University of St. Gallen

School of Economics, Law, Social Sciences, International Relations and Computer Science

CAIML Group Project:

Maximising TelCo Company Profits With Customer Churn Models

By

Tobias Jucker, Jingxuan Song, Mathias Steilen, Dan Wang

Matriculation N°: / 22-602-593 / 19-608-512 / 22-621-452

Course: Concepts, Applications, and Implications in Machine Learning and Artificial Intelligence

Lecturer: Prof. Dr. Clemens Stachl

28.11.2022

Contents

List of Figures

List of Tables

1 Why is it useful to predict customer churn?

TBD: Tobias

(*Comment*) Notes what I think should be mentioned in this section, feel free to mention what you think is important as well: - Customer churn -> target customers that want to leave and retain them -> often less costly than marketing efforts to gain new clients (find quote online that said something about 9x more expensive or something) - Directly affects the bottom line of a company (that is profits) - Especially important in competitive industries like TelCo with price competition

2 Packages and Data

2.1 Packages

The packages used for this project include `Tidymodels` for modelling; `Tidyverse` and `Broom` for data wrangling; `doParallel` for parallelisation of hyperparameter tuning; `vip` for variable importance plots; `stacks` for creating linearly stacked model; `themis` for dealing with class imbalance. Smaller, less relevant packages not directly tied to model output and evaluation were not separately listed. This final paper was written in `RMarkdown` and compiled using `knitr` and `tinytex`.

2.2 Data

TBD: Tobias

(*Comment*) Can you go into detail where the data came from? Links: Data Source: Kaggle More Info from IBM: IBM

Before the modelling process begins, certain steps have to be taken: Firstly, most dummies are encoded as categorical predictors, so the ones that are still encoded as binary variables are made consistent upon reading of the data. Additionally, there was one unnecessary level "No internet service" in six of the categorical dummy variables, which was changed to "No", as a one hot encoding of these variables would lead to perfect collinearity with the existing variable `internet_service` and is redundant. Additionally, character columns are converted to factors, as `Tidymodels` often requires factor columns, for instance for evaluation metric computation. The last step is removing missing variables, as there is only one variable with around 0.15% missingness. Given the absolute size of data, this is negligible and does not warrant imputation and extensive reasoning on the type of missingness, therefore it is just dropped. Importantly, none of the above steps lead to data leakage, as no metrics are computed on the aggregate data set and the shape and content of the data is not (negligibly) impacted by the operations.

3 Exploratory Data Analysis

Having introduced the origin of the data, the purpose of our modelling task and the initial preprocessing, we now turn to a short section on exploratory data analysis (EDA). Given the size limit for this paper, we limit our EDA to looking at both distributions and relations of nominal and numeric variables with the target variable in four plots.

Figure 1 shows the variable counts for all nominal variables, including the target variable. Notably, there is a considerable class imbalance in the target variable `churn`. In the preprocessing process, we use the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) which relies on creating new samples through k-nearest neighbours in the feature space to deal with this problem. `Tidymodels` provides great support for simple integration into the preprocessing pipeline with the `themis` package (*Apply SMOTE Algorithm*, n.d.).

TBD: Potentially go into why oversampling is necessary here: Models being biased towards majority class (cite paper?)

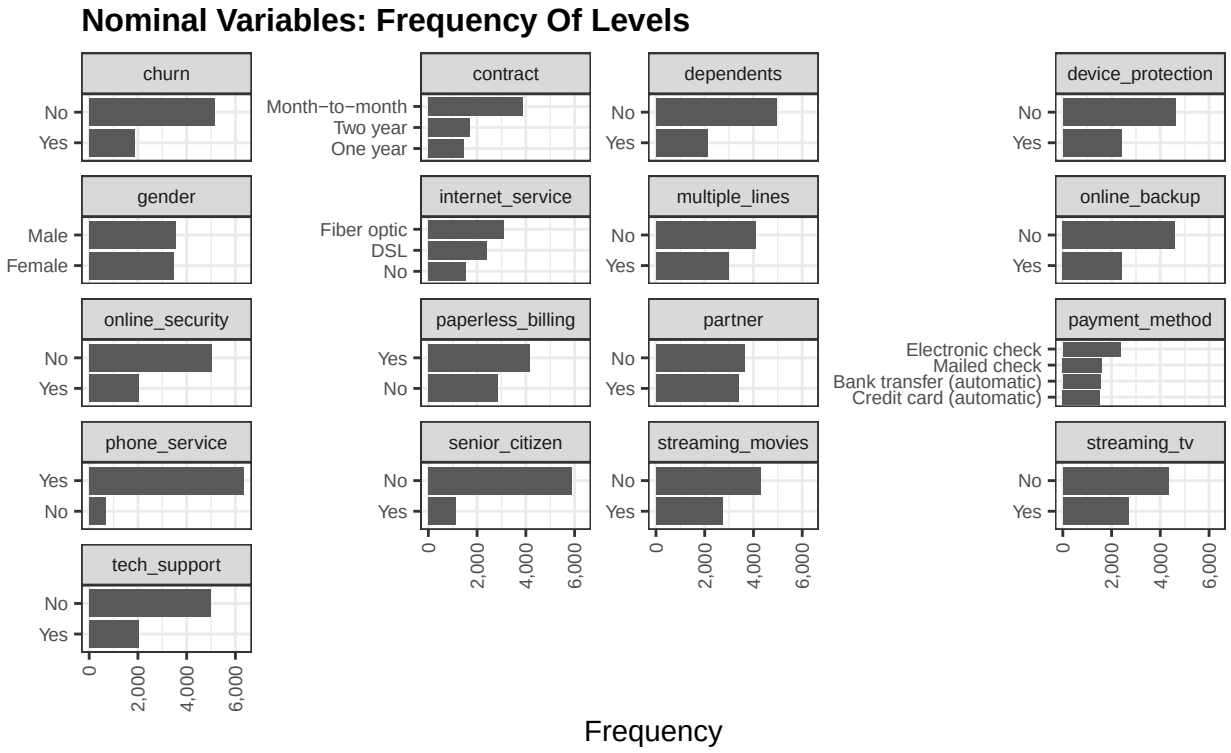


Figure 1: Counts of nominal variables

Similarly, Figure 2 shows the distribution of numerical features. It can be observed that monthly charges almost has a bimodal distribution, whereas tenure and total charges are mostly decreasing in time, which is likely a by-product of company growth, though tenure has a peak at its maximum value, which are clients that have been with the company since inception.

Next, the relationship of the predictors with the target variable `churn` is depicted in Figure 3 and

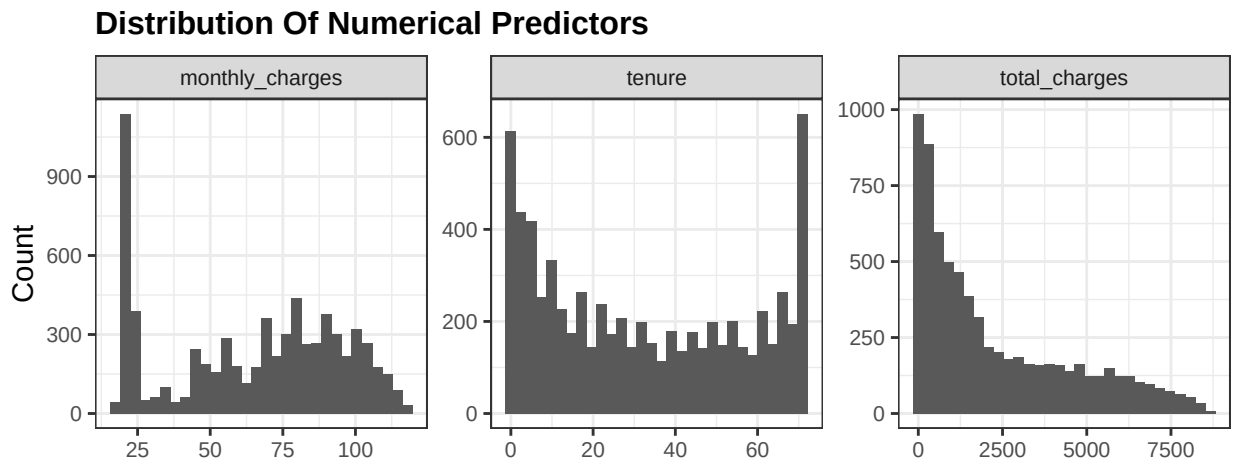


Figure 2: Distribution of numerical variables

Figure 4. Note that the variable of customer IDs has been left out, as it is a randomly generated categorical value with n levels, hence bearing no predictive power by definition. For the nominal variables, churn rates were computed for each level in each of the most important variables, which are shown in Figure 3. The main insights that EDA gives us here, is that customers with month-to-month contracts, which have no dependents, fastest internet service and electronic checks, as well as no additional support or security services are most likely to churn. These are likely signs of young, single customers like students or young adults that are highly price sensitive and more prone to changing providers due to their familiarity with technology. In contrast, customers that have no internet service and are generally more conservative, i.e. showing signs of being older, are less likely to change providers.

