HarvardX PH125.9x Data Science: Capstone - Telco Churn Prediction

Edgar Pampols

7/13/2021

Contents

1 Introduction to the project

This report is written within the frame of the "HarvardX - PH125.9x Data Science - Capstone" course. The ojective is to choose a project, and apply machine data learning techniques that go beyond standard linear regression. The project chosen is to predict customer churn for a Telco company, based on a dataset made available at Kaggle.com by BlastChar: https://www.kaggle.com/blastchar/telco-customer-churn

Telco companies spend a lot in the process of acquiring customers, hence the importance of predicting churn and mitigating it if possible. The aim of the project will be to test multiple classification algorithms with the aim of predicting churners, and then have a sensible plan to retain those customers with a positive total company impact.

A typical way of retaining potential churners is to offer discounts, however offering a discount to a wrongly predicted churner will also dilute the companies revenues. In this project it is crucial to "cover a maximum of churners" but also "offer discounts to a minimum of non-churners".

2 Summary of achieved results

The dataset has been first slighlty cleaned and adapted. Next, it has been partitioned into "training","test" & "validation" partitions, the last one will only be used to compute the result of a final model, and not for training, tuning nor choosing models.

The sequence of this project will then consist on i) exploring different models, ii) evaluate them on the "test" partition accross different metrics, iii) decide for a final model and evaluate its performance on the "validation" set. To compare the different performance of the models, as well as computing the final result, multiple aspects of the "confusion matrix" will be used, such as Accuracy, Precision and Recall.

After some initial data exploration of the data properties, and the construction of multiple models based on the course material and extra research, the model of choice will be "gamLoess". The accuracy of the model is among the highest (81%) and has also a balanced precision (66%) vs. recall (57%).

This type of model will not only provide a predicted churn ("Yes" or "No"), but also will provide a probability of every customer to churn (between 0 and 1). In a real world situation, this allows us to rank the predictions, and demonstrate that our model would rather be applied only to the first 3 deciles (ordering by predicted churn probability) in order not to cannibalize existing revenues of the lower deciles. Our simulation reveals potential for saving 4.8% of the Telco's revenues.

The final complete results on the validation test show an accuracy of 81%, precision of 68% and recall of 53% and the simulation shows also how targetting the first 3 deciles only with a discounted offer would be the best thing to do. All in all, the Telco company will have saved about 4.5% of their revenues by applying the model on the validation data set.

3 Data Loading, Preparation and Exploratory analysis

3.1 Initial data loading

The dataset used for this project has been uploaded into the following github repository:

https://raw.githubusercontent.com/ecamats/edx-chooseyourown-churn/main/data/WA_Fn-UseC_-Telco-Customer-Churn.csv

After installing and loading some handy packages, and downloading the data, here's a glipmse of the raw dataset and its original 21 variables labelling:

```
## 'data.frame':    7043 obs. of  21 variables:
##  $ customerID      : chr  "7590-VHVEG" "5575-GNVDE" "3668-QPYBK" "7795-CFOCW" ..
##  $ gender          : chr  "Female" "Male" "Male" "Male" ...
##  $ SeniorCitizen   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Partner         : chr  "Yes" "No" "No" "No" ...
##  $ Dependents      : chr  "No" "No" "No" "No" ...
##  $ tenure          : int  1 34 2 45 2 8 22 10 28 62 ...
##  $ PhoneService    : chr  "No" "Yes" "Yes" "No" ...
##  $ MultipleLines   : chr  "No phone service" "No" "No" "No phone service" ...
##  $ InternetService : chr  "DSL" "DSL" "DSL" "DSL" ...
##  $ OnlineSecurity  : chr  "No" "Yes" "Yes" "Yes" ...
##  $ OnlineBackup    : chr  "Yes" "No" "Yes" "No" ...
##  $ DeviceProtection: chr  "No" "Yes" "No" "Yes" ...
##  $ TechSupport     : chr  "No" "No" "No" "Yes" ...
##  $ StreamingTV     : chr  "No" "No" "No" "No" ...
##  $ StreamingMovies : chr  "No" "No" "No" "No" ...
##  $ Contract        : chr  "Month-to-month" "One year" "Month-to-month" "One y"..
##  $ PaperlessBilling: chr  "Yes" "No" "Yes" "No" ...
##  $ PaymentMethod   : chr  "Electronic check" "Mailed check" "Mailed check" "B"..
##  $ MonthlyCharges  : num  29.9 57 53.9 42.3 70.7 ...
##  $ TotalCharges    : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ Churn           : chr  "No" "No" "Yes" "No" ...
```

Basically each record represents a unique subscriber monthly data, with a series of demographic information, or some Telco data such as tenure, contract duration and charges. Multiple "Yes/No" flags and categorical variables are also available describing which type of products and services the subscriber has subscribed to.

The complete dataset consists of 7,043 subscribers.

3.2 Further data preparation and partitioning

In order to easily uniform data for exploration purposes we defined a couple of handy functions:

```
YesToOneFunction <- function(x){
  ifelse(x=="Yes", 1,0)
}

NoServiceToNo <- function(x){
  ifelse(x=="Yes", "Yes","No")
}
```

Before getting into exploration and modeling, a few mutations and reformatting of the data have been done to ease the analysis:

- Removal of 11 records containing NAs (too small of a portion to worry about them)
- Uniforming product flags for better visualization (mix of 1/0's and Yes/No fiels)
- Creation of new "insightful variables" (i.e. % of current contract completion)

In terms of data partitioning, we will split the clean dataset into three parts:

- A part for "validation" as 20% of all dataset, this will be used as final hold-out test
- The remaing 80% will be further split into a 20% for "testing" and 80% for "training"

```
#split of validation set out of the initial set

val_index <- createDataPartition(transformed_churn$Churn, times = 1, p = 0.2, list = FALSE)
clean_churn = transformed_churn %>% dplyr::slice(-val_index)

#clean_churn set will be kept for exploration purposes keeping original labels

test_index <- createDataPartition(clean_churn$Churn, times = 1, p = 0.2, list = FALSE)

#re-split of the non-validation part into training and test
```

For exploration purposes we will use the data as previously prepared, that's for all except "validation" data (please note we are treating validation set as an unknown portion of data till the final test). However, since we want to test multiple models using the convenience of "caret package" we will do a further transformation to all sets.

This will be particularly relevant for categorical variables that will be transformed into "dummy variables" consisting on 1's or 0's for every class in that variable. We can use the "dummyVars" function for that purpose, and our data will be ready to apply to several models at once.

```
dummies_model <- dummyVars(Churn ~ ., data=transformed_churn, fullRank = TRUE)
transformed_churn_mat <- predict(dummies_model, newdata = transformed_churn)

transformed_churn <- data.frame(transformed_churn_mat)
transformed_churn$Churn <- y

str(transformed_churn,strict.width="cut")
```

```
## 'data.frame':    7032 obs. of  29 variables:
##  $ genderMale                           : num  0 1 1 1 0 0 1 0 0 1 ...
##  $ SeniorCitizen                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PartnerYes                           : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ DependentsYes                        : num  0 0 0 0 0 0 1 0 0 1 ...
##  $ tenure                               : num  1 34 2 45 2 8 22 10 28 62 ...
##  $ PhoneServiceYes                      : num  0 1 1 0 1 1 1 0 1 1 ...
##  $ MultipleLinesYes                     : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ InternetServiceFiber.optic           : num  0 0 0 0 1 1 1 0 1 0 ...
##  $ InternetServiceNo                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ OnlineSecurityYes                    : num  0 1 1 1 0 0 0 1 0 1 ...
##  $ OnlineBackupYes                      : num  1 0 1 0 0 0 1 0 0 1 ...
##  $ DeviceProtectionYes                  : num  0 1 0 1 0 1 0 0 1 0 ...
##  $ TechSupportYes                       : num  0 0 0 1 0 0 0 0 1 0 ...
##  $ StreamingTVYes                       : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ StreamingMoviesYes                   : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ ContractOne.year                     : num  0 1 0 1 0 0 0 0 0 1 ...
##  $ ContractTwo.year                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PaperlessBillingYes                  : num  1 0 1 0 1 1 1 0 1 0 ...
##  $ PaymentMethodCredit.card..automatic.: num  0 0 0 0 0 0 1 0 0 0 ...
##  $ PaymentMethodElectronic.check        : num  1 0 0 0 1 1 0 0 1 0 ...
##  $ PaymentMethodMailed.check            : num  0 1 1 0 0 0 0 1 0 0 ...
##  $ MonthlyCharges                       : num  29.9 57 53.9 42.3 70.7 ...
##  $ TotalCharges                         : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ tenure_years                         : num  0.0833 2.8333 0.1667 3.75 0.1667..
```

```
##  $ Contract_length                    : num  1 12 1 12 1 1 1 1 1 12 ...
##  $ Contract_cycles                    : num  1 2.83 2 3.75 2 ...
##  $ Contract_remainig                  : num  1 2 1 3 1 1 1 1 1 10 ...
##  $ Contract_completion                : num  0 0.8 0 0.8 0 0 0 0 0 0.2 ...
##  $ Churn                              : Factor w/ 2 levels "Yes","No": 2 2 1 ..
```

```
val_churn = transformed_churn %>% dplyr::slice(val_index)
model_churn = transformed_churn %>% dplyr::slice(-val_index)

test_set = model_churn %>% dplyr::slice(test_index)
train_set = model_churn %>% dplyr::slice(-test_index)
```

