

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

04.12.2022

Contents

1	Why is it useful to predict customer churn?	1
2	Packages and Data	1
2.1	Packages	1
2.2	Data	1
3	Exploratory Data Analysis	2
4	Creating The Models	5
4.1	Splits	5
4.2	Data Preprocessing	5
4.3	Model Specifications	6
4.3.1	Gradient Boosting	6
4.3.2	K-Nearest Neighbours	6
4.3.3	Random Forest	6
4.3.4	Logistic Regression	7
4.3.5	Support Vector Machines	7
4.4	Hyperparameter Tuning	7
4.5	Digression: Stacked Models	9
5	Model Performance on Holdout Data	9
6	Business Case: Model Evaluation	11
7	Ethics	15
8	References	17

List of Figures

1	Counts of nominal variables	3
2	Distribution of numerical variables	3
3	Churn Rates for different levels of nominal predictors	4
4	Churn rates associated with binned numeric predictors	4
5	Hyperparameter tuning results for various evaluation metrics	8
6	Out-of-sample performance metrics from predicting on the testing set	10
7	Area under the ROC Curve for all models	10
8	Variable importance for random forest model	11
9	Profit curves for all models depending on classification threshold	13
10	Highest achieved business profit by each model including modified classification threshold	14
11	Highest achieved business profit by fitting one logistic regression model on each evaluation metric	14
12	Profit curve for stacked model depending on classification threshold	14

List of Tables

1	One example of a model prediction	11
2	Maximum impact on forecasted profit for each model	15

1 Why is it useful to predict customer churn?

The fierce competition in the telecom industry leads companies to increasingly switch from having the product at the core to having the customer at the core (Zhao, Zeng, Chang, Tong, & Su, 2021, p. 2), as retaining a dissatisfied customer is about 6 to 7 times cheaper than attracting a new one (Karanovic, Popovac, Sladojevic, Arsenovic, & Stefanovic, 2018, p. 1). To effectively target the dissatisfied customers and convert them into loyal ones, a reliable prediction model is necessary. Machine learning models seem to be ideal as they continuously learn from new data and thus detect newly emerging patterns and consumer trends (Karanovic et al., 2018, p. 4284). In a first step, several models are created to calculate the churning probability for each customer. The churn probability is the likelihood that a customer no longer buys the company's products or services (Ahn, Han, & Lee, 2006, p. 553). In a second step, the models are evaluated and compared using traditional evaluation metrics. Lastly, the direct impact of applying the model on the company's bottom line is demonstrated with profit curves in a simple business setting, based on which we recommend the best model, also taking into consideration practical implications like explainability.

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 a 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

The Telco customer churn dataset was retrieved from Kaggle and was originally posted in a data science challenge by IBM to predict customer behaviour and thus help improve the companies ability to retain them (BlastChar, 2018). The dataset contains information about a fictional telecommunications company that provided home phone and Internet services to 7043 customers in California in Q3 (*Cognos analytics with Watson*, 2019). The dataset does not contain a time variable and all observations are from a fictional third quarter. Hence, seasonality is not expected to play any meaningful role and will be neglected.

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, 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 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.). 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 4. Note that the variable `customer_id` 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 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

