Advanced Machine Learning — Project 2 Report

Maja Andrzejczuk, Jakub Kasprzak, Maciej Orsłowski

Contents

1 Methodology

Our approach for finding the best model for the specified task (i.e. accurate but based on a few variables) was divided into three steps. In the first step, we would evaluate different feature selection strategies on three to four models. In the second step, we pick the best strategy or strategies for each model, and for each, we optimise the model's hyperparameters. This way we obtain the final models, from which, as a third step, we pick the best one to make our final prediction.

The most important aspect of models evaluation is picking the right metric to evaluate models quality. This metric needed to be similar to the metric which will be used to grade the project, as this would ensure that we optimise our models in the most desired way for this project. Unfortunately, a metric specified in the task description was designed to work only for data with 5000 observations, and in our case, we needed a way to evaluate predictions on cross-validation splits of a size of 1000. That is why we came up with a metric, which we called the adaptive score. This score attempts to imitate the target metric for varying numbers of observations, hence its name. It does so by assuming the following:

- The company can send offers to max. 20% of the customers. This comes from the fact that in the task description, it would send offers to 1000 out of 5000 customers;
- For each customer that used the offer, the profit is equal to €10000 divided by the number of customers that we sent the offer to. This caps the profit at €10000, which is in fact the same maximum profit as in the task description;
- Since the number of features is still 500, the penalty for each used feature remains unchanged and is equal to €200.

The formula used to calculate this score is then the same as for the one defined in the task description.

In our project, we distinguish selection methods from selection strategies. We define selection strategies as combinations of different selection methods. The methods we implemented are as follows:

- Correlation-based method - finding pairs of highly correlated variables (according to Spearman correlation, threshold 0.8), and removing the one which is less correlated with the target variable (according to Kendall correlation);
- ANOVA;
- Forward selection;
- Backward elimination;
- Permutation-based feature importance.

Each of them is designed as a function, which accepts the data, the number of features to keep (except for correlation-based), sometimes the model, and sometimes also some extra parameters related to the method. These functions return a list of features to keep, which makes it easy to combine different methods to form selection strategies.

The reason for the distinction between methods and strategies is because most of the implemented selection methods are not feasible to be used on their own. Correlation-based method discards way too few variables, while forward and backward selections are too computationally expensive for a dataset with 500 variables.

2 Strategies

The first step involved designing and evaluating different selection strategies. Each strategy was tested on three to four models and run with different numbers of features to keep. These strategies are as follows:

1. ANOVA;
2. Correlation-based, and then ANOVA;
3. Correlation-based, then ANOVA to select 15 features, and then forward selection;
4. Correlation-based, then ANOVA to select 15 features, and then backward elimination;

5. Permutation-based;
6. Correlation-based, and then permutation-based;
7. Correlation-based, then permutation-based to select 15 features, and then forward selection;
8. Correlation-based, then permutation-based to select 15 features, and then backward elimination;

For models, we used Gaussian Naive Bayes, XGBoost, Random Forest, and Logistic Regression. However, the Logistic Regression often failed to train in methods which required training on numerous variables, namely permutation-based variable importance and backward elimination. This is why it was omitted for strategies from 4 to 8 inclusive. The remaining models were tested with every strategy.

For summarised results for each method, as well as full evaluation results for each strategy, please refer to Appendix A. The respective strategies IDs are as specified earlier in this section.

In the case of logistic regression, combining ANOVA with forward selection helped to make results more stable, as the score never went below €4500. For the other strategies, this pattern was very similar. However, the best results were achieved using strategies involving permutation VI. Interestingly, combining it with backward elimination significantly worsened the results. However, the other three strategies utilising permutation VI clearly outperformed all strategies involving ANOVA. When looking at those three best strategies, however, they all performed on a similar level. This, for example, means that using forward selection, which makes the selection process much more time-consuming, does not help to improve the results.

Comparing results between different models, the Naive Bayes algorithm performed the best, usually reaching adaptive scores of around €6500 to €7000 with the best strategies. The worst model was Logistic Regression, which usually scored between €4000 and €5000, however it worked only with some ANOVA-based strategies, which turned out to be inferior to permutation-based strategies. The other two models, Random Forest and XGBoost, performed similarly, usually achieving scores between €6000 and €6500 with their best strategies. While they performed worse than Naive Bayes at this stage, one has to keep in mind that these two models have a much higher potential when it comes to hyperparameter optimisation than Naive Bayes, meaning they could still produce even better results later on.

3 Hyperparameter optimisation

Based on the results obtained from the experiments in section 2, we selected the following model + feature selection strategy combinations:

- Logistic Regression — strategies 2, 3;
- Naive Bayes — strategies 6, 7;
- Random Forest — strategies 5, 6;
- XGBoost — strategy 6;

making up for **seven** combinations total. We decided to pick the best combination for each model, but picked two for the models for which the results were similar. We also usually omitted strategies involving forward or backward selection, unless they significantly improved results, due to their high computational complexity.

