# Response Model: A Real World Binary Classification Problem

Piąstka[1], Leszko[2], Lorenz[3] & Früchtnicht[4]
1 Konrad Piąstka M20180038, konrad.piastka@gmail.com
2 Dominika Leszko M20180077, leszkodominika@gmail.com
3 Joana Lorenz M20180412, M20180412@novaims.unl.pt
4 Lukas Früchtnicht M20180067, lukas.f@me.com

**Abstract**
In order to support a retail food company in the wise spending of its Marketing budget, the objective of the project was the development of a predictive model to appropriately classify their customers as valuable vs. non-valuable targets of the next Marketing Campaign. To build the model, a campaign involving 2240 customers was carried out and the data appropriately labeled, if a customer bought the product. The provided dataset further includes 24 socio-demographic and firmographic features on the customers. The following report outlines the project work consisting of data preprocessing, model development and fine-tuning as well as the interpretation of the results.

# 1. Data Preprocessing

## 1.1. Null Values

Considered options for the removal of the 24 detected null values in the income variable were the imputation with a measure of central tendency, such as the variable's mean or median value, as well as the imputation using regression. In order to introduce minimal possible error through imputation and predict with the most probable values, we decided to go with the latter and impute via linear regression. Independent features used for the regression model were determined by performing backward elimination to ensure of their significant impact on the Income value.

## 1.2. Outlier detection and handling

Outliers were removed class-wise, based on a cut-off point of standard deviations from the mean (sigma rule). The class-wise outlier removal was preferred in order to remove contextual outliers - such data points that deviate strongly from the other data points in the same class. This way, we hope to reach more consciousness within the classes and to be able to capture the essence of the class members much better[1].

Different sigma levels were tested with the best results in regards to the amount of outliers, which we tried to keep below 3 % of the data, with a particular focus on not removing too many data points from the under-represented class 1. Furthermore, we observed the performance of the prediction models at multiple sigma values. The best results could be observed at a sigma value of 3.5, which is in accordance with common practice of the sigma rule. Figure 1 shows that removing observations beyond 3.5 standard deviations from the mean outperforms all other considered sigma values on both metrics: profit per capita and F1 Score and across all tested models.

[1] Antonio J. Tallón-Ballesteros, José C. Riquelme (2014)

