Havardx Data Science - Capstone Project: Churn Prediction

Daniel Handojo

04/14/2021

Contents

1. Introduction

This data science project is all about churn. To be particular, we use the telecom users dataset provided from Radmir Zosimov by Kaggle in order to predict whether a customer will renew his contract based on various variables such as: their demographic characteristics, the services they use, the duration of using the operator's service, the method of payment, and the amount of payment. We are provided with data of almost six thousand customers and 19 different variables. As for any business, it is especially costly to lose a subscriber or customer to its competition. Therefore, predicting whether a customer will renew their contract or cancel is the first step for a company in order to take necessary measurements such as new attractive deals, discounts or even cross sell.

Thus, the goal for this project is to develop a classification model that is capable of especially predicting customers who will churn. For this reason, we won't look only at the accuracy of our models but rather focus on the balanced accuracy. That is, the average of the sensitivity and specificity. By doing so, we will give the correct predictions of churners as well as non-churners equal weight.

In this project, we will first clean and analyze the data to get an overall understanding of the variables and what they could mean. Further, we use our insights to develop some of our own features by combining variables. Next, we are ready to build our model. In this project, logistic regression, random forest, boosting, lasso regression, ridge regression and naive bayes will be performed. We are also going to combine all models into an ensemble model and see whether that will outperform the individual models. Of course we will fine tune every model as best as we can. Throughout our data science journey we will see which hypotheses and which variables will be most important to our models and which variables are not important at all. With that knowledge, our telecom business has a better understand which variables are good predictors for a churn. In the end, the model with the highest balanced accuracy wins.

1.1 Data cleaning

First glimpse into the dataset. Our Data was imported correctly.

Here is a description of our variables in the dataset.

customerID - customer id
gender - client gender (male / female)
SeniorCitizen - is the client retired (1, 0)
Partner - is the client married (Yes, No)
tenure - how many months a person has been a client of the company
PhoneService - is the telephone service connected (Yes, No)
MultipleLines - are multiple phone lines connected (Yes, No, No phone service)
InternetService - client's Internet service provider (DSL, Fiber optic, No)
OnlineSecurity - is the online security service connected (Yes, No, No internet service)
OnlineBackup - is the online backup service activated (Yes, No, No internet service)
DeviceProtection - does the client have equipment insurance (Yes, No, No internet service)
TechSupport - is the technical support service connected (Yes, No, No internet service)
StreamingTV - is the streaming TV service connected (Yes, No, No internet service)
StreamingMovies - is the streaming cinema service activated (Yes, No, No internet service)
Contract - type of customer contract (Month-to-month, One year, Two year)
PaperlessBilling - whether the client uses paperless billing (Yes, No)
PaymentMethod - payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
MonthlyCharges - current monthly payment
TotalCharges - the total amount that the client paid for the services for the entire time
Churn - whether there was a churn (Yes or No)

```
## spec_tbl_df [5,986 x 22] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ X1              : num [1:5986] 1869 4528 6344 6739 432 ...
```

```
##  $ customerID      : chr [1:5986] "7010-BRBUU" "9688-YGXVR" "9286-DOJGF" "6994-KERXL" ...
##  $ gender          : chr [1:5986] "Male" "Female" "Female" "Male" ...
##  $ SeniorCitizen   : num [1:5986] 0 0 1 0 0 0 0 0 0 1 ...
##  $ Partner         : chr [1:5986] "Yes" "No" "Yes" "No" ...
##  $ Dependents      : chr [1:5986] "Yes" "No" "No" "No" ...
##  $ tenure          : num [1:5986] 72 44 38 4 2 70 33 1 39 55 ...
##  $ PhoneService    : chr [1:5986] "Yes" "Yes" "Yes" "Yes" ...
##  $ MultipleLines   : chr [1:5986] "Yes" "No" "Yes" "No" ...
##  $ InternetService : chr [1:5986] "No" "Fiber optic" "Fiber optic" "DSL" ...
##  $ OnlineSecurity  : chr [1:5986] "No internet service" "No" "No" "No" ...
##  $ OnlineBackup    : chr [1:5986] "No internet service" "Yes" "No" "No" ...
##  $ DeviceProtection: chr [1:5986] "No internet service" "Yes" "No" "No" ...
##  $ TechSupport     : chr [1:5986] "No internet service" "No" "No" "No" ...
##  $ StreamingTV     : chr [1:5986] "No internet service" "Yes" "No" "No" ...
##  $ StreamingMovies : chr [1:5986] "No internet service" "No" "No" "Yes" ...
##  $ Contract        : chr [1:5986] "Two year" "Month-to-month" "Month-to-month" "Month-to-month" ...
##  $ PaperlessBilling: chr [1:5986] "No" "Yes" "Yes" "Yes" ...
##  $ PaymentMethod   : chr [1:5986] "Credit card (automatic)" "Credit card (automatic)" "Bank transfer
##  $ MonthlyCharges  : num [1:5986] 24.1 88.2 75 55.9 53.5 ...
##  $ TotalCharges    : num [1:5986] 1735 3973 2870 238 120 ...
##  $ Churn           : chr [1:5986] "No" "No" "Yes" "No" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double(),
##   ..   customerID = col_character(),
##   ..   gender = col_character(),
##   ..   SeniorCitizen = col_double(),
##   ..   Partner = col_character(),
##   ..   Dependents = col_character(),
##   ..   tenure = col_double(),
##   ..   PhoneService = col_character(),
##   ..   MultipleLines = col_character(),
##   ..   InternetService = col_character(),
##   ..   OnlineSecurity = col_character(),
##   ..   OnlineBackup = col_character(),
##   ..   DeviceProtection = col_character(),
##   ..   TechSupport = col_character(),
##   ..   StreamingTV = col_character(),
##   ..   StreamingMovies = col_character(),
##   ..   Contract = col_character(),
##   ..   PaperlessBilling = col_character(),
##   ..   PaymentMethod = col_character(),
##   ..   MonthlyCharges = col_double(),
##   ..   TotalCharges = col_double(),
##   ..   Churn = col_character()
##   .. )
```