These combinations of strategy and model were then optimised using the Bayesian optimisation algorithm. Aside from hyperparameters for each model, we also optimised the number of selected features. The number of iterations in the algorithm was determined manually, based on the execution times of selection strategies and model training. The evaluation metric used was the adaptive score, averaged across 5 cross-validation splits.

This section does not discuss search spaces nor final values of hyperparameters other than the number of features to select. For details on model-specific hyperparameters, please refer to Appendix B.

3.1 Logistic Regression

Logistic regression was optimised in combination with two strategies: Correlation + ANOVA with and without forward selection. In the case of the strategy without forward selection, we set up the Bayesian optimisation to sample 10 random points from the hyperparameter search space, and then run 100 optimisation iterations. For the other strategy, due to higher computational complexity, we only sampled 4 random points and then run 20 iterations of optimisation. The search space for the number of features was from 1 to 9 in both cases.

Unfortunately, during the training of the second strategy, the logistic regression algorithm often raised warnings indicating that it had failed to train. This resulted in very poor results. However, the first strategy yielded significantly better results, and we ended up with the best logistic regression model reaching the score of €5140, with 2 variables selected.

3.2 Naive Bayes

The next model we optimised was the one that performed the best in the previous stage: the Gaussian Naive Bayes. The selection strategies we used in the optimisation process was correlation + permutation VI and correlation + permutation VI + forward selection. We arranged the Bayesian optimisation algorithm to first draw 4 random hyperparameter sets and then run 20 iterations of optimisation. The search space for the number of features to select was defined as $\{1, 2, ..., 9\}$.

The optimisation process for Naive Bayes did not improve its results from the previous stage. Still, it achieved very strong results, reaching the evaluation of €6890 using 3 variables.

3.3 Random Forest

For Random Forest, we optimised the model for two strategies with permutation VI, one with correlation and one without it, as both produced similar results in strategy evaluation. In both cases, the Bayesian optimisation first sampled 4 random hyperparameter combinations and then optimised them for 20 more iterations. The number of features to select was being chosen from 2 to 7.

Eventually both strategies produced similar results, however the model obtained by optimising the strategy with correlation and permutation VI reached the evaluation of €6890, which was around €100 more than in the case of the other strategy. The best model was trained on 3 features.

3.4 XGBoost

The final model we used and optimised was XGBoost. The strategy we chose to use for the hyperparameter optimisation process was correlation + permutation VI. Again, the optimiser first sampled 4 random hyperparameter values and then run 20 iterations of optimisation. The number of features to select was drawn from 2 to 7.

The final model used 2 variables and reached the CV score of €6840. This was a significant improvement compared to the previous step, which proves the significance of hyperparameter optimisation for XGBoost.

4 The final models

To select the best model, we compared three different top-performing models, each with parameters chosen using Bayesian optimization.

- **Naive Bayes Model** - the top 3 features were selected through correlation + permutation + forward selection
- **Random Forest Model** - the top 3 features were selected through correlation + permutation
- **XGBoost Model** - the top 2 features were selected through correlation + permutation

To ensure robust evaluation, for each model, the dataset was split into training and test sets 100 times using stratified sampling and varying random states. This method mitigates bias introduced by a single random split, providing a more reliable assessment of model performance.

The following metrics were calculated for each model: Accuracy, Precision, Recall, F1 Score, and Custom Adaptive Score. The average values of these metrics were computed to provide a comprehensive evaluation of each model's performance.

Metric	**Naive Bayes**	**Random Forest**	**XGBoost**
Accuracy	0.6313	0.6323	0.6002
Precision	0.6358	0.6356	0.6030
Recall	0.6313	0.6323	0.6002
F1 Score	0.6280	0.6298	0.5974
Custom Adaptive Score	€6887	€6501	€6618

Table 1: Average Performance Metrics for Each Model

It's surprising that the Naive Bayes model outperforms the Random Forest model in Custom Adaptive Scores despite having comparable or slightly lower traditional metrics like accuracy, precision, recall, and F1 score. This discrepancy stems from how the Custom Adaptive Score is computed and the different priorities it places on various aspects of model performance.

The Naive Bayes model's strength likely lies in its ability to effectively identify the top 20% of customers who are most likely to respond positively to offers. This proficiency in targeting high-value customers can lead to increased earnings, significantly boosting its Custom Adaptive Score. To better observe which models perform best, we visualized the distribution of scores for each model using a boxplot:

