Feature Engineering vs Feature Selection vs Hyperparameter Optimization in the Spotify Song Popularity Dataset

Alan Cueva Mora
School of Computer Science
Technological University Dublin
Dublin, Ireland
e-mail: d20125565@mytudublin.ie

Brendan Tierney
School of Computer Science
Technological University Dublin
Dublin, Ireland
e-mail: brendan.tierney@tudublin.ie

***Abstract*—Research in Featuring Engineering has been part of the data pre-processing phase of machine learning projects for many years, but we sometimes forget its importance. It can be challenging for new people working with machine learning to understand its importance along with various approaches to find an optimized model. This work uses the Spotify Song Popularity dataset to compare and evaluate Feature Engineering, Feature Selection and Hyperparameter Optimization. The result of this work will demonstrate that Feature Engineering has a greater effect on model efficiency when compared to the alternative approaches.**

Keywords-feature engineering; language identification; feature selection; hyperparameter optimization; cross-validation.

I. INTRODUCTION

Feature selection and hyperparameter optimization are two sophisticated machine learning techniques with a strong research background. For an early-stage data scientist, it is very easy to think that they are the best alternatives for a machine learning task.

Feature engineering is an important, but labour-intensive take on machine learning [1]. Most machine learning performance is heavily dependent on the representation of the feature vector. As a result, much of the actual effort in deploying machine learning algorithms goes into the design of pre-processing pipelines, data transformations, domain and metadata knowledge [1].

Kaggle competitions and the Knowledge Discovery and DataMining (KDD) Cup have seen feature engineering play a very important part in several winning submissions [2]. Additionally, the Kaggle Algorithmic Trading Challenge was won with an ensemble of models and feature engineering. The features engineered for these competitions were created manually by the data scientist, utilizing their domain knowledge.

This paper is structured as follows. Section 2 shows a brief exploration of related work. Sections 3, 4 and 5 step through using feature engineering, feature selection and hyperparameter optimization of a regression machine learning task over the Spotify Song Popularity dataset [3]. The influence of each step over the machine learning task is measured using a Cross-Validation (CV) [4] and the Root Mean Square Error (RMSE) loss function. Finally, Section 6 presents the conclusions and future work.

II. RELATED RESEARCH

There are some common research topics in machine learning literature. One of them is about comparing different machine learning methods to solve specific tasks [5] [6] and identify the best scenario for a method.

Another common machine learning research topic is to compare similar techniques, as is the case of feature selection and feature extraction [7]. The objective of both methods is to reduce the feature space to improve data analysis. Feature selection performs the reduction by selecting a subset of features without transforming them, while feature extraction reduces dimensionality by computing a transformation of the original features to create other features that should be more significant.

In featuring engineering research, it is common to find comparisons between combinations of different engineered features and methods to identify which methods generally benefit from the same set of engineered features [8].

Considering that in every machine learning task the objective is to reduce the error, there is no reason not to compare completely different techniques, such as those proposed in this work.

III. FEATURE ENGINEERING

Feature engineering involves calculating new features, based on the values of the other features, and it is primarily a manual, time consuming task [8].

The Spotify Song Popularity dataset [3] consists of 129 thousand rows and 17 independent variables of which three are strings and cannot be used in the machine learning task. For this type of data, feature engineering focuses on generating numerical variables from these.

Every new variable was evaluated with a correlation (spearman) test to validate its relationship with the target and its criterion could be very weak (0.0-0.19), weak (0.2-0.39), moderate (0.4-0.59), strong (0.6-0.79), and very strong (0.8-1.0). Some variables could be generated in different ways, or their values were similar to those of another variable. In these cases, the correlation test was used to compare all variants and the best one was used. All statistics generated for this step were statistically significant (p-value < 0.05).

 ISBN: 978-1-61208-891-4

A. Numerical Release Date

This variable is the numerical representation of the 'release_date' variable (YYYY-MM-DD or YYYY if the data is incomplete). The integer part is the year and the float part is the elapsed percentage of the year:

$$year\_rd = year(release\_date) + (day\_of\_year(release\_date)/365)$$

This new variable is similar to the variable 'year'; both variables are moderately correlated with the target variable, but 'year' (r=0.5386) is a better option than 'year_rd' (r=0.5391). This is why this variable was discarded.

B. Number of Artists

This variable was created from the 'artists' variable which is a string formatted as a Python list. Each value was transformed to a list object, and its length is the value for this new variable. Its test shows a very weak negative correlation (r=-0.1968). This fits in with an observed pattern where songs with many artists tend to be unpopular

C. Artists' Mean Popularity Value

This variable was calculated from the 'artists' and 'popularity' variables. First, a dictionary of artist's popularity was created. Each song's popularity is used to calculate the artists' mean popularity value.

