Technische Universität Berlin
Fakulät IV
Institut für Softwaretechnik und Theoretische Informatik
Fachgebiet Elektronische Mess- und Diagnosetechnik

Praktikum Grundlagen der elektronischen Messtechnik
Betreuer: Felix
SS 2020

Feature Selection for Data-Error Robustness

Project summary

Tomasz Tkaczyk (368998)
Amit Nautiyal (408787)

2022-08-17

Contents

1 Introduction

With the explosive growth of high-dimensional data, feature selection has become a critical step in machine learning(ML) assignments. Though most of the possible works[GE03] focus on devising selection strategies that are efficient in recognizing small subsets of predictive features, new research has also highlighted the relevance of examining the robustness of the selection method concerning sample variation. As most of the state-of-the-art feature selection (FS)[GE03] methods are focused solely on accuracy or correlation with the target variable, there is no well-defined strategy to address the error robustness of the model in the feature selection step. Here, we address this challenge of finding a feature representation that is accurate and robust.

1.1 What is the problem statement?

A data scientist is driven by business needs. For instance, the ML application requires a certain classification accuracy, low inference time, fairness, and robustness against data errors. FS is one important part of the ML pipeline and in fact, by selecting the right features, the data scientist can ensure to achieve these ML application constraints. Unfortunately one of those metrics, namely error robustness, is not trivial to measure and thus optimize.

2 Solution Overview

Formally, for a given dataset $D = \{t_1, t_2...t_n\}$ with N tuples t_i, $M = \{m_1.., m_j\}$ feature columns, we want to find a feature representation $F \subset M$ to maximize the mean accuracy and minimizes the sensitivity to perturbations caused by data errors. For that, we need to develop a new strategy that takes both objectives into account and efficiently yields the most promising feature representation.

2.1 Big picture of our solution

In our project, we want to address the challenge of finding a feature representation that is both accurate and error robust. In our solution, we want to define a score that enables us to assess the collective sensitivity of the whole feature representation to data errors. This can be a very helpful metric to find an Error Robust representation

3 Approach

Our research we want to design an ensemble approach using combination of both Filter and Wrapper methods [Tal05]to feature selection. This section describes the

tools we used in our experiments and their purpose.

3.1 Data Corruptor

For our analysis, it was crucial to obtain different sets of corrupted data. Thus, we have decided to control the quality of the 'clean' data by artificially generating errors. Based on the taxonomy of data quality errors from[Abe+16], we implemented a 'Data Corruptor' (DC) which is a data loader that is capable of controlled introducing various data quality issues depending on the data type. DC supports both numeric and categorical data. In the first case it is introducing outliers, adding noise, introducing NaNs and switching values between cells. For cardinal data, our Data Loader is randomly replacing cell values with an empty string, randomly changing characters, flipping boolean values, and switching values between cells.

Furthermore, DC enables us to adjust how many data quality errors are introduced at each iteration. One can choose to corrupt the whole dataset (cell-by-cell) or a set of features in a feature-by-feature manner. It allows us to define how many cells in a dataset are going to be corrupted i.e 90% cells in a feature. One can also control the error distribution i.e 40% NaNs, 40% noise perturbations & 10% outliers.

3.2 Prepossessing pipeline & Estimator

In most ML projects the data that one has to work with is unlikely to be in the ideal format that is ready to be used by the ML algorithm. Usually, it is necessary to apply various transformations before the data is ready to be feed into the algorithm. While some transformations like scaling might be optional, others like feature encoding or data imputation are necessary. This makes a preprocessing pipeline an inherent part of the whole ML application.

For our analysis, we have developed a very generic pipeline that can be applied to different datasets and convert them into a desirable format. Firstly, numeric features are being scaled utilizing 'StandardScaler' which standardizes features by removing the mean and scaling to unit variance. Secondly, missing values are imputed, using "SimpleImputer" which replaces the missing and NaN values with the average of the feature column. Similarly, in the case of cardinal data, missing values are being imputed with a constant value - ''Missing'', and then encoded using a one-hot encoding scheme which creates a binary column for each category. To make the pipeline module portable, it was implemented using sklearn's preprocessing module [skl], which enables to conveniently specify a standardized list of sequential transformation for multiple features and pass the results the final estimator.

As the objective is to find a suitable feature selection strategy regardless of a predictive model we have decided to focus on Logistic Regression as it is relatively simple, adjustable, and sensitive to errors.