3.3 Selected exploratory analysis

Since the event we are trying to predict / prevent has a binary outcome "Yes" or "No" it will come handy to define an event rate, that we will call "churn rate". Churn rate is defined as the proportion of subscribers churning out of a group or a segment.

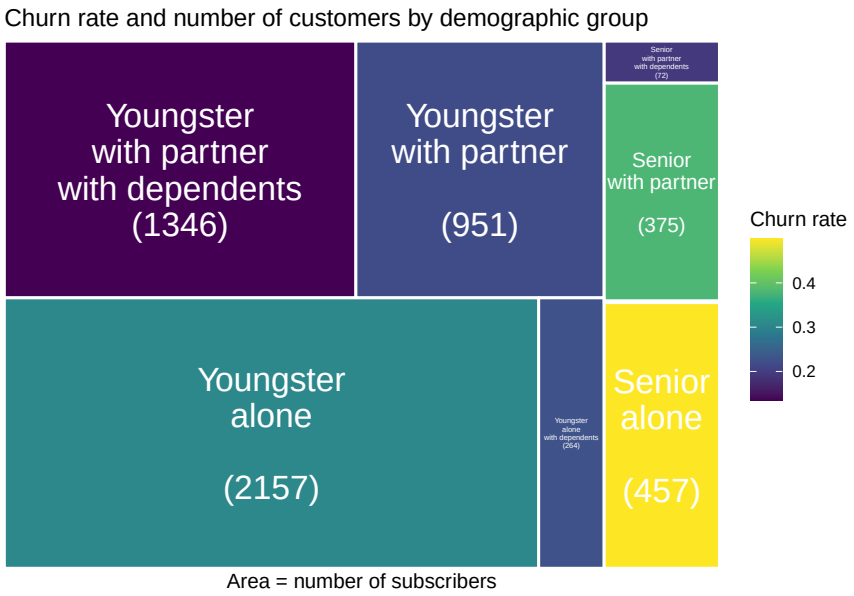
For example, in the code below we can see that, out of the 5,625 subscribers in our dataset (we have excluded "validation" here), 1,495 will churn. This leads to an average "churn rate" in our dataset of 0.266 or ~27%.

```
## # A tibble: 1 x 4
##      No   Yes CustomerCount churn_rate
##   <dbl> <dbl>         <dbl>      <dbl>
## 1  4130  1495          5625      0.266
```

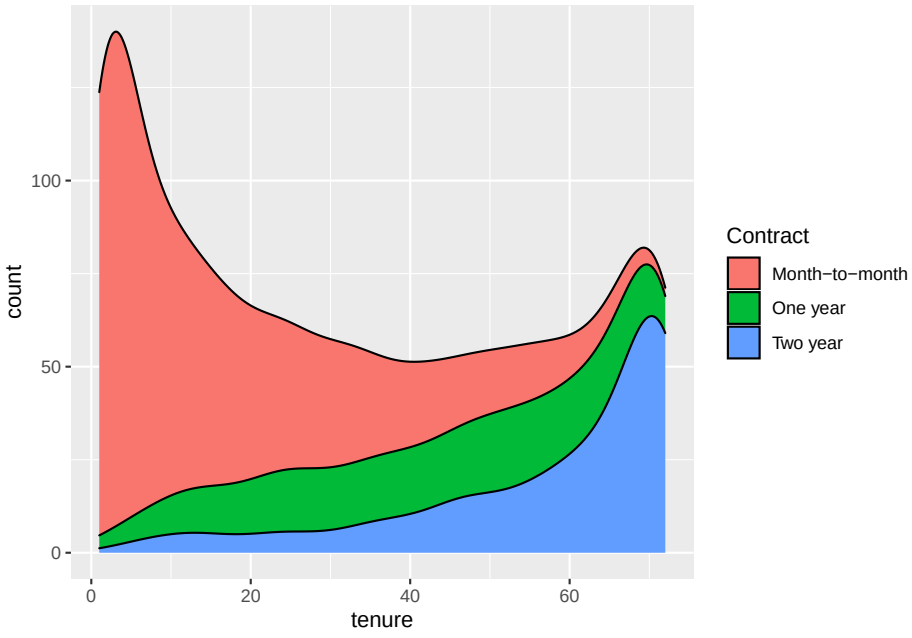
Following this logic, we can slice our data in different dimensions and compare the "churn rates" accordingly, like in the below example showing gender split. In here we see that "Female" customers have a slightly higher churn rate than "Male" customers.

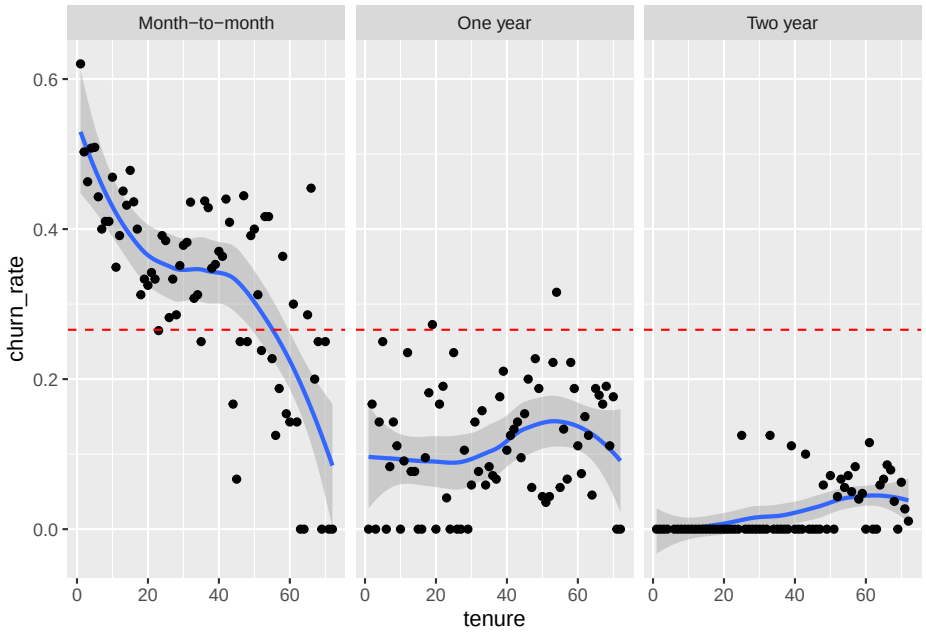
gender	No	Yes	CustomerCount	churn_rate
Female	2026	760	2786	0.2727925
Male	2104	735	2839	0.2588940

To explore both the distribution and the churn rate among different socio-demographic variables, we will use the handy "geom_tree" plot. It seems clear that "Senior citizens" have higher churn rates, and this phenomena is exaggerated in the case of 1-person households, without partner nor dependents (where churn rates are around ~50%, that's double the base average).