Nominal Variables: Frequency Of Levels

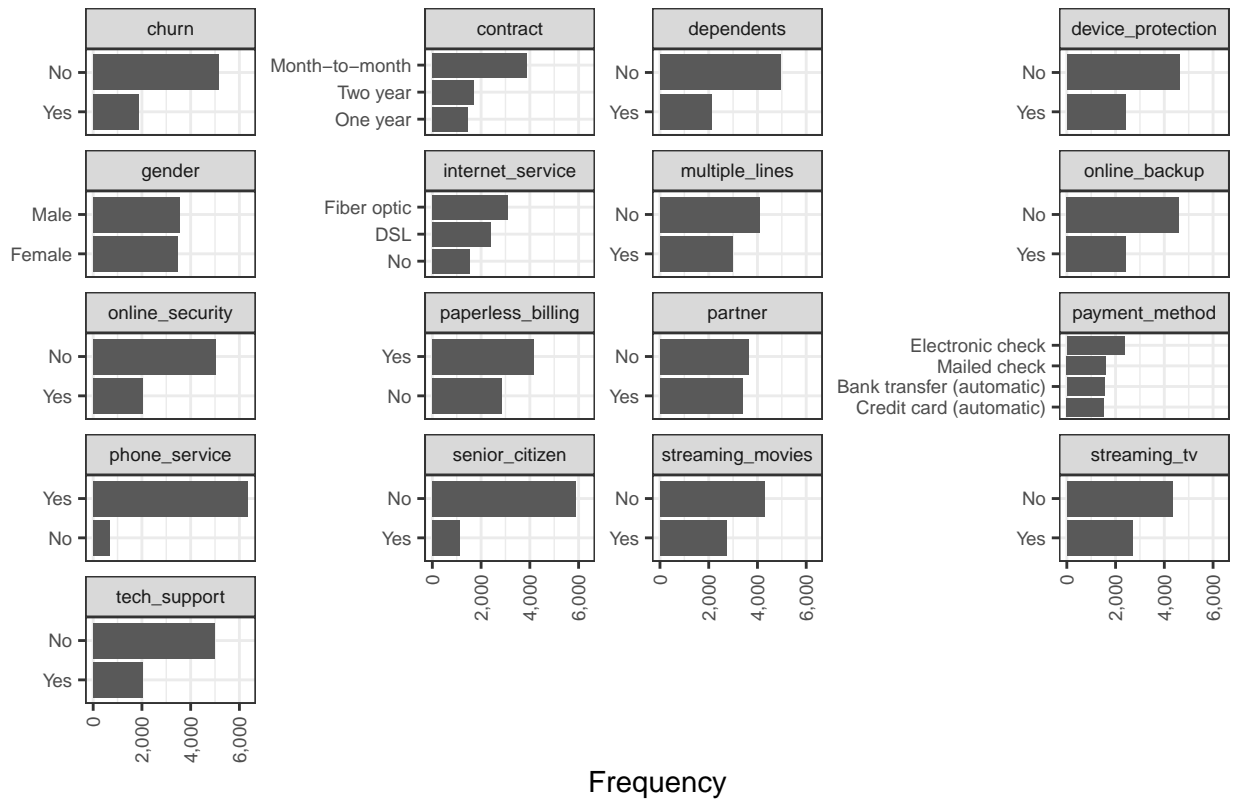


Figure 1: Counts of nominal variables

Distribution Of Numerical Predictors

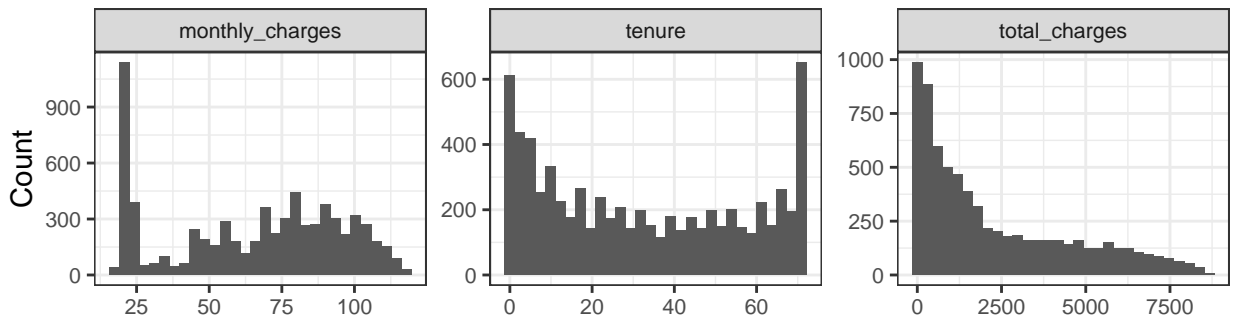


Figure 2: Distribution of numerical variables

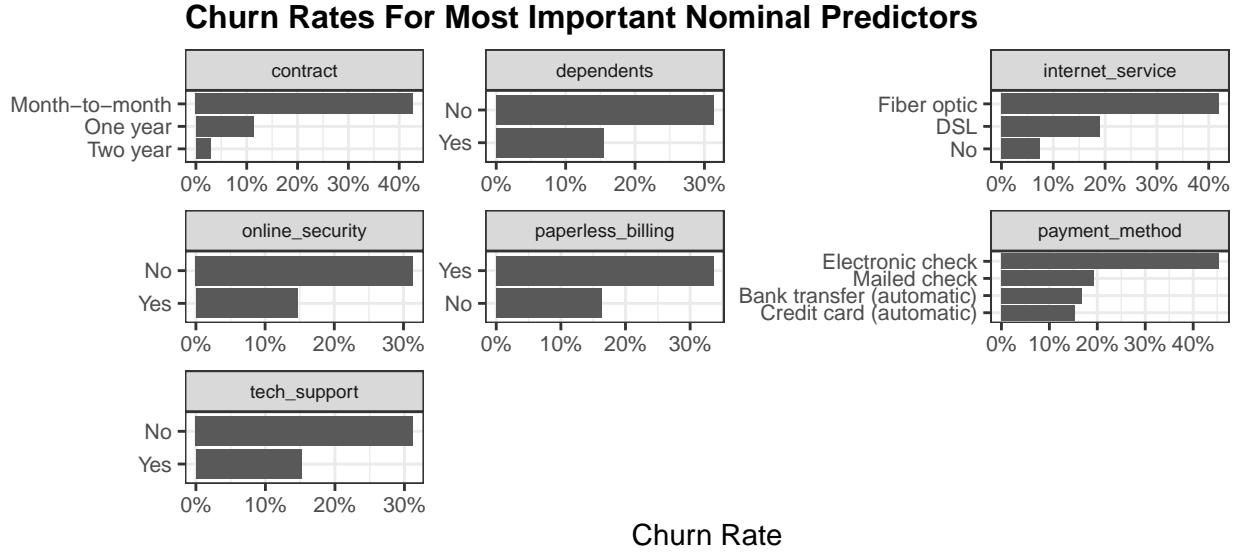


Figure 3: Churn Rates for different levels of nominal predictors

charges in time. Monthly charges does not reveal a linear relationship: In general, it looks like there is a positive relationship, but there are exceptions at 60 USD and beyond 100 USD. Potentially, for this variable, there will be a marginal benefit of allowing for more flexible methods over a logistic regression for instance.

