DSL Report
Wine Quality Regression

Armando La Rocca
Politecnico di Torino
s279401
s279401@studenti.polito.it

***Abstract*—This report contains a possible approach to the regression task on the wine quality dataset. Firstly, the problem is briefly explained. In the second section the proposed approach is presented and described. The last two sections contains a presentation of the results obtained and some considerations also on possible improvements.**

I. Problem overview

The following report present a possible solution to the competition related to the estimation of wine quality. Each wine is characterized by 8 categorical features: four related to the geographical provenience and three related to the winery, variety and designation. The last feature is a textual description of the wine. The problem is a regression task and the aim is to predict the quality of the wine that is represented by a value from 0 to 100.
The proposed dataset is divided in 2 splits:

- *development set* containing 120738 labelled samples.
- *evaluation set* containing 30186 samples.

First of all the development set is analyzed in order to extract useful information.
In the dataset are encoded a lot of wines of different types and provenience. However it is difficult that a dataset represents all the existing (or at least a large part of the) wines produced in the world. So, to increase the generality, data should be encoded in a way that allow to work also with categories that are not present in the development dataset or with missing values. Missing values, in fact, are quite frequent in the dataset, above all in certain features as shown in Fig 1.

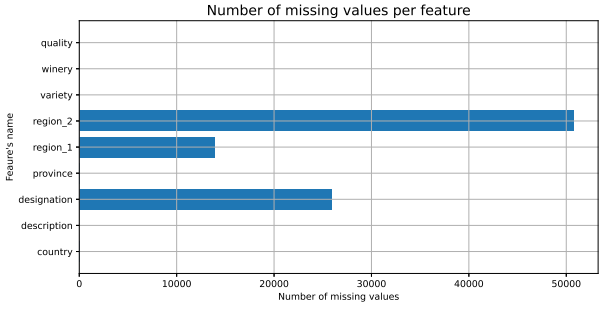

Fig. 1. Number of Missing values for each feature.

The most interesting feature is probably the description. For each wine is provided a textual description of the wine, however is not easy to analyze this kind of reviews. First of all because the descriptions are very similar even with a great difference of quality. An example is reported below :

'the nose on this single-vineyard wine from a strong, often overlooked appellation is tight and minerally before showing a slightly tropical kiwi element. brightly acidic on the lively palate, flavors range from key lime and meyer lemon to pear skins and apple flesh.' $\rightarrow quality = 100$

'clean as anyone should reasonably expect given the almost unheard-of price. it offers solid cherry and blackberry flavors, while the mouthfeel is round and totally inoffensive. drink it now; cook with it; make a sangria with it. just enjoy it.' $\rightarrow quality = 0$

In fact, between the two presented wines there is an high difference of quality but from the descriptions is not easy to catch that.

The choice of the encoding and a good management of the description features are crucial steps to obtain good performances.

II. Proposed approach

A. Pre-processing

First of all, a data cleaning step is performed on the development set. Duplicates are founded and dropped out from the dataset because represent not useful and redundant information. After this operation the development set accounts 85028 samples.
Then all the feature values are transformed in lower case to avoid mapping mistakes generated by different cases letters.
Another transformation is related to the designation category in which same wine categories appear in different languages. In Figure 2 can be seen as the feature *reserve* appears also as riserva and *reservé*. To exploit that the wine category is the same all the different names are mapped with the same label. In Table 1 are presented the transformations made to solve this problem.

The dataset is also characterized by missing values, above all in three features as represented in Figure 1. Considering the real case, it is possible that a wine is not characterized by a specific designation or that the specific region provenience is not provided. However, the quality evaluation should be possible also without this data. For this reason the missing

TABLE I
DESIGNATION MAPPING

New designation	Old designation
reserve	*reserva,réserve*
grand reserve	*gran reserva*
special reserve	*reserva especial*
red wine	*red, rosso, tinto*
rosé	*rosé of, rosato, rosado*

values of the features *region_1*, *region_2* and designation are filled with the string *other*.
In contrast, the samples with missing values in *country* features are dropped out (just 3 samples not distinguishable in Figure1).

