# Regression analysis for wine quality estimation

Pierluigi Compagnone
*Politecnico di Torino*
Student id: s288301
s288301@studenti.polito.it

***Abstract*—The main goal of this homework is to build a regression model able to estimate the quality score of the wines. In this paper all the main steps performed are presented and motivated. At the end, the results obtained with the described approach are reported and briefly discussed.**

## I. Problem overview

The proposed task is a regression analysis on a dataset of wines reviews, with the purpose of estimating the quality of the wine.
Each review is described by eight attribute features and is associated to the quality score. It is a numerical target variable used to rate the wines in a range between 0 and 100. Most features are information about the wines and their provenance, such as the *variety* or the *country* that the wine is from. The only non-categorical feature is the *description*, that is a textual review on the wine, made by an expert.
The dataset is divided into two parts:

- The *development* set, containing 120744 reviews with also the quality score. This set is used to train and validate the model.
- The *evaluation* set, composed by 30186 reviews that do not contain the score. It is used to test the actual performance of the final models.

From the observation of the development set some interesting insights are extrapolated. First of all it contains about 35000 duplicates, that could affect in a bad way the training phase of the regression model. Another important information is the relevant number of missing values in different columns, as showed in Figure 1. It can be seen that the features most affected by their presence are *designation*, *region1* and *region2*. It is important to decide how to deal with missing values, because they can not be directly dropped, otherwise the size of the data would reduce significantly. Moreover, the massive presence of missing values suggests that they could be significant and they could probably be present in the test set too.
A fundamental step of the project is dedicated to the data transformation, because almost all the regression models can process only numerical variables. For this reason both the categorical and the textual features must be transformed in raw data.
After the data are ready, they are used in a first analysis to evaluate the performances of different regression models, in order to choose the ones most capable to predict the quality of the wines. Then the models are improved through hyperparameters tuning and finally they are used to estimate the quality of the wine review in the evaluation dataset.

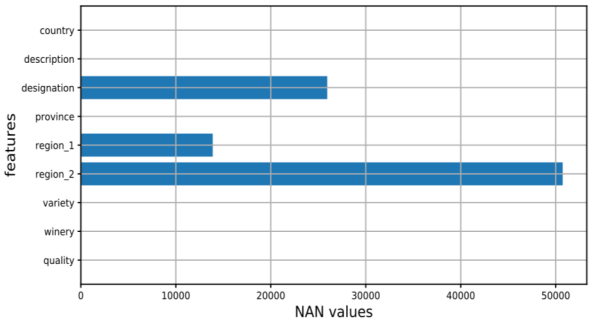


Fig. 1. Nan values over the features

## II. Proposed approach

### A. Preprocessing

In this section are presented the different operations computed to prepare the data before to use them to train and test the regression model.
How it was previously said, the development dataset contains duplicates, so firstly they are removed. In this way the final model is not trained on the same review more than once, otherwise it could loss the ability of generalization. The development dataset without the duplicates is composed by 85028 reviews.
The successive part is dedicated to the missing values which are principally located in the attributes *designation*, *region1* and *region2*. About these features some considerations can be done, in fact both the designation and the regions are attributes that do not characterize all the wines. For example many wines on the market do not report the designation and the second region on their labels. The feature with most null entries is the second region, because it is a more specific location, not often indicated. In rare cases the wines do not report information neither about the first region, but only about the country of provenance. For the reasons above, the absence or the presence of a value can be considered itself an information.
To solve the problem about null entries, firstly those in *region2* are filled with the values in *region1* (where they are

present), because it is observed that in some cases the first and the second regions are the same. Then the remaining null entries of the three features mentioned before are filled with the string "missing". After this process the records that yet contain missing values are dropped, reducing the size of the development set of only three rows.
The last *data preprocessing* step performed is the data transformation, in fact, as underlined in previous section, the categorical and textual features must be transformed in numerical data to be processed by the regression models.

