A Gradient Boosting approach to Regression with Qualitative Predictors and Textual Data

[redacted]
Politecnico di Torino
Student id: [redacted]
[redacted]@studenti.polito.it

***Abstract*—In this report we propose a possible method to address the problem of predicting a numerical target variable starting from categorical predictors and textual data. We show that the proposed approach, based on gradient boosting, performs significantly better than the random forest used as baseline.**

I. Problem overview

The objective of the competition was to predict the numerical quality score assigned to a wine in a range between 0 and 100. The data set provided consists of two different parts:

- A ***development*** set of 120,744 objects characterised by the numerical target, a review and a set of categorical features, containing information about the characteristics of the wine (*designation* and *variety*) and the producer (*country, province, region 1, region 2* and *winery*).
- An ***evaluation*** set of 30,186 data points with the same structure of the development set, except for the numerical target.

The original development set contains 35,716 **duplicate** objects. In this context the presence of duplicates is an artefact probably consequent to the merger of data coming from different sources, rather than an indication of frequently occurring patterns. Duplicates are a significative fraction of the development set and this may severely bias our model. For this reason we decide to discard them from our analysis keeping only the first occurrence of each object.

According to a preliminary inspection, summarised in table I, some features have a very large number of **unique** and/or **missing** values. The presence of missing values is a major problem for many regression models, while the high cardinality of the domain for some of the categorical attributes may lead to a very large and sparse feature matrix (more than 44,000 columns with one-hot encoding).

To address these two problems, we may try to consider only features with relatively few missing and unique values (i.e. *country*, *province*, and *variety*). On the other hand, we have already discarded roughly the 30% of the original records in the duplicates pruning step. This approach would reduce even more the amount of training data, leading to a model with very poor generalisation capabilities. Moreover, some of the discarded features may be very useful to discriminate high quality wines from low quality ones.

For instance, Figure 1 illustrates how the wine quality is distributed among the top 5 frequent countries and wineries.

TABLE I
Unique and missing values in development set

Feature	# unique values	# missing values
country	48	3
designation	27,800	25,944
province	444	3
region 1	1,206	13,889
region 2	18	50,734
variety	603	0
winery	14,105	0

Intuitively, we can notice that all the wines produced by a single winery tend to range in a relatively small quality band, while wines produced in a given country have a more evenly distributed quality. In other words, the quality of wines produced by each winery has a small variance, a the interquartile ranges of two different wineries are often non-overlapped. Conversely, the wine quality of each country has an higher variance. This trend is clearly visible on the entire data set and not only on the top-5 frequent attribute values.

This means that the *winery* field might be an effective predictor of our target variable. We can observe a similar behavior also for other features (*region 1, region 2, variety* and *designation*). In section IV we will show how this intuition turned out to be very accurate using the feature importance metrics produced by our model. In section II we propose a possible approach to efficiently handle these high-cardinality features, and extract useful information even in presence of missing values.

