

Feature Selection in Sparse Matrices

Rahul Kumar*, Vatsal Srivastava, Manish Pathak

MIQ Digital, Bangalore, India

Copyright©2019 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract Feature selection, as a pre-processing step to machine learning, is effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. There are two main approaches for feature selection: wrapper methods, in which the features are selected using the supervised learning algorithm, and filter methods, in which the selection of features is independent of any learning algorithm. However, most of these techniques use feature scoring algorithms that make some basic assumptions about the distribution of the data like normality, balanced distribution of classes, non-sparsity or dense data-set, etc. The data generated in the real world rarely follow such strict criteria. In some cases such as digital advertising, the generated data matrix is actually very sparse and follows no distinct distribution. For this reason, we have come up with a new approach towards feature selection for cases where the data-sets do not follow the above-mentioned assumptions. Our methodology also presents an approach to solve the problem of skewness of data. The efficiency and effectiveness of our methods is then demonstrated by comparison with other well-known techniques of statistics like ANOVA, mutual information, KL divergence, Fisher score, Bayes' error, Chi-square, etc. The data-set used for validation is a real-world user-browsing history data-set used for ad-campaign targeting. It has very high dimensions and is highly sparse as well. Our approach reduces the number of features to a significant degree without compromising on the accuracy of the final predictions.

Keywords Feature Selection, Sparse Matrices, Filters Methods

1. Introduction

High-dimensional data, in terms of number of features, is increasingly common these days in different domains. In order to extract useful information from these high volumes of data, one has to use statistical techniques in order to reduce the noise or redundant data. This is because

often we need not use every feature at our disposal for training a model. We can assist our model by feeding in only those features that are really important and are non-correlated. This is where feature selection can play an important role.

The objectives of feature selection at a broad level can be listed as:

- (1) To train the model faster
- (2) Reduce the complexity of the model and make it easier to interpret
- (3) Providing a better understanding of the underlying process that generated the data
- (4) To avoid overfitting of our model

In our use case, the data has high dimensions with thousands of features, high class imbalance and sparsity. This project was undertaken in order to reduce the model training time as well as to draw out insights from high performing features and to improve the accuracy of our models.

1.1. Common Filter Methods for Feature Selection and Their Drawbacks

Filter Methods select the features on the basis of their scores by various statistical tests for their association with the outcome variable [1]. Different tests are available depending on the nature of the data, the nature of the features and the type of outcome variables i.e. categorical/continuous.

Assumptions generally made by these algorithms are:

- Normality of variables
- Class balance of data-set
- Non-sparsity of data-set

Almost all the algorithms used for the purposes of feature selection or introducing separability in the data make a few basic assumptions [2] regarding the data characteristics such as, the dataset follows a normal distribution; the dataset does not have a very high class imbalance which means that if the data-set is labeled, then the different classes occur in almost equal proportions; the dataset is non-sparse which means that the data-set has a valid value for the majority of the columns of every row.

These assumptions might not be valid for all the real-world datasets. To overcome this problem, we have come up with some new techniques that use some common statistical methods to quantify the separability between the features and make the task of feature selection and dimensionality reduction easier.

2. Data

The dataset that we have is a user-browsing history dataset used for ad-campaign targeting. It has high dimensions and is highly sparse as well. The data is labeled into two classes – converts and non-converts. Converts are the users who fulfill our predefined criteria such as landing at the home-page of a website or making a purchase from the website, etc. and non-converts are the unsuccessful targets who do not satisfy the above-mentioned criteria. The dataset is highly skewed as the number of non-converts constitutes almost 99 percent of the entire dataset.

The dataset is stored in a *libsvm* format. A snapshot of the first 11 rows of the data is shown in Figure 1

```
0 127:2 160:2
0 164:1 45:1
0 20:2
0 280:1 45:4 20:2 42:2
0 49:5 160:2
0 31:2
0 33:1 119:1
0 65:7 82:24
0 160:1
0 141:4
1 31:3
```

Figure 1. First 11 rows of the dataset

2.1. Data Processing

Using the data shown in Figure 1, we create a frequency distribution for each feature. Let us assume that the dataset has N features. So, we will take all the values in *feature 0* of the *converts* dataset and get a frequency distribution of its values. Then we repeat the same process for *feature 1*, *feature 2*,..... and so on till *feature N*. At the end of the process we get one directory for each feature in the dataset. Then the same process is repeated for the *non_convert* dataset. In our example, we will have N different directories which will be named as *feature0*, *feature1*,, *featureN*. Inside each directory there will be two files – *converts.txt* and *non_convert.txt*. This means we are taking the frequency distribution of values for both the classes separately for each feature.