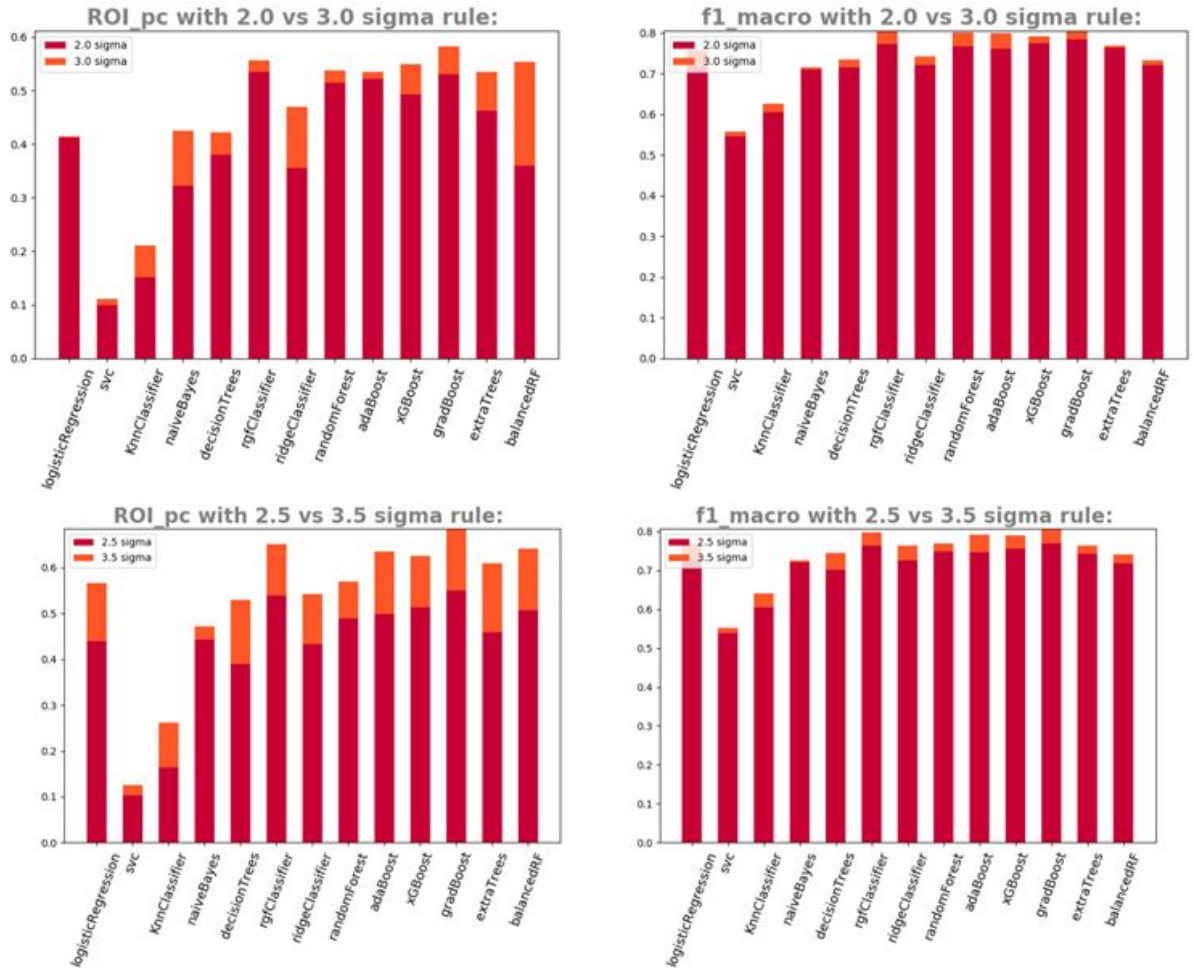


*Figure 1: Profit per capita, F1_macro per capita based on sigma rule*

Further methods considered and implemented include outlier detection via the IQR method, DBScan and the Stalactite plot. The clustering method DBScan was implemented specifically in order to detect outliers on a multivariate, rather than a univariate perspective. However, this lead to a loss of the intra-class perspective and no improvement in model performance.

The Stalactite plot[2] is a technique for outlier removal considering the Mahalanobis distances of data points in a multivariate normal distribution. The main limitation of is the assumed normality, besides having an exponential complexity in regards to the dataset length. To assess the multivariate normality we used QQ plots[3] which plots the squared mahalanobis distances of the data points in ascending order against the quantiles of the chi-square distribution. *A-priori consideration*: After finding continuous columns and transforming them to normal distribution outliers are deleted in a very efficient manner. *A-posteriori consideration*: We were not able to use the

[2] Atkinson, A.C. & Mulira, HM. Stat Comput (1993)

[3] Applied Multivariate Statistical Analysis by Johnson Wichern 2007

stalactite plot efficiently in production as only some columns met the assumptions needed and the plot suggested too many outliers considering our problem.

After detecting outliers, we considered to impute, delete or transform the respective values. For the imputation of outlier values, we considered and implemented KNN Regression. However, the error of the KNN remained high. Another considered option was the winsorization of values, which leads to the transformation of the x-th lowest and highest values to a defined quantile, thus changing values instead of removing them. However, many variables are skewed to one side and thus transforming variables on both sides of the distribution did not make sense.

Since we did not want to skew data points based on an improper imputation and few data points, especially with class label 1, were detected, we decided to drop the data points, compromising the risk of losing valuable information from the inconspicuous features of the data points for the soundness of our data.

### 1.3. Feature Engineering

In order to create maximum value from the existing information, multiple new features were created. Their importance of the newly created, as well as, the original features for the prediction was later evaluated using Recursive Feature Elimination through Random Forest. We decided to not transform any variables due to loss of interpretability.

The following feature engineering steps were undertaken:

- Lower the dimensionality of the categorical variables ‘Education’ and ‘Marital Status’ based on the value’s discrimination ability
- Capture the total count of children and teens in a household as well as the existence of any children or teens in the household
- Transform the accepted Marketing campaigns into a response rate and a binary value, indicating whether a customer ever accepted a Marketing campaign
- Calculating the web conversion rate which shows how many web visits are needed to generate a deal.
- Calculate average purchases per month in order to show the customers relative spending behaviour
- Calculate percentages of specific category and channel purchases of the overall purchases as potentially better indicators of the customer behavior
- Perform PCA on the subset of “Num*Purchases”, “Year_birth”, “Income”, “Recency” “DT_Customer” and “Mnt*” and include the PC’s accounting for =< 80% of the variability.