4 Experiments

In this project, we have conducted multiple experiments that gave us hints and insights about the process of feature selection. This section covers the two most important experiments that lead to the development of our feature selection strategy.

4.1 Data

Before we begin with the experiment, to validate our assumptions and develop the feature selection strategy, it is crucial to decide on the datasets which are going to be used. For the sake of simplicity, we have focused on binary classification and we decided to use open-source datasets that are also well researched by the community. Considering the above-mentioned aspects we have decided to use the Breast Cancer Wisconsin (Diagnostic) Dataset [WSM92] which has 569 rows and 30 numeric features and Airbnb Dataset [Li+19] with over 12000 instances and 40 features(numeric and categorical).

4.2 Experiment 1 - Naive Feature Ranking

In the first experiment, we have measured the impact of data errors on the accuracy of the model. Figure 1 is depicting the experiment setup. On the left-hand side of the diagram, one can see a clean dataset that is being split into test and train sets(in the ratio 7 to 3). The train set is being processed by the preprocessing pipeline (described in 3.2) and used to train the predictive model. In parallel, the test set is being passed to the DataCorruptor (Section3.1) and preprocessed. The result of data corruption is one 'untouched' clean test set and M test sets, where M is equal to the number of features in the data. In every test set, a different feature column was corrupted. To amplify the results of the experiment, we have decided to corrupt around 95% (to amplify the results) of the cells of the column.

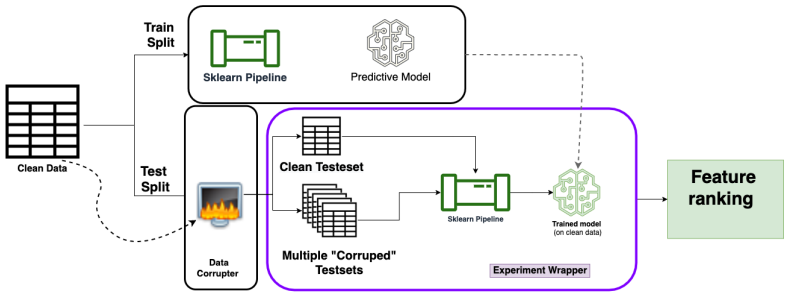


Figure 1: Experiment setup to generate 'naive' feature ranking based on sensitivity to corruption.

After the corruption phase, the 'baseline' accuracy is being calculated using an esti-

mator trained on clean train data, and the 'clean' test set. Then, every corrupted test set is being used to measure the accuracy of the model. The drop inaccuracy with respect to the baseline score is was collected after every run.

Results

The first experiment yields very promising results and proved our assumption that feature columns have various sensitivity for it comes to data quality issues. The results of the experiment, depicted in Figure 2 enabled us to create a ranking of features with respect to their sensitivity. The feature columns of the left-hand side of the diagram have proved to be immune to the data errors as the corruption of these did not influence the quality of the prediction. On the other hand in on the right side of the diagram, one can see that in most extreme cases, the corruption of just one column has contributed to a drastic (over 52%) loss of accuracy for the model.

These results show us that sensitivity for data quality error is one important characteristic of the feature that has to be taken in to account while doing feature selection. Furthermore, the above-described method proved to be very quick and relatively computationally inexpensive.

Nevertheless, this method had also some drawbacks. First of all, it works under a quite unrealistic assumption that most of the cells in a feature column are going to be corrupted. Secondly, using solely this method as an indicator for feature selection might be insufficient as it takes only the 'error robustness' into consideration. Lastly, we have also realized that the distribution of the introduced errors has a considerable influence on the order of feature ranking. Therefore, we have decided to focus on the NaN (for numeric data) and missing values (for categorical data) in future experiments.

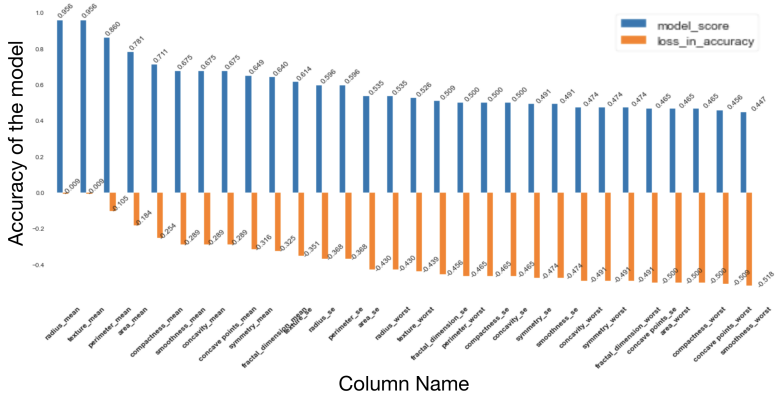
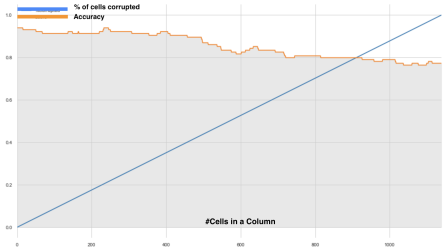