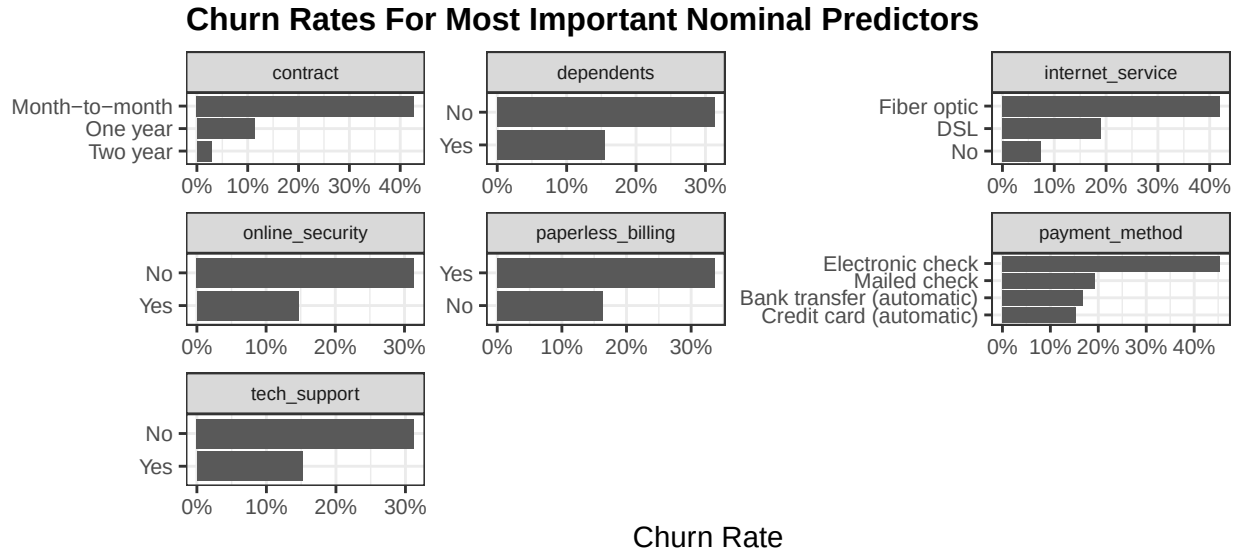


Figure 3: Churn Rates for different levels of nominal predictors

Figure 4 shows the distribution of numeric predictors and the target variable in scatter plots. The float variables monthly charges and total charges have been summarised into bins and average churn rates were calculated on them. Tenure, as an integer variable, did not require this transformation. For tenure and total changes, there is a clear negative relationship with the target. The latter is likely highly correlated with tenure, as monthly subscription models lead to linear growth of total charges in time. Monthly charges does not reveal a linear relationship: In general, it looks like there is an positive relationship, but there are exceptions at 60 USD and beyond 100 USD. Clearly, for this variable, there will be a benefit of using more flexible methods over a logistic regression for instance.

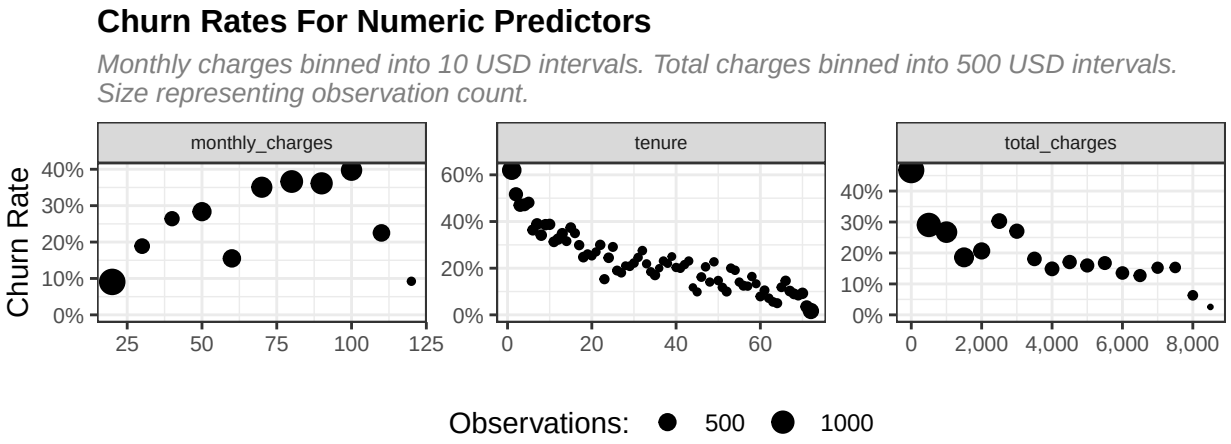


Figure 4: Churn rates associated with binned numeric predictors

A general note on dimensionality reduction: Given that there are only 19 predictors and over 7,000 observations, feature selection will likely not be necessary, as the data is quite long, implying that there will not be problems of dimensionality for linear models. Additionally, the nominal variables that were not shown above for space reasons also seem to explain some variance in the outcome variable, therefore the trade-off between the number of features, i.e. model complexity, and model performance does not seem to be necessary to accept.

4 Building, Tuning and Training Six Different Models

The first step in our modelling approach is creating the splits.

- Creating splits: stratified random splitting with a ratio of 0.75 train/total
- Go into benefits of stratified cross validation (cite paper)
- 5-fold cross validation, also stratified
- Preprocessing (go deep into benefits of tidymodels for preprocessing, cite official documentation, cite talk of Julia Silge and Max Kuhn)

  - step_novel(all_nominal_predictors()): allowing for novel factors levels
  - step_normalize(all_numeric_predictors()): normalising predictors (cite paper that shows benefit of doing this for linear models, also show benefit for variable importance calculations for tree-ensembles (prevents inflation of VI scores for higher value predictors))
  - step_dummy(all_nominal_predictors(), one_hot = TRUE): necessary for XGBoost and linear models, and random forest is indifferent (is it -> cite), so just going with one recipe, one_hot not for linear models due to multicollinearity
  - step_zv(all_predictors()) -> removing zero variance predictors, in case there are any (there shouldn't be)
  - step_smote(churn, skip = TRUE)

Model specifications: - boost_tree(trees = 1000, tree_depth = tune(), min_n = tune(), loss_reduction = tune(), sample_size = tune(), mtry = tune(), learn_rate = tune()) - nearest_neighbor(neighbors = tune()) - rand_forest(mtry = tune(), trees = tune(), min_n = tune())

- logistic_reg(penalty = tune(), mixture = tune())
- svm_linear(cost = tune())
- svm_rbf(cost = tune(), rbf_sigma = tune())
- Go somewhat into these models and on what packages they're based on behind the tidymodels interface (xgboost, ranger, kknn, glmnet, kernlab)

Hyperparameter tuning: Cross validation, parallel processing

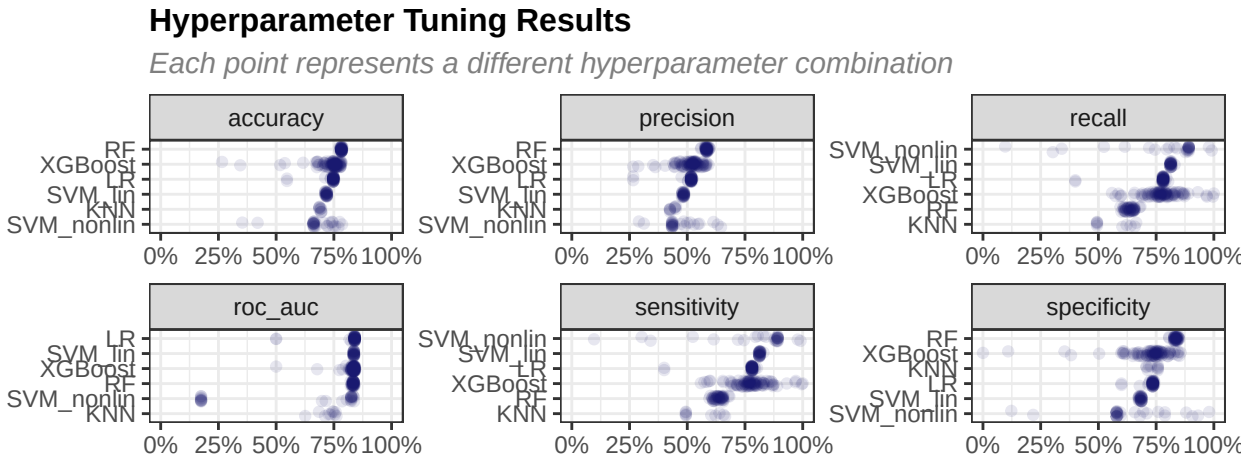


Figure 5: Hyperparameter tuning results for various evaluation metrics

Fitting the final on the training data, selecting the best model by roc_auc - Why ROC AUC? (cite some paper)

