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°: 16-611-485 / 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

29.11.2022

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?)

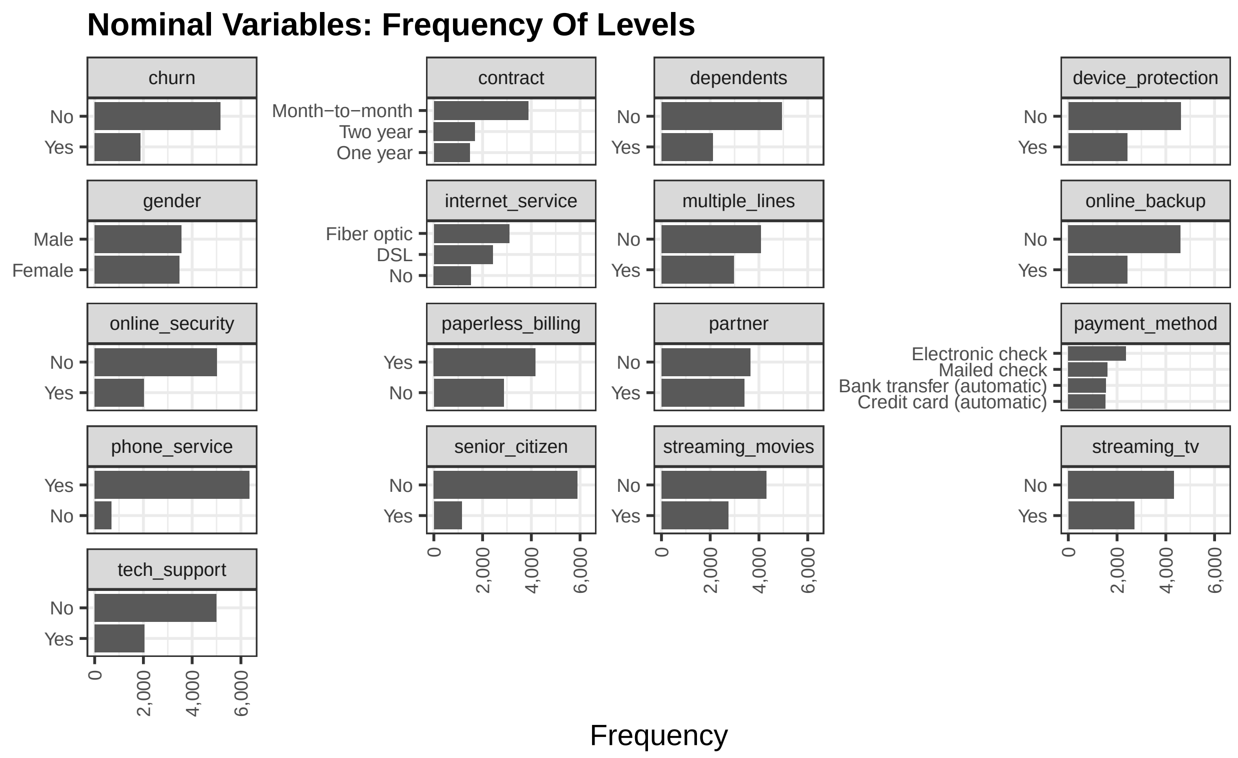


Figure 1: Counts of nominal variables

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

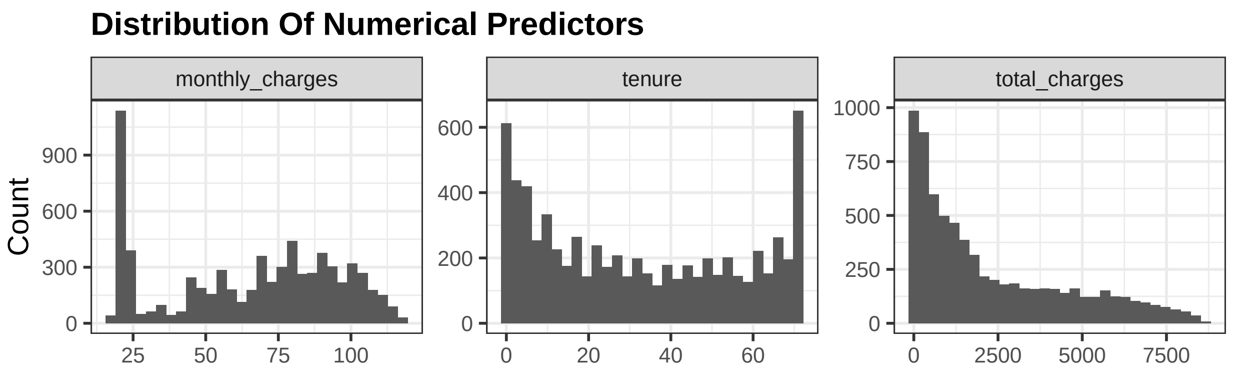


Figure 2: Distribution of numerical variables

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.