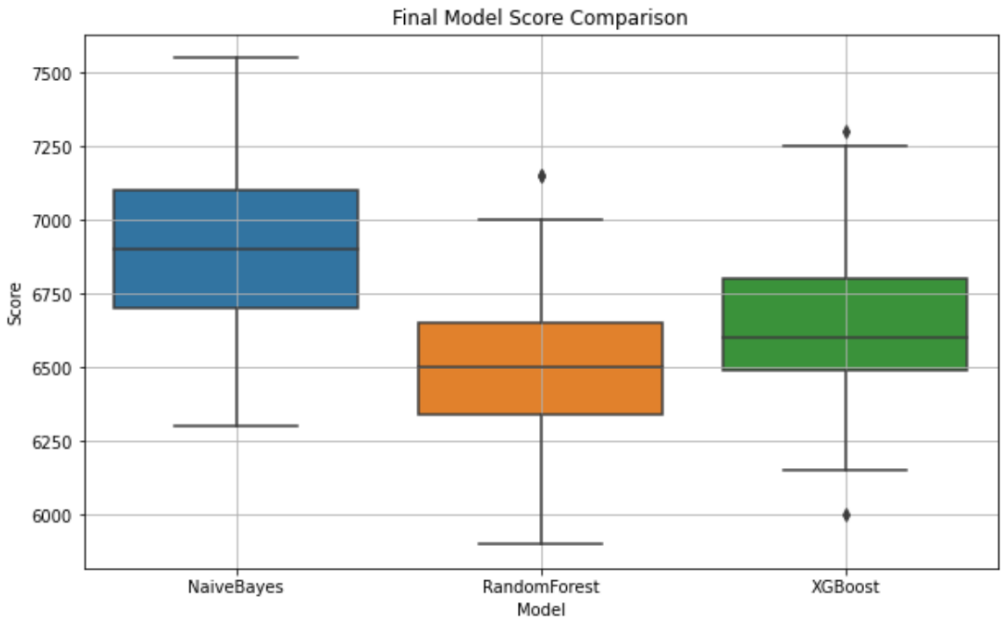

Figure 1: Evaluation of final models score

As shown in the table and box plot, the Naive Bayes model achieved the best results. It used three features: 102nd, 103rd, and 106th. Its scores ranged from approximately €6250 to over €7500, with an average of €6887. This thorough evaluation process ensured that the selected model not only performs well on average but also exhibits consistent performance across different data splits. Consequently, the Naive Bayes model was chosen for the final prediction due to its superior performance in the defined metrics.

A Appendix: Strategies evaluation results

This appendix contains graphs, which present the full results of strategies evaluation, discussed in section 2. Figures from 2 to 9 display detailed results for each strategy, while Figures from 10 to 13 show summarised results for each model tested.