The Telco customer base is predominantly composed by customers under monthly "pay as you go" contracts, that have a rather low tenure at this time of measurement. It doesn't come as a surprise that the churn rate of monthly contracts is far superior to the one of 1-2 year contracts (the red line indicates the average churn rate of the base). In fact the customer groups with low tenure have the highest churn rates.

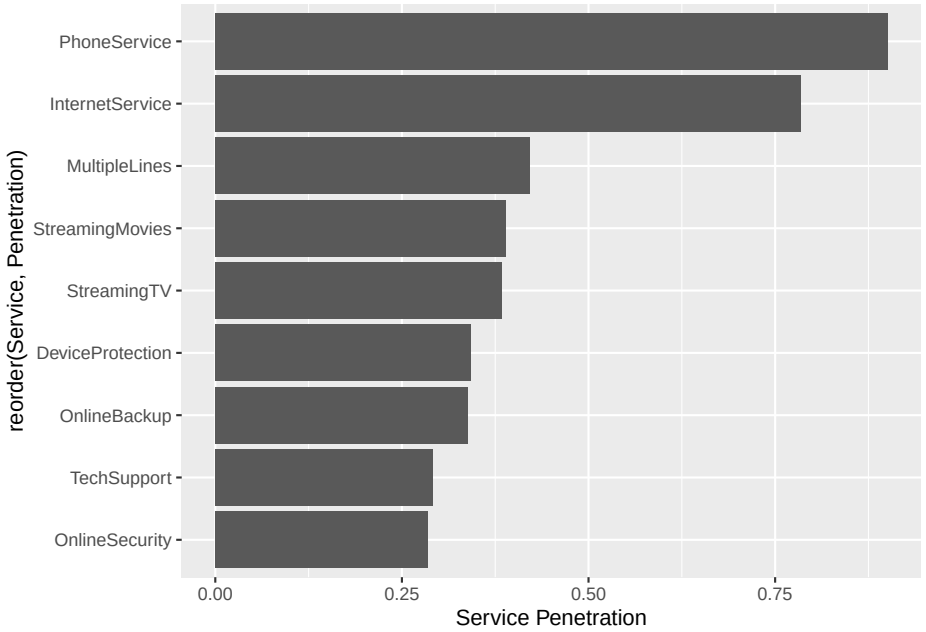




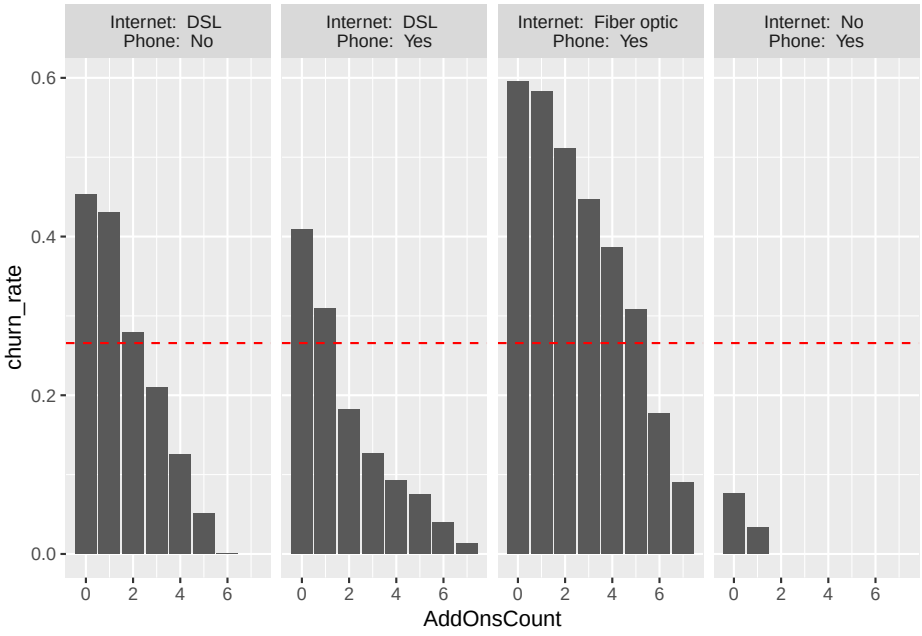
We now will take a look at the product-related information in the dataset. First we observe that each customer can have a different number of combined services:

- Two main services "Phone" & "Internet" (in different supports like "DSL" or "Fiber Optics")
- A series of "optional" services or add-ons (Online Backup, Streamin Movies or Multiple Lines...)
- Add-ons are conditional to some underlying services (i.e. to get Multiple Lines, Phone is needed)

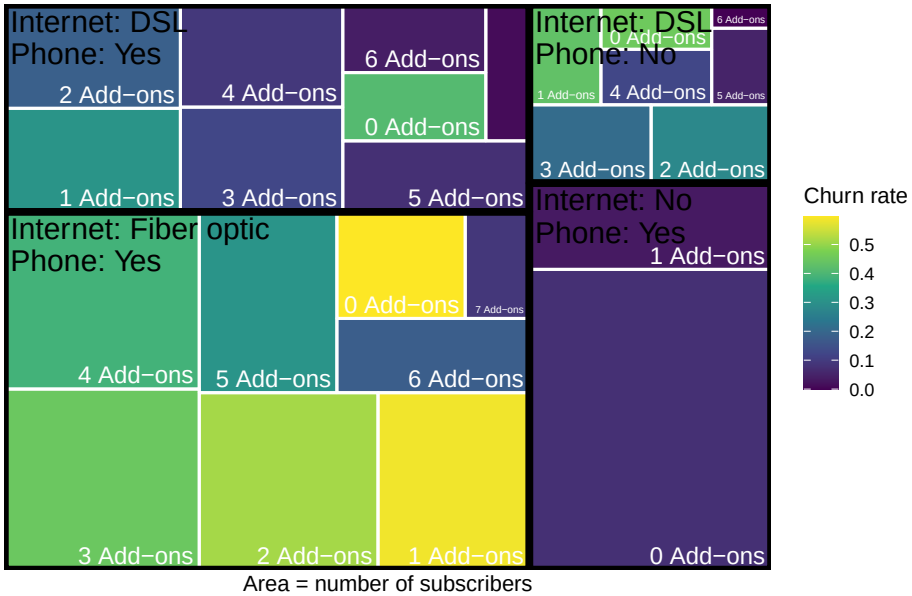
The popularity (or penetration in the base) of those add-ons oscillates between 30-40% of the base, which means they all could be meaningful for our analysis.



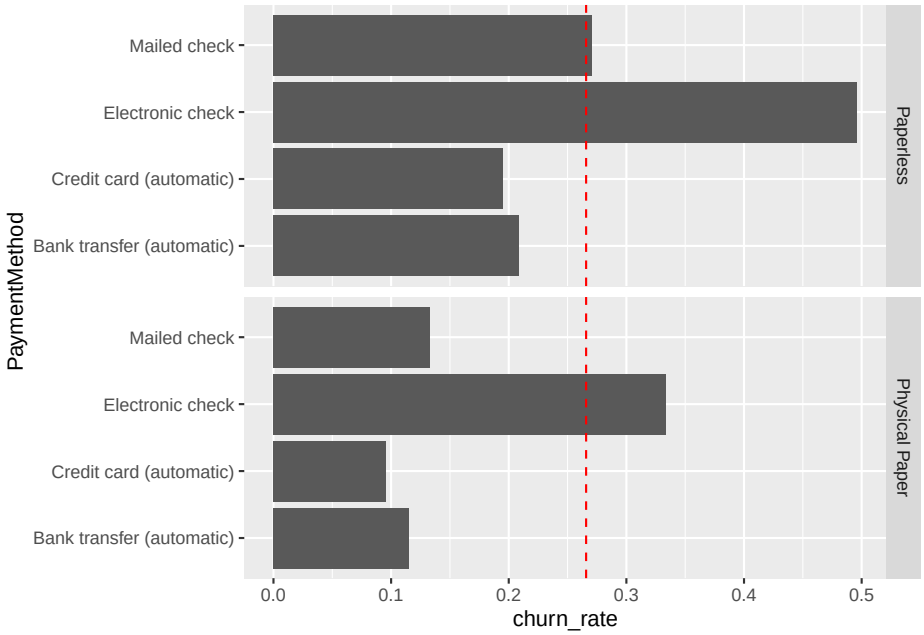
From the below couple of analysis it seems that the more add-ons a subscriber has, the less likely this customer will be to churn. As a matter of fact, the main service seems to matter less than the add-ons when it comes to drive churn, as we can see pretty simple products (Phone only) having less churn than Fiber Optics. The highest subgroup churn pockets in the base are from customers with Fiber Optic + Phone + 0 or 1 add-ons only.