- Perform LDA on the subset of “Num*Purchases”, “Year_birth”, “Income”, “Recency” “DT_Customer” and “Mnt*” and include the PC’s accounting for =< 80% of the variability. A priori this was thought as the best discriminator as accounting in for the class separability, however it has to be noted that assumptions of the LDA were consciously not verified. A posteriori only one LD accounts for more than 80% of the variability in the LD space and is also the most important feature for the baseline random forest.
- Perform K-Modes clustering on the columns “AcceptedCmp*”, complain and encodings for education and combined encoding for Child, Teen and Marital Status with number of clusters determined by chi squared discriminability for K in [1,5].

### 1.4. Feature selection

We defined our features for modeling in two steps. First, we determined the optimal number of features over 5 seeds using recursive feature elimination (RFE). Feature selection is based on recursively considering smaller and smaller sets of features when training an estimator until a defined number of features is reached. We trained a random forest on added data (SMOTE) and eliminated features stepwise and then performed cross validation on original data. We defined the number of features from 1 to 50 (all) and computed the mean of the five-fold cross-validated profit per capita over five different seeds. A-priori our considerations were that the profit would rise with increasing number of features until a certain threshold is reached, after which the profit per capita would fall again. A-posteriori we can conclude from the graph that our expectations were met, however we are surprised that only 13 features are needed for optimal performance.

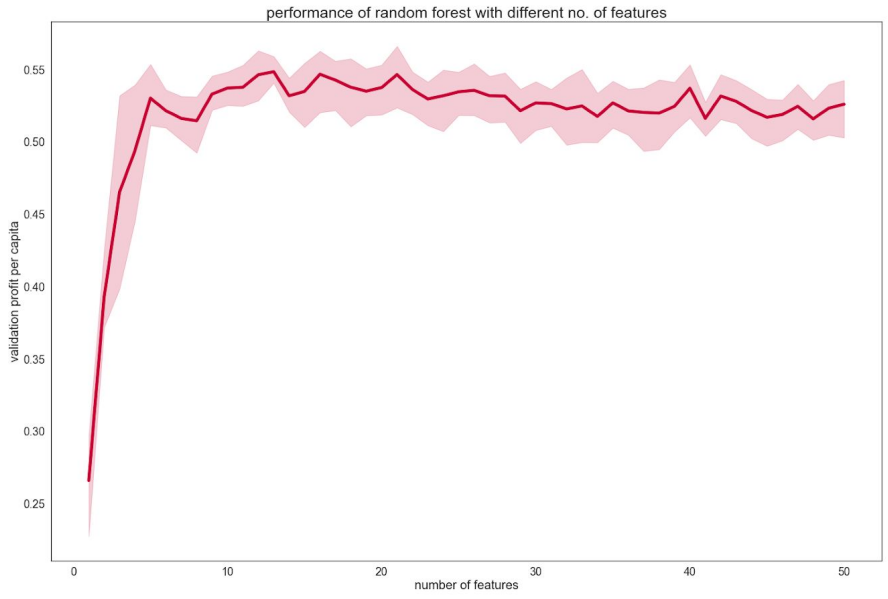


*Figure 2: Performance of Random Forest with different number of features*

In a second step we looked into what these 13 features are over five different train-test-splits and found that a total of 17 unique features were used over the different seeds. We decided to keep these 17 features for modelling (other models than Neural Networks) instead of selecting only the 13 most important features of the Random Forest in order to stay unbiased by the specific data split. Finally, we can see on the below bar charts that thanks to the applied feature selection, both profit per capita, and F1 Score increased across nearly all tested models, which only reassured us of keeping only the 17 selected features to train models.