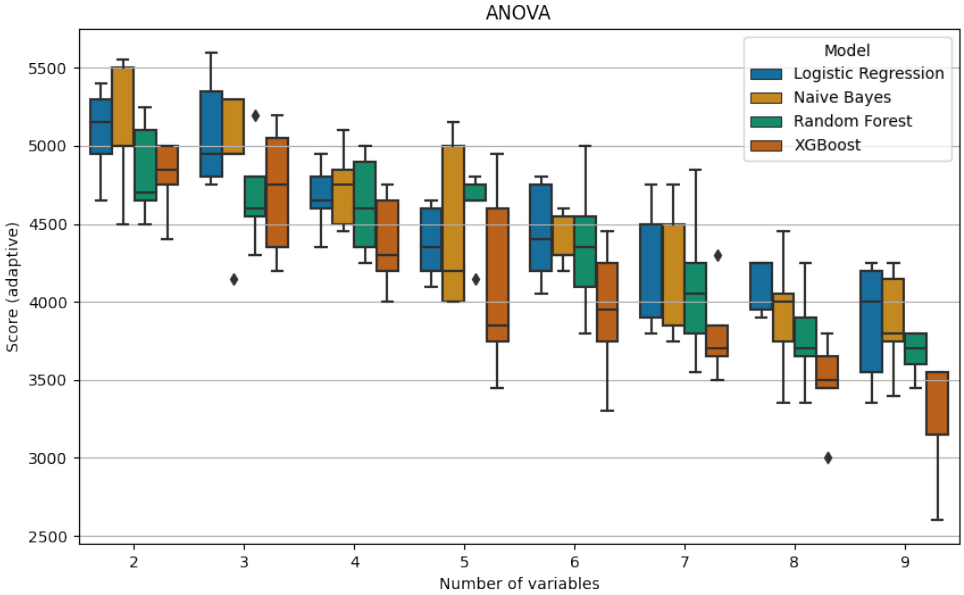

Figure 2: Evaluation of strategy 1: ANOVA

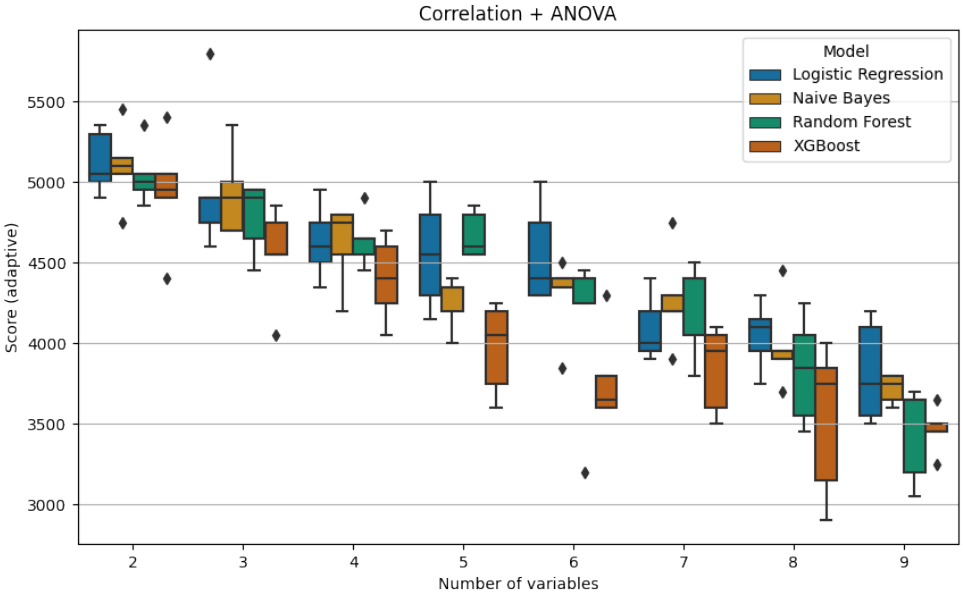

Figure 3: Evaluation of strategy 2: Correlation with target + ANOVA

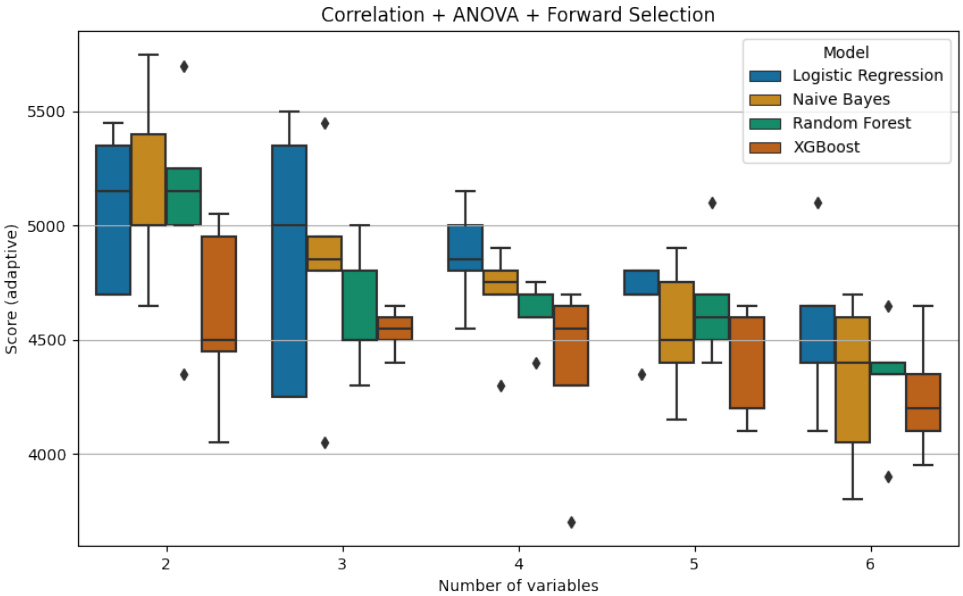

Figure 4: Evaluation of strategy 3: Correlation with target + ANOVA + Forward selection

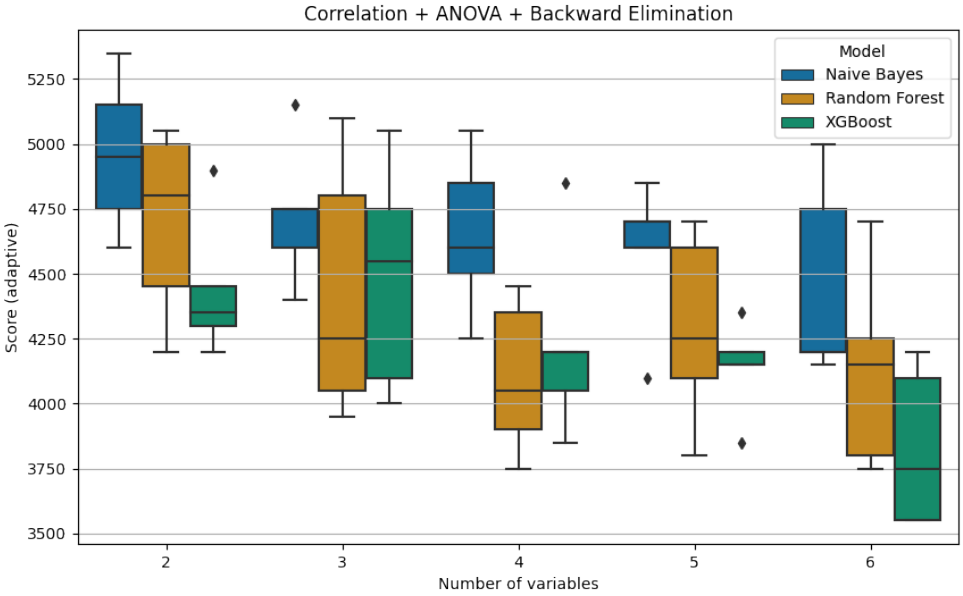

Figure 5: Evaluation of strategy 4: Correlation with target + ANOVA + Backward elimination