Figure 2: Naive ranking results for the Cancer dataset. Corruption of 95% cells using NaNs and missing values.

```
Algorithm:
Input: Estimator E, Data X
Output: Error AUC Score

1. Split Data into test & train.
2. Fit Estimator E with train
3. For i=0 and i < len(test):
    a. Corrupt one random cell of test
    b. Calculate & save accuracy of the model E
4. Measure Error AUC
```

(a) Pseudo Code of the algorithm



(b) Error AUC score plotted for every corruption with one column.

Figure 3

4.3 Experiment 2 - Error AUC Score

The formerly described method enables to create a meaningful ranking of features, however, the assumption that a column would be almost entirely corrupted is not only far-fetched but also a clear a very strong indicator that this feature column might not be usable in the production system.

To address these issues, we have designed another experiment. This time we measured the performance drop in a scenario where the data step-wise corrupted.

The algorithm we have used in this experiment is documented in Figure (a). While this experiment shares a lot of similarities with the predecessor, there are two main differences.

The first difference is the way of corrupting data which in this case occurs stepwise. In every iteration, one random cell of the dataset is being corrupted and passed to the predictive model to measure the performance drop. Because of that, one can obtain a score for every possible corruption percentage.

Secondly, the final score is not calculated based on just one measurement, but instead on a series of measurements done after each corruption. Figure 3 (b) shows the an example of step-wise degradation of performance(orange line) correlated with the increase of corrupted cells(blue line). To quantify this degradation, we decided to use trapezoid rule [Var16] which is an integral approximation method used in for discrete data. This score will be further referenced in this paper as 'Error AUC score'.

Results

As a result of this experiment, we have defined a score that enables us to assess the collective sensitivity of the whole feature representation to data errors. This can be a very helpful metric that helps to find an Error Robust representation.

However, the two most important drawbacks of this method are the lack of repeatably and vast computational cost. Considering that the cells are picked completely at random, this yields a different curve every time an experiment is repeated with a

standard deviation of around +/- 15%. We have addressed this problem using Cross-Validation and repeated runs to stabilize the end result what also rendered the method even more computationally expensive. Based on the Big O notation it is O(k*N*M) where k is number of Cross-validation Folds.

4.4 Solution - Feature Ranking & ErrorAUC

To validate the usability of this method, we have repeated this experiment for four different feature representations and the results are visible in Figure 4. We have used 'top 10' and 'top 5' features based on the ranking of the features from the previous experiment and compared it with the top 10 features selected using Recursive Feature Elimination and full feature representation. We made the decision to use 10 best features(optimal) based on the prior EDA research done by [Bha19] which showed that the data is highly correlated and the fact that it was second lowest number of features providing highest possible accuracy. This result shows us that the ranking used in combination with this scoring method enabled us to find a feature representation which is considerably less sensitive to low-quality data than i.e one selected by RFE.

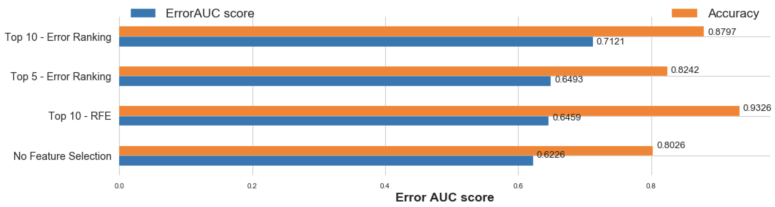


Figure 4: Error AUC Score computed for four different feature representations. RFE stands for Recursive Feature Selection

While RFE representation was superior when it comes to accuracy, 'Top10 Feature Ranking' was only marginally worst. Interestingly both RFE and 'Top10 Feature Ranking' representations share 6 features with each other.