5 Model Performance

Variable importance (only for RF due to space reasons):

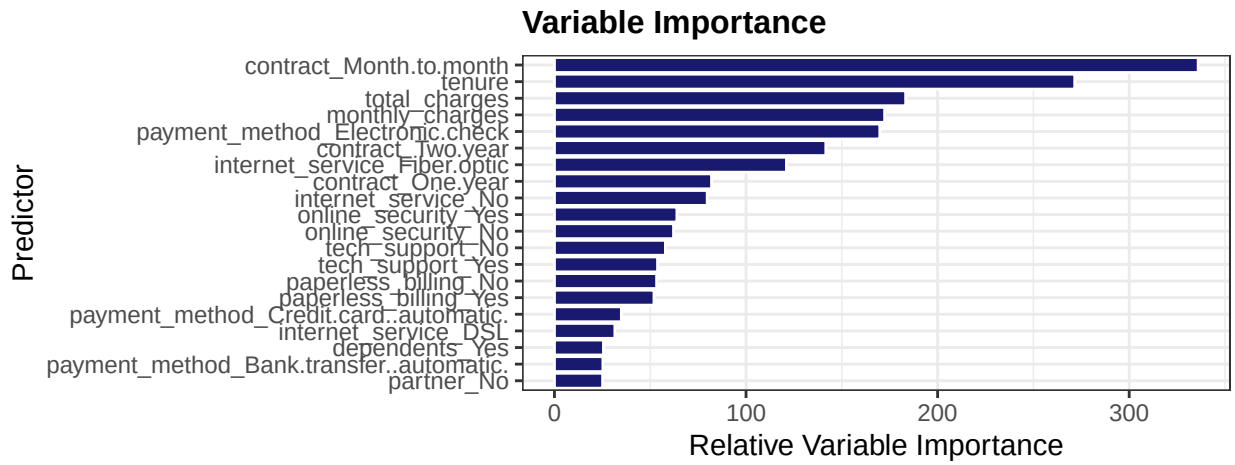
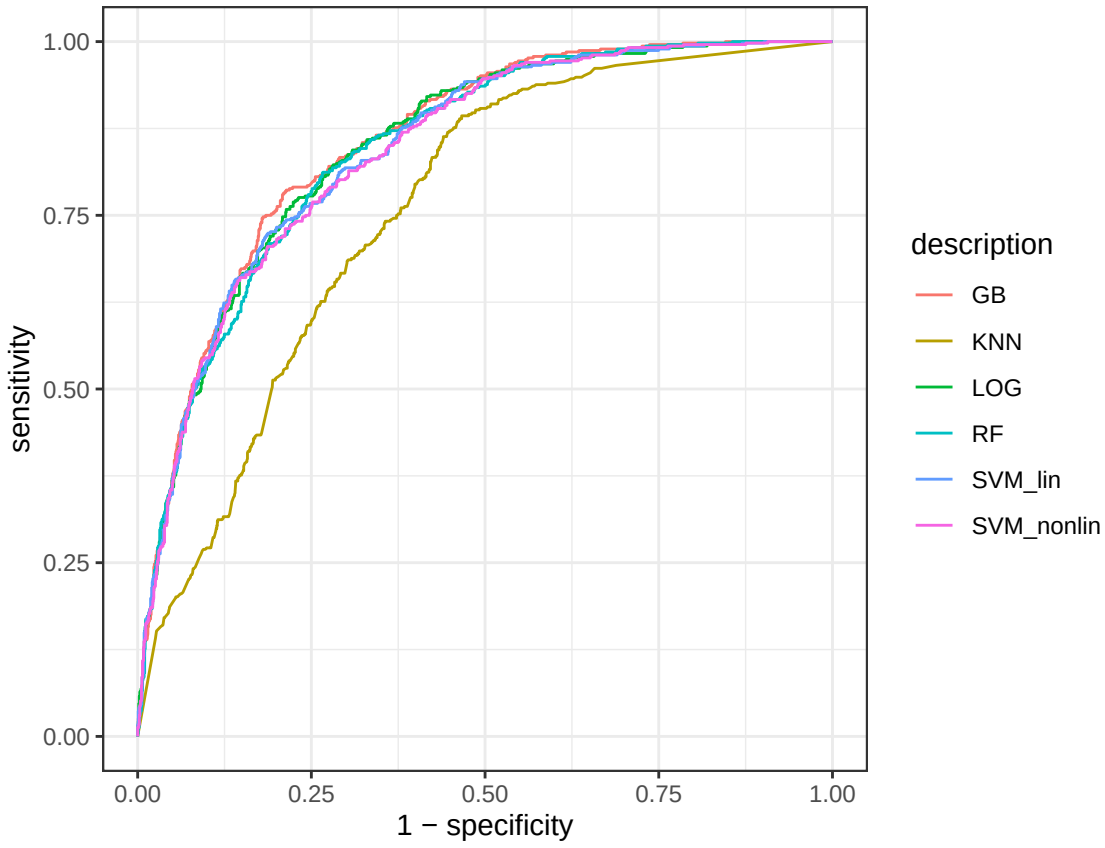


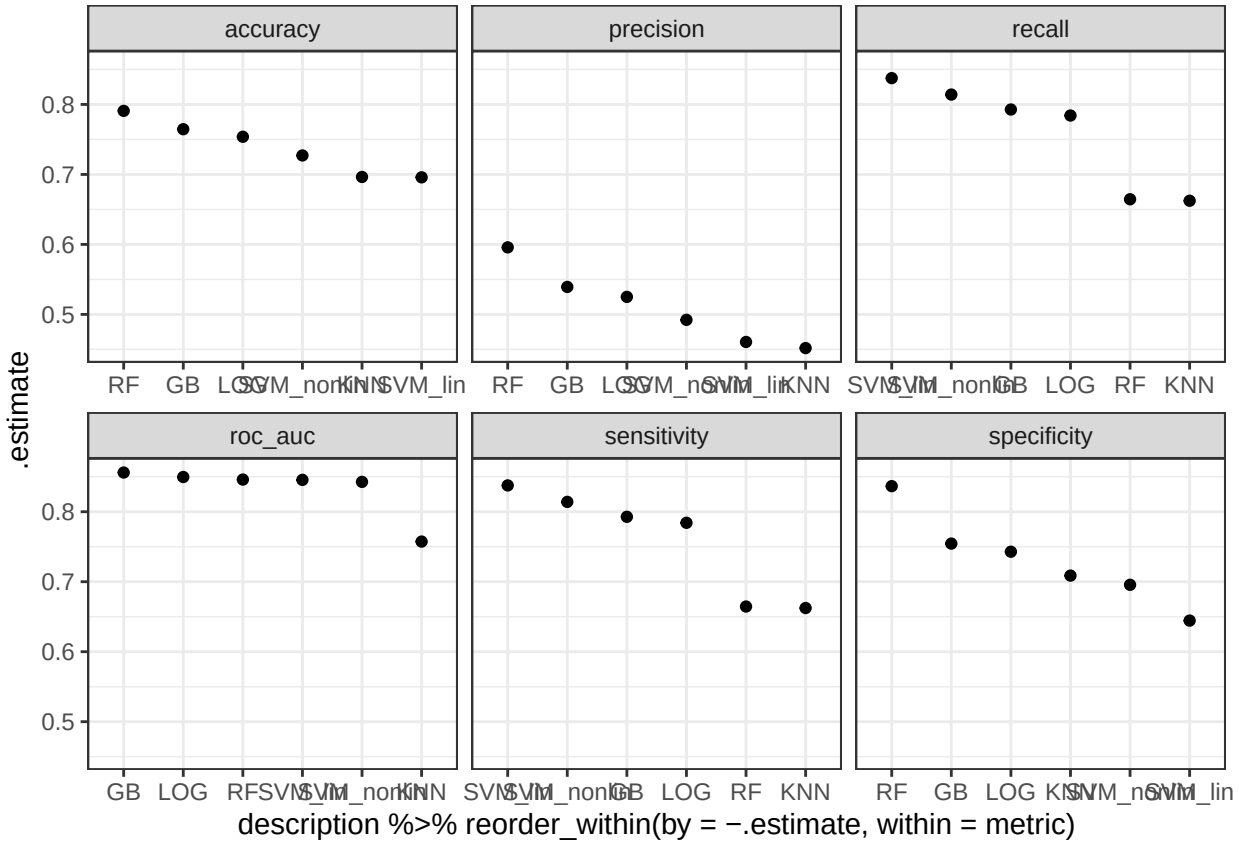
Figure 6: Variable importance for random forest model