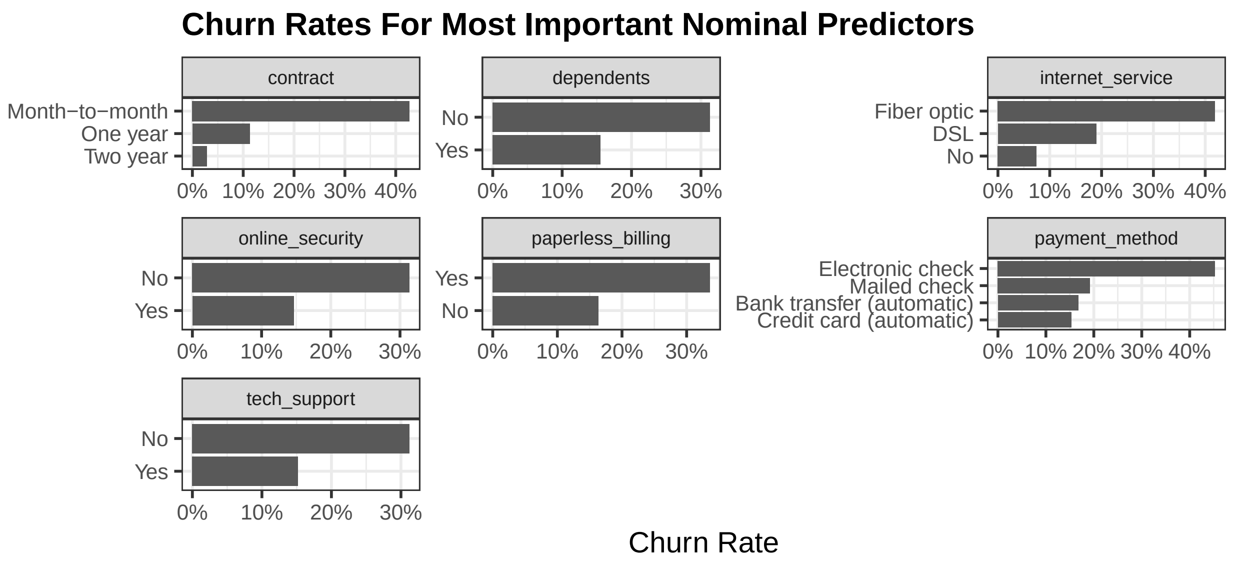


Figure 3: Churn Rates for different levels of nominal predictors

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.

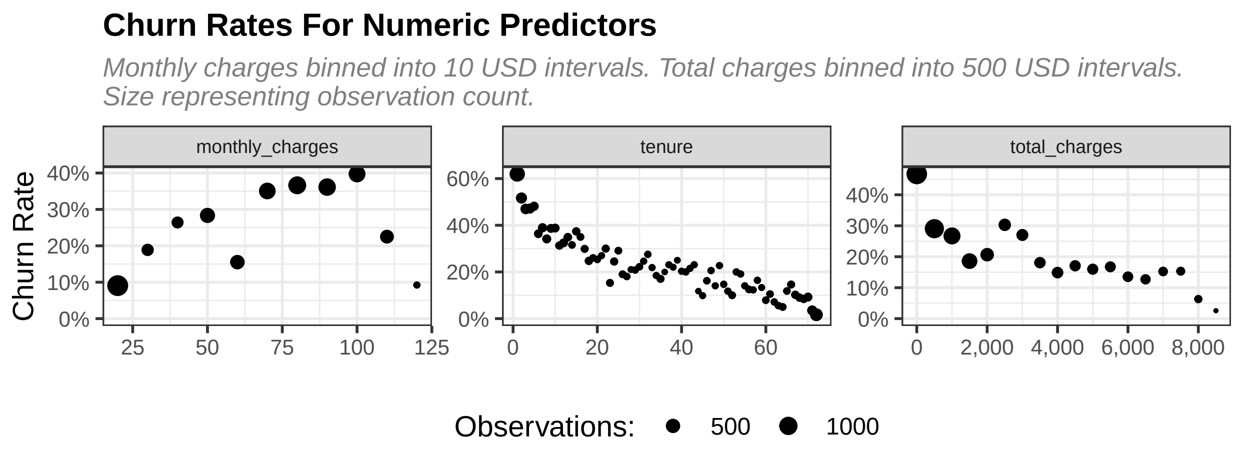


Figure 4: Churn rates associated with binned numeric predictors

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 Creating The Models

4.1 Splits

The first step in our modelling approach is creating the splits based on stratified sampling with a $\frac{n_{train}}{n_{total}} = 75\%$ ratio. Additionally, 5-fold stratified cross validation was employed for hyperparameter tuning the various models, where the same 5 folds are given to all models.

TBD Tobias: Citation on why stratified sampling is better?

4.2 Data Preprocessing

At this stage, the strength of `Tidymodels` fully shines through: With the `recipes` package, a single preprocessing pipeline can be specified with `R`'s formula notation and `dplyr`'s pipes. This pipeline will be executed upon hyperparameter tuning and model training on each fold and each split respectively and uniformly. This makes it very hard to commit preprocessing errors leading to data leakage, like not imputing separately on training and testing data.

```
recipe(churn ~ ., data = training(split)) %>%
step_rm(customer_id) %>%
step_novel(all_nominal_predictors()) %>%
step_normalize(all_numeric_predictors()) %>%
step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
step_zv(all_predictors()) %>%
step_smote(churn, skip = TRUE)
```

Going into detail for the preprocessing steps that have been taken: As a first step, the irrelevant customer IDs are removed. Next, we are allowing for previously unseen factor levels with `step_novel()`. Normalising the predictors allows for relative variable importance calculation without the inflationary effect of higher mean numeric variables, but is generally not necessary for the tree-based learners. However, as we will also be using other methods like kNN and support vector machines, which require normalisation, having it in the general recipe is the best option and does no harm to the models that do not require it. Next, all nominal predictors are one-hot encoded. Note that the linear models, that is logistic regression and SVMs with linear kernels have issues with multicollinearity, so these are dummy-encoded, i.e. leaving out a factor level. Additionally, a zero variance variable filter is applied, which removes zero variance predictors, though there should not be any in these data, after having inspected it in the EDA section. Lastly, the target variable is upsampled using the SMOTE algorithm, as mentioned previously.