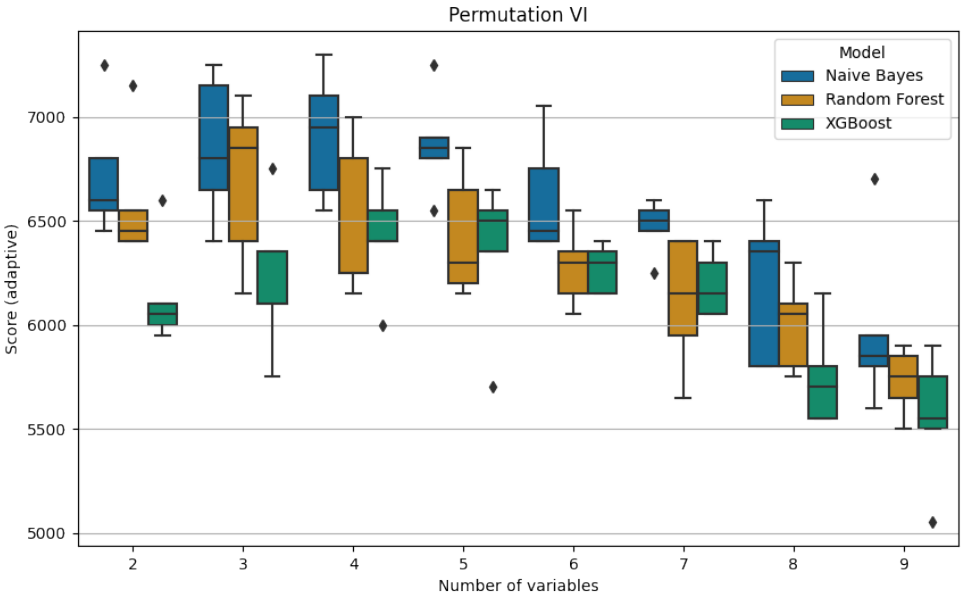

Figure 6: Evaluation of strategy 5: Permutation VI

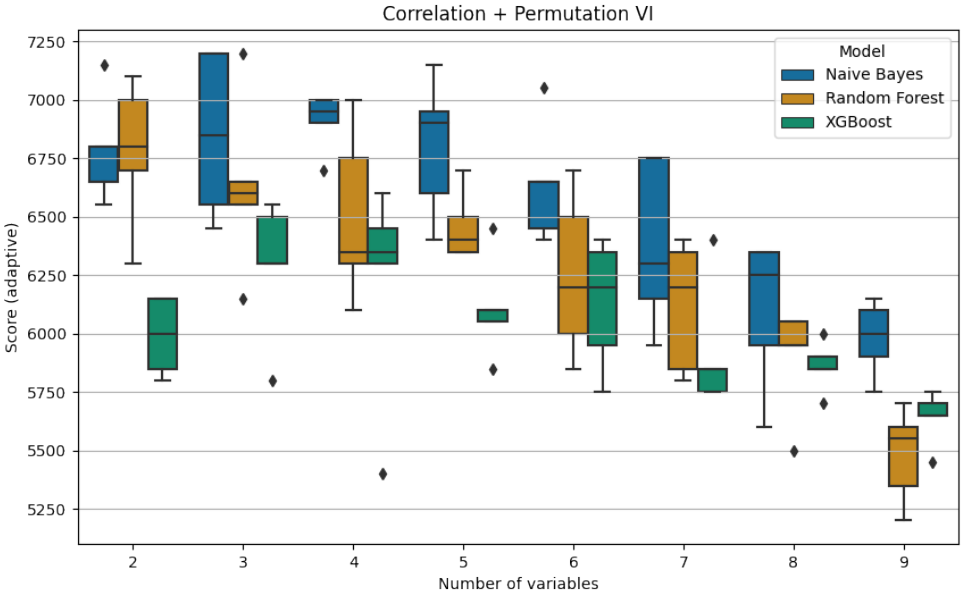

Figure 7: Evaluation of strategy 6: Correlation with target + Permutation VI

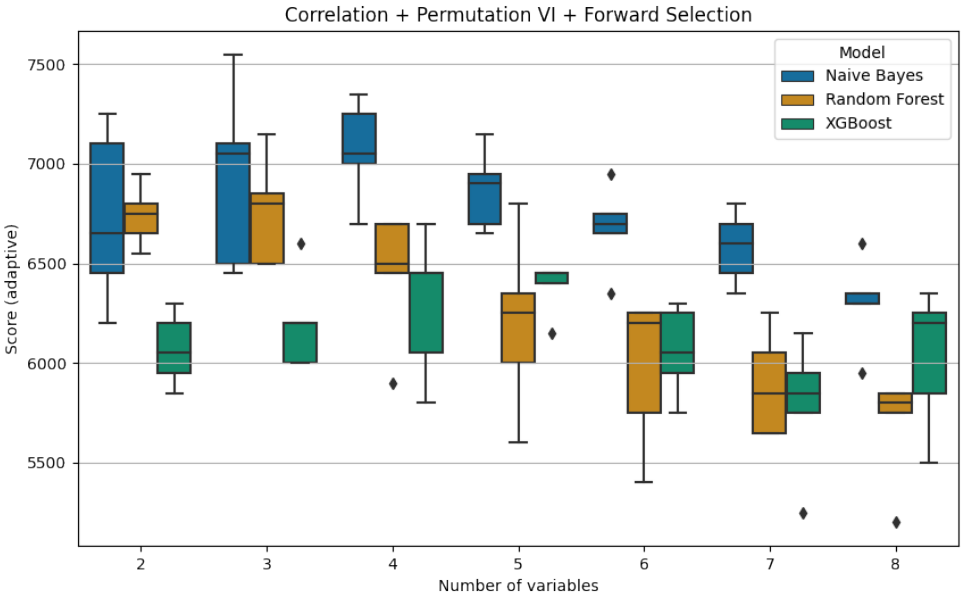

