Parameter | Values |
|---|---|---|
| Preprocessing | *feature selection* | {country, region_2} |
| Random Forest | *max_depth* | {None, 3, 5 ,7} |
| | *criterion* | {mae, mse} |
| | *n_neighbors* | $10 \rightarrow 50, step: 5$ |
| | *n_estimators* | {64, 128, 256} |
| | *max_features* | {auto, sqrt, log2} |
| KNN | *weights* | {uniform, distance} |
| | *p* | {1, 2} |

TABLE III: Hyperparameters considered: target encoding

Concerning *One-hot encoding* approach, description treated with tf-idf has significantly improved the scores. About the *TfidfVectorizer()*, a gridsearch on max_features was applied to choose the right number of features to encode. Instead of *MinMaxScaler()*, the *MaxAbsScaler()* was used to normalize the tf-idf generated columns. It allows to keep the sparsity of the matrix, having in this specific case the same behavior of the *MinMaxScaler()*. Furthermore, a sequential grid search on the minimum frequencies threshold for the categories of the attributes *winery* ($n_v$), *designation* ($n_d$), *region_1* ($n_{r1}$) was made. The less frequent categories of each of the attributes have been labeled as "other".

| Model | Parameter | Values |
|---|---|---|
| Preprocessing | *feature selection* | {country, region_2} |
| | $n_w$ | {2, 3, 4} |
| | $n_d$ | {2, 3, 4} |
| | $n_{r1}$ | $1 \rightarrow 20, step: 5$ |
| | *tf-idf max_features* | $500 \rightarrow 3000, step: 500$ |
| MLP Regressor | *activation* | {relu, tanh} |
| | *hidden_layer_size* | {(100,), (128,128), (256, 256)} |
| | *validation_fraction* | {0.1, 0.2} |
| LinearSVR | *epsilon* | {0,1,2,3} |
| | *C* | $1 \rightarrow 10, step: 1$ |
| | *fit_intercept* | {True, False} |
| | *max_iter* | {4000, 5000, 6000} |

TABLE IV: Hyperparameters considered: one-hot encoding

## III. Results

For *Target encoding* approach, the best configuration for both the regressors was obtained by keeping both *country* and *region_2*.

• The best configuration for Random Forest Regressor was found for: {*n_estimators=128*, *max_depth=None*, *max_features=sqrt*} ($R^2$ score $\approx$ 0.704).

• The best configuration for the KNN Regressor was found for {*n_neighbors=30*, *weights=distance*, *p=1*} ($R^2$ score $\approx$ 0.682).

The two regressors achieve satisfactory and similar results.

As for *One hot encoding* approach, the best configuration for both the regressors was obtained by removing both *country* and *region_2*. The other chosen preprocessing values are: {$n_w$=1, $n_d$=1, $n_{r1}$=10, *tf-idf max_features=2000* }

• The best configuration for MLP Regressor was found for: {*activation=tanh*, *hidden_layer_size=(100,1)*, *validation_fraction=0.1*} ($R^2$ score $\approx$ 0.745). The public score it achieved is ($R^2$ score $\approx$ 0.816).

• While for Linear SVR the best configuration was found for: {*epsilon=2*, *C=7*, *fit_intercept=True*, *max_iter=5000*} ($R^2$ score $\approx$ 0.737). The public score it achieved is ($R^2$ score $\approx$ 0.790).

Therefore the best regressor from an $R^2$ score point of view is the MLP Regressor, followed by: LinearSVR, Random Forest and KNN.

## IV. Discussion

We highlight how keeping duplicates leads to higher public scores on the platform. This is probably due to the fact that duplicates on which the model has trained, have attribute values very close to those of the evaluation set. In fact, all the models showed to have better performances on the public score, keeping the duplicates. For instance, the Random Forest Regressor with a greater number of trees (1028) to have an higher score, can reach ($R^2$ score $\approx$ 0.849). Also the MLP Regressor with its ($R^2$ score $\approx$ 0.846) increased its performance. We can imagine that for the private score things could change.

The score MLP Regressor achieved with duplicates is preferred (i.e. chosen for the score evaluation) to the Random Forest one, because it probably suffers less overfitting, having very similar scores even without duplicates. Therefore it is likely that it can get a better private score.

Anyway there is no doubt that MLP, removing duplicates, with its ($R^2$ score $\approx$ 0.816), is definitely the model to choose. For this reason it was chosen for the score evaluation (along with its version with duplicates).

Despite *Target Encoding* regression showed worse results than the one with *One-Hot Encoding*, however it showed satisfactory $R^2$ scores with a much faster and naïve approach.

Another consideration we can make is that, as expected, all the models outperforms MLP Regressor in terms of time using.

## References

[1] D. Micci-Barreca, *A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems*. SIGKDD Explorations, 3, 27-32, 2001.