The file structure after this step is shown in Figure 2

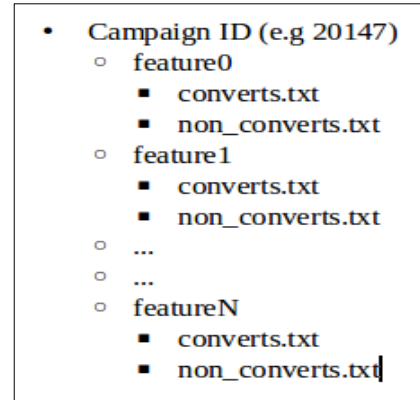


Figure 2. Directory structure after processing the data for frequency distribution

The *.txt* files will have data in the format given in Figure 3.

```
0.0, 31327
1.0, 427
2.0, 326
3.0, 138
4.0, 117
5.0, 89
6.0, 56
7.0, 43
9.0, 31
10.0, 37
13.0, 11
```

Figure 3. An example of the frequency distribution data for a feature

In the above image we can see that the value '0.0' occurs '31327' times, the value '1.0' occurs '427' times, etc.

2.2. Reasons for Sparsity - High Number of Zeroes

Most of the datasets are rectangular in shape with thousands of features and billions of rows. Often, the rows represent users and each of the elements of these rows represent the interaction of users with the features.

Now if the number of users and features are large than it is highly probable that a user may not have any interaction with a particular feature. Sometimes, a user may interact with a very small number of features. This potentially results in a highly sparse dataset.

In our case of predicting the conversion of a user based on the user browsing history, we face a similar situation. A user browses some webpages before making a purchase. We prepare a dataset with users on the rows and features made up of combinations of attributes like *website-url*, *keywords*, etc. on the columns. Since we have billions of users and thousands of features, it is very likely that a feature will have a very small fraction of all the users associated with it. This leads to a highly sparse dataset.

The sparsity can be seen from the Figure 1 where out of 8903 features in the data-set, the first row has only two with non-zero values (*feature127* and *feature160* with values 2 and 2 respectively).

2.3. Reasons for Class Imbalance

The meaning of class imbalance is the relatively high difference in the fraction of data points of different labels in the dataset. Class imbalance is quite common in a lot of real-world situations and in different industries. It is certainly a menace in Digital Advertising industry.

There are a variety of reasons for class imbalance. These could be flawed data collection procedure; improper sampling; inherent nature of the dataset and rigorous labeling criteria.

A website can have tremendous traffic. For an ecommerce website, if we only label the users who have made a transaction into positive (*converts*) class, it will most certainly lead to class imbalance as the ratio of visited users to those making a transaction is extremely high.

Another serious problem posed by class imbalance is the statistical validity of the model. If one of the labels in the dataset is extremely low, the consistency of parameters of the model may be affected. In simple terms, the consistency of a statistical parameter, is the inverse of the amount of deviance in the parameter estimation, as the sample size goes to infinity. This leads to high variance of the model estimated on the skewed data set.

A highly non-separable dataset is also probable due to skewness of the dataset. Many negatively labeled data points may be surrounding the positively labelled data points. Most of the models will not be able to identify the decision boundaries around these positively labelled points. This will lead to model with poor recall value.

3. Statistical Algorithms and Their Applications to Our Use Case

3.1. Ranking Features Based on the Anova Score and Effect Size

3.1.1. Concept of Anova

Anova [3] is one of the oldest methods to test the causality of a factor (categorical variable) on a continuous variable. The variance of the target variable is partitioned based on factor. The reduction in variance is judged based on a distribution to find out the effect of the factor.

Let y be a factor and x be a continuous variable. In our case, x represents the feature value in the dataset and y represents the class (*converts* or *non-converts*). The variance of x is v with cardinality n . The variable x is partitioned into x_1 for *converts* and x_2 for *non-converts* with cardinality n_1 and n_2 based on the factor y . The new variance v_1 and v_2 are calculated for x_1 and x_2 . The reduction in variance is expressed as a ratio of new and

initial variance as given by expression (1).

$$\frac{n_1 v_1 + n_2 v_2}{nv} \quad (1)$$

This statistic will follow a distribution based on the original distribution of x . The realization of statistic is judged based on distribution to come to the conclusion for the effect of the factor y on x .