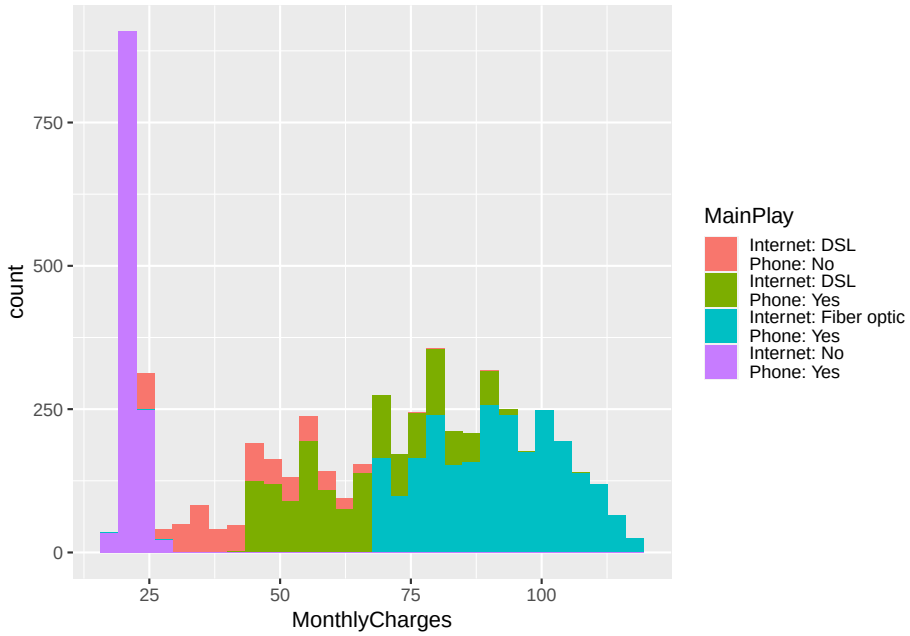
Churn rate and number of customers by product type and number of add−ons

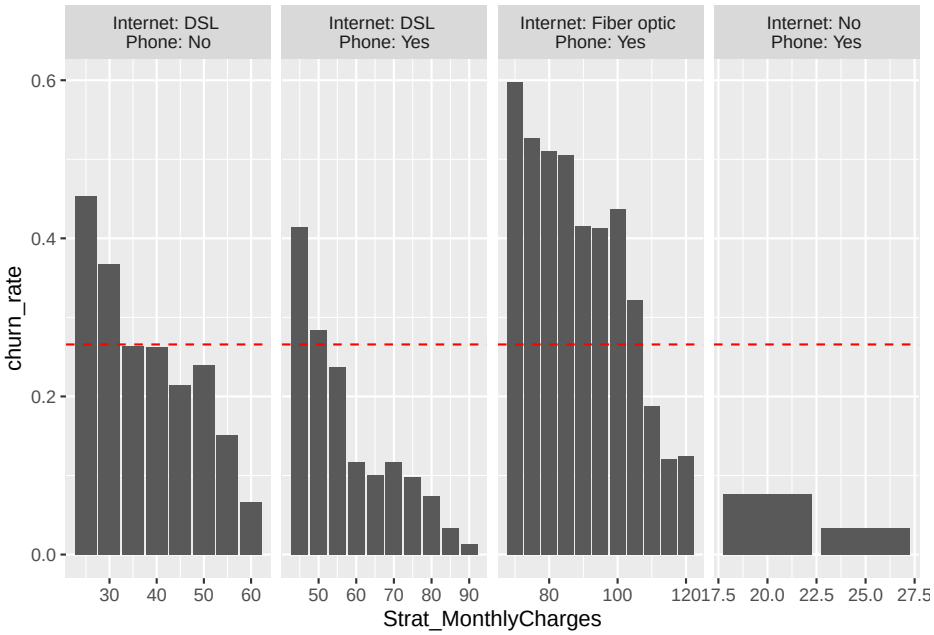


Our dataset details also 4 different payment methods for customers and two bill modalities (Paper and Paperless). We can observe how "Electronic check" seems to drive higher churn rate than other methods, as well as an overall higher churn rate for customers opting for "Paperless" billing when compared accross categories.



We will not be exploring Total Charges as they are mainly a function of MonthlyCharges and tenure, however Monthly Charges show insightful relationships. Monthly Charges are strongly linked to the type of products chosen and to the number of add-ons opted-in by every customer, so it makes sense that customers with lowest Monthly Charges among a main product also display the highest levels of churn.





All in all, we get a pretty good understanding of the variables in our dataset, and find meaningful relationships that should bring some predicting power to our models. Unless computational limitations arise, we will keep all possible variables as part of our models.

4 Initial evaluation of different models

We have already partitioned the data into "training" and "test" as well as prepared it in a friendly way to evaluate a bunch of models with the "caret" package.

Before going into more sophisticated models, we will create a simple model that will consist in randomly guessing if a customer will churn ("Yes") or not churn ("No"). We use this model to illustrate how we will collect model performance for subsequent tests. Being a classification problem it makes sense to look at accuracy, but since we have associated costs to False Predictions (False Positives and False Negatives), we will also be looking at a balanced Precision and Recall for each model.

```
prediction <- as.factor(sample(c("Yes","No"),nrow(test_set),replace = TRUE))
probability <- sample(c(1,0),nrow(test_set),replace = TRUE)

model_results <- data_frame(method = "random guessing",
        Accuracy = confusionMatrix(prediction, test_set$Churn)$overall["Accuracy"],
        Precision = confusionMatrix(prediction, test_set$Churn)$byClass["Precision"],
        Recall = confusionMatrix(prediction, test_set$Churn)$byClass["Recall"])

prediction_results <- data_frame(guess = prediction)

probability_results <- data_frame(guess = probability)

model_results %>% knitr::kable()
```

method	Accuracy	Precision	Recall
random guessing	0.5288889	0.2903811	0.5351171

We will create some data frames where we will be storing the results of our different models:

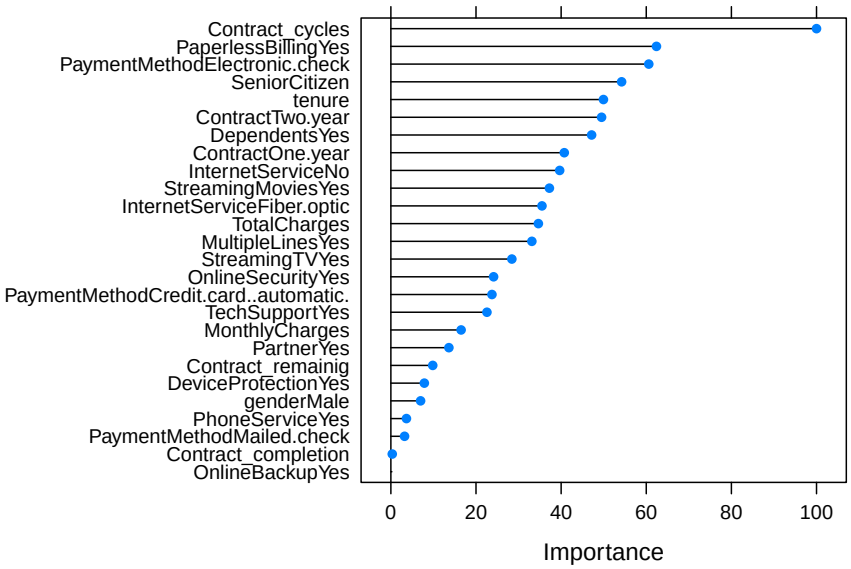
- Results dataframe: Accuracy, Precision and Recall
- Probabilities dataframe: here we will store the probabilities
- Predictions dataframe: the predicted "Yes" / "No" values (i.e. "Yes" if probabilty is >0.5)