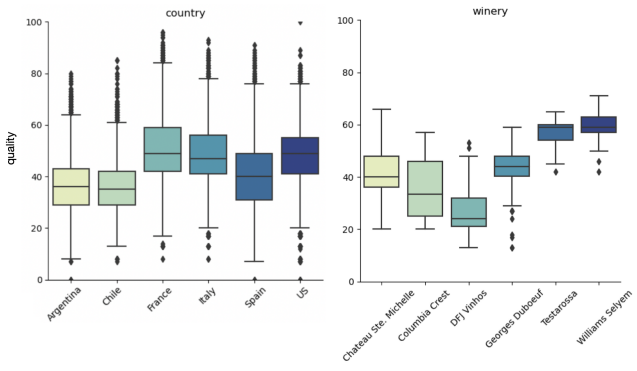

Fig. 1. Wine quality distribution in top 5 frequent countries and wineries

II. Proposed approach

In order to retain all the information contained in the data set without using thousands of dummy binary variables, we decide to encode each tuple (*country, province, region 1, region 2, winery*) with the corresponding latitude and longitude. To retrieve the coordinates we can make use of the APIs provided by services such as OpenStreetMap or Google Maps. We send a request to the API using the string obtained by concatenating all the elements of each tuple. Note that this solution can easily handle missing values. If one of the fields (e.g. region 2) is not available in the object, we simply do not include it in the request. We notice that OpenStreetMap does not return any coordinates for 30% of our requests. This is because some provinces in the data set do not actually correspond to real administrative regions (e.g. 'Central Italy' or 'Sicily & Sardinia'). On the other hand, Google Maps returns the coordinates even when the request is not completely clear, by providing the result with maximum likelihood. Visually inspecting the result, we can notice that in roughly 5% of the requests, Google Maps returned a completely wrong answer (e.g. some wineries in France or Italy are encoded with coordinates corresponding to the US soil, and vice-versa). Even if this adds noise to our data set, we decide to keep using this solution.

An alternative way to not discard any information while avoiding a high-dimensional representation, is to use a tree-based model that can make use of categorical data. Regression trees can deal efficiently with mixed predictors (qualitative and quantitative) and elegantly handle missing data through surrogate splits. However, their performance are often poor with complex data sets, they are subject to high variance and are naturally inclined toward overfitting [1]. To avoid these drawbacks we can use a tree-based ensemble model that averages the prediction of multiple shallow trees. In the following steps we will discuss the preprocessing steps and the results obtained with a standard random forest (used as base-line reference) and a gradient boosting regressor.

A. Preprocessing

We first focus on the data preparation steps needed for **textual** data. Each review is first transformed with a common preprocessing pipeline involving lower casing, removal of punctuations/stop-words, stemming and tokenization. After these standard steps, we compose a word presence binary matrix that simply indicates whether a word is present or not inside a review. Even if this approach may seem naïve, there are many empirical results showing its effectiveness for traditional sentiment analysis solutions [2]. With this method we have transformed the original set of reviews into a dataset with roughly 28,903 binary features (one for each unique word in the collection of documents). To reduce the dimensionality we decide to keep only the 1000 words most positively or negatively correlated with the target, according to the **word score correlation metric** described in [3]:

$$corr(t, D) = \frac{1}{|D|} \sum_{d \in D} \left[I(t, d) \cdot \left(q(d) - \frac{1}{|D|} \sum_{d' \in D} q(d') \right) \right]$$

where $q(d)$ is the *quality* score associated with review d, D is the collection of documents, and $I(t, d)$ is an indicator function whose value is 1 if d contains the term t and -1 otherwise. If $corr(t, D) > 0$, term t tend to appear more frequently in documents with above average quality, while if $corr(t, D) < 0$, term t is associated with below average quality reviews. Note that $\frac{1}{|D|} \sum_{d' \in D} q(d')$ is the average quality score. This metric implicitly penalizes words that occur both too rarely or too frequently to be useful for training.

TABLE II
Top 10 positively and negatively correlated words

Positive		Negative	
1.80	tannis	-1.00	flavors
1.45	years	-0.95	aromas
1.19	rich	-0.82	finish
1.18	wine	-0.80	citrus
1.12	black	-0.78	palate
0.87	oak	-0.78	fresh
0.78	ripe	-0.72	clean
0.77	cherry	-0.72	apple
0.75	drink	-0.63	nose
0.70	cabernet	-0.62	light

Table II shows the top 10 positively and negatively correlated terms over the entire collection of reviews.

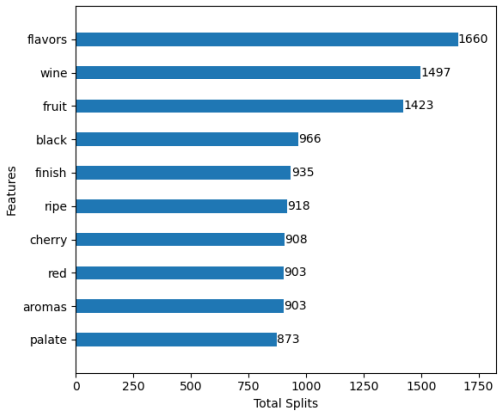

Fig. 2. Word importance according to total number of splits

To verify the effectiveness of this words selection method, we can train an unoptimized version of the the model (described in paragraph B) using the word presence matrix as features, and inspect the resulting feature importance. Figure 2 shows that words with a strong correlation score (close to 1 or -1) listed in table II are likely to produce a high number of splits, and consequently a high overall gain.