Evaluating performance on training data:

- ROC AUC



- Out of sample metrics:

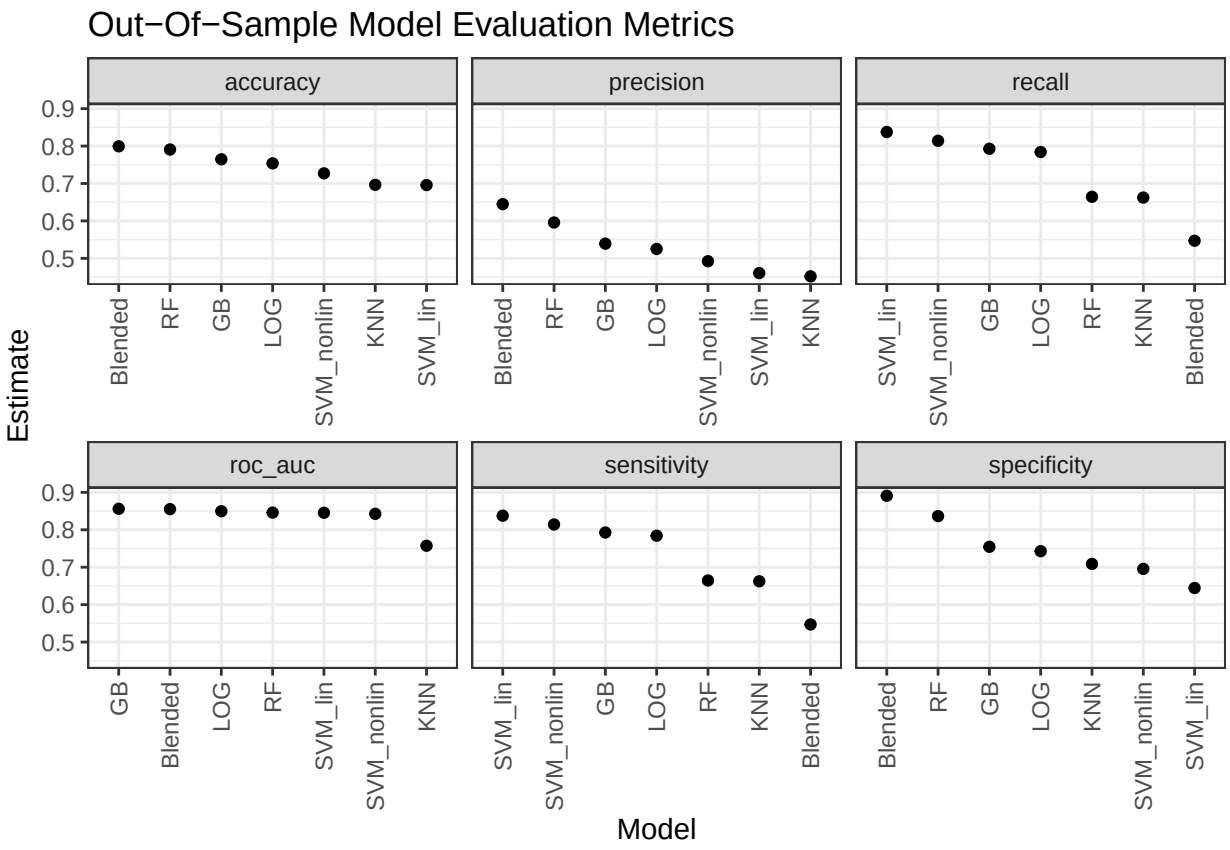


- Confusion matrices: Probably no need to include

```
##           Truth
## Prediction Yes  No
##        Yes 371 317
##        No   97 974
```

```
##           Truth
## Prediction Yes  No
##        Yes 310 376
##        No  158 915
```

- Digression: Creating a stacked model with the stacks package



6 Business Case

The real question following the above is: How can we use any of the above to create business value? That's were the gain curve comes in.

Classification models work by assigning probabilities of each class to any individual observation. For instance, one specific customer might be categorised with 75% probability of churning and 25% probability of not churning. In that case, the model would predict the customer to churn, as the probability is higher than 50%. This can be seen below. The XGBoost model predicted around 70% probability of the customer churning and around 30% of them not churning based on the available variables.

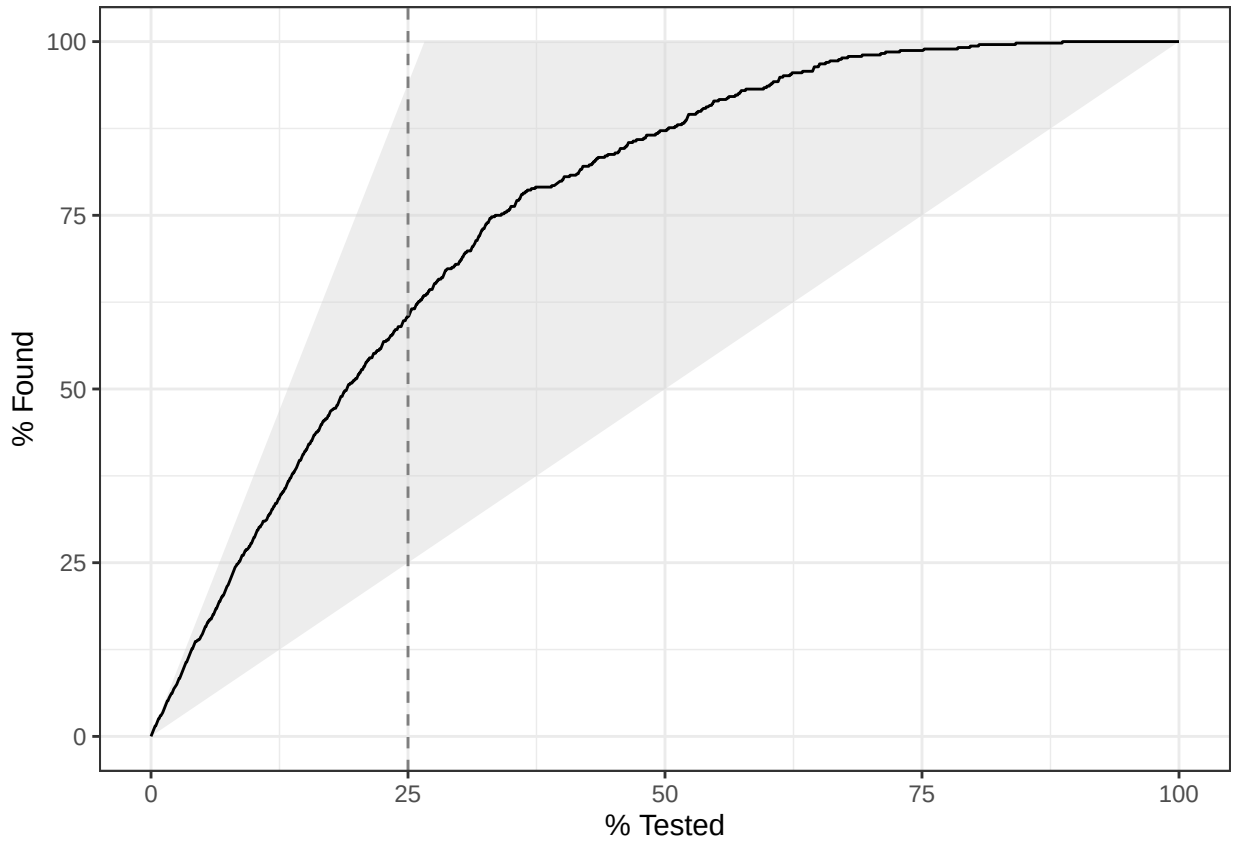
```
## Rows: 1
## Columns: 24
## $ customer_id       <fct> 4124-MMETB
## $ gender            <fct> Male
## $ senior_citizen    <fct> No
## $ partner           <fct> No
## $ dependents        <fct> No
```

```
## $ tenure            <dbl> 22
## $ phone_service     <fct> Yes
## $ multiple_lines    <fct> Yes
## $ internet_service  <fct> Fiber optic
## $ online_security   <fct> No
## $ online_backup     <fct> No
## $ device_protection <fct> No
## $ tech_support      <fct> No
## $ streaming_tv      <fct> Yes
## $ streaming_movies  <fct> Yes
## $ contract          <fct> Month-to-month
## $ paperless_billing <fct> Yes
## $ payment_method    <fct> Credit card (automatic)
## $ monthly_charges   <dbl> 94.65
## $ total_charges     <dbl> 2104.55
## $ churn             <fct> Yes
## $ .pred_class       <fct> Yes
## $ .pred_Yes         <dbl> 0.7002382
## $ .pred_No          <dbl> 0.2997618
```

