FRI Data Science Project Competition 2023
Final report

Feature selection on large and sparse datasets

Leon Hvastja, Amadej Pavšič

Abstract

This project centers on feature selection in large datasets, specifically within Outbrain's domain of operations. We engaged various selection methods and established an evaluation pipeline for efficient testing across multiple models, sampling regimes and other parameters. Our performance assessments were founded on a tailored metric we developed from the Jaccard index.
Two base truth rankings, one derived from a simultaneous ranking of all features and the other from a forward feature selection process, were provided by the company. Our objective was to align our methods with these reference rankings. Random forests emerged as a superior approach, balancing accuracy and computation time. We developed a novel technique, using the Borda count voting system on rankings from random data subsamples to determine feature importance. This method enabled us to use extremely small samples of the data to achieve greater accuracy than rankings derived from the entire dataset.
Using this technique, we observed an increase in accuracy with larger subsamples, the smallest subsamples recording the most significant gain. This approach also led to markedly lower variance in prediction accuracy, strengthening our evaluation confidence.
Our method offers a more time-effective solution while simultaneously being more accurate with lower variance. These findings suggest a new method that utilizes small data samples to yield high accuracy and low variance in feature selection, enabling Outbrain to conduct more reliable feature evaluations swiftly, while using less data, thereby enhancing their existing strategies.

Keywords
feature selection, feature ranking, machine learning, big data, sparse features

Advisors: Jure Demšar, Blaž Mramor, Blaž Škrlj

Introduction

Feature selection is the process of reducing the number of input variables when building a predictive model. The aim is to reduce computational cost and, in some cases, improve model accuracy.

Our project aimed to explore feature selection methods for Outbrain, a web recommendation platform. They provided us with data from their regular operations, which was typically large, sparse, and required time-efficient use. The dataset underwent minimal preprocessing specific to our project. We were also provided with their internally generated feature scores to use as a reference.

We hypothesized that simpler models could produce quick feature selection results, but models that accounted for feature interactions would be more accurate at the expense of longer computations.

The main goal of our project was thus the development of an efficient and intelligent feature selection approach given the specifics and constraints of Outbrain's domain of work.

Methodology

Rank and Evaluation Pipeline

We developed a framework for evaluating features that streamlines our algorithm testing process. This pipeline handles data reading, pre-processing, feature score calculations, and evaluations against our two ground truth rankings. Its flowchart is shown in Figure 1.

To reduce the computational load, we performed subsampling by rows. This means selecting a random subset of data points to work with. The aim is to use this approach to gain insight into the behavior of our feature scoring algorithms datasets of varying sizes, and possibly identify an effective subsampling strategy for very large datasets.

We also perform factorization as a preprocessing step.

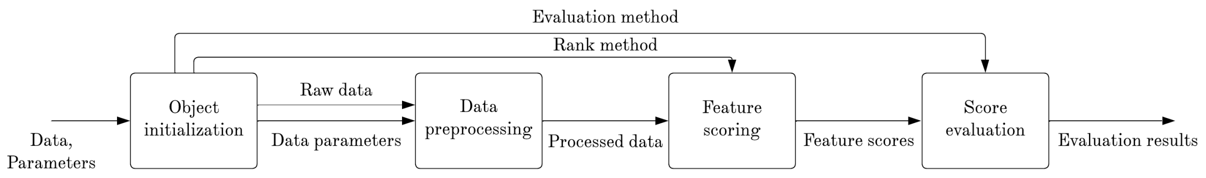

Figure 1. A visualization of our feature selection pipeline.
This figure visualizes the steps required for feature selection in our framework.

This involves transforming the original dataset's categories, which are represented by large integers, into a categorical form and re-encoding them as integers from 0 to n, where n is the number of unique values for a particular feature.

Ranking algorithms

The algorithms we chose to base our approaches on fall into the following two categories: a) **Filter methods** (rank features independently of any machine learning algorithm) and b) **Wrapper methods** (use ML algorithms to implicitly rank features).

Our selection of filter methods includes widely-used approaches that rely on statistical tests, along with a suite of feature ranking algorithms derived from the Relief algorithm. The latter is capable of identifying feature interactions, but its time complexity is significantly higher than that of other methods.

The wrapper methods use some of the most common and simple ML models such as random forests and XGBoost. The aim here is to build simple models and extract the implicit feature importance from the model.

A complete list of used algorithms can be found in Figure 3.

Baseline

We define our baseline performance using Equation 1, where $E[k]$ represents the expected value of the Jaccard score obtained from a random permutation evaluation of the top k features from a set containing a total of T features. In practical terms, this baseline performance reflects the expected outcome when guessing, which is not zero.

$$[!h]E[k] = \sum_{i=0}^{k} \frac{\binom{k}{i} \cdot \binom{T-k}{k-i} \cdot \left(\frac{i}{2k-i}\right)}{\binom{T}{k}} \tag{1}$$

Ranking Evaluation

We employ the **Jaccard index** to evaluate our results. The output of our pipeline consists of an array of 100 scores, representing assessments for all possible top-k features. The score assigned to the i-th position indicates the proportion of top i features in our generated ranking that inter with the ground truth ranking. For a visual representation of this evaluation and its corresponding curve, refer to Figure 2.