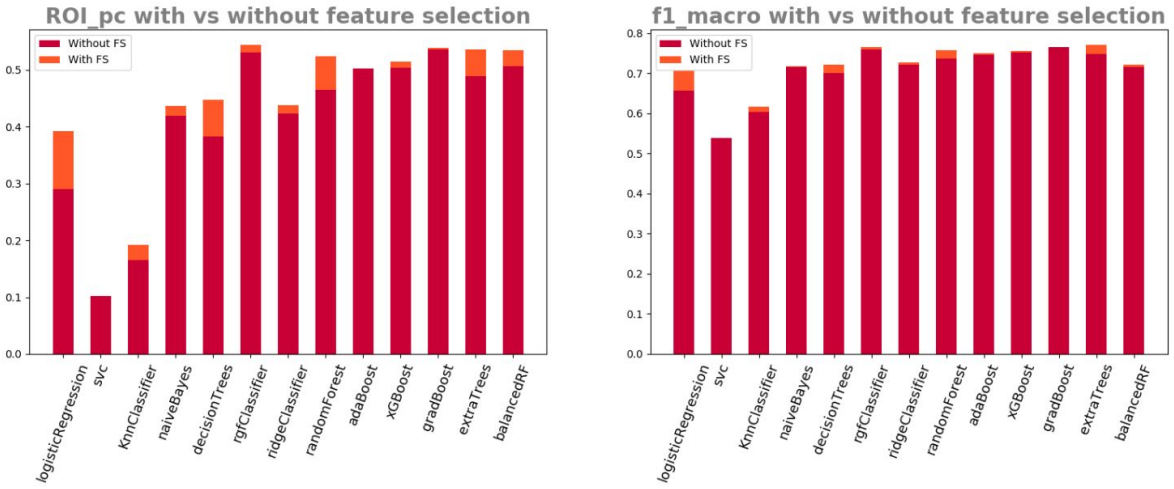


*Figure 3: Profit per capita, F1_macro with and without feature selection*

## 2. Modeling

### 2.1. Handling of data imbalance

Within the full data set, the important class 1 is highly scarce with only ~15 % of all observations. Since their classification is of greatest importance in order to make profit, measures had to be taken in order for the models to be able to also classify the minority class correctly.

First, we made sure that the proportion of classes with the train and test data set was equal through the use of the train_test_split stratify function. Next, we evaluated methods for undersampling of the common class or oversampling of the rare class. Undersampling was implemented using RUS sampling, whereas oversampling was achieved through the creation of synthetic data using SMOTE.

Since the dataset was already relatively small, we did not want to lose further data points by removing class 0 data points through undersampling. Using SMOTE, however, results in terms of profit per capita and F1 Score improved significantly across all tested models, as can be seen in Figure 3. We found a good ratio of artificial observations of class 1 to original observations of class 1 of 50% shifting the ratio of class 1 data points to 30%. When using k-fold validation, we made sure that

the models were trained on the added artificial and original data (added data), but evaluated on the original train data split only (original data).

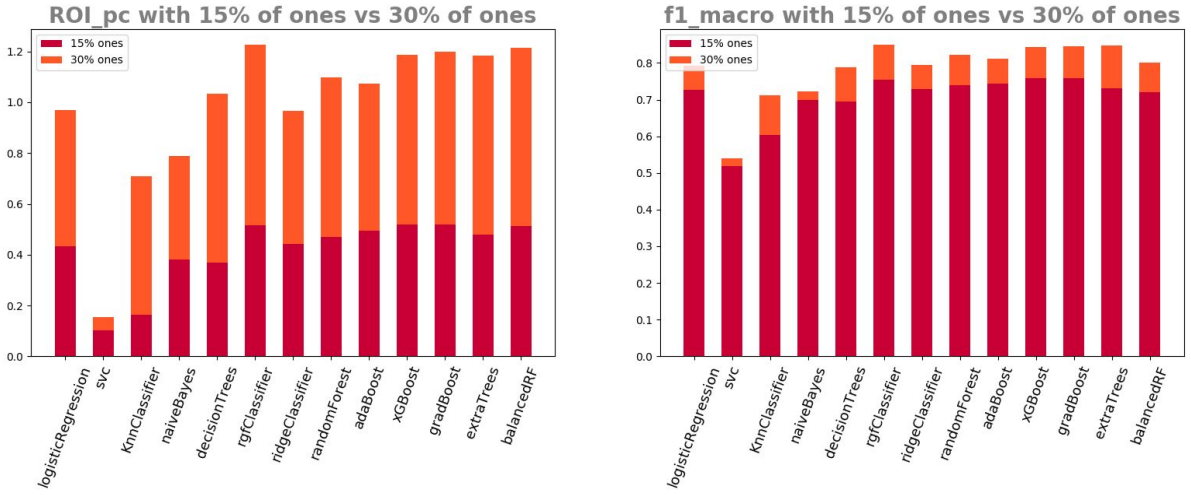


*Figure 4: Profit per capita, F1_macro with and without SMOTE oversampling*

### 2.2. Measures of model performance

Given the described imbalance of the data set, measures of model performance other than the accuracy had to be considered. This is because the accuracy will naturally be very high through a large amount of True Negatives, even if a high percentage of False Negatives lead to a poor target business value, the resulting profit of the Marketing campaign.

Considered methods for a statistical evaluation of the model performance other than the accuracy included the AU-ROC Score, Precision, Recall and the F1 Score, which represents the harmonic mean of Precision and Recall. In the underlying problem we have a higher focus on detecting the positive class, based on the profit/loss distribution of False Positives (-3 Euros) and True Positives (+8 Euros) as well as the strong imbalance of labels. Due to a relatively high amount of constantly present True Negatives, the ROC FPR measure does not capture differences in model performance accurately. Thus, we put a focus on Recall and Precision, which much better capture the quality of the prediction. Since both Recall and Precision are of importance in our case, we decided to use the F1 score as a measure of model performance. In order to account for the class imbalance, the F1 score was averaged over classes (F1_macro).