In a business setting, a logical consequence for a model as reliable as this one predicting a customer to churn would be to target said customer with a retention programme, for instance via selected benefits only attributed to customers at risk of churning (e.g. coupons, discounts etc.). However, it would likely not be economically viable to target all customers that have a greater than 50% chance of churning, as it would most likely be a waste of resources. If a customer has only a 50.00001% chance of churning, according to the model, and assuming that the model is right, then the customer should not deserve a discount equal to the one given to a customer with >90% probability of leaving. After all, there is an around 50% chance that the first might not intend to leave in the first place. Then the discount would be wasted.

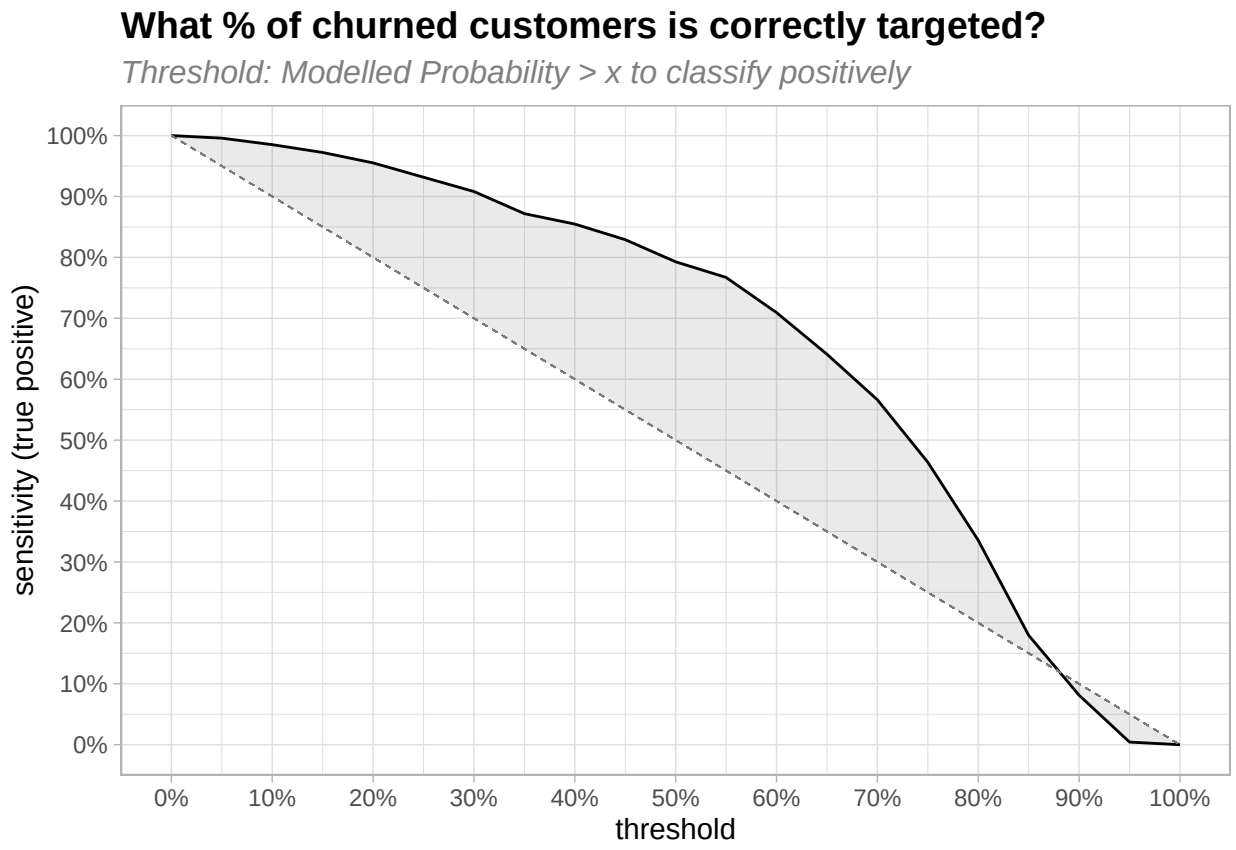
The business only wants to give costly retention programmes to customers that are at a high risk of leaving, not to the ones who were more likely going to stay anyway.

Therefore, businesses must find a threshold: Where do you set the minimum probability proposed by the model to classify a customer as *at risk of churning*? There exists an inherent trade-off in wanting to prevent customers from leaving the business, and not wanting to accumulate costs giving out retention programmes to many clients, who were not at high risk of leaving. As an example, the bank might decide that it targets the top 25% of customers with highest probability. This can be visualised with a gain curve, which comes with the *tidymodels* package in R:

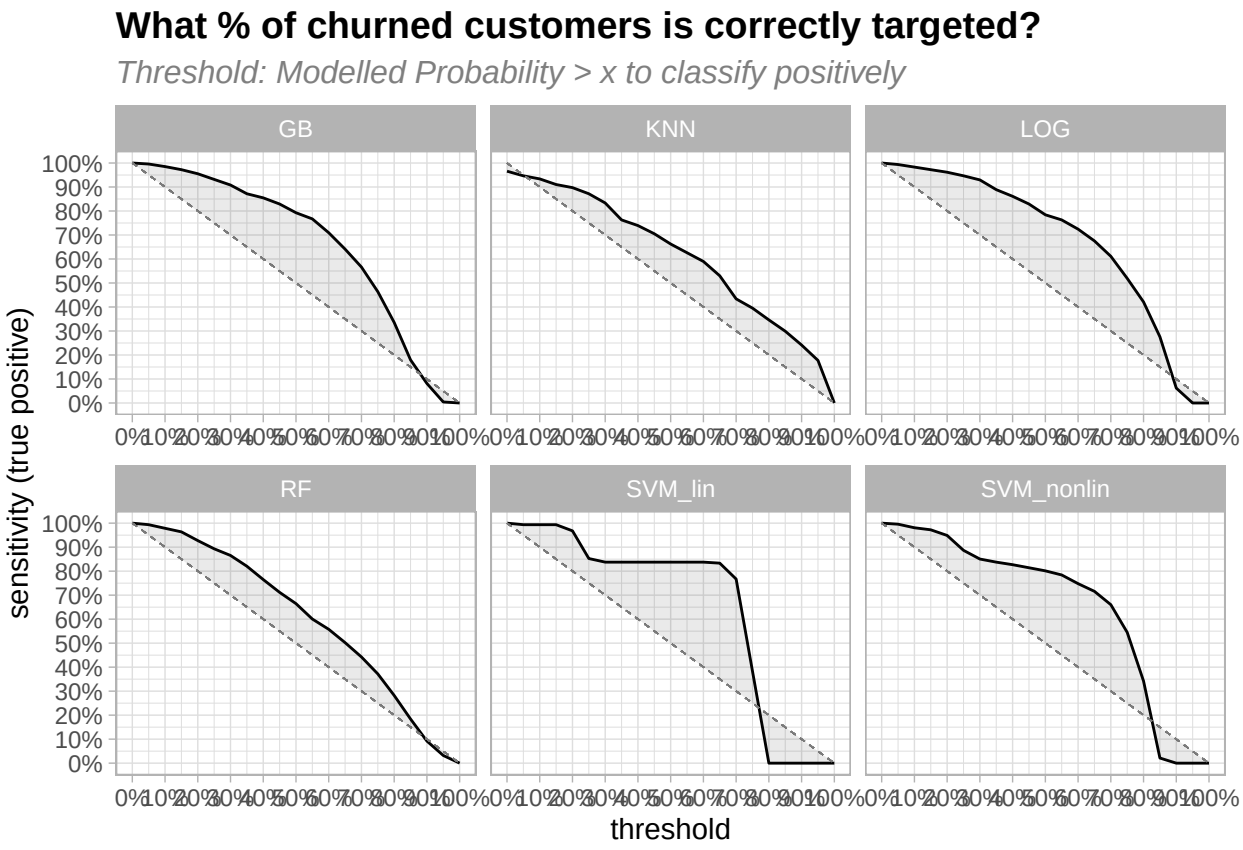


The curve can be read the following way: If targeting the x customers with highest probability of leaving, how many % y do we get right of the customers who will actually leave? In the case of targeting the 25% of customers with highest modelled probability of leaving, we would get close to but not 100% of the customers with an intention of leaving right. The curve being very close to the upper edge of the grey area indicates that the underlying model works very well.