Let's do some data cleaning by first changing all categorical variables to factors.

```
df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)], as.factor)
```

Cleaning the data.

```
#Transofrm back customerID into character
df$customerID <- as.character(df$customerID)
```

```r
#Transform senior citizen into factor
df$SeniorCitizen <- as.factor(df$SeniorCitizen)

# eliminate X1 variable as we don't need it
df$X1 <- NULL

# Change the Churn into binary variable
df$Churn <- as.factor(ifelse(df$Churn =="Yes", 1, 0))

# checking NA's
sum(is.na(df))
```

```
## [1] 10
```

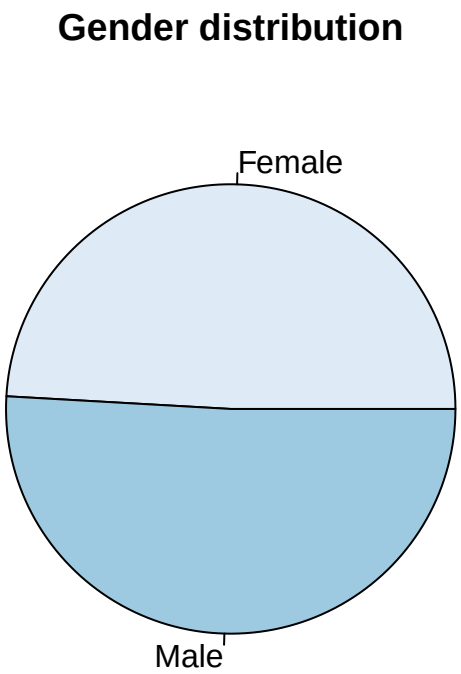
```r
# removing rows with NA's
df <- df[!is.na(df$TotalCharges),]
```

2. Analysis

2.1 Data exploration

Now we will get our feet into the cold water by exploring the dataset and form hypothesis out of our plots and statistics.

Let's start by checking out whether the gender is equally distributed. We see an almost equal distribution in gender, which is quite surprising!



In sum there are 1587 customers that churned.