4.1 Fitting a group of models using caret package

```
models <- c("glm","lda","naive_bayes","rpart","knn","gamLoess")

fits <- lapply(models, function(model){
  train(Churn ~ ., method = model,data = train_set)
})
```

method	Accuracy	Precision	Recall
random guessing	0.5288889	0.2903811	0.5351171
glm	0.8044444	0.6525097	0.5652174
lda	0.7946667	0.6297710	0.5518395
naive_bayes	0.7093333	0.4736842	0.8428094
rpart	0.7955556	0.6385542	0.5317726
knn	0.7644444	0.5726496	0.4481605
gamLoess	0.8071111	0.6601562	0.5652174

Some of those give us already pretty good results, we will conduct an additional test to check how relevant are some of the variables involved. If we run, for example, glm with only top 5 variables, we observe how our accuracy results decrease from ~80% to ~78%, still very close but worth having those extra ~2 percent points.

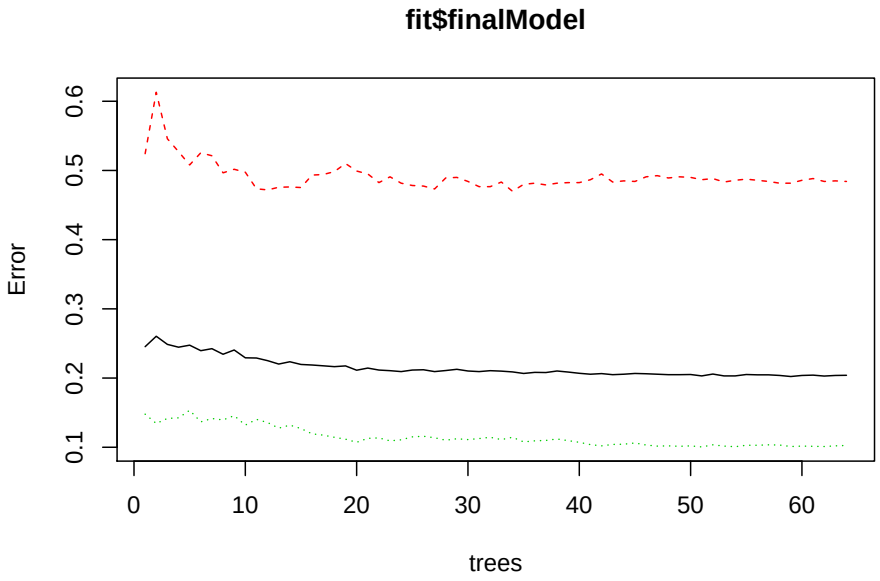


We conclude that using more variables has useful predicting power for our project, but if in need to decrease calculation efforts, a few variables can already provide good results.

method	Accuracy	Precision	Recall
glm	0.8044444	0.6525097	0.5652174
glm (5 top vars)	0.7760000	0.6192893	0.4080268

4.2 Additional set of models tested

To add to our first test, we also test the following models with dedicated pieces of code. For some of those we have fixed some parameters to reduce computation time (i.e. number of trees) or we have created additional pieces of code to accomodate data to the non-caret functions used. These are Random Forest, Support Vector Machine and Adaptative Boosting.



Here's an example of our Random Forest error reducing with number of trees and stabilizing after ~64 trees. We didn't excel the results of our previous group of models, and those last techniques took a lot of training time comparatively.

method	Accuracy	Precision	Recall
random guessing	0.5288889	0.2903811	0.5351171
glm	0.8044444	0.6525097	0.5652174
lda	0.7946667	0.6297710	0.5518395
naive_bayes	0.7093333	0.4736842	0.8428094
rpart	0.7955556	0.6385542	0.5317726
knn	0.7644444	0.5726496	0.4481605
gamLoess	0.8071111	0.6601562	0.5652174
glm (5 top vars)	0.7760000	0.6192893	0.4080268
rf (ntree = 64)	0.8035556	0.6547619	0.5518395
svmLinear2	0.8017778	0.6623932	0.5183946
adaboost (nIter = 100)	0.7920000	0.6235741	0.5484950

4.3 Combining our models (voting ensemble)

We already got a few candidates to explore further, but we will also test combining all our models in one by creating a simple voting system. The results are good but there is no drastic improvement when compared to our best models.

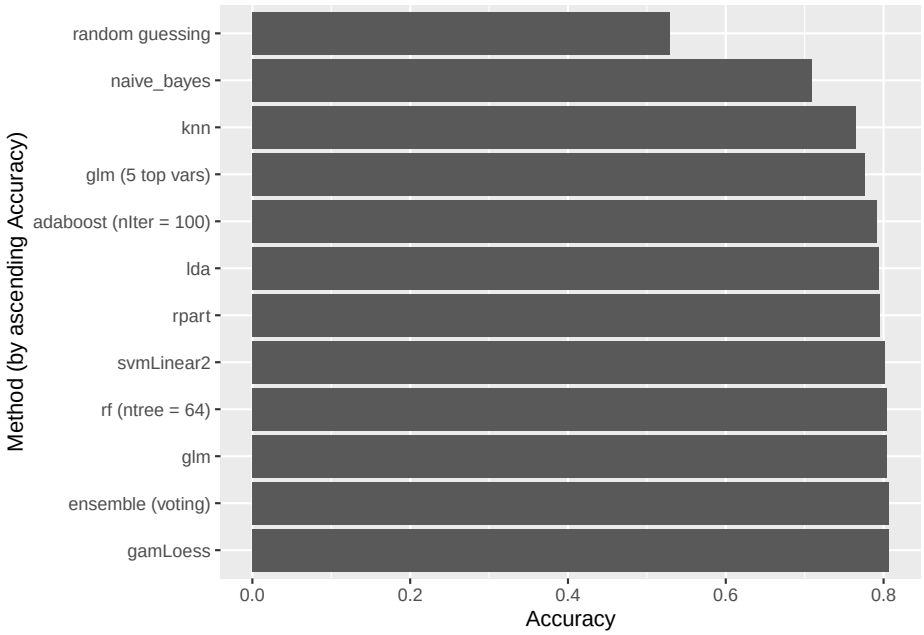
```
votes <- data_frame(votes = prediction_results %>% select(-guess) %>%
  mutate(across(.cols = everything(), YesToOneFunction)) %>% rowMeans)

prediction <- votes %>% mutate(prediction = as.factor(ifelse(votes > 0.5,"Yes","No"))) %>% pull(predict
```

At this stage, we will proceed to compare our models and decide on a good candidate to pursue into our final predictions.