4.3 Model Specifications

As our goal is to predict customer churn as best as possible, we not only want to test configurations of any single model, but test multiple models and compare performances. This section will shortly go into each model, how it works, what needs to be taken care of and which hyperparameters were tuned. Notably, at this stage, `parsnip` allows us to use the same syntax for specifying each model, independe_nt of the underlying package. The detailed description can be found *here*. Hence, for all our six different models, we can use the following syntax:

```
specification <- model(parameters = ...) %>%
set_engine("package") %>%
set_mode("classification")
```

4.3.1 Gradient Boosting

The gradient boosting relies on the `xgboost` package, which allows for the tuning of the number of trees (`trees`), the tree depth (`tree_depth`), the minimum number of observations in a node required for the node to be split further (`min_n`), the number for the reduction in the loss function required to split further (`loss_reduction`), the number of observations that is exposed to the fitting routine (`sample_size`), the number of predictors that will be randomly sampled at each split when creating the tree models (`mtry`) and the rate at which the boosting algorithm adapts from iteration-to-iteration (`learn_rate`).

4.3.2 K-Nearest Neighbours

The kNN algorithm relies on the k-nearest neighbours in the feature space and generates a weighted prediction based on the weighting method, which can also be called the kernel. In `parsnip`, the engine package is `kknn` and only one hyperparameter can be tuned, namely the number k neighbours. For this application, we are going with a Gaussian weight function, which was superior to the equally weighted rectangular weight function on the holdout data.

4.3.3 Random Forest

Our implementation of the random forest algorithm relies on the `ranger` package and allows for the tuning of the number of predictors that will be randomly sampled at each split when creating the tree models (`mtry`), the number of trees in the forest (`trees`) and the minimum number of observations in a node required for the node to be split further (`min_n`).

4.3.4 Logistic Regression

The implementation of the logistic regression with the `glmnet` engine allows for two tunable hyperparameters of the number between zero and one (inclusive) giving the proportion of L1 regularization (i.e. lasso) in the model (`mixture`) and the regularisation parameter (`penalty`). Notably, we're using Ridge and Lasso regularisation in order to increase the ability of our model to generalise to out-of-sample observations. As we are tuning both parameters, it remains to be seen, whether regularisation actually benefits the model performance, or whether best performance comes from a standard logistic regression model.

4.3.5 Support Vector Machines

For support vector machines, we are implementing both a linear and a radial basis function kernel. The latter allows for non-linear relationships with the target variable, and it remains to be seen whether the a performance benefit is associated with that. However, from the EDA, it seems that some non-linearity exists, therefore it will likely be the case. The linear specification has just one hyperparameter of the cost of predicting a sample within or on the wrong side of the margin, which separates classes (`cost`). The non-linear specification also has the `cost` parameter, but additionally, we can tune the sigma of the radial basis function (`sigma_rbf`), which inversely proportionally affects the weights used in the kernel.

4.4 Hyperparameter Tuning

For the hyperparameter tuning of the six different models, there are several points to address. Starting with the cross validation, the `tune_grid()` function in `tune` package enables us to fit every hyperparameter combination five times, once on each fold and compute cross validated performance metrics. Performance metrics can be specified using a `metric_set()` function from the `yardstick` package, which allows for computation of all desired performance metrics in one function call. The results returned from each tuning object are the cross validated performance metrics, which then enables us to analyse dependency of model performance on hyperparameters. One additional tool used for hyperparameter tuning is the `doParallel` package, which enables us to use any number of cores on our CPU, greatly improving tuning speed over just using one core. Regarding the actual hyperparameters, all models have been tuned with hyperparameter combinations from a space-filling design, specifically a latin hypercube, which enables us to cover the space of possible hyperparameter combinations optimally. The number of combinations are $n = \{100, 10, 50, 50, 30, 30\}$ in the order as presented in the model specification section above.

The evaluation metrics we have chosen to look at are *accuracy*, *roc_auc*, *precision*, *recall*, *sensitivity* and *specificity*. Figure 5 presents the tuning results visually. Each dot is one cross validated metric mean for one hyperparameter combination for a given model. The point clouds are sorted in descending order by their median value to be able to compare the general trend of each model given each metric better.

TBD Findings: - Tree based ensembles best accuracy, but RF has terrible sensitivity, but high precision, hence is a little conservative on predicting that a customer is going to leave - SVM RBF is best for recall, but worst for precision, which is because it is best for sensitivity, but worst for specificity, i.e. it is too trigger happy -> predicting customer is going to leave way too often - KNN looks overall disappointing across the board, also lowest ROC AUC - Logistic Regression looking very promising with midfield for all metrics, but very high ROC AUC

After investigating the hyperparameter tuning results, we have decided that the ROC AUC metric gives us the best trade-off between specificity and sensitivity. As we have the business perspective in mind, we have to balance the amount of false positives and false negatives, as either of them can hurt our business case. If we give out discounts to falsely positives, we waste money on loyal customers. If we give out vouchers to false negative, we missed the opportunity to potentially retain a customer who then churned. Therefore, both cases are costly and we cannot just optimise for either one. ROC AUC appears to be able to strike the balance, which is exactly what we seek. Therefore, we proceed by setting the hyperparameter combination for each model on maximum ROC AUC and proceed with training all models on the training data.