3.1.2. Effect Size

Effect size is a simple way of quantifying the difference between the two groups. It has many advantages over using tests of statistical significance alone. Effect size emphasises the size of the difference rather than confounding this with sample size. The effect size is just the standardised mean difference between the two groups, and is expressed as,

$$Effect\ Size = \frac{\mu_1 - \mu_2}{v} \quad (2)$$

We can use the effect size to rank the features as it is a direct measure of the magnitude of the difference in the two classes. Finding the effect size of each feature and ranking them can give us the knowledge of the importance of each feature.

3.2. Ranking Features Based on Fisher Score

3.2.1. Concept of Fisher Score

Fisher Score [4] is one of the most widely used supervised feature-selection methods. It selects each feature according to their scores under the Fisher criterion.

The key idea of Fisher Score is to find a subset of feature, such that in the data space spanned by the selected feature, the distances between data points in different classes are as large as possible, while the distances between data points in the same class are as small as possible.

We compute a score for each feature independently according to the criterion F . Specifically, let μ_k^j and σ_k^j be the mean and standard deviation of k^{th} class, corresponding to the j^{th} feature. Let μ^j and σ^j denote the mean and standard deviation of the whole data set corresponding to the j^{th} feature. Then the Fisher score of the j^{th} feature is computed below,

$$F(x^j) = \frac{\sum_{k=1}^c n_k (\mu_k^j - \mu^j)^2}{(\sigma^j)^2} \quad (3)$$

where,

$$(\sigma^j)^2 = \sum_{k=1}^c n_k (\sigma_k^j)^2 \quad (4)$$

In our case, there are two classes, so $c = 2$, n is the number of data points in k^{th} class and j represents a particular feature in the dataset. The value of $(\sigma^j)^2$ calculated for each feature can be used for the purposes of feature ranking.

3.3. Reduction of Zeroes for Anova and Fisher Score

As in Anova, we partition our dataset based on the factor

and test the effect size that is the difference in the mean divided by variance for each partition. Since our feature has a high number of zeroes, the mean of both partitions tends to zero. This creates the problem of very low effect size.

To overcome this problem, we try reducing the number of zeroes without disturbing the distribution of the overall dataset. We find the proportion of each of the values in the partition is shown in Table 1,

Table 1. Proportion of Values in each partition

Values	Percentage
0.0	99.53
1.0	0.02
2.0	0.03
3.0	0.22
4.0	0.15
33.0	0.3
61.0	0.1
75.0	0.1

The data above is skewed towards 0 as it constitutes more than 99 percent of the entire dataset. The mean will therefore, also tend to 0. We try to adjust the concentration of 0's in the data without disturbing the distribution of other values.

Let us take the concentration of each value x to be c_x . Now taking new concentration of 0 to be c'_0 , we define the modified concentration c'_x of each of the other values in the following manner.

$$c'_x = \frac{(1-c_0)c_x}{\sum_{c_x, x \neq 0} c_x} \forall x, x \neq 0 \quad (5)$$

The new set of c_x has less concentration of 0's but the distribution of other values remains the same as their relative concentration is unchanged.

The justification for changing the concentration of c_0 is not difficult. If we look at the way how data has been prepared, the high number of zeroes is because of non-interaction of a user with a feature. In ANOVA and Fisher, we are testing the difference in the mean of the counts converted group and non-converted group. It makes little sense to include those users who have not interacted with a feature at all.

3.4. Ranking the Features Based on Mutual Information

3.4.1. Concept of Mutual Information

Mutual Information [3, 5] is a measure of dependence between two random variables. In probability theory and information theory, the mutual information (MI) of two random variables is a measure of the mutual dependence between the two variables. More specifically, it quantifies the "amount of information" obtained about one random variable, through the other random variable. The concept of

mutual information is intricately linked to that of the entropy of a random variable, a fundamental notion in information theory that defines the "amount of information" held in a random variable.

The Mutual Information between two random variables for feature k is defined as,

$$I(X_k; Y) = \sum_{y \in Y} \sum_{x \in X_k} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \quad (6)$$

In our case, $p(X_k, Y)$ the joint distribution of the feature X_k and Y which is the class label (converts or non-converts). $p(X_k)$ and $p(Y)$ are the marginal distributions of feature k and class label Y . We can use the value of $I(X_k; Y)$ calculated for each feature for the purposes of ranking.