In addition to the average F1_makro score, we created a profit measure in order to accurately depict the ratio between the loss through falsely contacted customers (False Positives) and the profit through correctly contacted customers (True Positives).

Profit = FP * (-3) + FN * 0 + TP * (11-3) + TN * 0

Furthermore, in order to compare performance over different train-test-splits, we measured the profit ratio, given by the achieved profit divided by the maximum possible profit ((TP+FN) * 8).

### 2.3. Implemented Models

Generally, according to the No Free Lunch Theorem, no algorithm is best for every problem. Thus, no specific algorithm could be chosen a-priori all algorithms suitable for binary predictions were implemented and our objective in model selection was to try and evaluate multiple approaches, evaluating their strengths, weaknesses and their suitable for the underlying problem. Furthermore, different (even weak) classifiers, especially returning uncorrelated predictions, could be valuable for the later building of ensemble methods. Figure 4 gives an overview of the implemented methods.

| **Linear Models** | **Tree-based Models** | **Others** | **Ensemble** |
|---|---|---|---|
| ● Logistic Regression<br>● Ridge Classifier | ● Decision Tree<br>● Random Forest<br>● Regularized Greedy Forest<br>● Extremely Randomized Trees<br><br>Gradient Boosted<br>● Ada Boost<br>● Gradient Boost<br>● Xtreme Gradient Boost | ● KNN<br>● (SVC)<br>● Naive Bayes<br>● Neural Network | ● Voting Ensembles<br>○ equal weights<br>○ correlation based weights<br>○ accuracy based weights<br>● Bagging<br>● Stacking |

*Figure 5: Overview of implemented models*

#### 2.3.1. Linear Models

Generally, linear models have their biggest advantage in their transparency and thus interpretability of results. However, they fail to model non-linear relationships within data and their simplicity may significantly lower the predictive performance on complex relationships.

*Logistic Regression* allows to model the output which can be interpreted as probability of each datapoint belonging to either class. The *Ridge Classifier* is based on the Ridge Regression, which enables the regularization, i.e. avoiding of over and underfitting, of a linear regression through the introduction of an error term.

### 2.3.2. Tree-based Models

*Decision trees* represent a non-parametric supervised learning method, where splitting criteria are selected based on the optimization of a function to measure the quality of a split, such as the impurity decreased after split. Decision Trees are simple to interpret but prone to overfitting. Tree-based *Boosting methods* represent the combination of multiple Decision Trees, with the objective of combining a number of weak learners to a strong learner. The combination of multiple models lowers the chance of overfitting.

*Random Forest* adds additional randomness to the boosting model by taking a random subset of features into consideration at each node. The *Regularized Greedy Forest* is a rather novel method which makes use of the tree structures themselves by looking for the one-step structural change to the current forest that minimizes the loss function.

*Extremely Randomized Trees* usually outperform Random Forest in case of multiple noisy features present in a training set. The main difference of this model opposed to the Random Forest is the fact that the splits of the trees happen in a random rather than deterministic way.

### 2.3.3. Others

Further models implemented include the distance-based *K-Nearest Neighbor Classification*, the *Support Vector Classification*, which tries to find the best hyperplane for class-separation, as well as the probabilistic *Naive Bayes Classification.*

Also, we implemented single *Neural Networks* using Keras as well as a combination of Neural Networks in order to create a voter.

### 2.3.4. Ensemble Learning

Boosting models implemented include Adaptive Boosting Classifier, Gradient Boosting Classifier and Extreme Gradient Boosting Classifier. Adaptive Boosting Classifier(AdaBoost) is based on improving the areas where the base learner fails by exaggerating the weights of the previously misclassified data points. *Gradient Boosting* has been attracting attention, especially for its prediction speed and accuracy with complex data. The target is calculated based on the gradient of the error with respect to the prediction and each new model tries to minimize the prediction error. *XGBoost* is an implementation of Gradient Boosting, optimized for speed and performance.

In order to combine the strengths of multiple models, we used *Ensemble Models,* combining several model instances and then combining results via hard-voting or

soft-voting with weights based on average correlation scores (higher correlation: lower voting power) and accuracy.

Additionally, we implemented Bootstrap Aggregating *(Bagging)*, where multiple models are fitted on different samples (with replacement) of the population and then aggregated via a (weighted) average or voting system. Bagging reduces the interpretability of the model but creates a more robust model that should be better able to identify the underlying patterns of the data.