5 Related Work

Some notable work has been done related to FS and robustness.We see work done on detecting data errors [Xu +16] which discuss robustness to capture most errors in real-world data sets. Also, we see various methods of feature selections[Tal05].In [Wur+19], we see a combined feature selection approach to attractively learn a robust well-fitted statistical model and rule out irrelevant features. But, this fails to address the multivariate robust setting.Huanjing.W [WKW11], investigated the robustness of six commonly used feature selection techniques on 16 datasets from three real-world software projects. And their results show that making smaller changes to the datasets has less impact on the stability of feature ranking techniques applied to those datasets.

6 Conclusion

Two main contributions of our research are, development effective feature ranking strategy and design and evaluation of an Error Robustness score. We have shown that using these two methods combined with classical model performance evaluation can yield a more robust feature representation that is also robust.

Along the way, we have made several observations that enabled us to acquire a deep understanding of feature selection. Firstly, we realized the importance of this step in the ML lifecycle and understood that it's not a trivial task, especially when the accuracy is not the only objective. We have observed that current state-of-the-art methodologies are mainly focused either on optimizing prediction accuracy which highlights a need for methods and algorithms which can incorporate Error Robustness as a second objective. One of the key assumptions in the field of ML requires having similar distributions of the data in the train and production setting. This applies not only to the similar distribution of classes but also to data errors as patterns in data perturbations can be considered as features by itself. For instance, missingness pattern in blood pressure measurement might be conditional on another variable, like the age of patients.[Sci] Thus, our observation is that the preprocessing pipeline, especially the imputation strategy is an inherent part of the ML system and it has to be extensively tested using low-quality data in order to preserve this information.

References

[Abe+16] Ziawasch Abedjan et al. "Detecting Data Errors: Where Are We and What Needs to Be Done?" In: *Proc. VLDB Endow.* 9.12 (Aug. 2016), pp. 993–1004. ISSN: 2150-8097. DOI: 10.14778/2994509.2994518. URL: https://doi.org/10.14778/2994509.2994518.

[Bha19] Saptashwa Bhattacharyya. *Understanding PCA (Principal Component Analysis) with Python*. July 2019. URL: https://bit.ly/3hWPoOZ.

[GE03] I. Guyon and A. Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* (2003). ISSN: pp.1157-1182.

[Li+19] Peng Li et al. "CleanML: A Benchmark for Joint Data Cleaning and Machine Learning [Experiments and Analysis]". In: *CoRR* abs/1904.09483 (2019). arXiv: 1904.09483. URL: http://arxiv.org/abs/1904.09483.

[Sci] ODSC - Open Data Science. *Missing Data in Supervised Machine Learning | Medium*. https://medium.com/@ODSC/missing-data-in-supervised-machine-learning-b6df0f02a731. (Accessed on 07/28/2020).

[skl] sklearn. *sklearn.pipeline.Pipeline — scikit-learn 0.23.1 documentation*. https://bit.ly/39DYtZU. (Accessed on 07/25/2020).

[Tal05] Luis Talavera. "An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering". In: *Advances in Intelligent Data Analysis* 6 (2005). ISSN: pp 440-451. URL: https://bit.ly/3fhAv7U.

[Var16] Various. *Trapezoidal rule - Wikipedia*. https://en.wikipedia.org/wiki/Trapezoidal_rule. (Accessed on 07/27/2020). Dec. 2016.

[WKW11] Huanjing Wang, Taghi M. Khoshgoftaar, and Randall Wald. "Measuring robustness of Feature Selection techniques on software engineering datasets". In: *IEEE International Conference on Information Reuse Integration* (2011). ISSN: ISBN: 978-989-758-377-3. DOI: 10.1109/IRI.2011.6009565. URL: https://ieeexplore.ieee.org/abstract/document/6009565/authors#authors.

[WSM92] William H Wolberg, W Nick Street, and Olvi L Mangasarian. "Breast cancer Wisconsin (diagnostic) data set". In: *UCI Machine Learning Repository [http://archive. ics. uci. edu/ml/]* (1992).

[Wur+19] Alexander Wurl et al. "Exploring Robustness in a Combined Feature Selection Approach". In: *8th International Conference on Data Science, Technology and Applications* (2019). ISSN: ISBN: 978-989-758-377-3. DOI: 10.5220/0007924400840091. URL: https://pdfs.semanticscholar.org/7fe9/49ea417c52213a20cadf07d0f8ccc8a5a4c5.pdf.

[Xu +16] Ziawasch Abedjan ans Xu Chu' et al. "Detecting Data Errors: Where are we and what needs to be done?" In: *vldb* vol.9 (2016). URL: http://www.vldb.org/pvldb/vol9/p993-abedjan.pdf.