3.5. Kullback-Leibler Divergence

3.5.1. Concept of Kullback-Leibler Divergence

The Kullback-Leibler Divergence [3] or KL Divergence measures the difference in two probability distributions. The mathematical formulation is for an individual feature k given as

$$D_k(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad (7)$$

where $p(x)$ and $q(x)$ are probability distributions of converts and non-converts for feature k . It can be observed that if $p(x) = q(x)$ for all x , then the measure is 0. In Mutual Information, we are measuring the difference between the joint distribution of two variables, $p(X, Y)$ and the marginal distribution, assuming the two variables are independent. The joint distribution of X and Y will be $p(X) * q(Y)$. In KL Divergence we are calculating the difference between two different distributions $p(X)$ and $q(X)$.

The score $D_k(P||Q)$ calculated for each feature can be used for the purposes of ranking our features.

3.6. Ranking Features Based on Bayes Error

3.6.1. Classification Using the Discriminant Function

Let X be the feature vector space and K be the set of classes to be classified. To make things simpler, we take two classes C_1 and C_2 one for converts and the other for non-converts respectively. The conditional probability density function of feature space for two different classes will be

$$P(X = x | C = C_1) \quad (8)$$

And

$$P(X = x | C = C_2) \quad (9)$$

Now to find the discriminant function, we equate the two distributions given by equations 8 and 9. This gives us a function in x , as shown in equation 10, which is the desired discriminant function.

$$P(X = x | C = C_1) = P(X = x | C = C_2) \quad (10)$$

We classify the regions of the domain to particular class on the basis of their probability. The data-point is assigned to the class with the highest probability among all the classes.

3.6.2. Concept of Bayes' Error

Bayes' error is the cumulative measure of the risk taken while classifying the regions to a particular class. If we classify $X = a$ as class C_1 then the risk we are taking is $P(X=a|C=C_2)$. Summing over the domain, we find the total risk of the classification aka Bayes' error.

Bayes' error depends upon the decision boundaries. It is imperative that such decision boundaries should have minimum Bayes' Error.

3.6.3. Ranking Features Using Minimum Bayes' Error

For each feature, we have the values $x_1, x_2, x_3, x_4, \dots, x_n$. Taking each value as a discriminant function we form our decision rule:

$$\text{predicted class} = \{0, x < x_k, 1, x \geq x_k\} \quad (11)$$

We find the Bayes' error using this rule mentioned above and find the minimum error over all x_k . The value x_k is chosen such that there is a minimum level of wrong classification.

Repeating this for all the features and finding the minimum Bayes' Error for all of them, we can rank them based on the minimum Bayes' Error.

3.7. Kernel Density Smoothing for Mutual Information, KL Divergence and Bayes' Error

Kernel Density Smoothing [3] is a non-parametric method to estimate the density of random variable. The method relies on the smoothing the frequency distribution of the random variable. It uses a kernel to extrapolate the distribution at unknown values in the neighbourhood of the point. The following general expression is used for kernel,

$$K_h(X_0, X) = D\left(\frac{\|X-X_0\|}{h}\right) \quad (12)$$

Where, h is the kernel radius, X, X_0 are values in the domain space, $D(x)$ is real-valued decreasing function. For

$$E_j = \frac{\sum_{k \in K} (\text{converts across all buckets})}{\sum_{k \in K} (\text{converts across all buckets}) + \sum_{k \in K} (\text{non converts across all buckets})} \quad (14)$$

where K is the set of buckets after binning. and the observed frequency O for k^{th} bucket is taken to be

$$O_j^k = \frac{(\text{number of converts in bucket } k)}{\text{number of converts in bucket } k + \text{number of non converts in bucket } k} \quad (15)$$

Accordingly, the χ^2 parameter is calculated. The *degree of freedom (dof)* in this test is *(number of buckets-1)*. Higher the χ^2 parameter for a feature more is its importance in model training. Features can then be ranked with $(\chi^2)/dof$ parameter.

our purpose we have taken an exponential function for D . Now for a large h we can reduce the influence of abnormal values which in our case is the extremely high frequency of zero values.

3.8. Ranking Features on the Basis of Chi-Square Test

Chi-Square [3] Test is used in statistics to test the independence of two events. Chi-Square Score measures how much the expected count E and observed count O deviate from each other. In feature selection, the two events are the occurrence of the feature value and occurrence of a specific class in the dataset. This test is usually applied when both the feature values and the class labels are categorical.

In other words, we want to test whether the occurrence of a specific feature value and the occurrence of a specific class are independent. If the two events are dependent, we can use the occurrence of the feature value to predict the occurrence of the class. We aim to select those features whose values are highly dependent on the occurrence of the class.