Figure 8: Evaluation of strategy 7: Correlation with target + Permutation VI + Forward selection

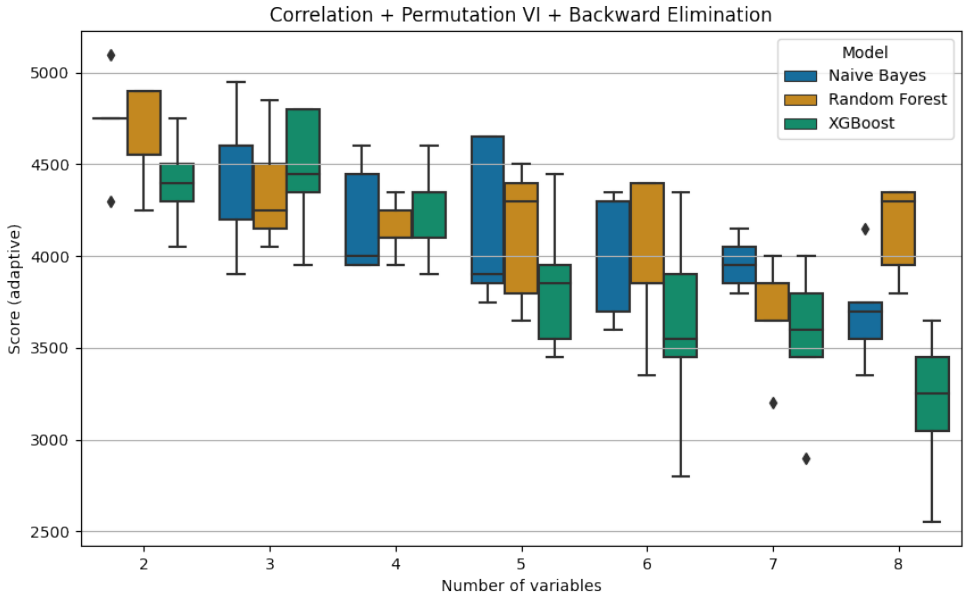

Figure 9: Evaluation of strategy 8: Correlation with target + Permutation VI + Backward elimination