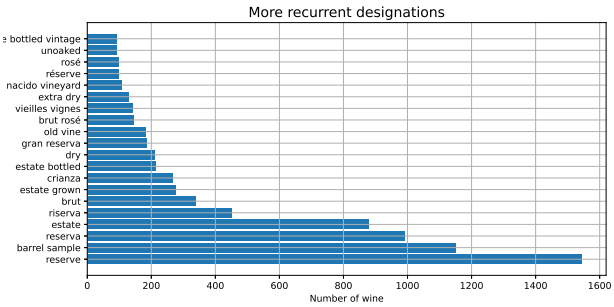

Fig. 2. Number of wines per designation

As said before, all the features are categorical except the description that is a textual review. Categorical features are transformed with the one-hot technique. This method produce for each feature a number of elements equal to the number of different values assumed by the features. Then put to 1 the bit associated to the value assumed by the feature for that sample and to 0 the others. Moreover, this approach allows to encode values that are not in the development set putting all the elements to 0. This is a good point for this analysis because also new or unknown wines can be encoded. However this encoding increases consistently the number of features of the dataset even if the data is quite sparse.
The preprocessing step for the description feature is more complex. Fist of all, each description is tokenized and the tokens containing numbers or punctuation are removed. Then, the tokens are lemmatized in order to avoid repetitions caused by inflected forms. Finally, all the obtained tokens are transformed to lower case and stop words are removed. This cleaning step is performed to exclude tokens that are not useful as numbers or parts of the text without specific meaning.
At the end to extract features from the obtained tokenized descriptions are used two different approaches :

- **Approach 1 (A1)** The descriptions are divided in three groups in according to their quality. From each group is created a vocabulary excluding the tokens that appear over 70% of documents and less that 5%. Then are selected only the words that are not in common with all the obtained vocabularies. This approach aim to find the words that most characterize wines belonging to a quality ranking of high,medium and low quality (as represented in Figure 3). At the end, features are constructed considering the count of the tokens of the obtained vocabulary in each sample description.
- **Approach 2 (A2)** The tokens are filtered excluding the words that appears in more than 50% of descriptions samples and less than in 10 reviews. Then features are constructed considering the term frequency of each token in the descriptions.

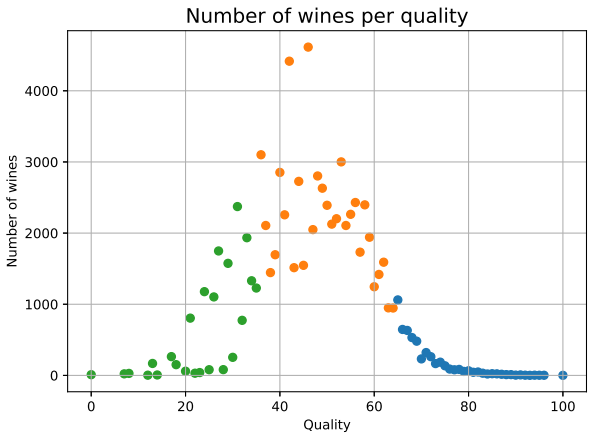

Fig. 3. Number of wines per quality. In green are represented wines associated with low quality, in orange medium and in blue high.

Fig. 4. Most recurrent words with approach A1

B. Model selection

For both the approaches proposed in the previous paragraph the number of features is high and data are quite sparse. This aspect can affect some algorithms for curse of dimensionality. For this reasons the models chosen for this tasks are:

- **Random Forest** [1]: this algorithm generates and train a given number of decision trees using the bagging method. Each tree works using one feature at time chosen between a subset of the original features. So the algorithm should be less affected by the high dimension of the data.

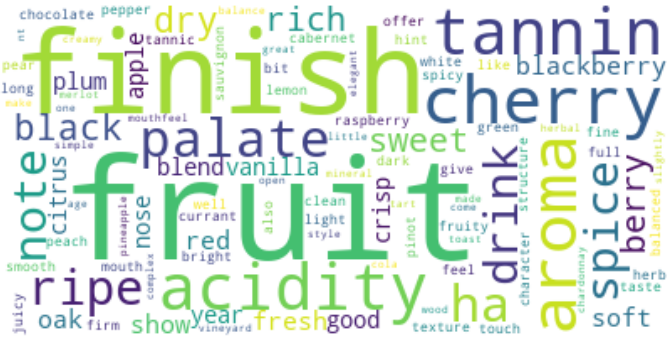

Fig. 5. Most recurrent words with approach A2

- **Multi Layer Perceptron** [2]: this algorithm is composed by a certain number of perceptrons (or neurons) interconnected and divided in three main layers: input, hidden and output. The model is trained updating the weights and biases of each neuron using a gradient descent method. The goal is to minimize the loss function that in this case is a *L2 loss* function. Also this method should works well on high dimensional data. Moreover, this algorithm seems quite popular on more complex data as text.

C. Hyperparameters tuning

The experiments are performed using a hold-out validation. The development test is splitted in two parts, the training set that accounts the 80% and the validation set that accounts the remaining 20% . This technique is chosen due to the time required for the training. In general, techniques as cross-validation are preferred as less biased, but in this case to exploit a good number of experiments this approach is preferred.