- **Categorical attributes:** The features are extracted using the *One Hot Encoding* technique, that transforms each categorical feature to a group of bits: each one is related to a different value assumed by that features. For each record, the only features set to 1 are those corresponding to the values assumed by the different categorical attributes. The chosen technique allows also to deal with data that are not seen by the encoder in training phase, in fact all the new unknown data are mapped to the zero vector. It results to be an advantage because in the evaluation set some categorical features contain values that are not present in the development.
  With this encoding technique over 40.000 columns are generated by the seven original features. As directly consequence of the increasing size of the dataset also the computation time needed to process the data increases.

- **Description:** For the textual descriptions, two different data transformations are evaluated: the *Bag of Words* and the *tf-Idf*.
  These are textual representations that transforms the sentences in vectors of the size of the entire vocabulary, composed by all the words appeared in the descriptions. In the first method, the values assumed by the features extracted are the occurrences of the corresponding words in that particular review. While for the *tf-idf* the values are calculated in order to penalize the words that appear in many documents.
  These methods can be also extended to the *n-grams*, that are the combinations of $n$ adjacent words.
  Firstly, the descriptions are cleaned up by the stop words. Then they are processed in order to reduce all the words to their lemma. At the end, the textual data are transformed with one of the two methods mentioned above.
  With both the *Count Vectorizer* and the *Tf-idf Vectorizer* the columns would increase too much and along with them also the computational time. So, some hyperparameters are tuned in order to ignore some n-grams, in relation to their frequency along the documents.

Of course, both the categorical encoder and the Vectorizers are trained only on the train subset in the model building phase and on the entire development set when the final model is performed.

### B. Model selection

The analysis on the performances of different regression models is computed, with the aim to find the ones most capable to estimate the quality score of the wines. The results of the models are compared considering the *R2 score* as benchmark. All the experiments, made to select and improve the models, described in this section and in the successive one, are performed using as train and validation sets two portions of the development set. The 80% of the original data is used as train, and the remaining part as validation. This technique of model evaluation is known as *hold-out*. The same train and validation sets are used for all the tests computed in this homework, to make the results more comparable.
For the selection of the model the features associated to description are not used: the idea is to find the model that performs better using only the one hot features, generated by the categorical attributes; then try to improve the results, using the features generated by the descriptions.
Some methods tested in this phase are the *Linear regression*,the *SVR*, the *Multi-layer perceptron (MLP)* and the *Random forest* [1]. They are compared with their default configuration, except of *MLP*, whose configuration is explained successively.
The only models that obtain competitive *R2* values are *MLP* and the *Random Forest*, with respectively **0.7205** and **0.6962**.

- **Random Forest Regressor:** This is an ensemble model, composed by multiple decision trees. Each one is trained on a different subset of data and features of the original training set, created randomly with replacement. In the prediction phase the target value assigned to a record is the average of the results of all the decision trees. In this way the performances of the model are improved and also the overfitting is limited.
- **MLP:** The *multi-layer perceptron* is one of the most simple neural networks. It is composed by three levels of neurons: the input, the output and the hidden layers. The number of neurons in the hidden layer can be decided in the building phase, while the neurons in the external ones are fixed and they respectively depend by the numbers of features in input and values assumed by the output.
  It is decided to use this model in order to make a comparison between the performances of the classical models, respect to a more advanced approach, such as a neural network. In particular, the *MLP* is selected, because it seems to work well with text data and because of its simple structure.
  The configuration used for this model in the selection phase differs to the default one because of the *early_stopping* set to *True*.
  - **early_stopping:** It stops the training phase until the fixed number of iteration are completed, if the score on the validation does not improve over a certain tolerance for ten consecutive iterations. In this way, the time needed for the training is reduced and it gives an idea about the right number of iterations to

use in the final model. The score, used as stopping condition, is computed on an internal validation set obtained as a split of 10% of the train set.
So, using this configuration, the model is fitted each time only on the 90% of the original train set.

### C. Hyperparameters tuning

Two main sets of hyperparameters are tuned:

- **Preprocessing hyperparameters**:
  - **ngram_range:** consider the n-grams for all the n integers in *ngram_range*.
  - **min_df:** consider only the n-grams that appear at lest in *min_df* documents.
  - **max_df:** consider only the n-grams that appear at most in *max_df* documents.