When the two events are independent, the observed count is close to the expected count resulting in a small chi-square score (χ^2). So, a high value of χ^2 indicates that the hypothesis of independence is incorrect. In other words, the higher the value of the χ^2 score, the higher the likelihood that the feature is correlated with the class hence it should be selected for model training.

To apply this test on our dataset, first the continuous data values needed to be converted to categorical values. In order to convert them to categorical value, we take floor of \log_2 of all the values and aggregate their frequencies. This gives us a dataset of the format shown in Figure 2.

Theoretically, the Chi-Squared parameter for a distribution is given by:

$$\chi_j^2 = \sum_{k \in K} \frac{(O_j^k - E_j)^2}{E_j} \quad (13)$$

where O represents the observed frequency and E represents the expected frequency.

Here, in this case, the expected frequency E for j^{th} feature is given as

4. Results

The dataset we have used for testing has 8903 features. It has the same format as that shown in Figure 1. We randomly sampled twenty percent of the campaign dataset as test data. The testing dataset has around 52.8 million users, out of which around 2600 users are in the positive or converts class.

Using the feature ranking techniques as mentioned in Section 3, we give a score to each feature of our dataset. Based on this score, we select the top 10 percent, 20 percent, 30 percent, 40 percent and 50 percent features. Next, we train a Naive Bayes model using only these top n percent features. This model is trained to predict the users who will fall in the converts class. We then compare our results with the model trained without applying any of the above-mentioned feature selection techniques.

We observe that the models trained after applying the feature selection techniques can filter out a significant amount of noise from our dataset. This makes our model more robust and generalized. As can be seen from the figures below, there is a great improvement in the ability of our model to extract the signal from noise when only 10 percent to 20 percent of the highest scored features are taken for each algorithm.