Table 1: Number of Churns

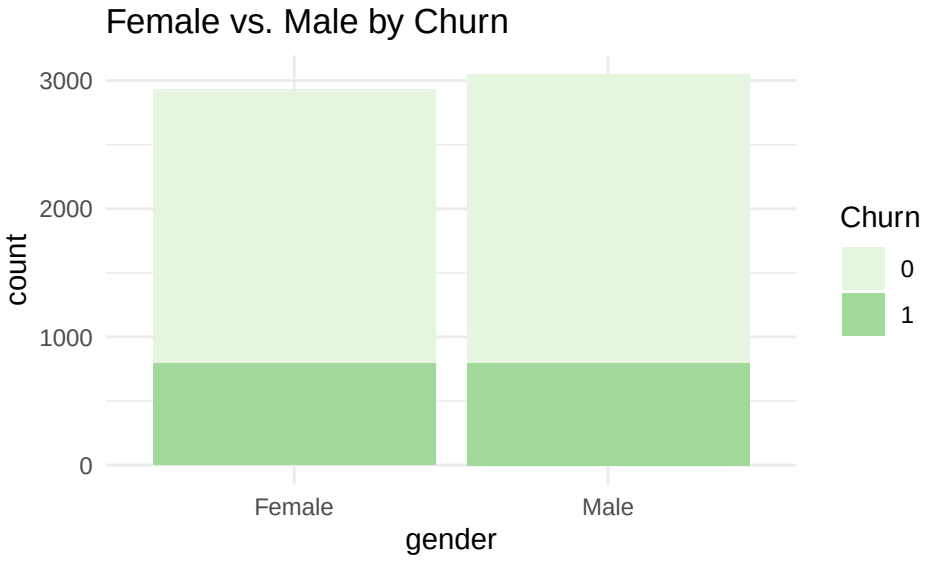
x
1587

Both male and female have quite the same amount of Churns, differing only by 5 Churner.

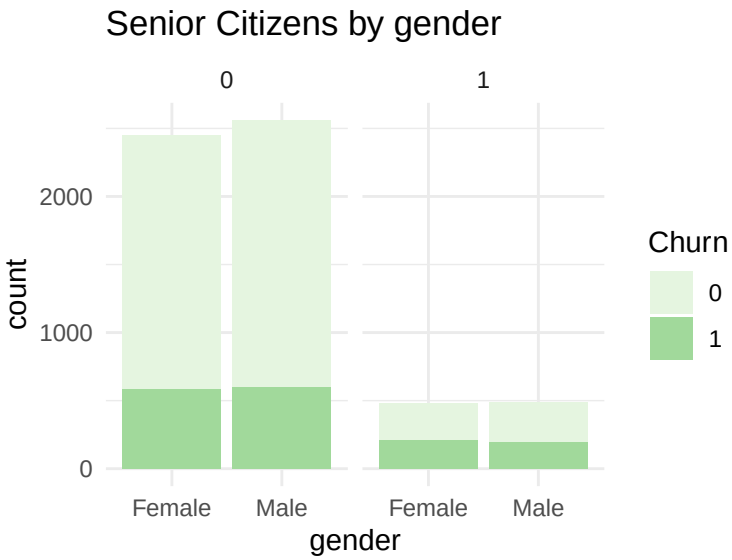
Table 2: Churn divided by gender

gender	n
Female	791
Male	796

It doesn't seem that gender has an impact on the churn rate.



Checking out senior citizens we see that if customers are senior, their churn rate is higher than not being senior over both gender types.