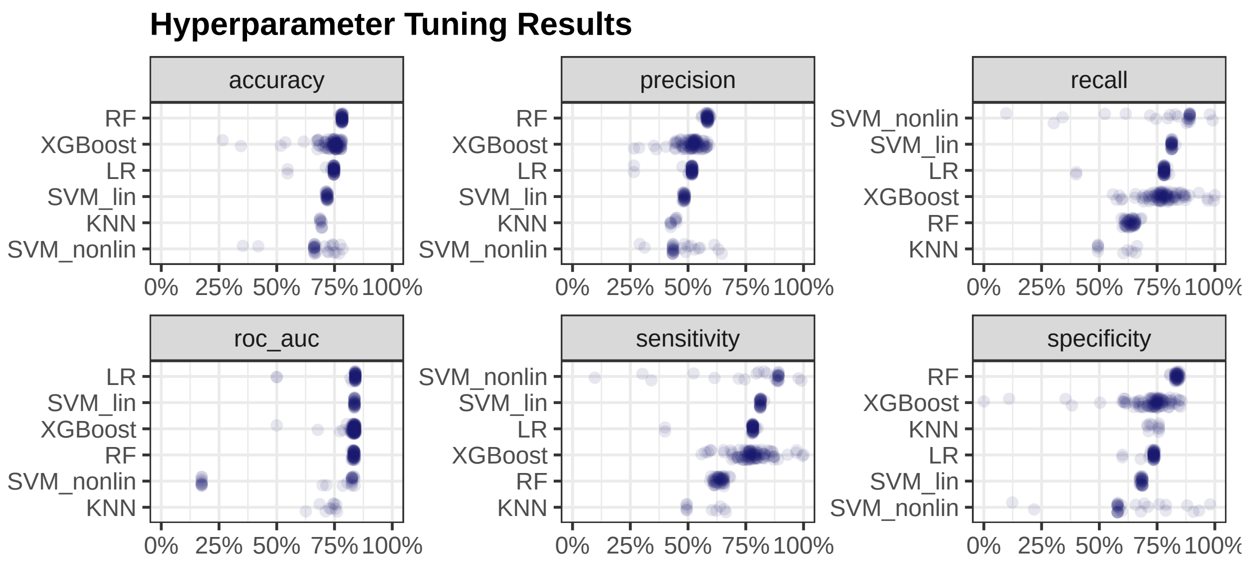


Figure 5: Hyperparameter tuning results for various evaluation metrics

4.5 Digression: Stacked Models

Before we proceed with the model evaluation section, we quickly want to digress into stacked models. The `stacks` package enables us to easily use a LASSO regression to linearly blend model predictions into a stacked model. The `stacks` package then returns weights from zero to one, being able to drop models due to the nature of LASSO, for each sub-model from the hyperparameter tuning process, during which we saved predictions. We investigate the performance of the stacked model together with the others in the next section.

5 Model Performance on Holdout Data

Firstly, we make predictions with all models on the holdout data and compute the same number of evaluation metrics with the previously initialised metric set. This time, we do not fit on the cross validation folds, but make predictions once on the holdout data, therefore we will receive one number for each model and each metric. Alternatively, we might also have a look at the confusion matrices, but as we have calculated all interesting metrics based on exactly these, it will be redundand at this point.

TBD Evaluate Metrics: - Looking at accuracy alone: blended model seems to have worked, however, looking at sensitivity and recall, it looks like the model is just way too conservative and always predicting the customer is not going to leave - GB has highest ROC AUC, also a great tradeoff between sensitivity and specificity, which makes sense from the business tradeoff perspective again - the SVM models are a little trigger happy, low precision, but high sensitivity - logistic regression, surprisingly, also very very good, almost as good as grdient boosting, even slightly better on sensitivity, which might be useful, also much more interpretable, strong competitor! - generally:

precision lower than recall, might be smart to dial up the classification threshold, if we need different ratio for profit maximisation

