Data Science

Advanced Machine Learning

Project 2

Weronika Plichta (no. 335725)
Maja Wasielewska (no. 335210)
Jakub Kubacki (no. 313494)

June 3, 2024

1 General idea of the project

Our general idea was to search through the possible feature subsets to build model on, trying to maximize the score measure mentioned in the task description (later referred to as the "score") and once the subset for which the score is the highest (for the test model), produce the best possible model on that subset of features, score-wise.

However, that approach posed several problems. First, we needed a way to do this search intelligently, as the number of all possible subsets grows exponentially with the number of features in the experiment matrix. Second, even with an intelligent way of searching, trying to meticulously search through the possibilites with all 500 features in the picture was way too demanding on our hardware, so the need for a preliminary feature selection that didn't require fitting any models arose. Finally, we obviously needed to design a testing system that will allow us to compare models and feature subsets without running to problems stemming from variance and overfitting to a given validation set.

Our solutions for each of those problems will be discussed in the following report.

2 Preliminary feature selection methods

In order to filter out the irrelevant variables, we attempted several feature selected methods.

2.1 BORUTA

First of said methods was BORUTA. We used the available implementation in the **boruta_py** package. The algorhithm initially had a lot of trouble deciding on any variable, but during the 8-th iteration it pushed through that plateau and eliminated all but 26 features. Several iterations later it reached a conclusion, labelling all but 12 features as irrelevant. Below are the indices that BORUTA selected:

The top26: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 64, 100, 101, 102, 103, 104, 105, 109, 155, 158, 285, 335, 351, 403, 458, 498]

The top12: [0, 2, 3, 4, 8, 9, 100, 101, 102, 103, 104, 105]

After some minor testing of the bigger subset in the second stage of feature selection, we decided that feature number '6' was incorrectly thrown away and should still be considered for further selection, alongside the features BORUTA didn't eliminate.

Features forwarded to the second stage: [0, 2, 3, 4, 6, 8, 9, 100, 101, 102, 103, 104, 105]

2.2 QDA

Second method was QDA. We implemented QDA from sklearn.discriminant_analysis package and later applied random forest classifier to assess the importance of feature.

The top15 and features forwarded second stage: [104, 106, 103, 101, 102, 105, 10, 221, 352, 324, 286, 274, 238, 68, 440]

2.3 RF

The third selected method was feature selection using a Random Forest classifier from the scikit-learn package. This method trains a Random Forest to assign importance scores to each feature. The SelectFromModel class then selects the top features based on these scores. Listed below are the 10 and 5 features with the highest importance score:

The top10: [3, 5, 6, 8, 100, 101, 102, 103, 104, 105]

The top5: [100, 101, 102, 103, 105]

3 Second stage of feature selection

Our methodology of the extensive search was based on three components.

3.1 Finding candidates

We selected candidate subsets using a bidirectional search starting from a random subset of the subset determined in the preliminary feature selection stage, at each iteration trying to increase the score of the testing model the most. Each search had its own random seed present throughout to ensure that it won't loop due to variance. We tried different models for testing, for instance the Random Forest Classifier with 120 trees and KNN Classifier with $K = 50$.

This way we determined several dozen local optimas (meaning subsets that couldn't be improved by either adding or taking out any single feature with the currently set seed). Those candidates were then moved to the validation part of the process.

3.2 Validating results

Every candidate for the global optimum was then tested across 100 different random seeds (with the score averaged) in order to eliminate variance as much as possible, using much bigger model (Random Forest Classifier increased to 360 trees, KNN Classifier was applied with $K = 150$) to come closer to the version of the model we were going to train later on.

3.3 How did we calculate the score?

That leaves one question - how did we fairly assess the score of the model? Our general idea was to perform a crossvalidation and average the results. To avoid overfitting, data splitting into folds was randomized, reliant on a different seed every time it was possible.

When we had a given train-val split, the score was assessed by having the model predict the probabilities of validation observations belonging to the positive class and selecting top 20% observations that had this probability the highest. Then those observations were checked for belonging to the positive class and the model was awarded the score ranging from 0 (not guessing anything correctly) to 10000 (all observations chosen were from the positive class). Finally the model was penalized by 200 for each of the features used, effectively simulating the real calculation of the score.

We are aware that this is by no means perfect, as the score might be lower due to a reduction in the number of training observations or the size of the model and higher due to a more generous fraction of observations belonging to the positive class in the training dataset compared to the test dataset we will need to make predictions on with the final model, but it was the closest approximation we came up with.

3.4 Global optimum candidates

The highest-scoring subsets of variables were [101, 102, 105], [102, 103, 105] and [101, 102, 103, 105]. We ultimately decided to build our models using the 103rd, 104th and 106th feature only, for a score penalty of 600.

4 Creating the model

In order to create the best possible model on the features we have selected, we performed several experiments.

4.1 Model selection

First, we have considered many different models, including Logistic Regression, Random Forest, K-Nearest-Neighbors and Feed-Forward Neural Networks. Out of those:

Logistic regression was way too simple to capture the nature of the data.

FNN failed to understand the relationships in the data, suffering from serious overfitting and/or trouble in learning.

Random Forest and KNN were both interesting, so they were then examined in details.

4.2 Hyperparameter selection

Training hundreds of models with different random seeds we have determined that Random Forest works best with around 300-400 trees and KNN works best with K having the value of around 200. Out of those two, KNN scored about 200 points higher, so we decided on predicting the probability of a given observation being positive based on how many of its 200 nearest neighbors are positive.

4.3 Outlier prelabeling

One thing that we have noticed, when looking at visualizations of the dataset (as it was restricted to three columns, it was possible to visualize using python libraries for interactive 3D plots) is that most of the observations at the outskirts of the dataset belonged to the positive class. Moreover, it's well known that KNN will have higher variance when predicting outliers. That's why we decided to test what's going to happen if we modify our model so that it believes that outliers are all positive.

For outlier detection, we have used Isolation Forest. Model that was assigning a probability of being positive equal to 1 to all found outliers turned out to have about 200 higher median score than the model that did not perform that additional step. Thus this step made it into the final model.

4.4 Failed experiments

We tried augmenting the data with column products, that failed completely across all models considered.

5 Summary

Our final model is based on only three columns, the 103rd, 104th and 106th one (so indices [102, 103, 105]). On these columns, first an isolation forest is fit, then a K-Nearest-Neigbors classifier with the $K = 200$ is fit. KNN predicts the probabilities of belonging to the positive class for all the test observations and then isolation forest modifies them, assigning probability of 1 to all the outliers. Finally, we order the observations by the probability of being positive and select 1000 most likely candidates.

During our crossvalidation-based tests (mentioned in the subsection 3.3), our final model reached a median score of about 6940 €.