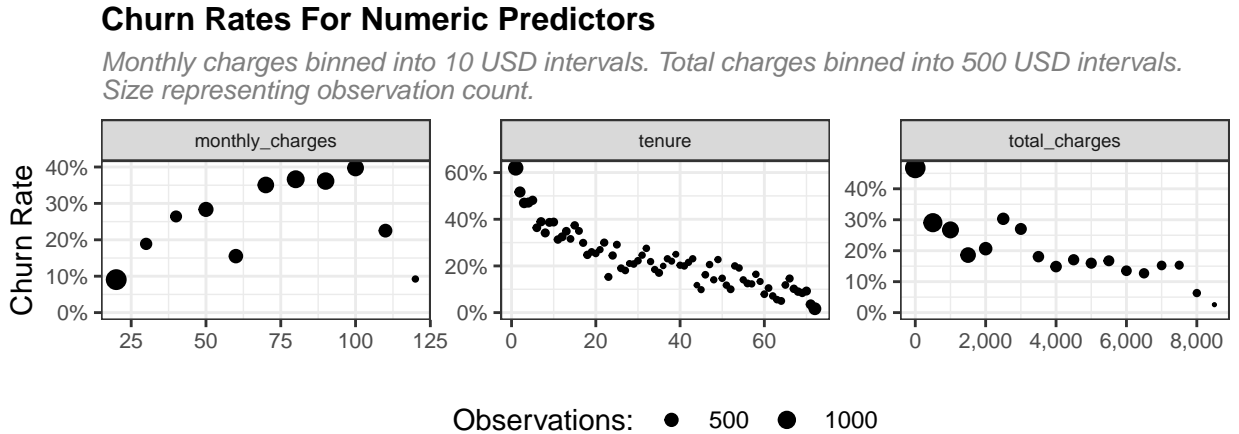


Figure 4: Churn rates associated with binned numeric predictors

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

When classifying a dataset with imbalanced classes, stratified resampling is generally a good idea (Kohavi et al., 1995, p. 7). Setting churn as the stratification variable will keep the proportion of churning and not churning customers in each training set equal to the proportion in the complete dataset. Therefore, the first step in our modelling approach is to create the splits based on stratified sampling with a $\frac{n_{train}}{n_{total}} = 75\%$ ratio. Note that any other ratio which allows for the trade off between having enough training data and being confident about your out-of-sample performance estimates would be fine. As we have more than 7,000 observations, going with a 3 to 1 ratio seems reasonable. Additionally, 5-fold stratified cross validation was employed for hyperparameter tuning the various models, where, importantly, the same 5 folds are given to all models to allow for consistent estimation. Additionally, the 5-fold cross validation allows for better estimation of our performance metrics, which might potentially have higher variance.

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 missing values separately on training and testing data would.

```
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, independent of the underlying package. The detailed description can be found *here*. Hence, for all our six different models, instead of using the idiosyncratic syntax of each package, we can use the unified syntax of **Tidymodels**:

```
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 **kkn** and only one hyperparameter can be tuned, namely the number k neighbours. For this application, we are going with a Gaussian weight function, which prevailed over 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 are 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. We are not limiting ourselves to either one, but judge from the hyperparameter tuning results instead.

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, leading to high time efficiency in tuning. The number of combinations are $n = \{100, 21, 50, 50, 30, 30\}$ 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. There is a different number of points

for each model, as only 21 hyperparameter combinations were tuned for kNN and 100 for gradient boosting, for instance. The reason for the latter is that there is only one hyperparameter for kNN, and therefore, fewer combinations are required to cover a sufficient range. 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. Generally, the tree-based ensembles show highest accuracy, but random forests show low sensitivity. Couples with its high precision, but comparatively low recall, we can deduct that it is a little conservative on predicting that a customer is going to leave. Furthermore, the non-linear SVM has highest recall, but lowest precision. Looking at the sensitivity (high) and specificity (low), it becomes clear that it is a very trigger happy in predicting a customer to churn. kNN looks disappointing across the board, also with regard to ROC AUC. Surprisingly, logistic regression holds up very well, landing in midfield for most metrics, placing it very high on ROC AUC.