The gain curve is really useful and quick to make in R, however it only says "target the 25% of customers with highest probabilities". This being dependent on the customers that predictions are being made on, I wanted to create a function that says "target only customers that have a probability x of leaving or higher". This can be seen here (Code for the function call on the chart can be inspected by clicking the "Code" button to the right below):



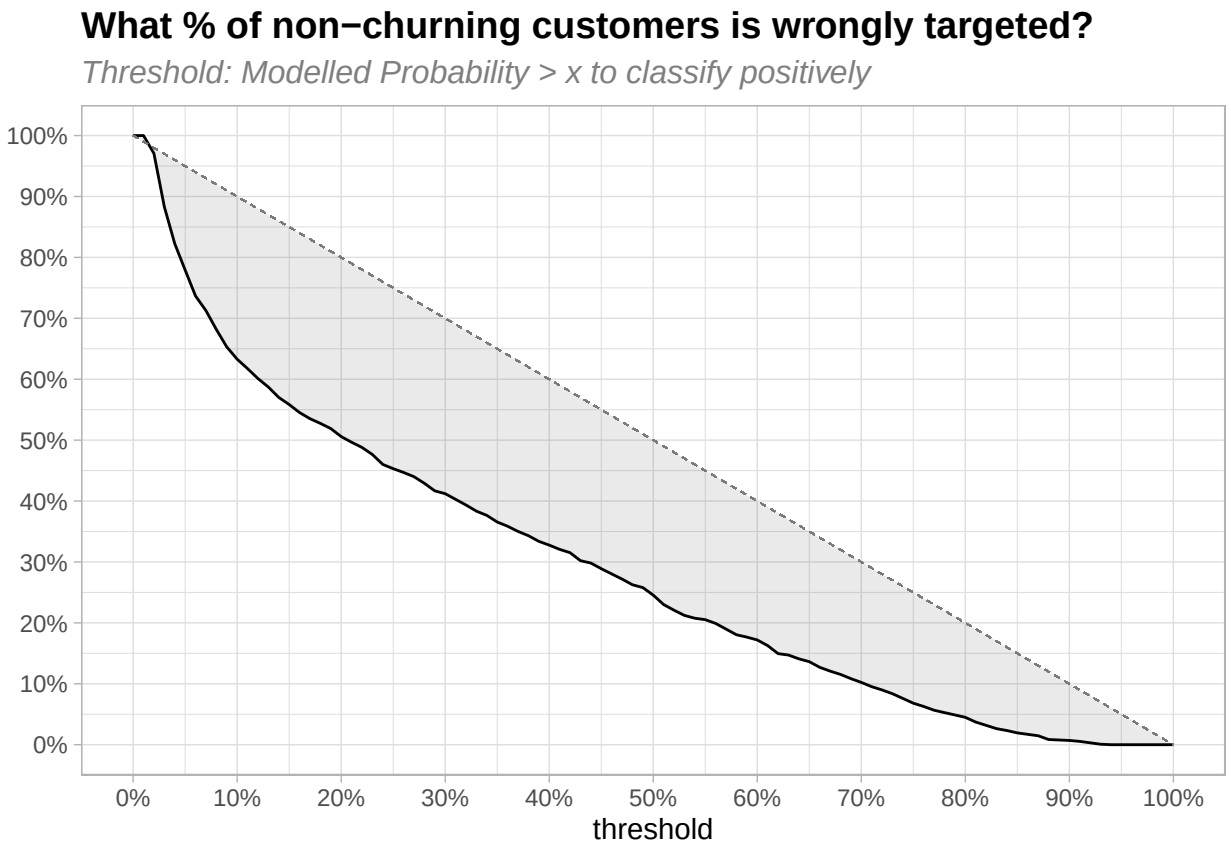
Doing this for all models:



This curve now demonstrates the real trade off of setting the classification threshold better. At 50%, the default threshold for the model to classify clients above 50% as *will churn* and below 50% as *will not churn* has a higher sensitivity. However, at the same time, we are classifying more clients on an absolute level as *will churn*. Therefore, there are likely also more clients in our predicted *will churn* class, that are not actually going to leave us. Remember, we didn't want to spend additional money on them, as they are not going to leave.

Therefore, while we, as the business, want to maximise the number of churning customers we target with retention programmes, we also want to minimise the non-churning customers we wrongly give the coupons/discounts.

This can be seen below: With an increasing threshold, i.e. the "stricter" we make our model, the fewer % of actually non-churning customers we target with coupons that were designed for the customers at risk of churning.



With this curve, we can look at an actual business case: Let's make some quick and dirty assumptions about profit generated per customers.

- Let's say, a regular, non-churning customer generates USD 500 of profit for us.
- We are going to give out a discount of 33.3% to customers we believe will churn in the next period. It is effective, but not perfectly effective, so only 50% of those customers, who were going to leave, stay after getting the discount. The others still leave and leave us with USD 0.
- Customers who leave us do not spend any money anymore, so we get USD 0 from them.

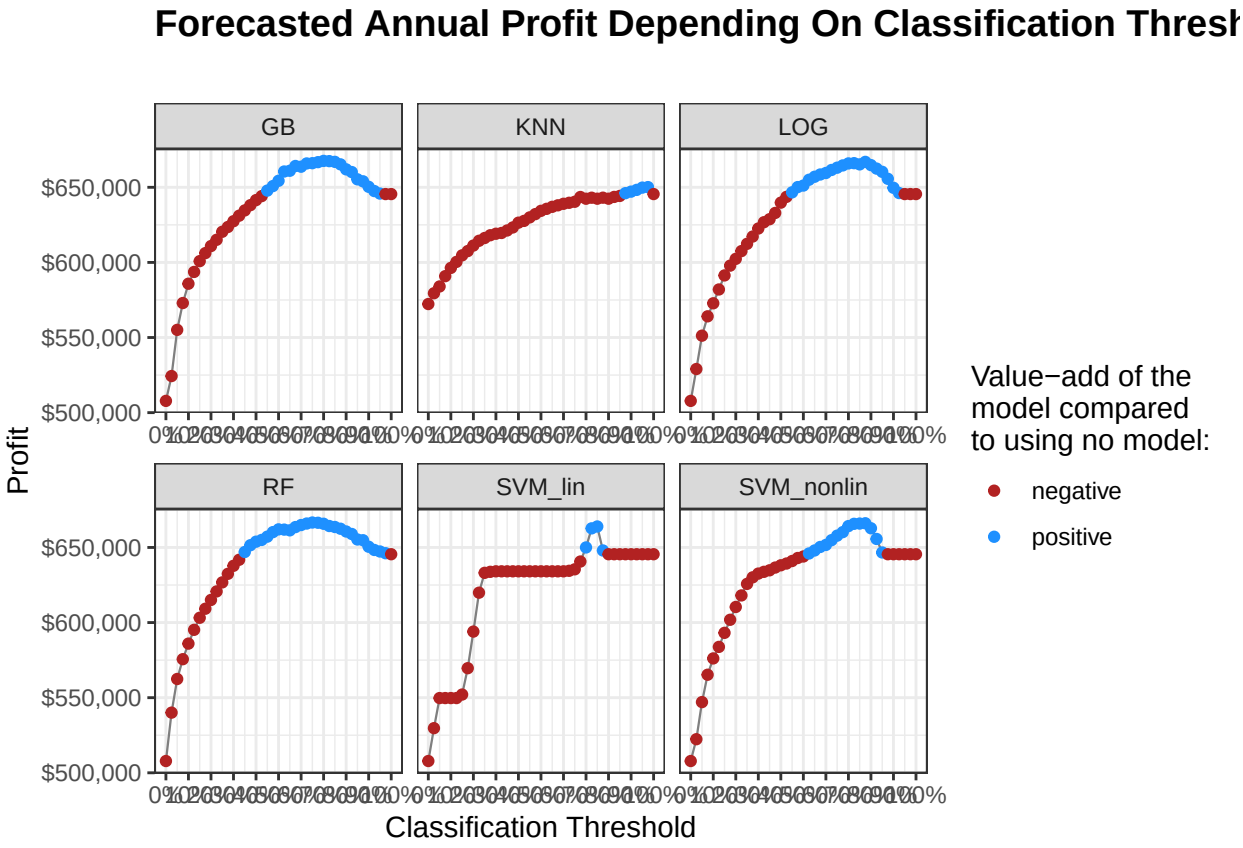
In model terms this implies:

- TP = True Positive: We predicted the customer leaves, we gave out a 33.3% voucher. 50% of them stay and create profit of USD 500, the rest leaves. Our profit from this group is $N_{TP} * 500 * 0.5 * 0.666$.
- FP = False Positive: We predicted the customers leaves, but they weren't planning on leaving. We gave them a 33.3% discount, all of them stay and our profit from this group is $N_{FP} * 500 * 0.666$.
- TN = True Negative: We predicted the customer is not going to leave, they actually didn't leave. We like those customers because of their loyalty and because they give us the most money, namely $N_{TN} * 500$.

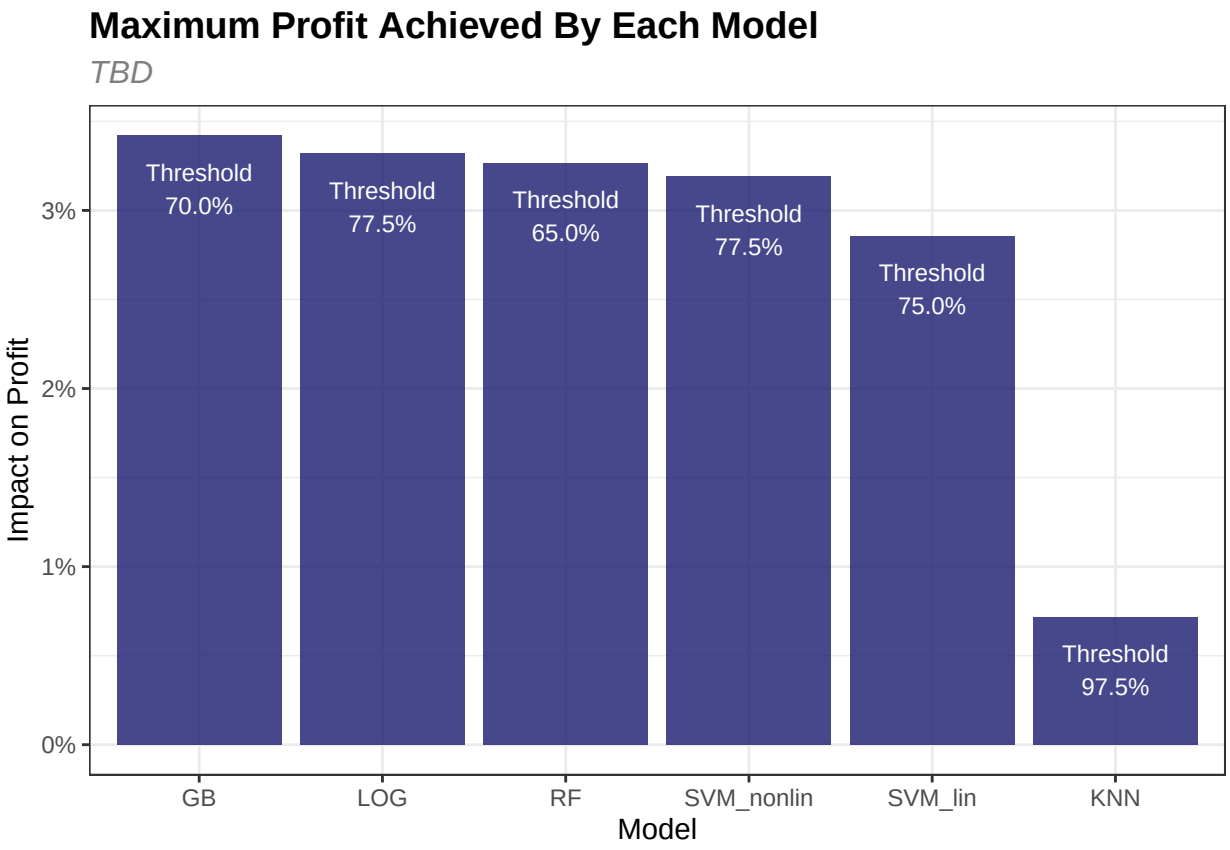
- FN = False Negatives: We predicted the customer is not going to leave, but they actually left. These are bad, because we didn't target them with a voucher. Ouch: The profit from this group is 0.

Now I can go ahead and write a function, which counts our TP, FP, TN and FN and calculates the profit based on the sum of all of the four points above, for each threshold we could use in our model.

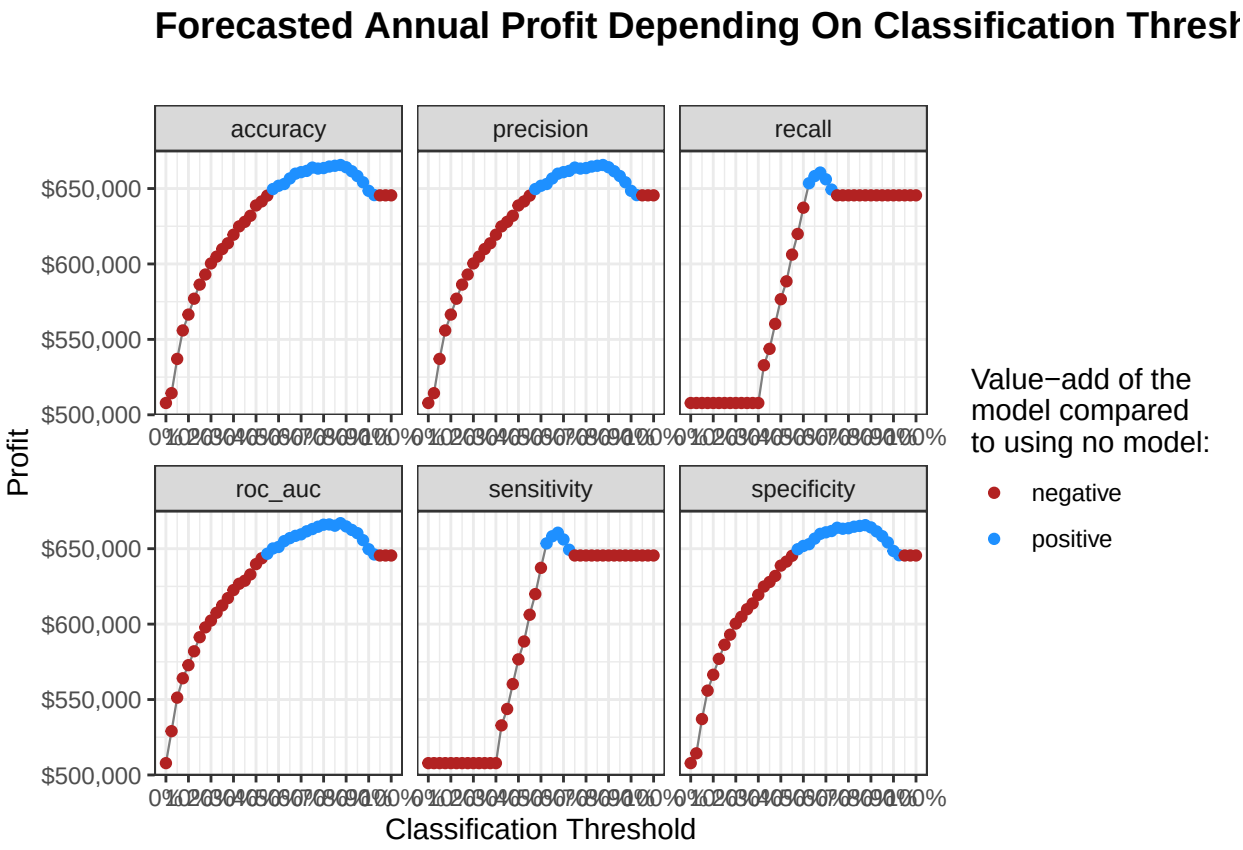
Profit curve for all models:

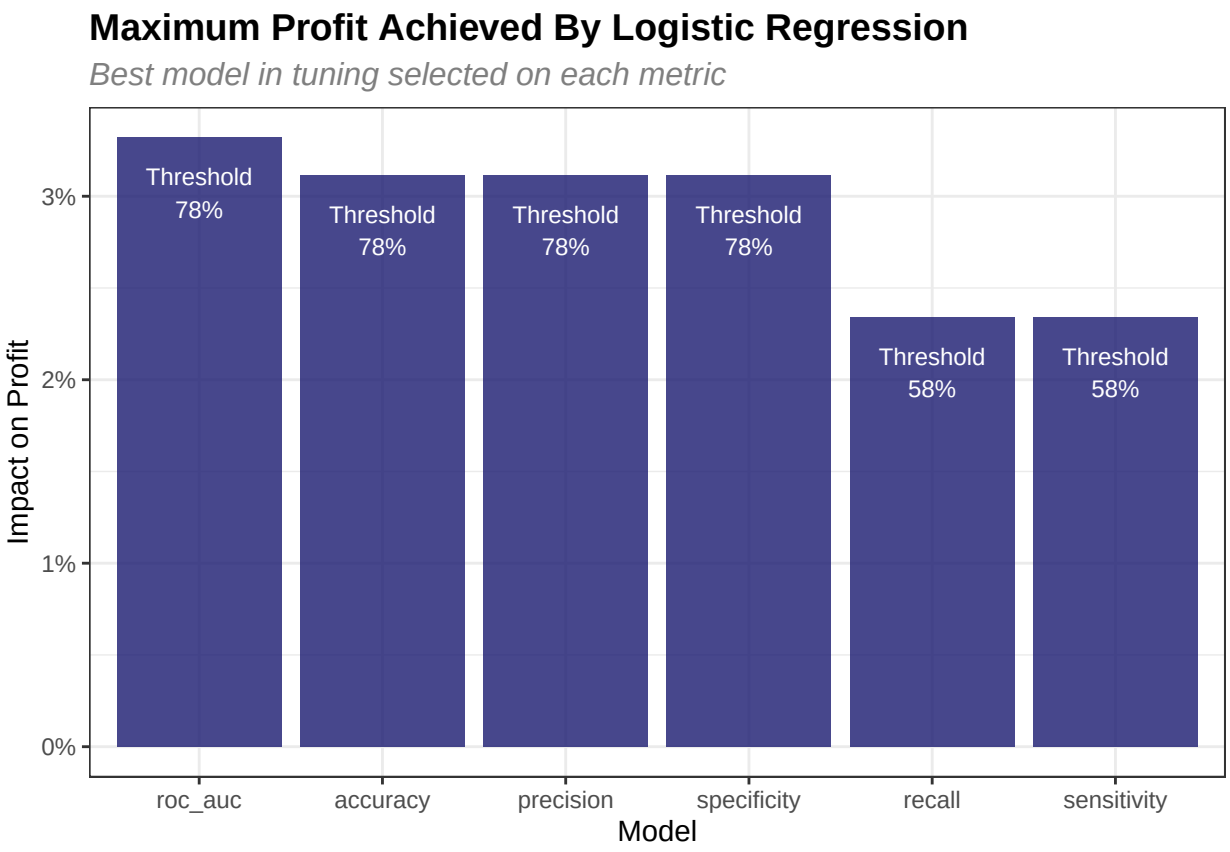



```
## # A tibble: 6 x 3
##   description profit_delta threshold
##   <chr>              <dbl>     <dbl>
## 1 GB                0.0342      0.7
## 2 KNN               0.00719     0.975
## 3 LOG               0.0332      0.775
## 4 RF                0.0327      0.65
## 5 SVM_lin           0.0286      0.75
## 6 SVM_nonlin        0.0319      0.775
```



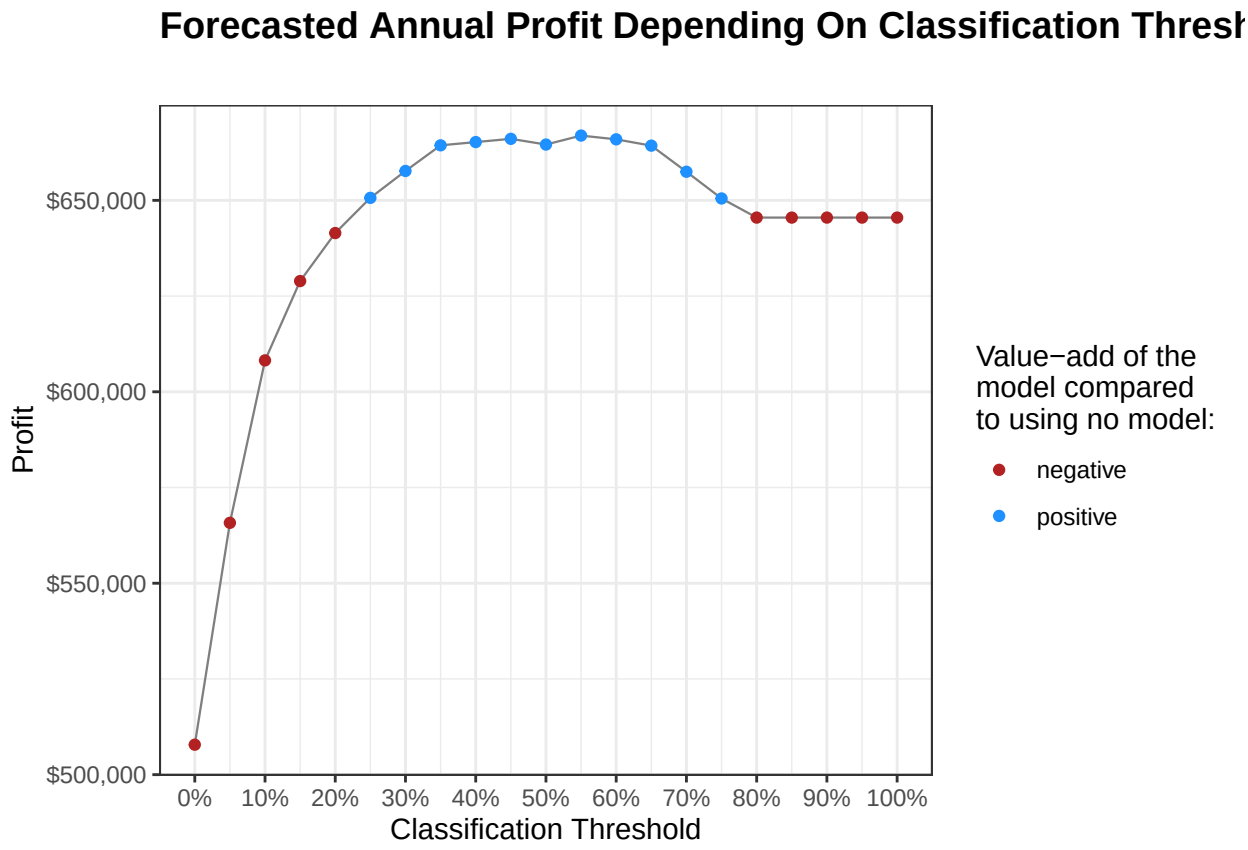
- Checking which metric would be best for fit: ROC AUC should be best


Forecasted Annual Profit Depending On Classification Thresh
accuracy
precision
recall
roc_auc
sensitivity
specificity
$650,000
$600,000
$550,000
$500,000
Profit
Classification Threshold
Value−add of the model compared to using no model:
negative
positive



Looking at the blended model last:

- Profit curve for blended model:




```
## # A tibble: 1 x 1
##   profit_delta
##          <dbl>
## 1       0.0332
```

Gradient boosting has 0.0342, 0.1 ppts better than blended model

Conclusion: No benefit in blending models, much more computationally expensive and slow. Logistic regression with regularisation and shrinkage brings the same performance, considerably faster

7 Ethics

TBD: Tobias

8 References

Apply SMOTE Algorithm. (n.d.). `https://themis.tidymodels.org/reference/step_smote.html`. (Accessed: 2022-11-18)

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.