Wine Review Dataset Regression Problem

Paolo Rabino
Politecnico di Torino
Student id: s283282
paolo.rabino@studenti.polito.it

***Abstract*—In this report we introduce an approach to the *Wine Review Dataset* Regression problem. The proposed approach consists in the extraction of keywords through vectorization and lemmatisation from the description and the encoding of categorical features from the other attributes. The approach obtains overall satisfactory results.**

I. Problem overview

The competition consists in a regression problem on the *Wine Review Dataset*, a collection of reviews of different wines. The goal is to accurately assign a quality score to the wines. The Dataset is divided into two parts:

- a *development set*, containing 120000 reviews for which the quality score is present.
- an *evaluation set*, containing 30000 reviews.

We will use the development set to build a regression model to score accurately the reviews of the evaluation set. The database is composed of 8 columns, plus the quality one that we need to predict.

TABLE I
Databset Attributes

Name	Type	cardinality
Country	Nominal	48
Description	Text	85005
Designation	Nominal	27800
Province	Nominal	444
Region_1	Nominal	1206
Region_2	Nominal	18
Variety	Nominal	603
Winery	Nominal	14105

The attributes are all strings of text, the description is a small review of the wine while the others are all categorical. The problem appears well balanced as the development set qualities are approximately distributed as a normal distribution centered on 46, the quality score goes from 0 to 100. We can understand better the contents of the description column through manual inspection, the reviews often contain informations about the taste of the wine and critiques about the quality, this can be highly meaningful for our goal.

II. Proposed approach

A. Preprocessing

From table 1 we can see that there are less unique Descriptions then total number of reviews, this means that probably there are duplicates in the development set, dropping duplicate rows we reduce the set to 85000 unique entries.

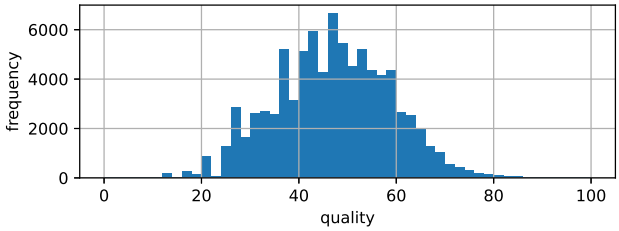

Fig. 1. Distribution of the quality

There are many null values in the *designation*, *region_1*, *region_2* columns. We map the null values to a ”0” nominal value as the absence of the attribute could be meaningful for our prediction.
The categorical attributes can be all encoded through the OneHotEncoder technique, this means generating a new feature column for every possible value.
Assuming that a single cell is saved as an integer value (4 Bytes), a normal matrix would occupy approximately 14 GB.

$$\sum_{i=0}^{n} Card(column_i) * 85000 rows * 4 Byte \approx 14\ GByte$$

Fig. 2. Memory usage of a the feature matrix

A possible solution is using a sparse matrix representation, where for each row only the non-zero cells are saved.

$$n * 85000 rows * 4 Byte \approx 2.5\ MByte$$

Fig. 3. Memory usage of the feature matrix with a sparse matrix representation[2]

The *description* is encoded with a bag of word representation, the *TfidfVectorizer* is used to select the most significant words. The steps used are:

1) Tokenization: the *description* is separated into a word list

[2]This is an approximation and the effective size depends on the implementation

2) Lemmatization: the words are lemmatized, they are reduced to their "basic" form through morphological analysis. (eg. {wines, wine, wine's} $\implies$ wine)
3) Stopwords Removal: all words that are considered useless for the core meaning of the phrase are removed (eg. conjunctions, prepositions)
4) Terms that exceed or are below set cutoff Document frequencies are removed as they are less significant

The nltk library has been used for the Tokenization, Lemmatization and Stopwords Removal steps as it outperforms the scikitlearn standard implementation. The values of the features extracted from the description are the tf-idf measure of the term in relation to the training set. The number of features extracted in this step is very high, we can again benefit greatly from the sparse matrix representation. The number of features ($n\_features$) extracted in this step and the cutoff frequencies ($freq_w$) can be considered hyperparameters for the preprocessing step.

B. Model selection

The following algorithm have been tested:

- *Random Forest Regressor*: The main feature of the algorithm is the possibility of computing the feature importance, this has been very useful in the data exploration phase but it does not scale well with the high number of features.
- *Ridge*: This model handles well high feature data that suffers from multicollinearity [1]. Multicollinearity happens when there is high correlation between predictor variables in a regressor model, this is the case as most of our features are negatively correlated, because they were obtained through One-Hot-Encoding. Ridge is also optimized for the usage of sparse matrixes in the scikitlearn standard implementation.

Only the *Ridge* classifier has been used on the whole dataset.

C. Hyperparameters tuning

The two main sets of hyperparameters to be tuned are:

- $freq\_min$, $freq\_max$ $n\_features$ for the preprocessing
- the *ridge* parameters

The K-Fold cross validation technique with k = 10 is used with a grid search approach. The evaluation is done with the mean r2 score of the 10 predictions.

TABLE II
HYPERPARAMETERS CONSIDERED

Model	Parameter	Values
Preprocessing	$freq\_min$	{0,0.005,0.01}
	$freq\_max$	{ 0.9,0.95,1}
	$n\_features$	{1000,2000,3000,4000}
Ridge	α	{0.1-1.2}

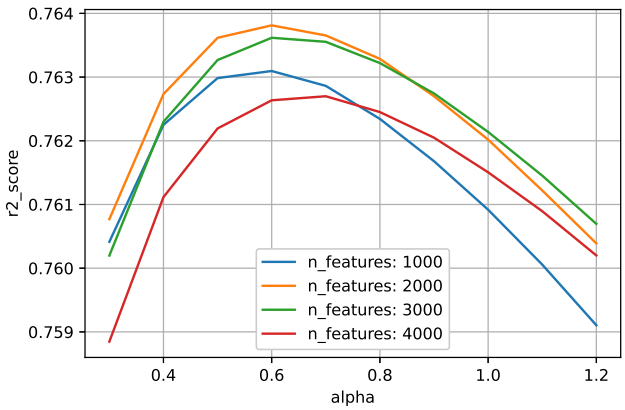

Fig. 4. Performance of Ridge for different number of features extracted as α varies

III. Results

The cutoff frequencies were not found to be useful, probably because the tf-idf weighting schema already addresses the problems derived from outliers and too frequent terms. The tuning of the parameters α and $n_features$ is summarized in Figure 4. The best configuration found was $\{n_features = 2000, \alpha = 0.6, freq\_min = 0, freq\_max = 1.0\}$.
We trained a new Ridge classifier with the best hyperparametrs found, on the whole training set and classified the evaluation set. The public score obtained is of 0.848. Using other configurations, mainly by increasing the number of extracted features we can increase the score even further, up to 0.858.

IV. Discussion

The difference in the scores obtained probably means that we overfitted on the evaluation set. To address this problem the final submissions sent will be the best performing in the cross validation step and the best performing in the leaderboard.
Cross validation with k-fold decreased the likelyhood of Overfitting due to the high complexity of the model selcted. The solution found performs well above the baseline. It does so by leveraging the compactness of the sparse matrix representation that allows the use of an high number of features. The results obtained are promising and there seems to be little room for improvement, some experimentation can be still done on some aspects:

- Use other Regression techniques
- Run a more thorough grid search, testing different configurations for the vectorization step of the *Description* attribute
- Encoding differently *null* values

References

[1] A. Bager, M. Roman, M. Algedih, and B. Mohammed, "Addressing multicollinearity in regression models: a ridge regression application.," 2017.