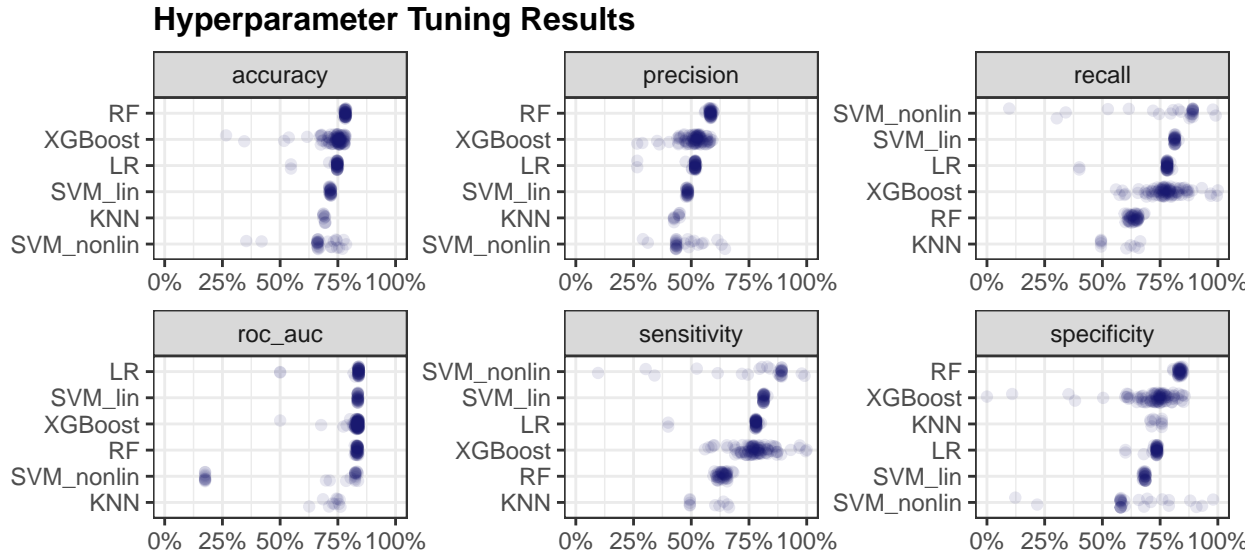


Figure 5: Hyperparameter tuning results for various evaluation metrics

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 from the cross-validation holdout evaluation and proceed with training all models on the full training data set.

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 the LASSO penalty, 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. Note that we could have used cross validation as well, however, given that we have sufficient observations in our testing set, the variance of the holdout estimator are fairly low. 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 redundant at this point. Looking at accuracy alone, the blended model seems to have worked best. However, looking at sensitivity and recall, it looks like the model is just way too conservative and always predicting the customer not to leave. Gradient boosting has a great trade off between sensitivity and specificity, also indicated by the highest ROC AUC. Both SVM models are a little too trigger happy, as they have high sensitivity and low precision, but low specificity and high recall. Logistic regression, surprisingly, almost sees eye to eye with gradient boosting. Given its explainability compared to the tree-ensemble, this makes it a great candidate. Generally, precision is lower than recall for all models. If we are interested in having balanced models, as we want to avoid both false positive and false negative cases, it might be smart to investigate the effects of changing the classification thresholds. At this stage, we hold out on final conclusions, as we want to look at the performance of the models in the business context first. 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 from each other.