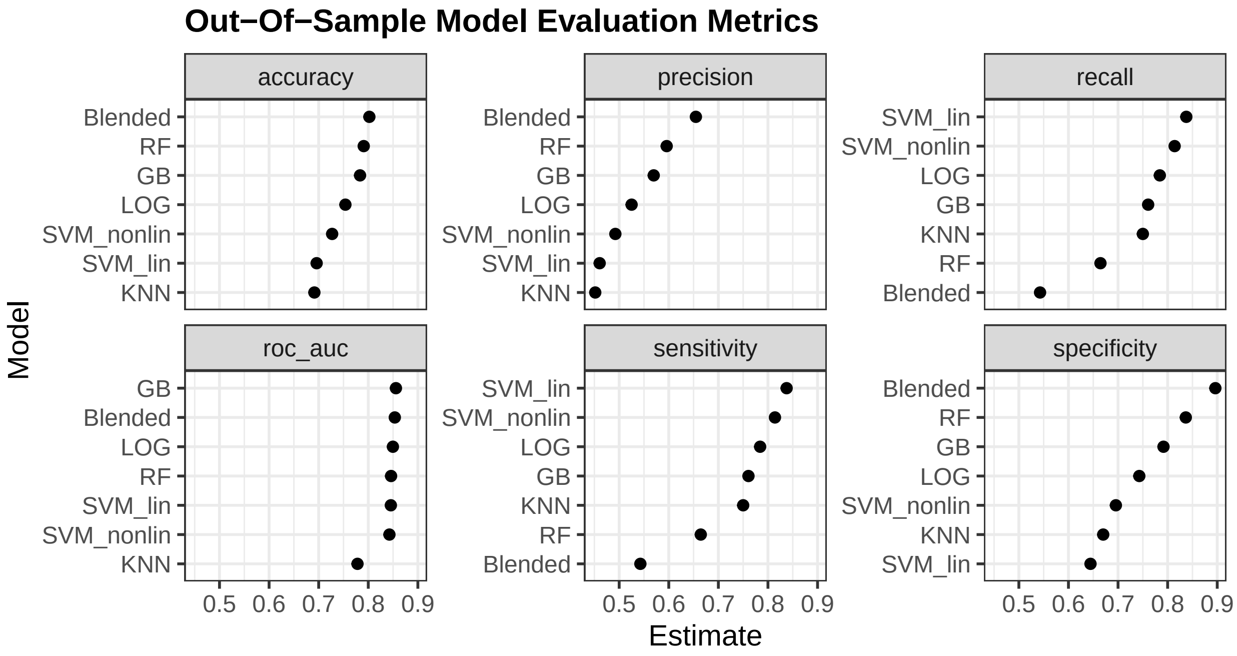


Figure 6: Out-of-sample performance metrics from predicting on the testing set

Another tool to evaluate the performance is ROC AUC curves, as seen in Figure 7, though it is rather visual. It shows the behaviour of models when varying the classification thresholds. Generally, the higher the curve lies towards the upper left corner, the better the model. If one curve is strictly to the left of the others, the model is strictly superior. In this particular case, the kNN model is visibly worse than all others, but it is hard to distinguish the other ones.

At this stage, we could look at the variable importance of our two best models: gradient boosting and logistic regression. As the latter might suffer from collinearity and omitted variable bias though, we will not show the directional effect, but only the absolute importance of the metrics. We want to stress that we are not doing inference, hence the level of these metrics should not be important, as long as the regression model generalises well to out-of-sample data, which is all we care about. The variable importance for the gradient boosting model was calculated based on Gini impurity, which measures how well the impurity of child nodes in a decision tree is reduced by using the variable as the split variable at this given split. Figure 8 shows that both models have comparable variables in their top 5.

6 Business Case: Model Evaluation Based On Business Metrics

The real question following the above is: How can we use any of the above to create business value? That's were the profit curves comes in. Classification models work by assigning probabilities of

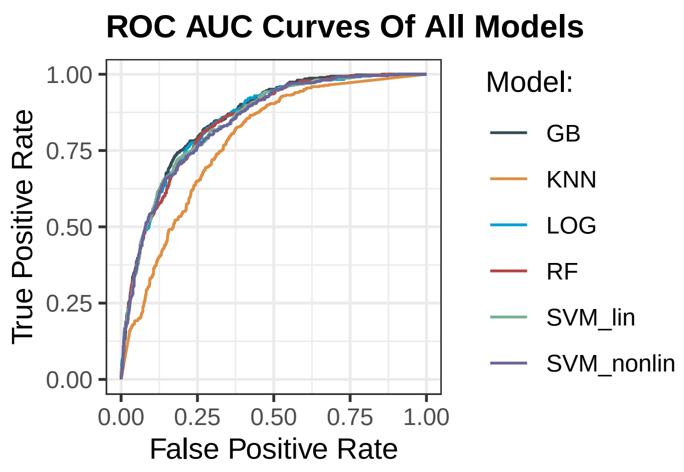


Figure 7: Area under the ROC Curve for all models

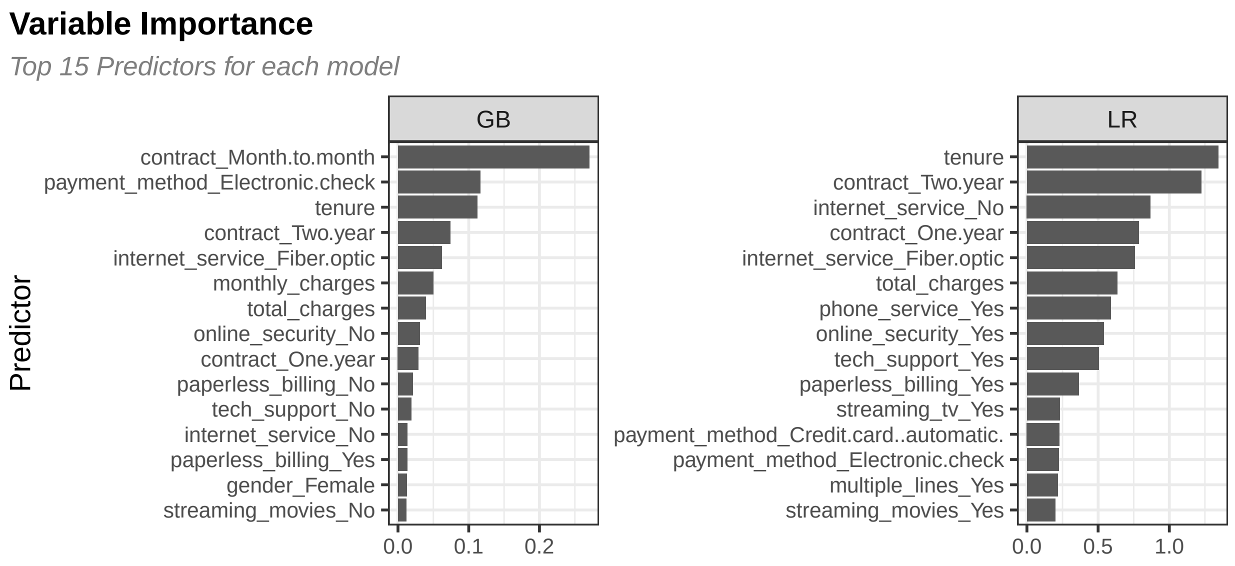


Figure 8: Variable importance for random forest model

Table 1: One example of a model prediction

customer_id	churn	.pred_class	.pred_Yes	.pred_No
4680-KUTAJ	No	Yes	0.5500337	0.4499663

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 in Table 1. The gradient boosting model predicted around 55% probability of the customer churning and around 45% of them not churning based on the available variables.