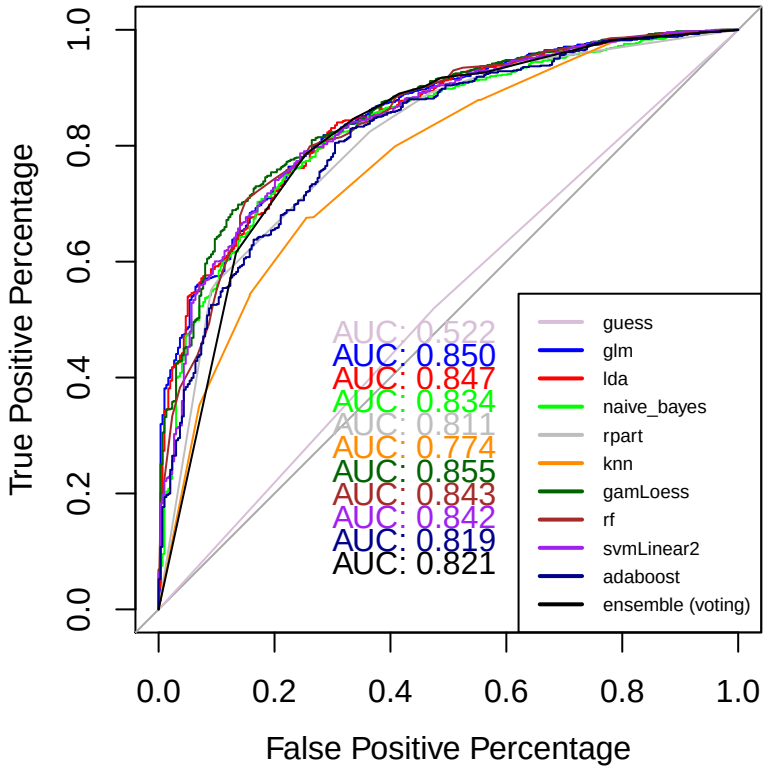
5 Model comparison, selection and tuning

5.1 Accuracy, ROC curves and AUC

Below is a comparison of all our models and ensembles sorted by achieved accuracy. Besides our ensemble voting, GamLoess, glm and Random Forest seem to show the highest accuracy levels on our test dataset.



Since we do not only care about accuracy but about good balance between positive and negative predictions, we will also use the ROC and AUC visual methods to compare our models. The pROC package provides a very useful function to display and overlay ROC curves for distinct models.



Our ROC plot confirms that GamLoess is the best performing individual model, with the greatest AUC ("Area Under the Curve") and has a good and steep "learning curve" within the first portion of our dataset. This is a feature that we are looking for since in practical terms we will seek to prioritize which groups of customers in our predictions we want to target with a discounted promotion.

5.2 Further tuning of gamLoess model

By inspecting our previous default gamLoess model, we observe that the span is set at 0.5. In order to fine tune the model further we create a grid of span values and re-train the model. There is very marginal improvement with our new span value of ~0.5 as well, so we will stick to our default model for the final tests.

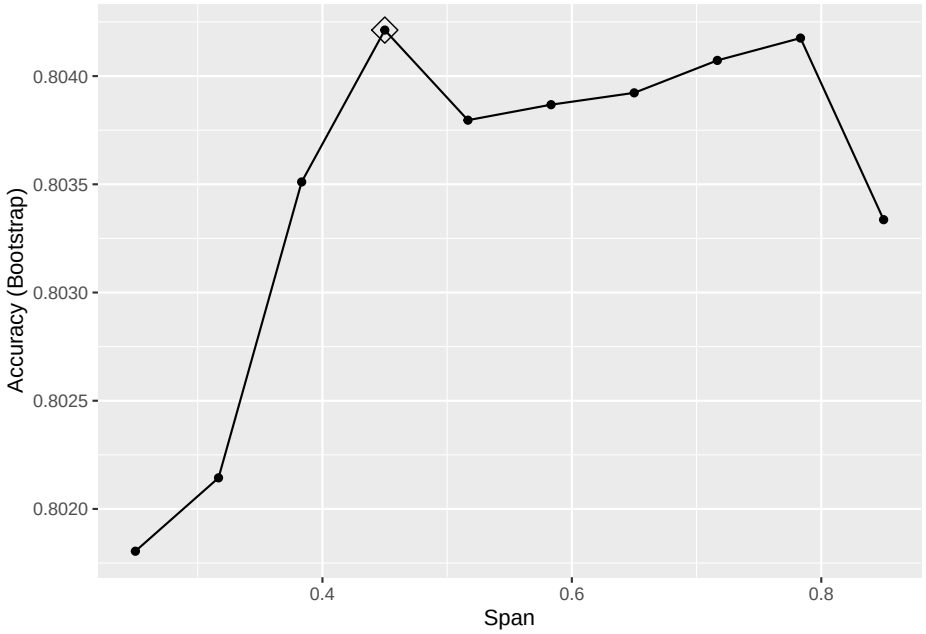
```
modelLookup("gamLoess")
```

```
##      model parameter  label forReg forClass probModel
## 1 gamLoess      span   Span   TRUE     TRUE      TRUE
## 2 gamLoess    degree Degree   TRUE     TRUE      TRUE
```

```
fits$gamLoess$bestTune
```

```
##   span degree
## 1  0.5      1
```

```
grid <- expand.grid(span = seq(0.25, 0.85, len = 10), degree = 1)
```

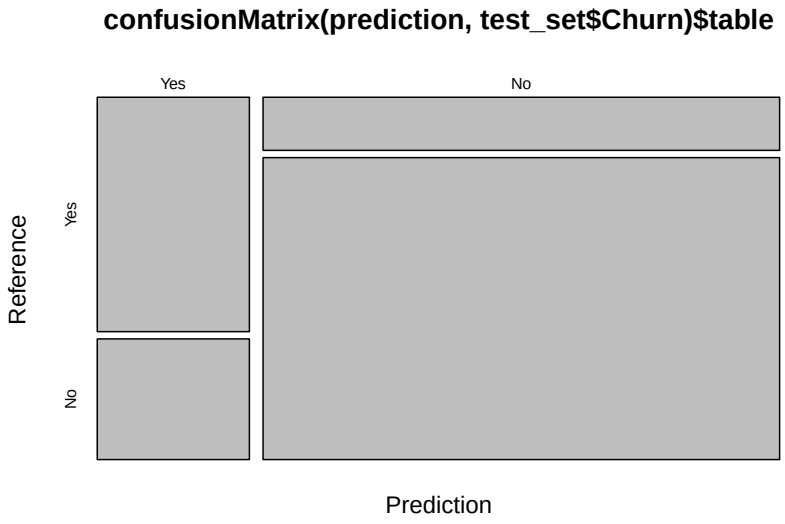


6 Final model: results and outputs

6.1 Confusion matrix and lift analysis

Since this is a classification problem, a very common way to illustrate the performance of our model is to use the cunfusion matrix. Below are the results for our selected model.