The preprocessing of **categorical** features for tree-based methods do not usually requires complex steps. The selected implementation of the gradient boosting regressor only needs

each category to be encoded with a different integer (-1 for missing values). This step do not increase the dimensionality of the dataset. The random forest regressor, requires additional steps, since each categorical feature has to be one-hot encoded. We then estimate the singular value decomposition with a truncated SVD (a method that can efficiently process large and sparse matrices and return an approximated SVD). We decide to keep the first 5300 components that explains more than 80% of the variance.

Numerical features (i.e. coordinates) can be used without any further preprocessing for both methods.

B. Model selection

We will now discuss more in depth the two models proposed for the regression task:

- **Gradient boosting machines** (GBM) are tree-based ensemble models that can be used both for classification and regression tasks. Each base model is trained on a bootstrapped version of the original data set, and produces splits only on a fraction of the original features. The main difference between GBMs and other tree-based methods is that the base models are not i.i.d.. Trees are grown sequentially, using information from previous iteration. Each base model is fit on a **reweighted** version of the training data set. Using this weighting (i.e. learning rare or shrinkage factor) we fit a tree using the current residuals, rather than the target variable [1]. In other words, at each iteration GBMs try to improve the performance on region of the dataset in which previous models are deficient.
- **Random Forests** are a tree-based ensemble method as well. Anyway, in this case the weak trees can be grown in parallel because the model do not learn from previous errors. Instead it relies entirely on the assumption of mutual decorrelation between base model.

Among the available libraries providing an implementation of gradient boosting regressors, **LightGBM** [4] proved to be very effective for general sentiment analysis on short texts [5]. Moreover, LightGBM performs extremely well with integer-encoded **categorical** features, applying Fisher (1958) [6] to find the optimal split over categories. Moreover, performance are not affected by **missing** values.

C. Hyperparameters tuning

For tuning our models we first need to split the development data set. We decide to keep 80% of the observations in the training set, and use the remaining 20% for the test set. For the training of the gradient boosting regressor we also need a small validation set (extracted from the training set), on which at each iteration we evaluate the performance of the model.

To optimized our baseline model (random forest) we have performed a simple grid search considering different combinations of parameters. The search space is summarized in table III.

The main drawback of GBMs is the very large number of hyperparameters to be tuned. Approaching this problem with a standard grid search is computationally prohibitive. **Optuna** [7] is an hyperparameter optimization framework that efficiently search large spaces and prune unpromising trials with a stochastic approach. We first run a tuning round with a K-fold cross validation strategy (K=3). The tuner returns a dictionary containing the initial suggested parameters. We then iterate the optimization process, each time selecting a narrower set of parameters to be optimized. The hyperparameter tuning of the GBM is, by far, the most computationally expensive step in the entire process.

We evaluate each configuration on the test set using the R^2 score, both for the RF and the GBM.

TABLE III
GRID SEARCH PARAMETERS FOR THE RANDOM FOREST

Parameters	Values
n_estimators	500, 750, 1000
max_features	sqrt, log2
criterion	mse, mae
min_sample_leaf	20, 40, 60

TABLE IV
HYPERPARAMETERS SELECTED FOR LIGHTGBM REGRESSOR

Parameter	Value	Parameter	Value
boosting_type	gbdt	bagging_freq	5
learning_rate	0.1	n_estimators	3000
metric	poisson	min_child_samples	20
num_leaves	255	lambda_l1	9.455
feature_fraction	0.42	lambda_l2	$1.158 \cdot 10^{-8}$
bagging_fraction	1.00	feature_pre_filter	False