  In Table 1 are shown the different configurations tested, along with the number of features obtained. These settings are used for both the *Count Vectorizer* and the *Tfidf Vectorizer*. This step is performed in particular for the practical computational limits, in such a way to keep the number of features extracted under a certain treshold, so that also the computation time does not increase too much. The idea is to ignore the *n-grams* that are the most or the less frequent above all the documents, because they could be the less significant for the analysis.

TABLE I
CONFIGURATIONS USED FOR THE *Vectorizers*

| ngram_range | min_df | min_df | features |
|---|---|---|---|
| (1,1) | 10 | 10000 | 6388 |
| (2,2) | 27 | 1000 | 6458 |
| (3,3) | 8 | 140 | 7267 |

- **Model hyperparameters**:
  - **Random forest regressor**:
    - **n_estimators:** number of decision trees in the model.
    - **max_features:** number of features considered for each split.
    - **max_depth:** the maximum depth of the trees.
    - **max_samples:** the size of the subsets used to train each tree.
  - **Multi-layer perceptron**:
    - **learning_rate:** it determines the step size at each iteration, while moving toward a minimum of a loss function. It is set to *adaptive* for all the experiments, so the *learning rate* is reduced each time two consecutive epochs fail to decrease the loss function of a certain tolerance.
    - **solver:** The algorithm used for the optimization of the network's weights.
    - **learning_rate_init:** the initial value of the *learning rate*.
    - **hidden_layer_size:** number of neurons in the hidden layer.

Firstly, it is decided the best data transformation configuration, this step is made separately for the two methods.
For the *MLP* only two alternatives are evaluated: the *tf-idf* and the *bag of words*, both at the word level. Bi-grams and tri-grams are not evaluated for this model, because considering only the uni-grams already gives acceptable improvements. Otherwise, for the *Random Forest*, all the combinations of the two textual data transformations with the configurations in Table 1 are evaluated.
To decide the textual features most appropriate for each model, these kind of features are used together with the *One Hot vectors*, generated by the categorical attributes. Then, the configurations that gives the highest improvement, respect to the *R2* obtained without textual feature, is selected.
For the selection of the best hyperparameters a grid search is computed, for both the *MLP regressor* and the *Random forest regressor*. The hyperparameters considered are shown respectively in Table 2 and Table 3.
How it can be seen from Table 2 different values of learning rate are evaluated for the different solvers, *adam* and *sgd*.
For the *Random forest* the grid search is splitted as shown in Table3, to reduce the number of combination evaluated. The second one is performed on the hyperparameters of the last two columns in addiction to the best configuration turned out from the previous research.

TABLE II
HYPERPARAMETERS OF THE *Multi Layer Perceptron*

| solver | hidden_layer_size | learning_rate_init | R2 |
|---|---|---|---|
| *adam* | 200 | 0.001 | 0.7311 |
| *adam* | 200 | 0.0001 | 0.7368 |
| *adam* | 300 | 0.001 | 0.7308 |
| *adam* | 300 | 0.0001 | 0.7391 |
| *sgd* | 200 | 0.01 | 0.7544 |
| *sgd* | 200 | 0.001 | 0.7485 |
| *sgd* | 300 | 0.01 | 0.7569 |
| *sgd* | 300 | 0.001 | 0.7505 |

TABLE III
HYPERPARAMETERS OF THE *Random Forest Regressor*

| first greed search | | second greed search | |
|---|---|---|---|
| n_estimators | max_features | max_depth | max_samples |
| 100 | 0 | 100 | 1000 |
| 200 | 5 | 500 | 5000 |
| 350 | 10 | 1000 | |
| 500 | | | |
| 600 | | | |

## III. RESULTS

- **Random forest regressor:** the best data representation turned out for this model is the one composed by only the One hot features. In fact all the attempts done considering also the features extracted by the descriptions reach a lower value of *R2 score*.
  The first grid search highlights a strange behaviour of the model, that is not so sensitive to the variation of the

number of trees as usual. Otherwise, the most incisive parameter pointed out by the research is the *max_features*. This unusual attitude is clearly shown in Figure 2. It can be seen that the best value founded for *max_features* (5) gives an *R2* improvement enough stable about the 0.034, while the numbers of estimators appears quite useless.
The second grid search, computed for this model, does not produce any improvement, instead anyone of the configurations explored produce much less performing models.
The final configuration chosen for the model is the following: *n_estimators* = 350, *max_features* = 5. That reaches a *R2* of **0.7340** on the validation set, with an improvement greater than 0.3 respect the default model.