We introduced the **Area Over Baseline (AOB)** metric, specifically tailored to our use case, in order to provide a concise representation of algorithm performance. Initially, we condense the array of Jaccard index scores obtained from our evaluation pipeline into a single value representing the area under the **evaluation curve**. Subsequently, we apply a linear transformation to ensure that the baseline performance corresponds to zero, while a perfect ranking receives a score of 1. Note that a negative score is possible if an algorithm performs worse than random.

Exploratory Data Analysis

The main purpose of exploratory data analysis (EDA) was to familiarize ourselves with the data and start developing ideas for intelligent preprocessing. The dataset logs ad clicks and contains over 1.5 million data points with 100 features and a binary label indicating whether the ad was clicked or not. All features were anonymized with numerical labels such as **feature0**. The data contains only categorical features whose values were transformed using feature hashing, making them impossible to interpret. There were no missing values in the dataset since these were also hashed into integers, but their specific values were unknown.

Cardinality was one of the crucial feature properties we analyzed in our EDA. We observed that 7 features contained only one unique value, rendering them uninformative. 8 features, as well as the label, were binary, while 20 features had more than 1000 unique values. However, some features had extremely high cardinality, which made them impractical for making predictions.

The label exhibited an 80-20 ratio, indicating a bias toward the ad not being clicked. It should be noted that our dataset was created by undersampling the original data, which had a label ratio closer to 200-1. Additionally, the features had varying value distributions, and we observed that 13 features had one highly prevalent value, occurring in over 90% of the samples. We inferred that these values very likely represented missing values, implying that the corresponding features were quite sparse.

Results

We used Pearson's correlation coefficient (PC) to examine the connections between features. Our observations unveiled sets of highly correlated attributes, some of which exhibited a negative association with the label.

It's worth noting that features 98 and 99 had a perfect correlation (PC = 1) with each other and the label, effectively

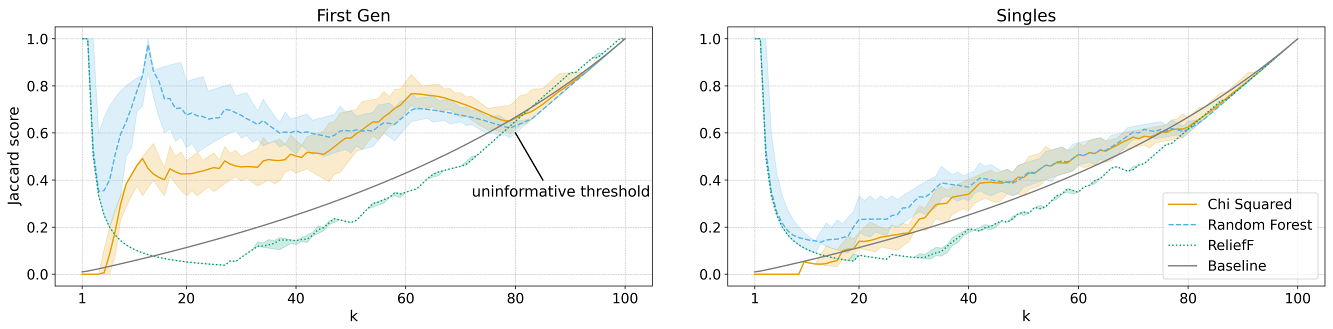

Figure 2. Ranking evaluation comparison. Jaccard score for each top k features for three different ranking algorithms and randomly permuted ranking as a baseline.

making them flawless predictors of the label. These characteristics were engineered into the data and were useful for the initial validation of our approaches as they are expected to be at the top of the feature ranking process.

Visualizing ranking performance

In Figure 2 we visualize evaluation curves three different ranking algorithms. The shaded areas represent a 95% confidence interval. We limited this visualization to three rankings for readability but they display characteristics that were recurring among all our algorithms.

First let us consider the FIRST GEN evaluation. The most prominent pattern is the initial ledge with a steep drop-off. This is caused by the two engineered features that correlate perfectly with the label and is perhaps the most basic indicator of algorithm coherence. We included the χ^2 metric to show that in fact not all ranking algorithms put these features in first and second place as we would like.

Another pattern we observe is the tail of our curve. All algorithms collapse to or very close to the baseline towards higher k values. This happens because the algorithms are encountering constant features, which hold zero information. To a ranking algorithm they are indistinguishable so the ordering has no structure, they are ranked randomly so our evaluation curve converges to the baseline, which is equivalent to guessing.

The last major pattern that had a high rate of recurrence among different ranking algorithms was the local peak in the values of k from 10 to 20. The cause is one of the correlated clusters of features that correlate negatively with the label that we found while performing EDA, many algorithms were able to rank these features as important.