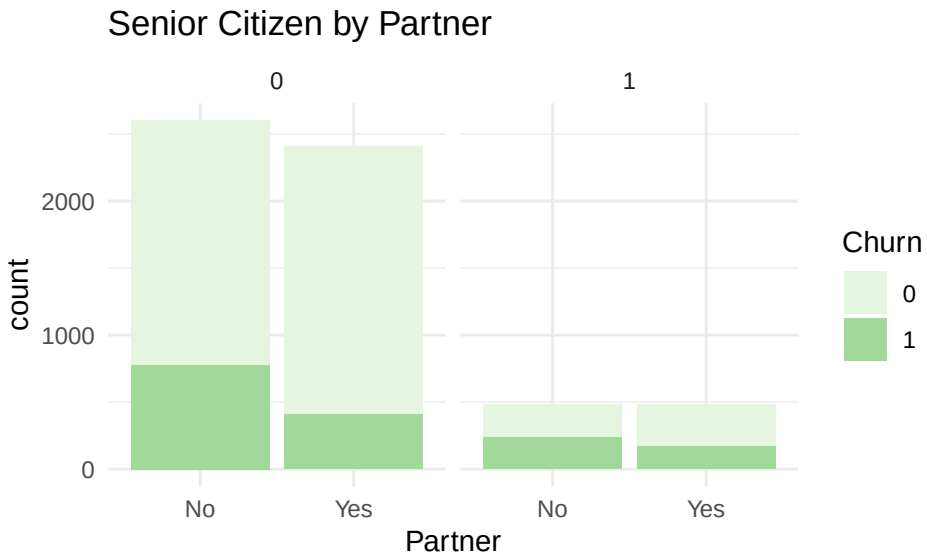


Senior Citizens account for 25.3% of the total churn

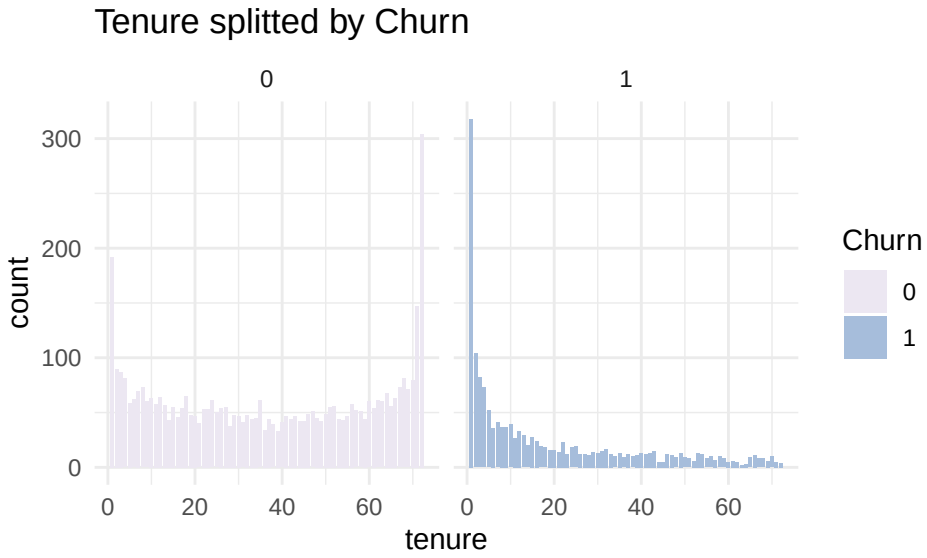
Table 3: Percentage Senior Citizen Churn

n
0.25331

When checking for the relationship status, we see that people without a partner and not being a senior citizen tend to have a higher churn share. Does having a partner reduce the churn?

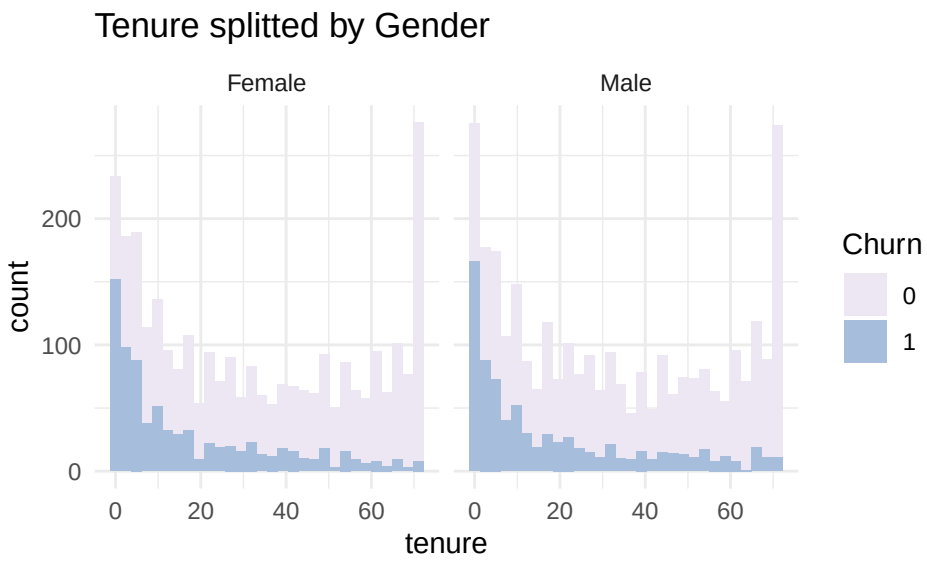


It is evident that the longer the tenure the less likely the Churn, which makes sense. People with a long tenure are probably satisfied with their contract or are just too lazy to look for a new one.