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

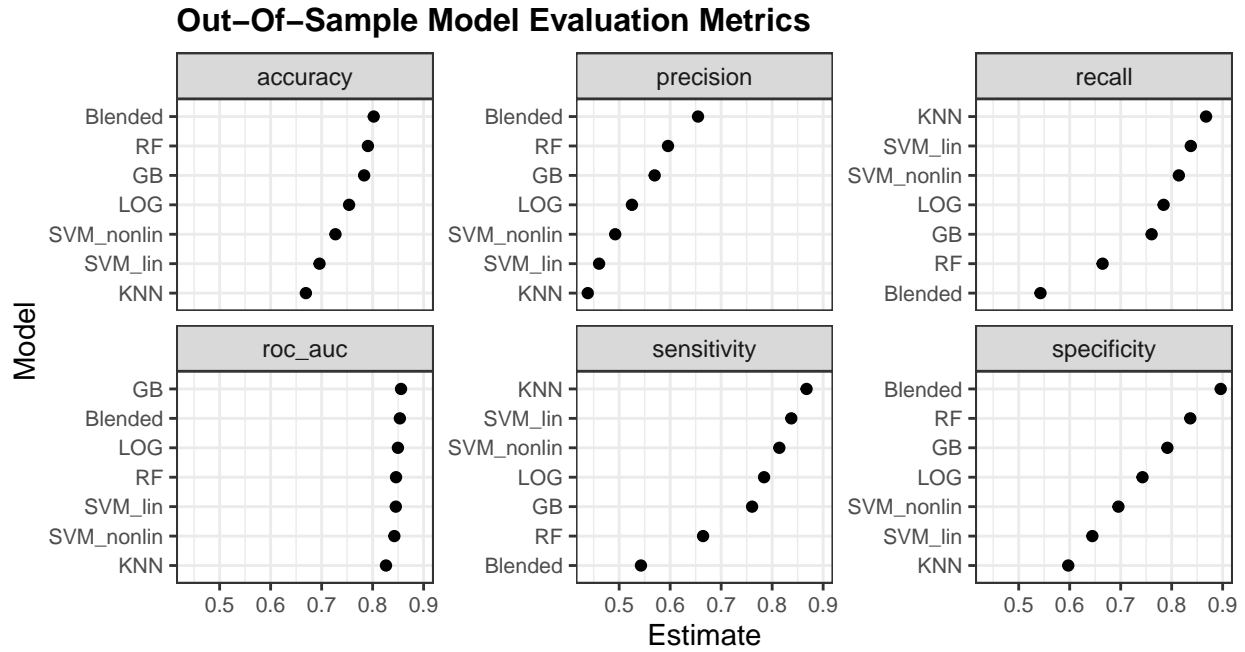


Figure 6: Out-of-sample performance metrics from predicting on the testing set

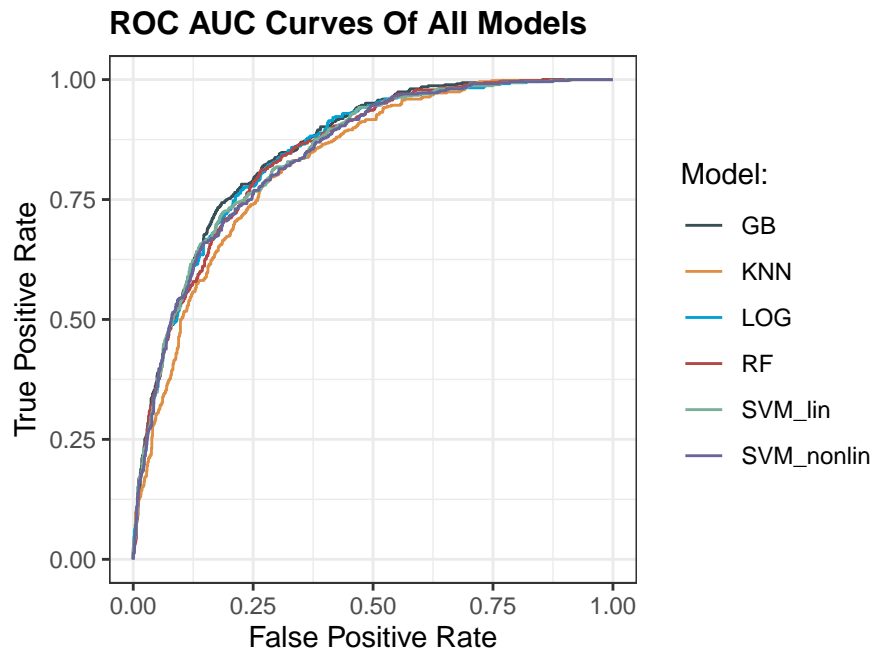


Figure 7: Area under the ROC Curve for all models

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

variable as the split variable at this given split. Figure 8 shows that both models have comparable variables in their top 5.

Variable Importance

Top 15 Predictors for each model

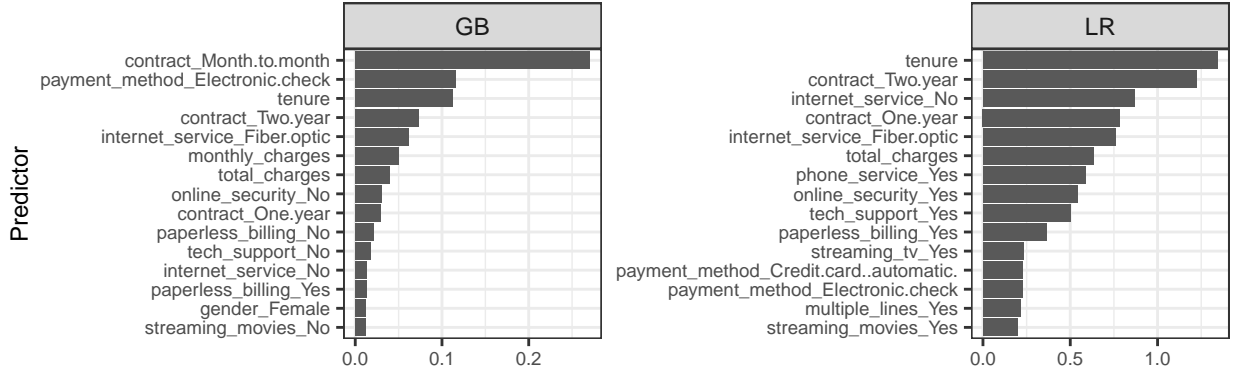


Figure 8: Variable importance for random forest model

6 Business Case: Model Evaluation

The real question following the above is: How can we use any of the above to create business value? That's where the profit curves come in. Classification models work by assigning probabilities of each class to any individual observation. For instance, one specific customer might be categorised with 55% probability of churning and 45% 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 the customer to leave with a probability of around 55%.