Unfortunately no significant patterns occurred on the forward feature selection evaluation for any of our tested algorithms. The only exception being detection of the two engineered features as before. In other words the only reason any of our rankings performed better than baseline is because of the intrinsic overlap between the two base truth rankings. At this point our conclusion was that none of our algorithms were able to identify the features as ranked by the forward selection method used by Outbrain, so we instead shifted our focus to optimizing performance for FIRST GEN.

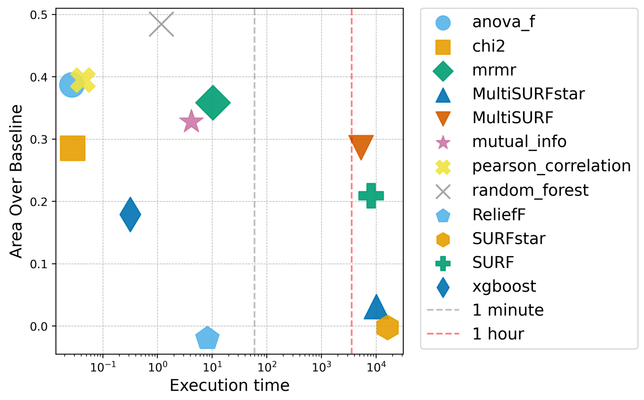

Figure 3. Results for 1% subsampling with baseline correction. All algorithms comparison.

Next we considered the algorithms' computational times versus their performance. This is shown in Figure 3. The x axis depicts time on a logarithmic scale so algorithms on the right are significantly slower than the ones on the left, a threshold for one minute and one hour are shown for a more intuitive understanding. The algorithm with the highest AREA OVER BASELINE was the random forest model from the `scikit-learn` library. Although some algorithms were much faster with a slightly lower performance the random forest model still fell within our threshold of feasibility. Thus it was chosen as the single candidate for further exploration.

We performed a number of evaluations with a wide range of different logarithmically spaced subsampling rates getting 100 different samples for each. The generated data's distributions is shown in Figure 4.

Ensemble technique

...

Conclusion

We have obtained results that contradict our initial hypothesis, which proposed that feature selection algorithms with higher complexity would yield more accurate results for our dataset. Surprisingly, we observed that these algorithms demonstrated some of the worst performance outcomes.

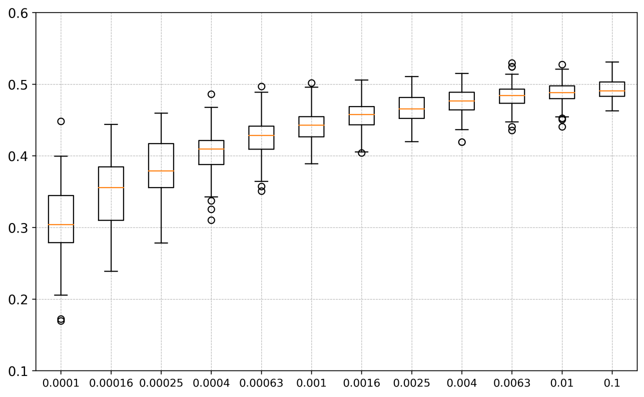

Figure 4. Subsampling performance with different subsampling proportions. .

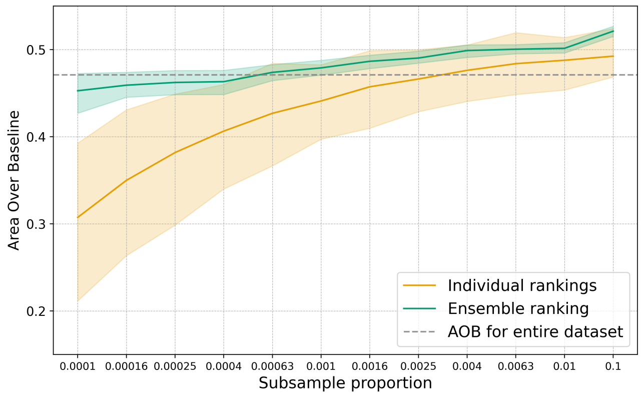

Figure 5. Comparison of individual and ensemble rankings. Ensemble rankings are obtained with bootstrapping.

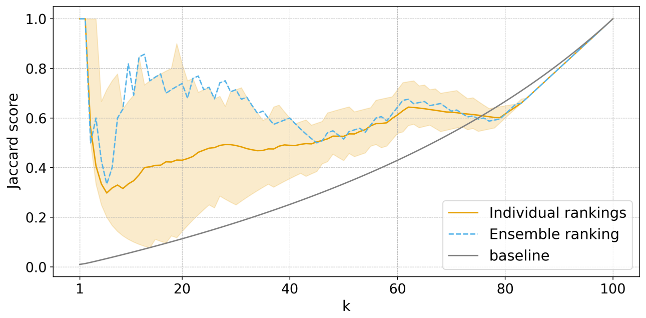

Figure 6. Random forest ranking performance. Comparison between individual rankings using different seeds and ensemble ranking from those individual rankings.

Fortunately, our employed ensemble technique proved to be highly efficient, generating accurate predictions even on very small subsamples of the data.

Discussion

...