In a business setting, a logical consequence 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 50.01% chance of churning, according to the model, and assuming that the model is right, then we will be losing out on revenue from a loyal customer around half of the time. After all, there is an around 50% chance that they might not intend to leave in the first place. In that case, the discount would be wasted. As a business, we only want 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. 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 discounts.

For the purpose of demonstration, we will make some simplifying assumptions about a business case and profit generated from each customer in that business. 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 any more, 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 very bad, because we did not target them with a voucher: The profit from this group is 0.

We write a function to count our TP, FP, TN and FN and calculate the profit based on the sum of all of the four points above for each classification threshold from zero to one in 500 basis points steps. This gives us the profit curves for each model, by which we can evaluate model performance depending on the classification threshold as shown in Figure 9. It can be seen that both SVM methods are erratic or show a very narrow window of value-add compared to the baseline of using no model and that the kNN algorithm is insufficient compared to the other models. This was to be expected after seeing the ROC AUC curves in the model evaluation section earlier. In contrast, the tree-based ensembles as well as the logistic regression look much more promising, with smoother lines and wider windows of value-add over the baseline. Figure 10 depicts the value-add in profitability of each model including the classification threshold that it is associated with. It can be seen that the classification thresholds of the tree-based models are much lower, indicating a better trade-off of these model at the default classification threshold. Gradient boosting is very close to the random logistic regression, however random forest tops them all. Had we not looked at the evaluation of the models from this business perspective, then we would have likely ruled out random forest. This goes to show the importance of optimising your models for the given task at hand.