There is a risk here. When evaluating a new song and one artist is not present in the dictionary, his/her popularity is zero. Considering the influence of this variable on the final result, an imputation should be made to avoid overfitting. The imputation used is the mean value of the artist's popularity. Its test shows a strong correlation (r=0.911).

D. Name Length

The length of the 'name' of the song produces a weak negative correlation (r=-0.2941). This fits with the observed pattern where songs with long names tend to be unpopular.

E. Name Language

Worldwide, English songs are more popular than other languages and recently thanks to Reggaeton, Spanish songs are popular too, but this information is not available in the dataset. One easy way to get the language is to detect the language in the title (name) of the song.

There are some libraries available in Python to detect language, some of them based on neural networks. For this new variable, five libraries were taken into consideration.

- LangDetect [9] is a direct port of Google's language-detection library from Java to Python.
- TextBlob [10] uses Google Translate API for language detection. It requires internet connection.
- FastText [11] is a text classifier that works with pretrained models.
- LangId [12] works with transductive learning and transfer learning techniques.
- CLD3 [13] uses a trained neural network model.

Five language detection tasks were performed using each library. LangDetect identified 46 languages, TextBlob 88, FastText 120, LangId 79 and CLD3 97, so there were five new high cardinality variables.

Some encoders were taken into consideration to evaluate the best way to represent these new variables.

- Label Encoder encodes a categorical variable with value between 0 and n_categories-1.
- Target Encoder [14] replaces features with a blend of the expected value of the target given a particular categorical value and the expected value of the target over all the training data.
- Leave One Out Encoder [14] is similar to target encoder, but excludes the current row's target when calculating the mean target for a level to reduce the effect of outliers.
- Min Hash Function [15] is inspired by the document indexation literature, and in particular the idea of Locality-Sensitive Hashing (LSH).

A correlation test was performed in every combination of library-encoder to identify the best one (see Figure 1). The results show that Text Blob with Target Encoder is the best option.

Between the libraries for language detection, TextBlob gives the best results. When performing a manual inspection of the results, it was evident other libraries confused Spanish and Italian, and this perception was supported by its statistics where TextBlob was the best, regardless of the encoder.

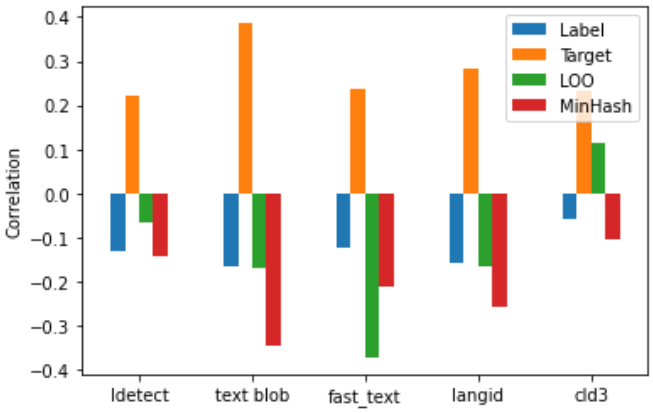

Figure 1. Libraries and Encoders Correlation Comparison

Finally, a Cross-Validation [4] (k=5) shows the result of the feature engineering process was a RMSE reduction from 17.1624, using only the original non-string features, to 8.9846 including the new variables.

IV. FEATURE SELECTION

In machine learning, feature selection entails selecting a subset of the available features in a dataset to use for model development. Among its advantages are generating better models and reducing computations cost [16]. The techniques considered in this section are Least Absolute Shrinkage and Selection Operator (LASSO) and Sequential Forward Selection (SFS).

First, an SFS task was executed using all features to detect any negative performance contribution to the model. Figure 2

 ISBN: 978-1-61208-891-4

shows there are no clear negative contributions to the model, but approximately 5 features seem to have a neutral contribution to the performance.

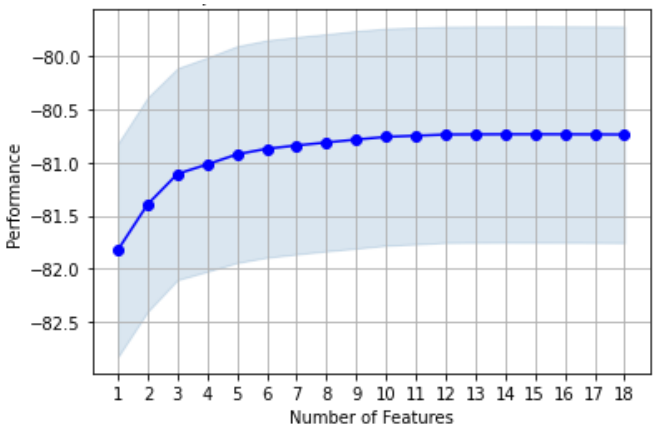

Figure 2. Sequential Forward Selection using all features

In order to get the number of features to consider based on evidence, a LASSO task was run using different regularization parameter values. The recommended values are 0.1, 0.01 and 0.001 and the one that includes more variables (regularization=0.01) reduces the number of features from 18 to 14, excluding 'danceability', 'energy', 'liveness' and 'speechiness'.