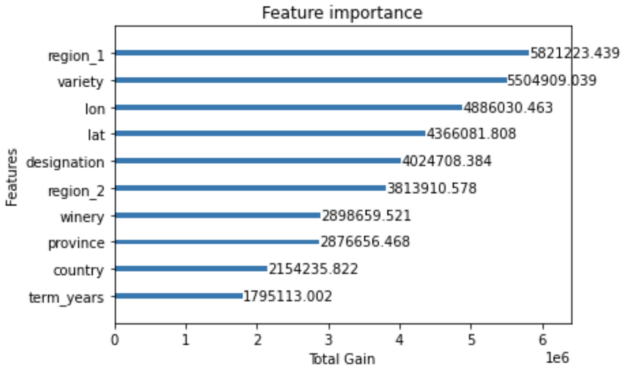

Fig. 3. GBM Feature Importance - Total Gain

III. RESULTS

We will now compare the results obtained with the two model, using the hyperparameter discussed in the previous section

- The best combination of parameters found for the **random forest** is {n_estimators=1000, max_features=sqrt, criterion=mse, min_sample_leaf = 40}. With this configuration we obtain $R^2 = 0.673$ on the test set, and

$R^2 = 0.645$ on the public evaluation set. We might consider to perform a deeper search to further optimize the this model. Anyway, since we are considering it only as a baseline, the score obtained is more than satisfactory.

- The configuration producing the best result for the **gradient boosting machine** is shown in table IV. With this set of hyperparameters we obtain $R^2 = 0.871$ on the test set, and $R^2 = 0.866$ on the public evaluation set.

IV. Discussion

The results obtained show that the gradient boosting machine outperforms the random forest approach, even if it requires a considerable effort in the hyperparameters tuning step. The GBM's capability of dealing with categorical data without any encoding, and effectively manage missing data, was crucial for the quality of our result.

Figures 3 and 4 show the feature importance for the GBM regressor. This result confirms our initial assumptions. Even if the attributes *region 1* and *designation* have many missing values, they carry a relevant amount of information, hence they produce a substantial fraction of the total splits. Discarding these features from the analysis would dramatically reduce the capacity of our model. Also the attribute *winery*, extensively discussed in section I, has major importance in the regression process.

We notice that the additional information on the coordinates retrieved with the Google API, are frequently used to produce splits in the weak trees composing the ensemble.

We have also confirmed that the correlation-based word selection method, introduced in [3], can be effectively exploited in similar use cases to reduce the dimensionality by pruning words that are irrelevant for the analysis.

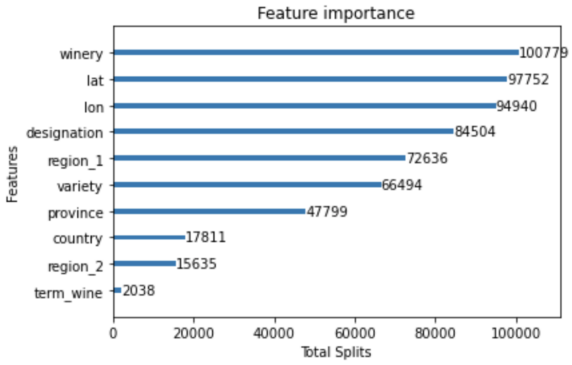

Fig. 4. GBM Feature Importance - Total Splits

References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer New York, 2009.

[2] C. C. Aggarwal, *Machine Learning for Text*. Springer Publishing Company, Incorporated, 1st ed., 2018.

[3] A. Drake, E. Ringger, and D. Ventura, "Sentiment regression: Using real-valued scores to summarize overall document sentiment," in *2008 IEEE International Conference on Semantic Computing*, pp. 152–157, 2008.

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 3146–3154, Curran Associates, Inc., 2017.

[5] F. Alzamzami, M. Hoda, and A. E. Saddik, "Light gradient boosting machine for general sentiment classification on short texts: A comparative evaluation," *IEEE Access*, vol. 8, pp. 101840–101858, 2020.

[6] W. D. Fisher, "On grouping for maximum homogeneity," *Journal of the American Statistical Association*, vol. 53, no. 284, pp. 789–798, 1958.

[7] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.