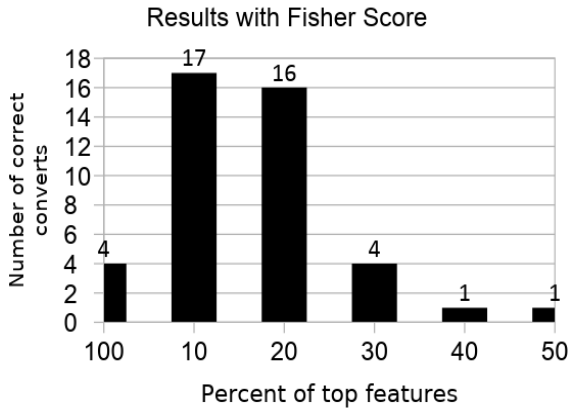


Figure 4. Results with Fisher Score

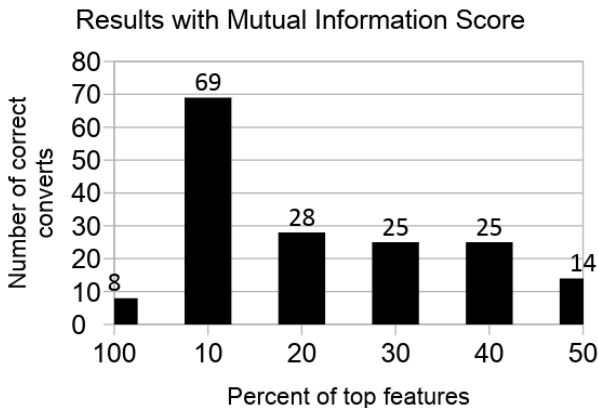


Figure 5. Results with Mutual Information Score

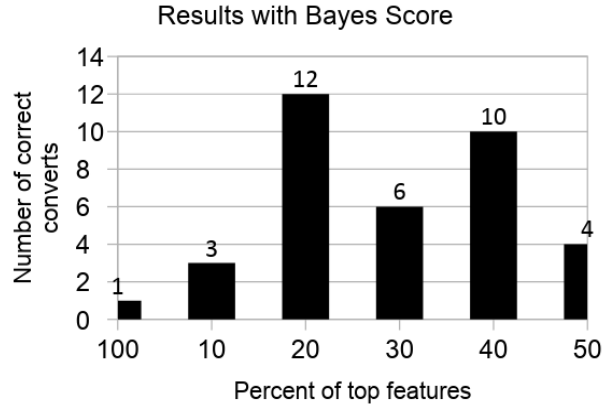


Figure 6. Results with Bayes Score

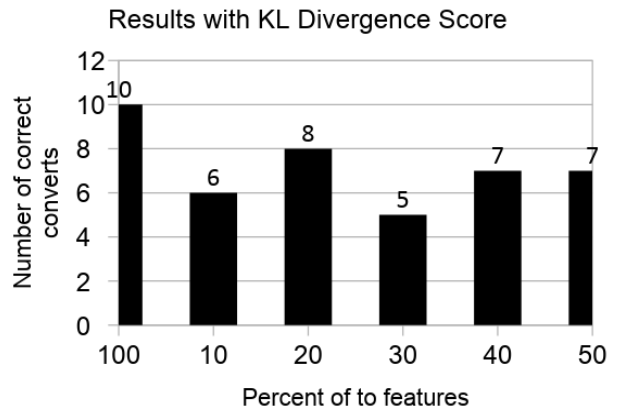


Figure 7. Results with KL Divergence Score

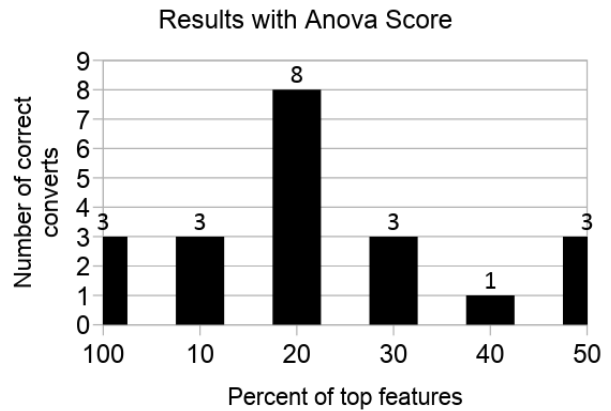


Figure 8. Results with Anova Score

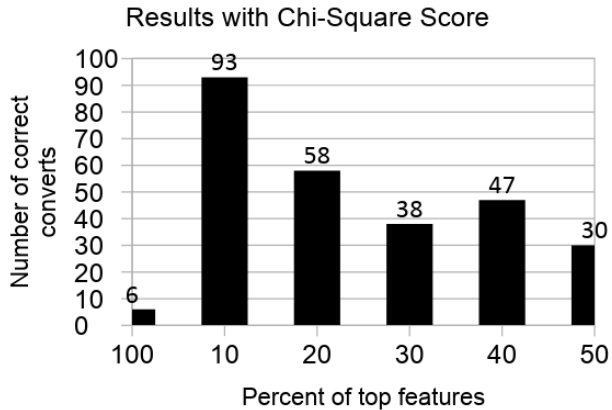


Figure 9. Results with Chi-Square Score

5. Conclusions

Most of the prominent filter methods used for dimensionality reduction makes some basic assumptions about the distribution of the data which might not hold true in the real-world datasets. The approach that we have proposed provides a solution towards feature selection for datasets that are highly sparse in nature with a high class imbalance. The efficiency and effectiveness of our methods can be demonstrated by the fact that we were able to correctly identify almost four times more users of the positive (converted) class using only one-tenth to one-fifth of the total number of features in the dataset. This has not only led to a significant performance boost for our model training pipeline by reducing the size of the training data, but has also made our model more robust and less prone to overfitting as can be seen from the higher number of correct predictions on the test dataset.

REFERENCES

- [1] Sánchez-Maróño N., Alonso-Betanzos A., Tombilla-Sanromán M. (2007). Filter Methods for Feature Selection – A Comparative Study. *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, 4881, 178-187. DOI: 10.1007/978-3-540-77226-2_19.
- [2] Avrim L. Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence* 97(1-2), 245-271. DOI: 10.1016/S0004-3702(97)00063-5.
- [3] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, Lotfi A. (2006). *Feature Extraction Foundations and Applications*. New York, N Y.: Springer.
- [4] Gu, Q., Li, Z., Han, J. (2011). Generalized Fisher Score for Feature Selection. *UAI, 2011b*, 266–273.
- [5] Doquire, G., Verleysen, M. (2011). A Hybrid Approach to Feature Selection for Mixed Categorical and Continuous

Data. *International Conference on Knowledge Discovery and Information Retrieval*, 2011, 394-401. DOI: 10.5220/0003634903940401.

- [6] Guyon, I., Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157-1182.
- [7] Yu, L., Liu, H. (2003). Feature selection for high-dimensional data: a fast correlation-based filter solution. *Proceedings of the twentieth International Conference on Machine Learning*, 856–863.