Considering the number of features proposed by LASSO, an SFS task was run using the same number, and it results with a different subset of features. The SFS task excludes variables 'mode', 'key', 'explicit' and 'danceability'. Only 'danceability' was excluded by both processes.

In order to choose the best subset, three regression tasks were run using CV (k=5). The results were RMSE=8.9846 using all non-string features, RMSE=8.9845 using the SFS subset and RMSE=8.9984 using the LASSO subset with SFS being the best subset.

The improvement is insignificant, so the advantage is to reduce the computational cost and omit features that do not contribute to the performance. It is important to mention that no new features were excluded by any methods.

V. HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization consists of testing a set of hyperparameters of a model and identifying the optimal values for them. In this section, five methods were taken into consideration. They can be divided in two groups: linear (1 and 2) and tree methods (3, 4 and 5).

1) Linear Regression (LR): creates a linear relationship between features and target.
2) Ridge Regression (RR): is a variant of LR where the loss function is the linear least squares.
3) Decision Tree Regressor (DTR): is the regression version of the decision tree method.
4) Extra Tree Regressor (ETR): is similar to DTR, but this method changes the way of splitting the nodes.
5) Random Forest Regressor (RFR): is an ensemble of a multitude of decision trees. It uses averaging to improve accuracy and control overfitting.

The hyperparameter optimization task was performed using a CV grid search (K=3). Unfortunately, there are no hyperparameters for the Linear Regression, for the Ridge Regression there is one, the regularization strength, but this only improves the RMSE by 0.000004, so this step focused on the tree methods.

In the tree methods, one parameter directly influences the results. This parameter is the max depth parameter, which specifies how many levels of nodes the tree could have. When this parameter is set to none, the tree will expand the nodes until all leaves are pure or until all leaves contain less than two samples.

Limiting the tree was clearly a good option, not only because the train for the entire tree takes too much time, but the results are better. After an exhaustive evaluation, the best values of max depth were 11 for DTR and 15 for ETR and RFR. Another parameter was the criterion which measures the quality of a split where the only options that worked were mean square error and mean squared error with Friedman's improvement score for potential splits, but the results prove that this parameter does not affect the metrics.

In the specific case of DTR and ETR, there was an option to add Bagging Regression (BR). The BR is an optimization to improve the stability and accuracy of the method. The Bagging splits the data and uses it in different decision trees and ensembles the result. In both cases the BR was the best option.

In order to compare the RMSE metrics with the previous section of this work, five CV tasks (K=5) were run using the best parameters for each method. Figure 3 shows that the RFR model gets the best metrics.

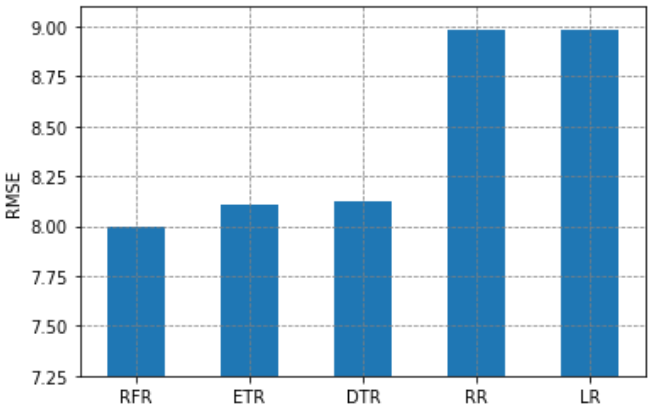

Figure 3. Model Comparison

The result of the hyperparameter optimization and the model comparison was a RMSE reduction from 8.98 to 7.99. Although the grid search task is automatic, it takes a lot of execution time, which requires monitoring because it can easily crash when computer resources are depleted.

 ISBN: 978-1-61208-891-4

VI. CONCLUSIONS AND FUTURE WORK

The analysis performed over the Spotify Song Popularity dataset involves feature engineering, feature selection, hyperparameter optimization and model selection using CV to validate each step.