Model averaging can be improved by weighting the contributions of each sub-model to the combined prediction by the expected performance of the submodel. This can be extended further by training an entirely new model to learn how to best combine the contributions from each submodel. This approach is called stacked generalization, or stacking and can result in better predictive performance than any single contributing model.

## 3. Model Performance Evaluation

### 3.1. Hyperparameter Tuning Methodology

In order to tune the hyperparameters of the implemented models, we used three methods: Grid Search, Hill Climbing and Genetic Algorithms. Grid search is a naive method of hyperparameter tuning, where a pre-defined number of configurations is tested for optimal performance. The difficulty of the Grid Search approach includes the a-priori definition of parameter ranges, which are difficult to estimate as well as the capturing of value nuances.

More suitable for optimization methods that were used are Hill Climbing and Genetic Algorithms. Both methods represent a more efficient approach to hyperparameter tuning, but are also much more time-consuming to implement and optimize. Thus, we used Hill Climbing for the optimization of the Random Forest Classifier, because of assumption of unimodality of search space. Grid Search was considered better for the optimization of the hyperparameters of the other models due to parallelization and performance optimization.

Beside the models individual hyperparameters, we also inspected the change in average profit when using different probability thresholds for the classification. This means that we artificially set the classification threshold on the prediction probability for class 1 to values other than the pre-set 0.5. A-priori, we hypothesized that the setting of a threshold to a lower value than 0.5 could be beneficial, because based on the profit/loss values, the inappropriate classification of more than two class 0 members (False Positives) can be outweighed by a single correct prediction of a class 1 member (True Positive) . Thus, it seemed valuable to classify more data points as class 1 overall.

### 3.2. Results

Hyperparameter tuning on the separate models using GridSearch improved performance on a great majority of models on both profit per capita and F1 macro score, which can be seen in Figure 5. It is worth noticing that the model which benefited the most from hyperparameter tuning was the simple Decision Tree, followed by the Logistic Regression.

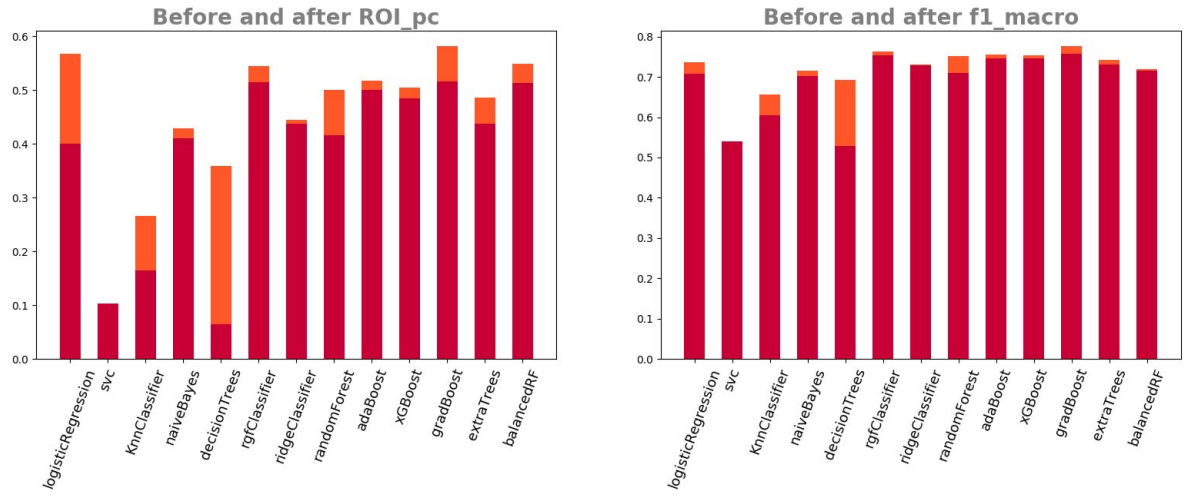


*Figure 6: Profit per capita, F1_macro per capita before and after hyperparameter tuning*

The Bagging approaches, which were mainly targeted at building a more robust classifier, in fact improved variance on all models significantly. However, the mean performance in terms of accuracy and standard deviation decreased, as can be seen in Figure 7, thus not making this approach one of the most promising ones.