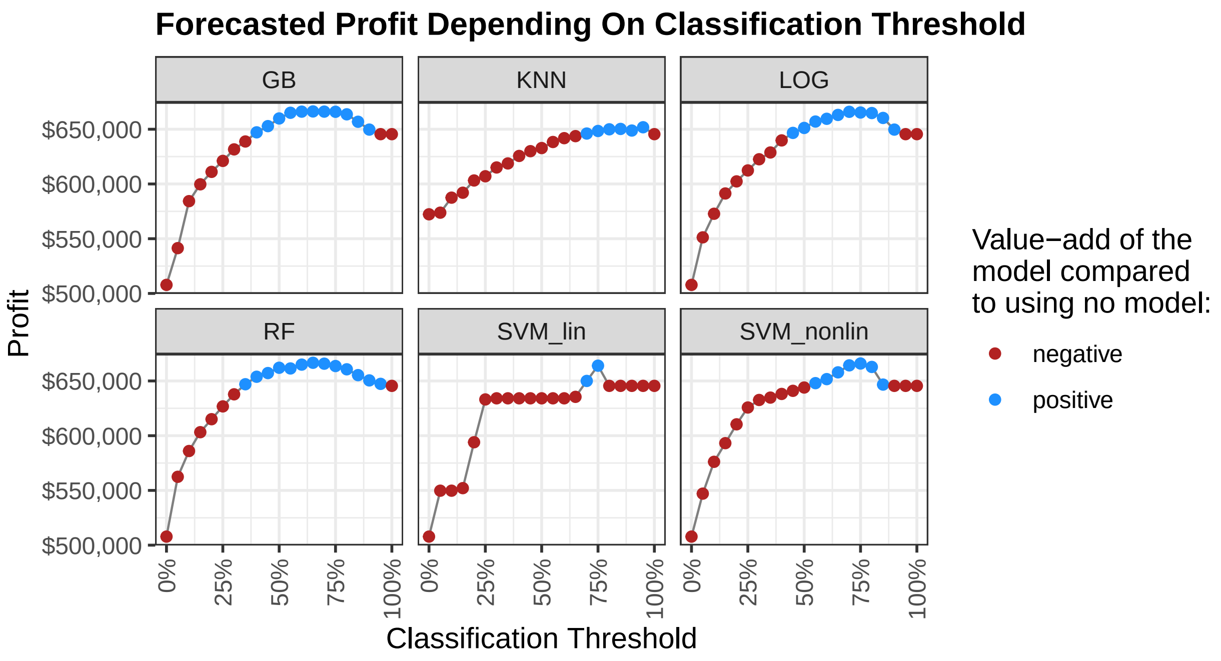


Figure 9: Profit curves for all models depending on classification threshold

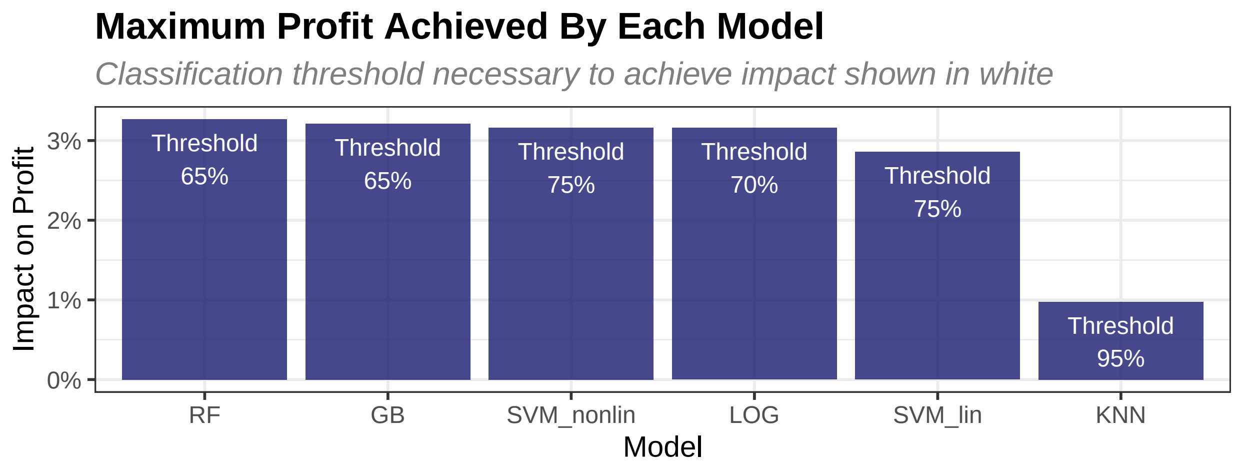


Figure 10: Highest achieved business profit by each model including modified classification threshold

In similar fashion, we can go back to model training at this stage and ask whether ROC AUC was the best metric for choosing the optimal hyperparameter combination. Taking the logistic regression workflow, due to computational speed, and refitting one separate logistic regression model one of each of the six evaluation metrics, we can analyse whether ROC AUC was indeed the best choice. Figure 11 confirms that ROC AUC is the metric leading to the highest expected business profit on the out-of-sample data, therefore our hypothesis from earlier is confirmed. Given the size constraint of this paper, we will not go into the details of each algorithm, nor plot the profit curves again, but for metrics.

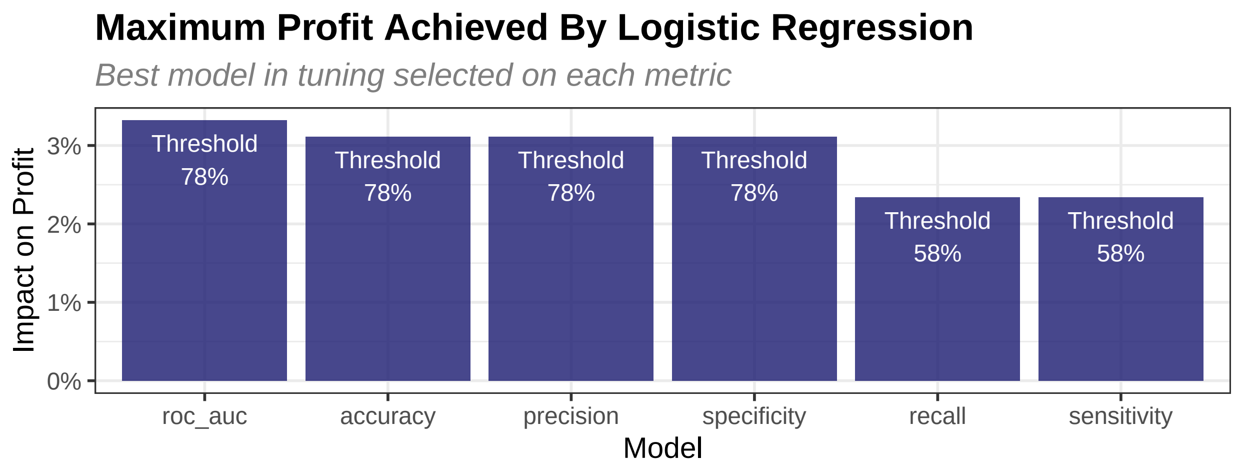


Figure 11: Highest achieved business profit by fitting one logistic regression model on each evaluation metric

Lastly, we can look at the stacked model in order to see whether the linearly blended combination performed better than any single model by itself. For this, we had to rewrite the two function calculating profits, which is why we did not include it in the above chart. From Figure 12, it looks like the blended model starts to be profitable very early and plateaus out after that. However,

Table 2: Maximum impact on forecasted profit for each model

Model	Profit Delta
Blended	0.0329597
RF	0.0326747
GB	0.0321294
SVM_nonlin	0.0316243
LOG	0.0316003
SVM_lin	0.0285786
KNN	0.0097738

from this alone, we cannot make comparisons to the other models. Numerically comparing both models in Table 2, it becomes clear that the blended model actually performs better from a profit standpoint.