Feature engineering was by far the technique that generated the best reduction of the RMSE metric. Figure 4 shows how this technique reduced the error to almost half, while the improvements produced by Feature Selection and Hyperparameter Optimization/Model Selection was not significant.

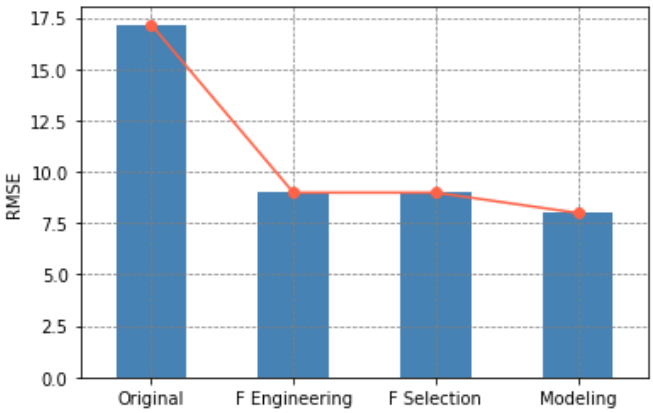

Figure 4. Sections Improvement Comparison

For this dataset, it can be concluded that a well performed feature engineering task has a greater impact on the model performance than more sophisticated machine learning techniques. Even when each step takes approximately the same time and resources, its value is not the same.

This experiment focused on using one particular dataset. Future work will look to expand to include more datasets from a variety of domains. This will be done to evaluate the effect of these tasks and to see if similar outcomes can be achieved.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] W. Zhou, T. D. Roy, and I. Skrypnyk, "The KDD Cup 2019 Report," SIGKDD Explor. Newsl. 22, Jun 2020, pp. 8-17, doi: 10.1145/3400051.3400056

[3] Kaggle.com, Spotify Popularity Prediction. Retrieved: Aug, 2021. [Online]. Available from: https://www.kaggle.com/c/spotify-popularity-prediction/ 2021.03.22

[4] D. Berrar, “Cross-Validation,” Encyclopedia of Bioinformatics and Computational Biology, Vol 1, pp. 542–545, Elsevier, 2019, doi: 10.1016/B978-0-12-809633-8.20349-X

[5] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, “A comparison of machine learning techniques for phishing detection,” Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit on - eCrime ’07, 2007, pp 60-69, doi: 10.1145/1299015.1299021.

[6] S. Pouriyeh et al., “A comprehensive investigation and comparison of Machine learning Techniques in the domain of heart disease,” 2017 IEEE Symposium on Computers and Communications (ISCC), Jul 2017, pp. 204-207, doi: 10.1109/iscc.2017.8024530.

[7] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," 2014 Science and Information Conference, 2014, pp. 372-378, doi: 10.1109/SAI.2014.6918213.

[8] J. Heaton, "An empirical analysis of feature engineering for predictive modeling", SoutheastCon 2016, 2016, pp. 1-6, doi: 10.1109/SECON.2016.7506650

[9] Pypi.org. langdetect. Retrieved: Aug, 2021. [Online]. Available from: https://pypi.org/project/langdetect/

[10] Readthedocs.io. TextBlob: Simplified Text Processing documentation. Retrieved: Aug, 2021. [Online]. Available from: https://textblob.readthedocs.io/en/dev/ 2020

[11] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of Tricks for Efficient Text Classification Retrieved: Aug, 2021. [Online]. Available from: https://arxiv.org/abs/1607.01759 2016

[12] M. Lui and T. Baldwin. “Cross-domain Feature Selection for Language Identification” Proceedings of the Fifth International Joint Conference on Natural Language Processing, pp. 553-561, Nov. 2011. Available from http://www.aclweb.org/anthology/I11-1062

[13] Pypi.org. pycld3. Retrieved: Aug, 2021. [Online]. Available from: https://pypi.org/project/pycld3/

[14] W. D. McGinnis, C. Siu, A. S, and H. Huang. “Category Encoders: a scikit-learn-contrib package of transformers for encoding categorical data” The Journal of Open Source Software, vol 3, pp. 501, Jan. 2018, doi: 10.21105/joss.00501

[15] P. Cerda and G. Varoquaux. Encoding high-cardinality string categorical variables. Retrieved: Aug, 2021. [Online]. Available from: https://arxiv.org/abs/1907.01860 2020

[16] P. Cunningham, B. Kathirgamanathan, and S. J. Delany. Feature Selection Tutorial with Python Examples. Retrieved: Aug, 2021. [Online]. Available from: https://arxiv.org/abs/2106.06437 2021

 ISBN: 978-1-61208-891-4