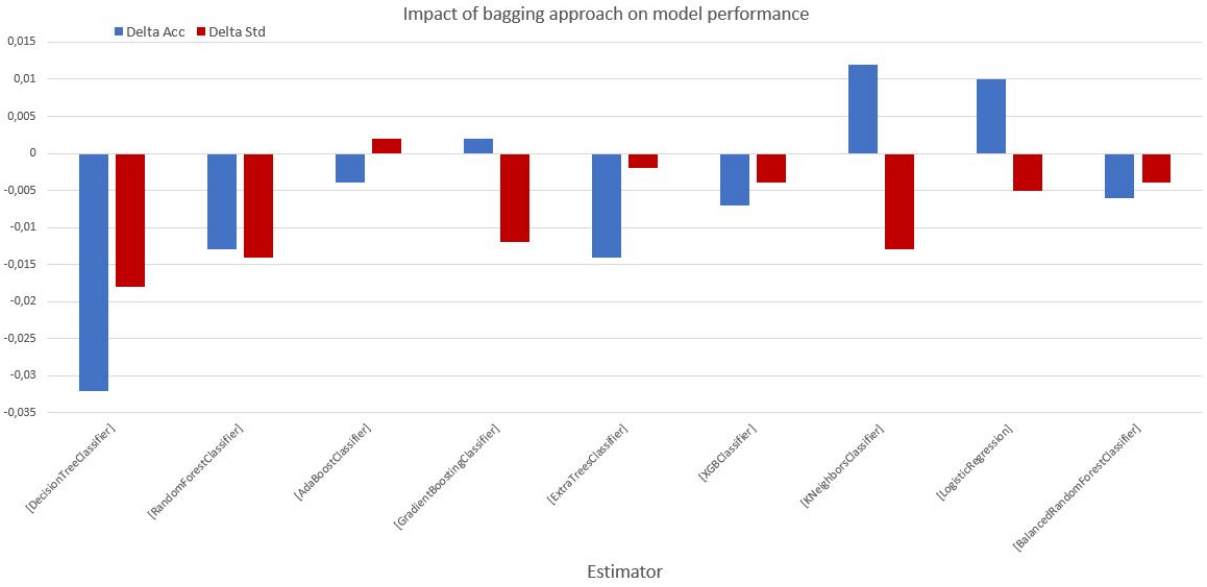


*Figure 7: Impact of Bagging on model performance*

For the ensemble voting approaches, we included the most promising models. Generally, the voting approach seemed promising and results seemed to confirm the co-functioning of multiple models as a stronger overall classifier. Equal weight voting performed worse than voting based on accuracy or correlation weights. Most interesting to us was the fact that for voting based on correlation weights, including the models with the lowest correlation with all other models indeed seemed to improve model performance significantly. This supports the hypothesis that different model errors balance each other and lead to a better overall classifier.

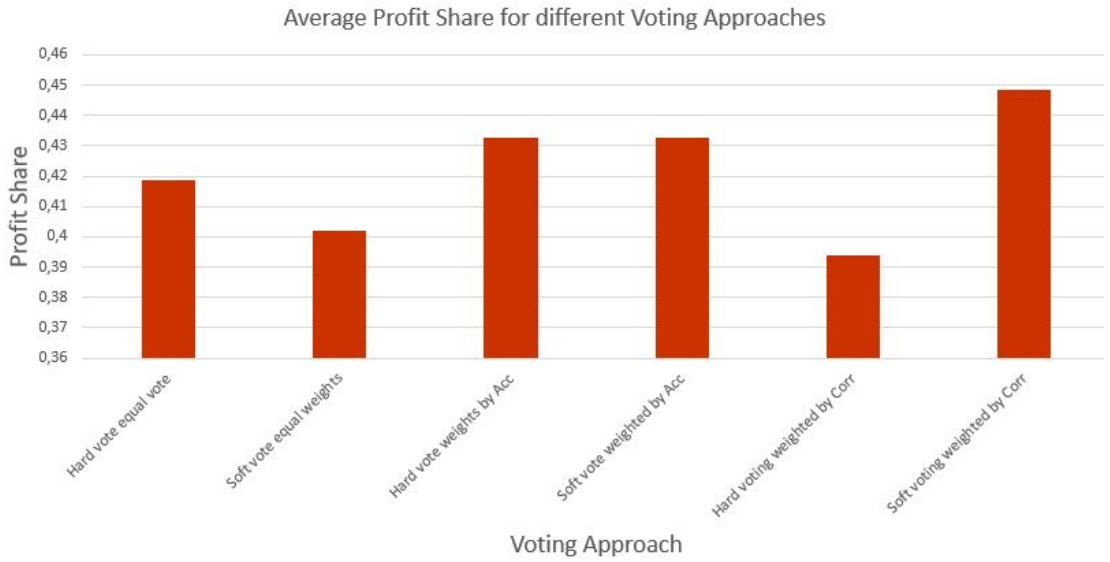


*Figure 8: Average profit share for different Voting Approaches*

For the Neural Network we tried different numbers of hidden neurons per layer, activation functions: relu and sigmoid, different dropout rates per layer, different regularizations, different optimizers: SGD, Adam, Nadam, RSMprop, different batch sizes and different number of epochs, as well as different scoring and loss functions, namly F1, weighted and not, as well as self defined profit share and profit per capita.