Now are presented the hyperparameters investigated with some gridsearch. The model is evaluated with different combinations of parameters and the performances are rated with the *r2 score*.
Concerning the Random forest the hyperparameters taken in account are :

- **Number of estimators** (n_e): number of decision trees used in the forest.
- **Max depth** (m_d) maximal depth of each tree.
- **Max features** (m_f) the number of features considered for each split.

For the MLP the choice of the hyperparameters can be very difficult .
Firstly, the maximum number of epochs is set to 200 and the early stopping technique is used. In this way the training is interrupted when a validation score that account the 10% do not improve for 10 consecutive epochs. This trick is performed to avoid a too long training that can cause overfitting. The stochastic gradient descent is chosen as solver. The hyperparameters varied in the gridsearch are:

- **Learning rate** (lr): this parameter set the step in updating model's weights. The choice of this parameter can affect significantly the results. In general, start with an higher learning rate and decrease it over the epochs make faster the training and improve the results. For this reason is used the "adaptive" configuration that decrease this value when the training loss do not decrease for at least two iterations.
- **Number of neurons in the hidden layer** (nn): in contrast with the input and the output layer, the number of neurons of the hidden layer can be set arbitrary. Higher is this number more complex is the model and able to learn more complex relations between input and output.

For both the algorithms the presented hyperparameters are investigated and tested considering the two preprocessing methods explained in section 2 and a configuration of features excluding the description feature (*ND*).

III. Experiments results

In tables II and III can be seen the results obtained from the performed gridsearch. From a first look can be seen that the random forest works better without features related to the description. In fact, with this algorithm the better configuration is trained without descriptions $\{ND, n_e = 500, m_f = 5, m_d = None\}$(*r2=0.7328*) while adding description's features the approach A1 works better.
Using MLP the results are slightly higher with all the presented set of features. However, in this case descriptions features increase the r2 score. The best configuration is obtained with the A2 approach $\{A2, nn = 500, lr = 0.01\ \}$ (*r2=0.7575*).
The MLP model with the configuration that achieve the best results is trained for 120 epochs on all the data of the development set and it is used on the evaluation set. The result obtained ($r2 = 0.891$) is quite higher respect to the validation so overfitting do not affect the model.

TABLE II
Random forest gridsearch

Hyperparameters			Features approaches		
n_e	m_f	m_d	*ND*	*A1*	*A2*
200	*5*	*2000*	0.6715	0.5492	0.4583
500	*5*	*2000*	0.6725	0.5499	0.4615
200	*10*	*2000*	0.7233	0.5484	0.4600
500	*10*	*2000*	0.7245	0.5485	0.4631
200	*sqrt*[a]	*2000*	0.7212	0.5468	0.5094
500	*sqrt*	*2000*	0.7230	0.5492	0.5111
200	*5*	*None* [b]	0.7318	0.5536	0.4579
500	*5*	*None*	**0.7328**	0.5545	0.4606
200	*10*	*None*	0.7303	0.5482	0.4602
500	*10*	*None*	0.7309	0.5494	0.4633
200	*sqrt*	*None*	0.7213	0.5470	0.5113
500	*sqrt*	*None*	0.7235	0.5496	0.5116

[a] *sqrt* stay for the square of the total number of features.
[b] *None* consider no pruning step for the tree.

IV. Discussion

The proposed approach works quite well obtaining competitive results respect to the proposed baseline. However there

TABLE III
MULTI LAYER PERCEPTRON GRIDSEARCH

Hyperparameters		**Features approaches**		
nn	**lr**	***ND***	***A1***	***A2***
100	*0.01*	0.7473	0.6776	0.7551
100	*0.001*	0.7369	0.7239	0.7504
500	*0.01*	0.7499	0.6641	**0.7575**
500	*0.001*	0.7417	0.7300	0.7524

are two main aspects that can be better analyzed to try improve the results.

- In the preprocessing two possible description's encoding are proposed. Text data are quite complex data and different other approaches can be tried. A first, very basic, possibility is to investigate the range of word excluded for high or low frequency in the sample descriptions. Also analyze group of words, as bigram and trigram, is an interesting ways. Word embeddings, actually, is considered one of the main technique to deal with text data and can be interesting understand if it can improve the results.
- Neural networks are characterized by a lot of different parameters that can affect the performances in a significant way. For this reason, a better study on the hyperparameters as other solvers(i.e. *adam*) or loss should be done. Also try more complex and deep models can be a way to improve the results.

REFERENCES

[1] Scikit-learn Multi layer regressor
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[2] Scikit-learn Random Forest regressor
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] Natural language toolkit
https://www.nltk.org