method	Accuracy	Precision	Recall
gamLoess (final)	0.8071111	0.6601562	0.5652174

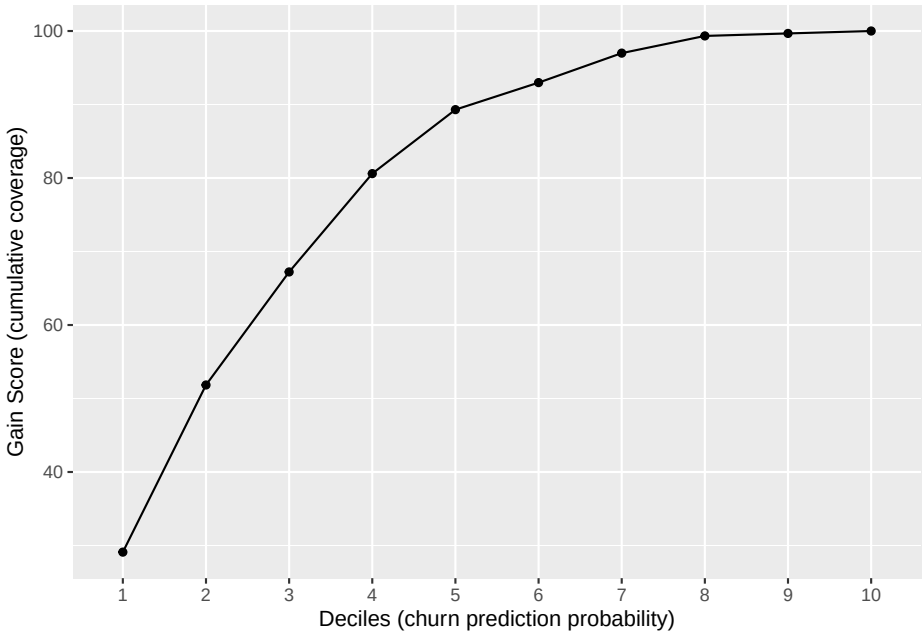


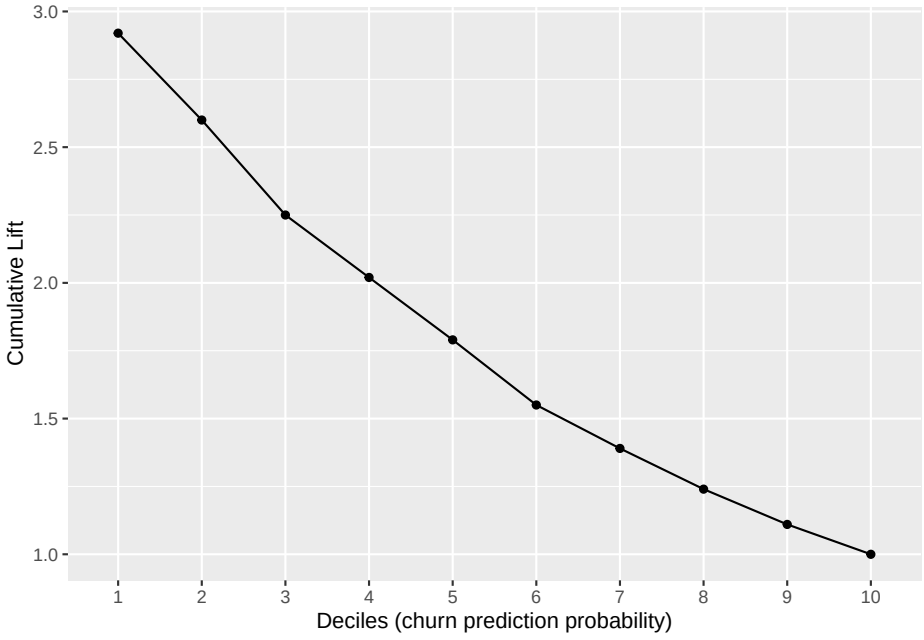
Once we will obtain our predictions, we can use the probabilities of churn to analyze our customer base in a sorted way. The below analysis is performed by deciles (i.e. decile 1 are the 10% customers with the highest probability to churn) and shows both gain and lift per decile:

- Gain score analysis tells us, for example, that first 4 deciles would already capture 80% of our "churners"
- Cumulative lift helps us understand how better a portion of our predictions are compared to the whole (i.e. the very first decile is 3x better than the overall "churn detection" on the entire set)

The differences of "churn rate" accross deciles are very noticeable: for example the first decile is composed of churners by 78% whereas last deciles have barely a 1% of churners and should not be disturbed with promotions.

Decile	Subscribers	Churners	Churn_Rate	Cum_Churners	Gain_Score	Cum_Lift
1	112	87	0.78	87	29.10	2.92
2	112	68	0.61	155	51.84	2.60
3	112	46	0.41	201	67.22	2.25
4	112	40	0.36	241	80.60	2.02
5	112	26	0.23	267	89.30	1.79
6	113	11	0.10	278	92.98	1.55
7	113	12	0.11	290	96.99	1.39
8	113	7	0.06	297	99.33	1.24
9	113	1	0.01	298	99.67	1.11
10	113	1	0.01	299	100.00	1.00
Total	1125	299	0.27	299	100.00	1.00





From such analysis we see how we have an opportunity to prioritize and focus our promotional campaigns into the first deciles of our base. In order to make the simulation more realistic, we have created a couple of assumptions to reflect real business performance and implications.

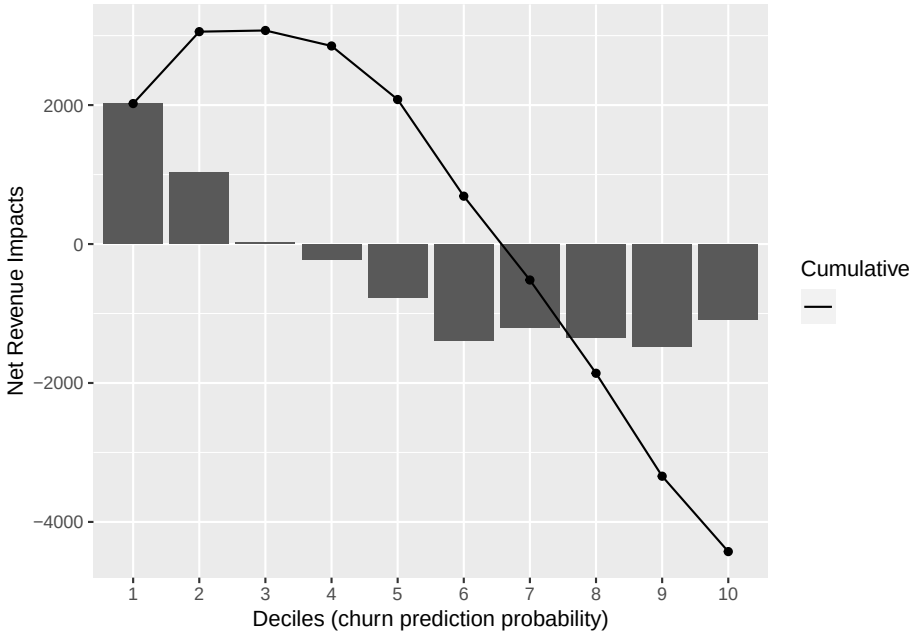
6.2 Real business impact simulation

In case a customer is likely to churn, we typically would offer a discount so that there are some chances of retaining that customer. For that purpose we create an hypothetical promotion of -30% on the current Monthly Charges, and also we will define some logical take-up rates for this offers:

- For a "churner", there will be around 50% chances to retain him/her with the -30% discount
- If we offer (by mistake) the discount to a non-churner, there will be greater chances (80%)

The below simulation results will help us find a "decile cut-off" for us to narrow down the communication of this offers and make a more efficient campaign.

Decile	RevLoss_DoNothing	RevLoss_DoSomething	NetImpact	Gain_Score	Cum_NetImpact
1	-7164	-5142	2022	29.10	2022
2	-5261	-4226	1035	51.84	3057
3	-3582	-3565	17	67.22	3074
4	-2903	-3126	-223	80.60	2851
5	-1700	-2471	-771	89.30	2080
6	-768	-2159	-1391	92.98	689
7	-758	-1965	-1207	96.99	-518
8	-393	-1735	-1342	99.33	-1860
9	-63	-1545	-1482	99.67	-3342
10	-46	-1131	-1085	100.00	-4427




```
## [1] 4.826432
```

The simulation indicates that it is not sensible to extend our churn prevention offers beyond decile 3 since we risk destroying more value on loyal customers than we create by saving a few churners. That means covering 67% of the churners and making less mistakes by discounting products to "non-churners" (false positives).

If correctly executed, our plan will impact the business by improving by 4.8% the current revenues, that's compared to doing nothing and let customers churn.