- **MLP:** Contrarily to the other method, its performances are improved with the usage of the textual features. In particular, the *Tf-idf* generates an improvement about 0,01 over the model used in the selection step, reaching **0.7310** as *R2*. The improvement is quite small, but it could increase with the tuning of the model's hyperparameters. So the tf-idf features are selected as textual feature for the successive phases computed with this model.
The results obtained with the grid search are shown in Table 2. It can be seen that the best configuration is: *learning_rate* = 'adaptive', *solver* = 'sgd', *hidden_layer_size* = 300, *learning_rate_init* = 0.01, that reaches a *R2* of **0.7569**. This result is obtained with *early_stopping* set to True and the training phase is automatically interrupted after 108 iterations. For the final model used to estimate the scores of the evaluation set, *early_stopping* is set to False, to use the complete development set in the training phase. Otherwise, the number of maximum iteration is set to 108 iterations, as turned out in the grid search.

At the end both the final configurations of the model are trained on the entire development set, then they are used to predict the wine quality scores of evaluation set. The *Random Forest regressor* obtains a *R2* of **0.858**, while the *Multi Layer Perceptron* reaches **0.889**.

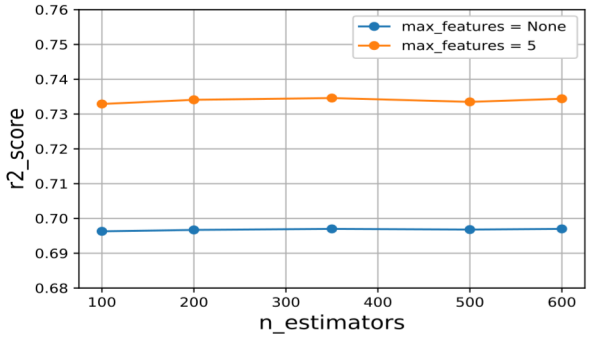


Fig. 2. R2 trend respect to the number of estimators

## IV. Discussion

The neural network based approach is resulted to be more adequate to analyze the proposed dataset, in particular because it is able to handle with the descriptions.
The *Random Forest* also gives competitive results, although using only the categorical attributes. From the experiments with this model, is turned out that it is not compatible with all the proposed text representations. Probably, it is also due to difficulty of the problem itself. In fact, it can be considered a difficult problem to extract useful insights, on texts all about the same argument.
A deeper analysis could be performed on both the two models, with the aim to improve their performances. For this purpose, the following are some aspects that might be investigated.

- For the *Random forest* it would be interesting to understand if the problem with the textual features, is due to the limited size of the dictionary chosen. Otherwise some alternative textual feature representations should be examined, in order to find a suitable one for the model. For this purpose the *Word Embeddings* should be elaborated.
- The *Multi-layer perceptron* shows to be able to learn from the descriptions, however also in this case other textual representations could give better results, so a deeper analysis in these sense can be performed.
Probably, for this model the performance could be easily improved exploiting other configurations. In fact there are a lot of hyperparameters that can be tuned in a neural network. About those analyzed in this homework, some other experiments could be done, increasing the number of neurons in the hidden layer to enhance the learning ability of the model. Also the learning rate could be object of a closer study, in this case probably lower values would be suggested.

## References

[1] *Scikit-learn* https://scikit-learn.org/stable/
[2] *Natural Language Toolkit* https://www.nltk.org/
[3] Shubham Jain (February 27, 2018) *Ultimate guide to deal with Text Data*
https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/