Moreover, with the results on the power of ensemble voting methods in the previous step, we combined three neural networks of different type to create a voter. The latter model turned out to give us the best (mean profit) and most reliable (standard deviation) results and was thus chosen as the final model.

Regarding the variation of the classification probability threshold for class 1 we could indeed find better model results, based on total profit, for a probability threshold of 0.4 or even 0.3. This was generally true for all models that return a prediction probability. For the final voting model of

Neural Networks, we tried lowering the threshold for the voting as well, but the model did not improve.

## 4. Final Model

Our final model is an Ensemble of Neural Networks in a majority voting scheme. This estimator consists of three Feed Forward Neural Networks. All three have two hidden layers and use the relu activation function on all layers except the output layer where sigmoid is used. Two networks use the same Adam optimizer and one RSMprop. The batch size is 64 for all three while the number of epochs is 50 for two and 20 for one. Each output value (uncalibrated) is counted as 1 above 75% and 0 below. The final voting is based on simple majority (if two or three models vote 1 the final prediction is 1 and 0 otherwise. Over 5 different train test splits on 8:2 the average profit was 249 mnt with a standard deviation of 16 mnt. Please find all the details in the source code.

## 5. Interpretation of the class characteristics

From the kernel density plots some conclusions can be drawn, albeit they do not provide in depth knowledge. Customers are more likely to accept the campaign if they:

- Are long customers
- Have a higher income
- Spent a lot on Meat products in the last two years
- Have not purchased items recently
- Spend more than the average customer in the last two years

Further, customers are more likely to accept if they have ever accepted a previous campaign. Considering the total responses to previous campaigns, the customers who accepted the test campaign have on average accepted also one previous campaign.

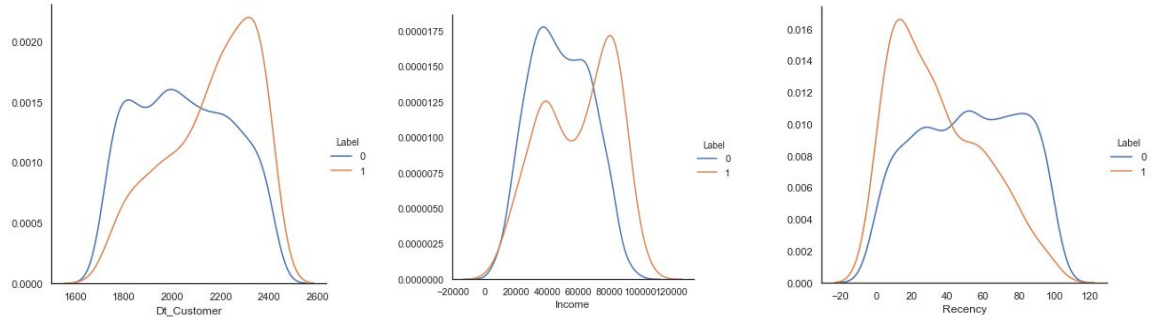


*Figure 9: Class-wise distribution of DT_Customer, Income and Recency*

## 6. Limitations

During the project, the main limitations we found was the optimization of hyperparameters. First of all, it was difficult to find appropriate ranges for a large amount of models, as combinations could not be tested extensively due to a lack of computational power and we mostly relied on online information and theoretical knowledge. Furthermore, the extensive testing of Neural Network and Stacking hyperparameter settings was difficult based on the package pre-set configurations and cross-validation settings.

Additionally, even though SMOTE sampling greatly improved our model's performance, it was difficult to handle in regards of metrics and cross-validation, as we did not want to evaluate too extensively on synthetic data.

Finally, time and computational power were also limiting taking into account the number of models and approaches that could be combined and explored more in depth for better results.

**Bibliography**


1. Antonio J. Tallón-Ballesteros, José C. Riquelme (2014)
2. Atkinson, A.C. & Mulira, HM. Stat Comput (1993)
3. Applied Multivariate Statistical Analysis by Johnson Wichern (2007)