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 55% chance of churning, according to the model, and assuming that the model is right, we will be losing out on revenue from a loyal customer around a little less than half of the time. After all, there is an around 45% 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 between trying to prevent customers from leaving the business and losing out on revenue by giving out retention programmes to loyal clients who were not planning on leaving.

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 us say, a regular, non-churning customer generates USD 500 of profit per period 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 were not 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 did not 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.

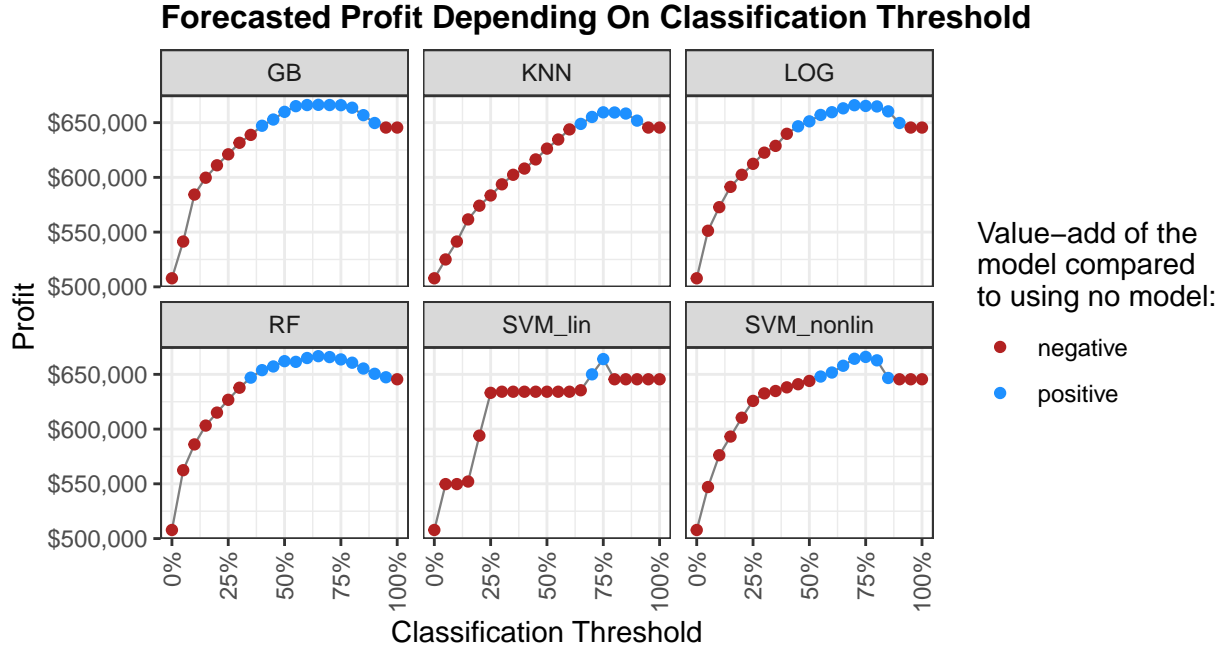


Figure 9: Profit curves for all models depending on classification threshold

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 for each model. As a final note, all models perform better than using no model, which results in USD impact, than making random predictions, which results in around USD 65,000 negative impact on average, than always predicting the majority class, which is the same as no model and than always predicting the minority class, which results in a negative impact of around USD 138,000.

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, 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.

At this stage, the tough decision must be made which model we recommend. Taking into consideration the marginal benefit of the more obscure and less explainable models in both absolute and

Maximum Profit Impact Achieved By Each Model

Classification threshold necessary to achieve impact shown in white

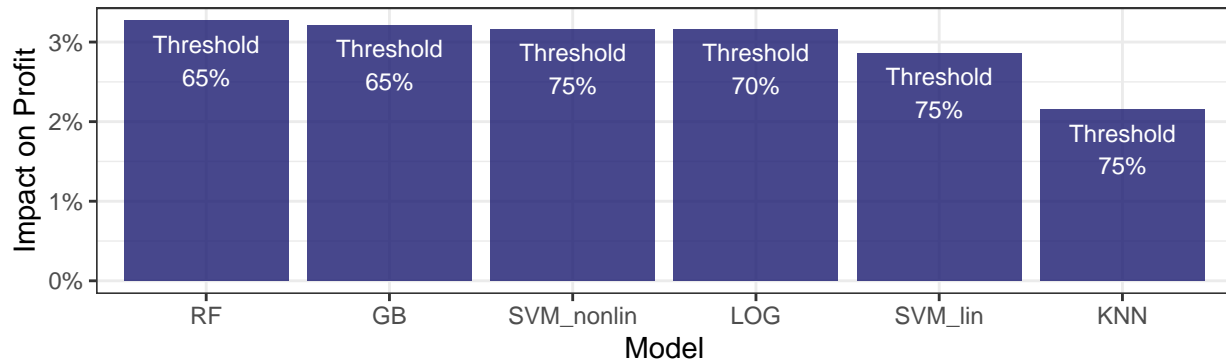


Figure 10: Highest achieved business profit by each model including modified classification threshold