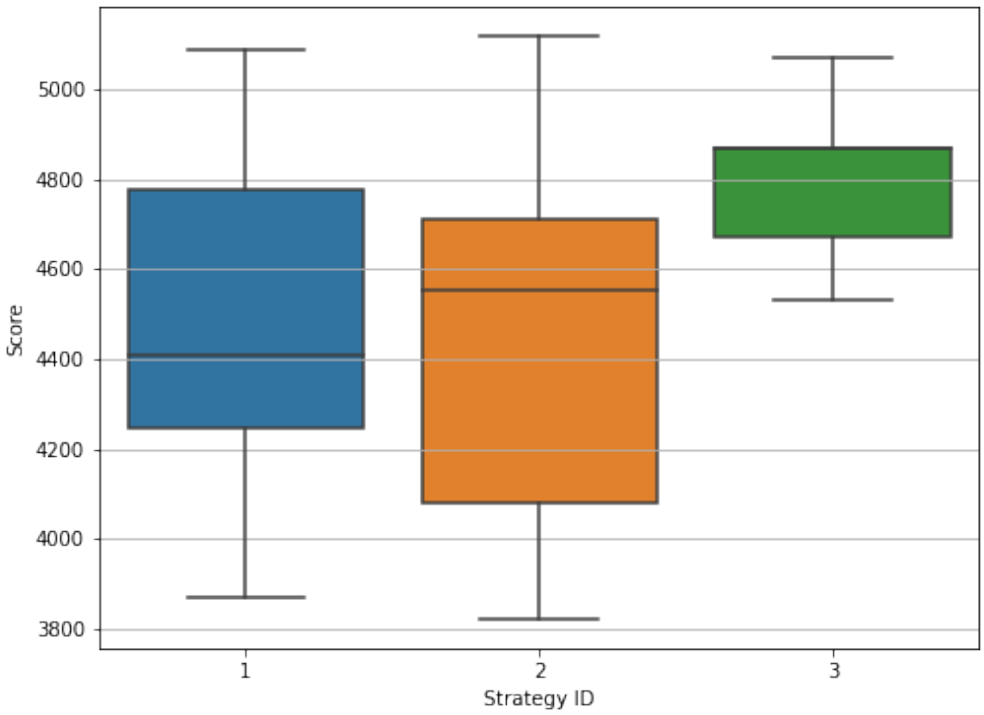

Figure 10: Evaluation of strategies for Logistic Regression

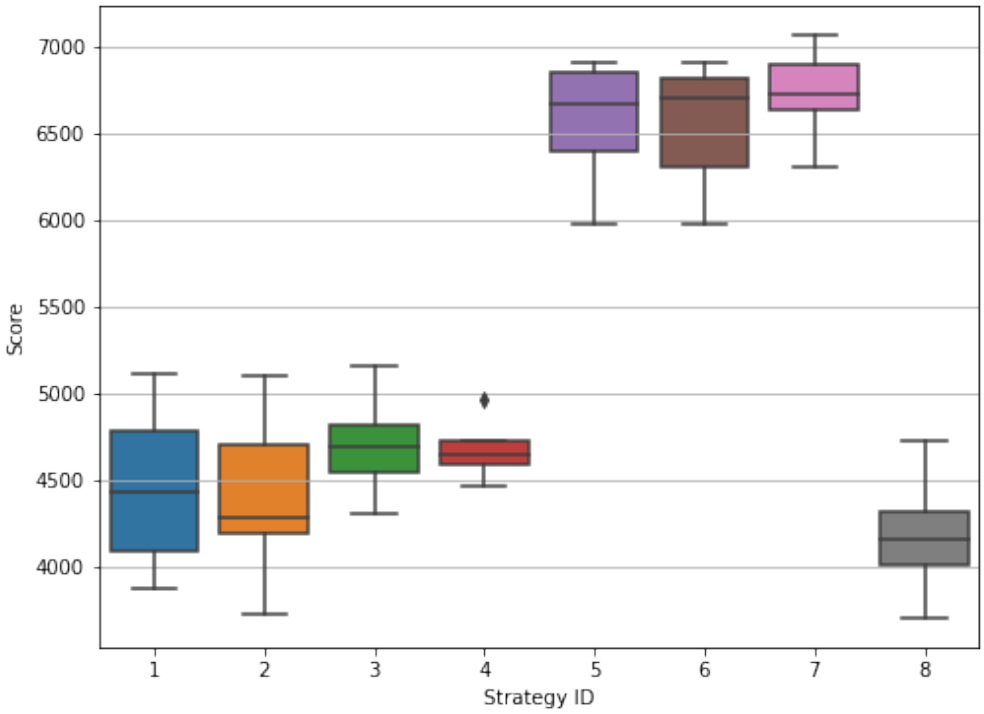

Figure 11: Evaluation of strategies for Naive Bayes

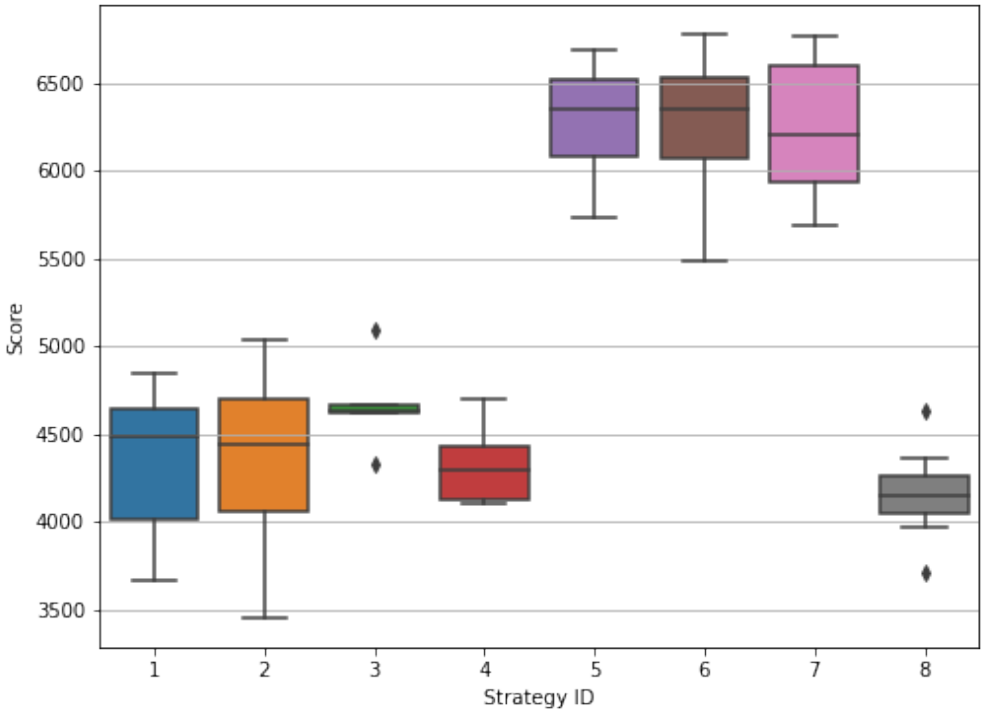

Figure 12: Evaluation of strategies for Random Forest

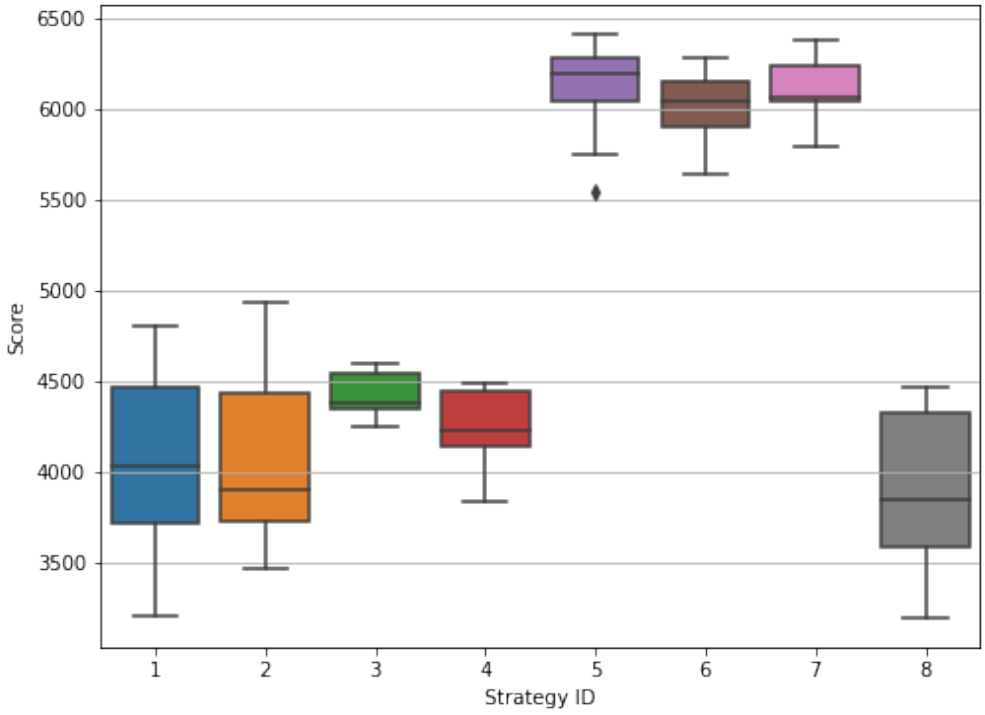

Figure 13: Evaluation of strategies for XGBoost

B Appendix: Details on hyperparameter optimisation

This appendix lists the hyperparameter values of the best models found through Bayesian optimisation for each model. The results are discussed in section 3.

B.1 Logistic Regression

Search spaces:

- number of features: $\{1, 2, ..., 9\}$;
- `C`: $[10^{-10}, 100]$.

Final hyperparameters:

- number of features: 2;
- `C` ≈ 15.6.

B.2 Naive Bayes

Search spaces:

- number of features: $\{1, 2, ..., 9\}$;
- `var_smoothing`: $[10^{-12}, 10^{-8}]$.

Final hyperparameters:

- number of features: 3;
- `var_smoothing` $= 10^{-8}$.

B.3 Random Forest

Search spaces:

- number of features: $\{2, 3, ..., 7\}$,
- `n_estimators`: $[1, 200]$;
- `max_depth`: $[5, 40]$;
- `min_samples_split`: $[2, 20]$;
- `min_samples_leaf`: $[1, 10]$.

The best hyperparameters:

- number of features: 3;
- `n_estimators` $= 178$;
- `max_depth` $= 5$;
- `min_samples_split` $= 20$;
- `min_samples_leaf` $= 1$.

B.4 XGBoost

Search spaces:

- number of features: $\{2, 3, ..., 7\}$,
- `n_estimators`: $[1, 200]$,
- `learning_rate`: $[0.01, 0.3]$,
- `max_depth`: $[3, 40]$,
- `min_child_weight`: $[1, 10]$,
- `subsample`: $[0.5, 1]$,
- `colsample_bytree`: $[0.5, 1]$,
- `gamma`: $[0, 5]$

The best hyperparameters:

- number of features: 2;
- `n_estimators` $= 38$;
- `learning_rate` $= 0.01$;
- `max_depth` $= 39$;
- `min_child_weight` ≈ 9.99;
- `subsample` $= 0.5$;
- `colsample_bytree` $= 1$;
- `gamma` ≈ 2.83.