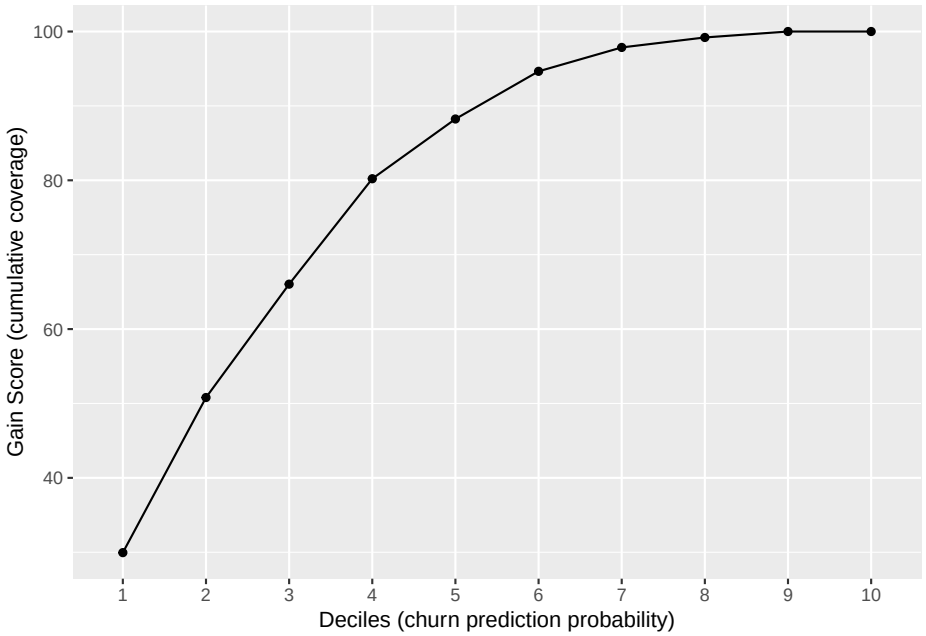
7 Final model training, hold-out test and results

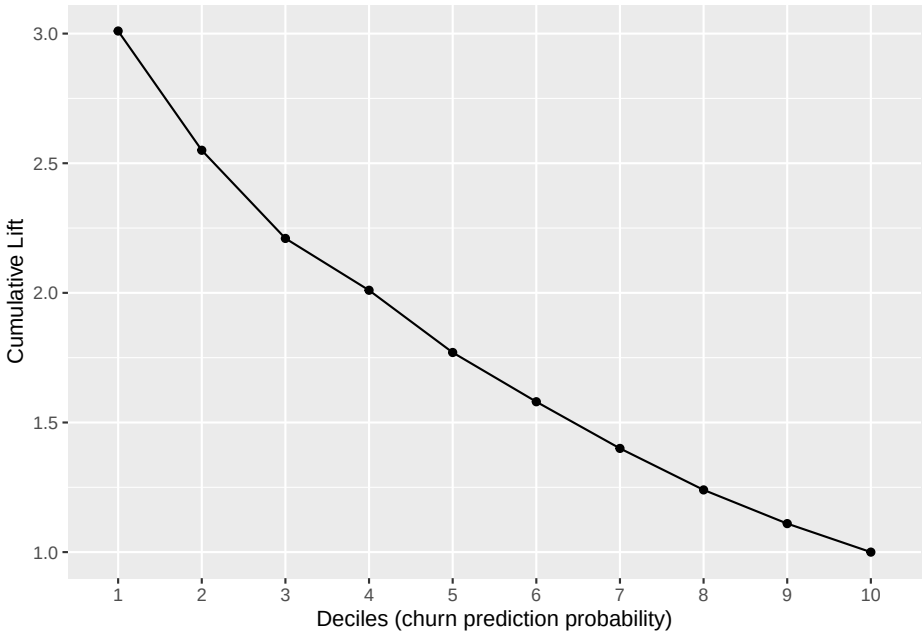
Once we have our churn strategy clear, we can proceed to retrain the selected model with all the possible data (except the validation chunk). This will make the most of the known data in terms of learning, and we will test our final model against the validation set that we isolated since the beginning of the project.

```
fit <- train(Churn ~ ., method = "gamLoess",data = model_churn)
```

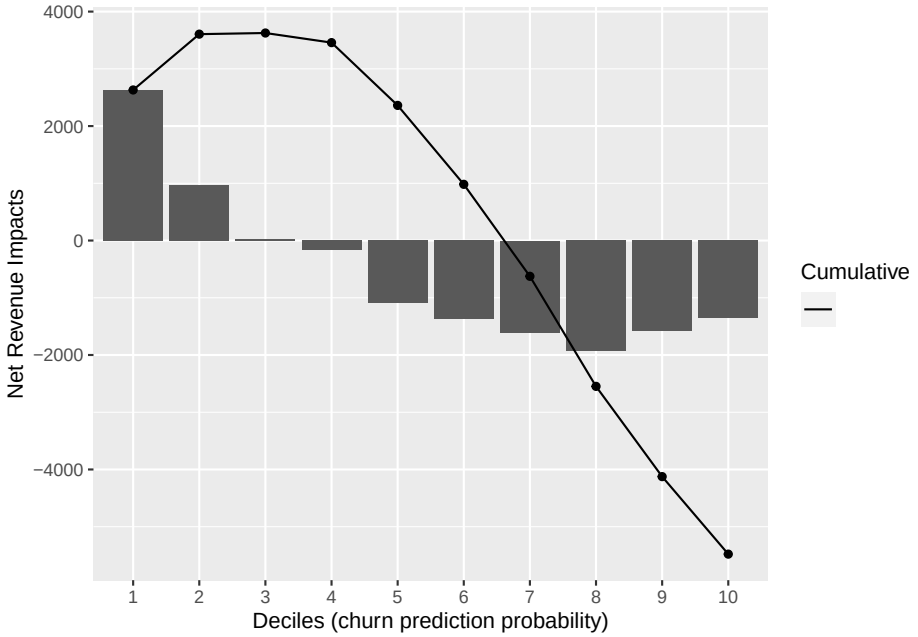
method	Accuracy	Precision	Recall
gamLoess (holdout)	0.8081023	0.6793103	0.526738

Decile	Subscribers	Churners	Churn_Rate	Cum_Churners	Gain_Score	Cum_Lift
1	140	112	0.80	112	29.95	3.01
2	140	78	0.56	190	50.80	2.55
3	140	57	0.41	247	66.04	2.21
4	141	53	0.38	300	80.21	2.01
5	141	30	0.21	330	88.24	1.77
6	141	24	0.17	354	94.65	1.58
7	141	12	0.09	366	97.86	1.40
8	141	5	0.04	371	99.20	1.24
9	141	3	0.02	374	100.00	1.11
10	141	0	0.00	374	100.00	1.00
Total	1407	374	0.27	374	100.00	1.00





Decile	RevLoss_DoNothing	RevLoss_DoSomething	NetImpact	Gain_Score	Cum_NetImpact
1	-9072	-6441	2631	29.95	2631
2	-6037	-5062	975	50.80	3606
3	-4305	-4285	20	66.04	3626
4	-4018	-4186	-168	80.21	3458
5	-1984	-3081	-1097	88.24	2361
6	-1678	-3057	-1379	94.65	982
7	-774	-2381	-1607	97.86	-625
8	-355	-2278	-1923	99.20	-2548
9	-138	-1715	-1577	100.00	-4125
10	0	-1354	-1354	100.00	-5479




```
## [1] 4.450808
```

The results on the validation set are very much in line with our last simulations, and are as follows:

- A model accuracy of ~81% (with ~68% precision and ~53% recall)
- A lift of 3x for the very first decile
- A positive net impact until the 3rd decile
- A total revenue impact of 4.5% (vs. ~4.8% on the test data)

8 Final conclusions

Several of the most common classification models provided usable results when ran on default parameters. Some of the models had a very long computing time for a marginal improved accuracy, so we could privilege simple models depending on the business requirements. Also, the consideration of multiple variables vs. a few seems to provide additional prediction power, hence value creation for the Telco company.

For such a critical customer event like churn, an approach to predict succesfully ~80% of churn status in the base proves to be very powerful, and the final revenue impact result (~4-5%) of revenues is considerable. The final approach also is quite simple as only 30% (3 deciles) of the base will be have to be addressed in order to preserve value, making the Telco save money and time (vs. a blanket approach).

Given the homogeneity of some initial results, a further study could explore tuning of different techniques or more fancy ways to ensemble a few models to improve results further.

Finally, the model should be monitored for subsequent months, as our available training data consists of only one period of one month. It could very well be that year seasonality has a role to play in churn as well. Our model could also be nurtured with additional data or from more sophisticated engineered variables.