Maximum Profit Achieved By Logistic Regression

Best model in tuning selected on each metric

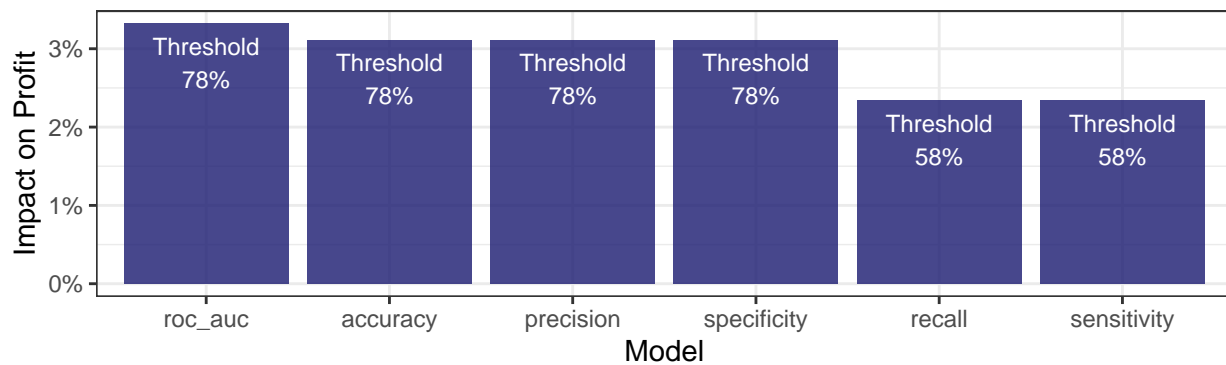


Figure 11: Highest achieved business profit by fitting one logistic regression model on each evaluation metric

Forecasted Profit Depending On Classification Threshold

Stacked Model

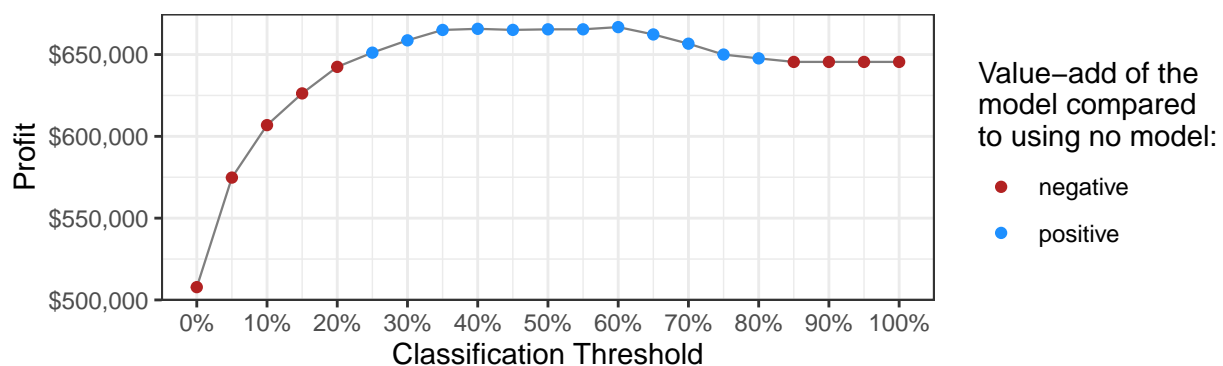


Figure 12: Profit curve for stacked model depending on classification threshold

Table 2: Maximum impact on forecasted profit for each model

Model	Profit Surplus	Profit Delta
Blended	\$21,275.50	3.30%
RF	\$21,091.50	3.27%
GB	\$20,739.50	3.21%
SVM_nonlin	\$20,413.50	3.16%
LOG	\$20,398.00	3.16%
SVM_lin	\$18,447.50	2.86%
KNN	\$13,892.00	2.15%