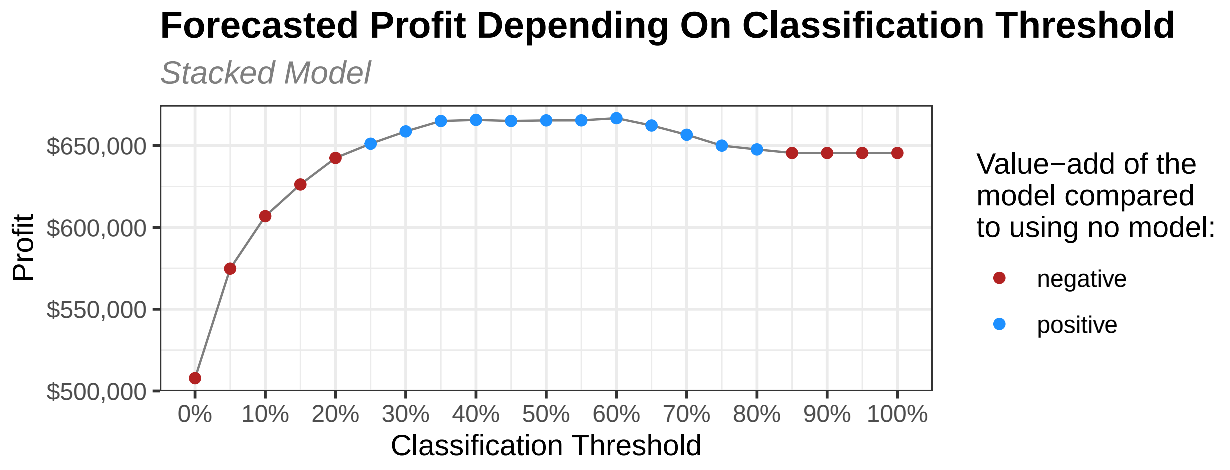


Figure 12: Profit curve for stacked model depending on classification threshold

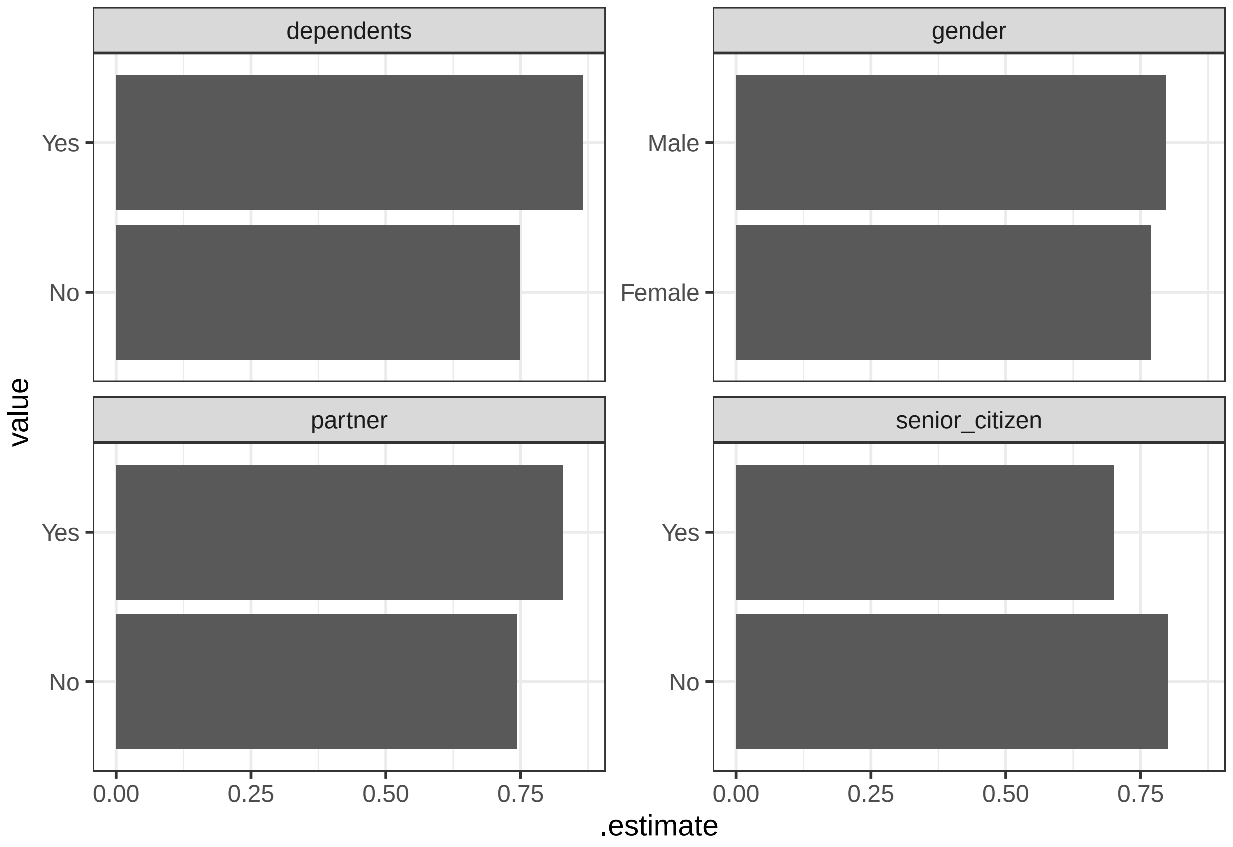
```
## Adding missing grouping variables: ‘description‘
```

TBD: - Tradeoff between very large and obscure models and just logistic regression, which works pretty much the same - Decide to go for logistic regression in the end: stakeholders will be happy to give up minuscule performance difference if they can understand the model and explain it to clients better

7 Ethics

TBD: Tobias

TBD: Is the accuracy of the model different for any subgroups?



TBD: Is it fair to give one subgroup discounts at higher rates? - Should we look at if model gives out more discounts on absolute level or on discounts per customer in group, i.e. ratios?

```
## # A tibble: 32 x 5
## # Groups:   name, value [8]
##    name       value  churn .pred_class     n
##    <chr>      <fct>  <fct> <fct>       <int>
##  1 dependents No     Yes   Yes           309
##  2 dependents No     Yes   No             82
##  3 dependents No     No    Yes           227
##  4 dependents No     No    No            610
##  5 dependents Yes    Yes   Yes            47
##  6 dependents Yes    Yes   No             30
##  7 dependents Yes    No    Yes            42
##  8 dependents Yes    No    No            412
##  9 gender     Female Yes   Yes           183
## 10 gender     Female Yes   No             49
## # ... with 22 more rows
```

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.