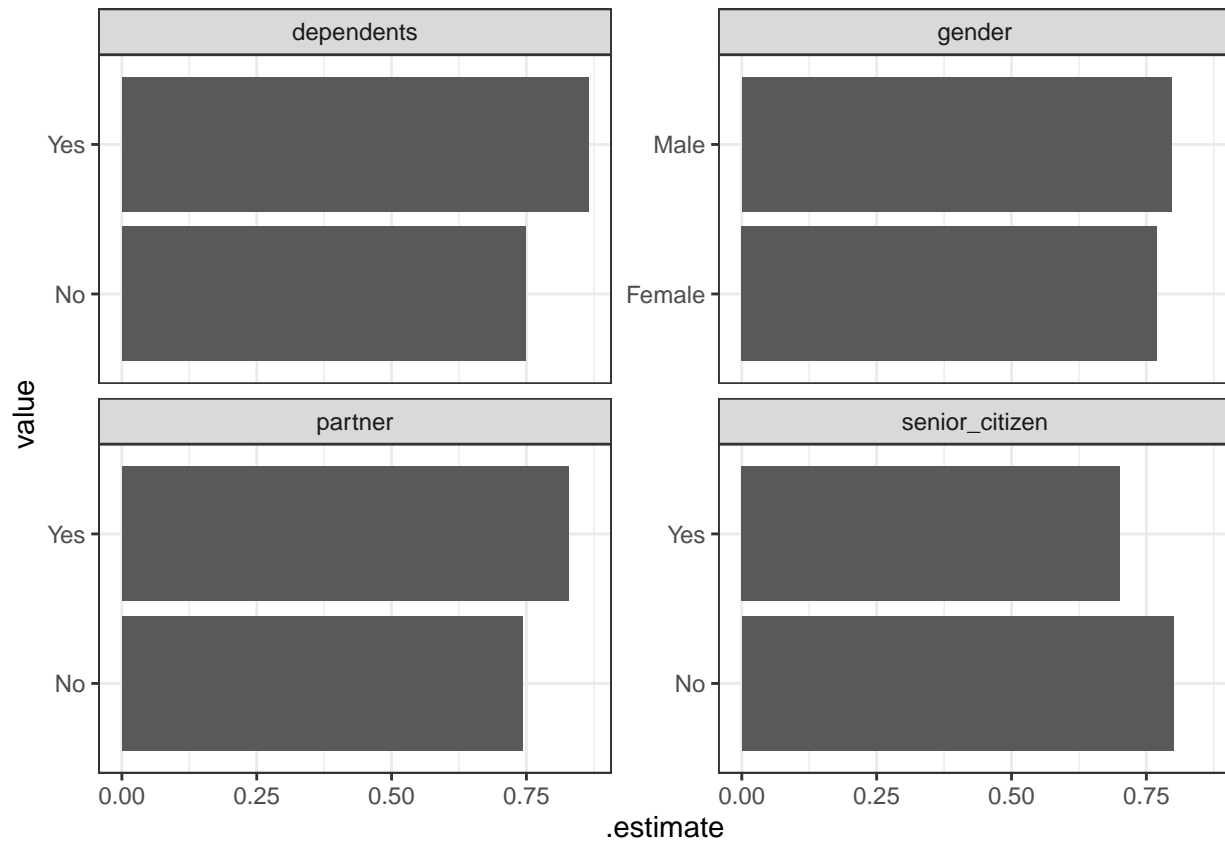
relative terms as seen in Table 2, we believe that the logistic regression would be a good starting point. When dealing with internal stakeholders in a company, explainability is often an important factor. We anticipate that the marginal monetary benefit of less than USD 1,000 will not be sufficient to make up for the increased difficulty to interpret and explain the model decisions to upper management and them having to justify it to potential external stakeholders. Furthermore, computational costs of retraining the tree-ensembles and especially the blended model, will likely make the projected economic difference between the models negligible going into the future. Therefore, as a final recommendation, weighing the pros and cons of performance evaluation and the business perspective, we recommend starting out with the logistic regression and potentially upgrading to more flexible methods, once relevant stakeholders have gotten used to our approach.

7 Ethics

TBD: Tobias

Kommentar: Habe diesen Satz hier drunter aus der Einleitung hier hin gepackt, da wir oben glaub noch nicht über Ethics reden müssen: “Due to the fictional nature of the dataset, we expect that IBM was mindful of potential biases and that the number of significant biases will thus most likely be limited.”

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

- Ahn, J.-H., Han, S.-P., & Lee, Y.-S. (2006). Customer churn analysis: Churn determinants and mediation effects of partial defection in the korean mobile telecommunications service industry. *Telecommunications policy*, 30(10-11), 552–568.
- Apply SMOTE Algorithm*. (n.d.). https://themis.tidymodels.org/reference/step_smote.html. (Accessed: 2022-11-18)
- BlastChar. (2018, Feb). *Telco customer churn*. Retrieved from <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>
- 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.
- Cognos analytics with Watson*. (2019). Retrieved from <https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-11113>
- Karanovic, M., Popovac, M., Sladojevic, S., Arsenovic, M., & Stefanovic, D. (2018). Telecommunication services churn prediction-deep learning approach. In *2018 26th telecommunications forum (telfor)* (pp. 420–425).
- Kohavi, R., et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, pp. 1137–1145).
- Zhao, M., Zeng, Q., Chang, M., Tong, Q., & Su, J. (2021). A prediction model of customer churn considering customer value: an empirical research of telecom industry in china. *Discrete Dynamics in Nature and Society*, 2021.