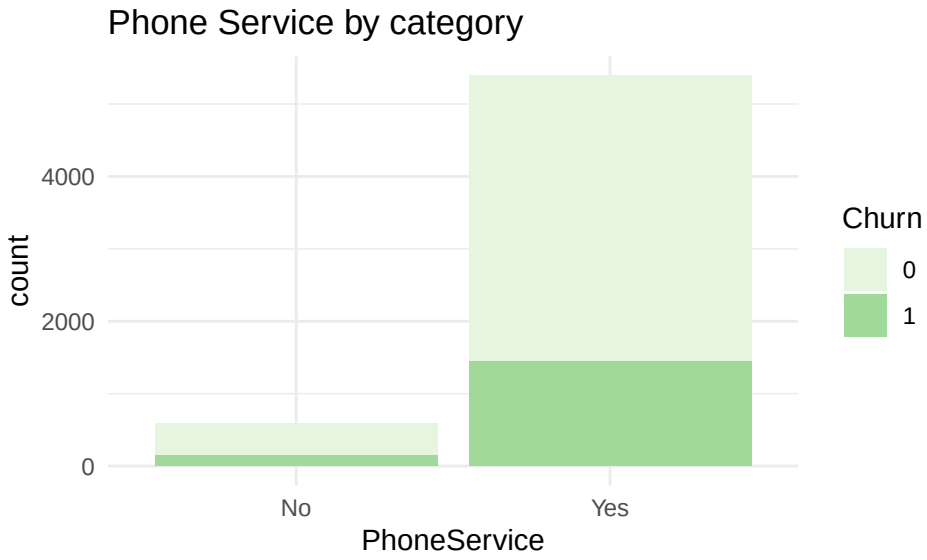
Let's check whether there is a difference in gender, e.g. if men have less patience.

There is no evident difference in the churn considering gender and tenure.



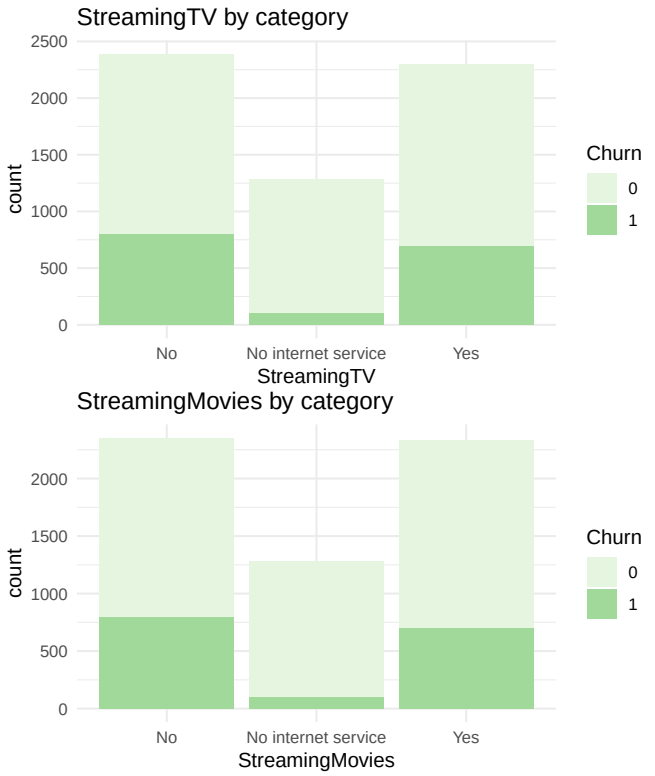
Does having a phone service change the churn?

It doesn't seem so, but only a fraction doesn't have a phone service. This seems reasonable, as phone services are most often offered to the internet connection anyway.

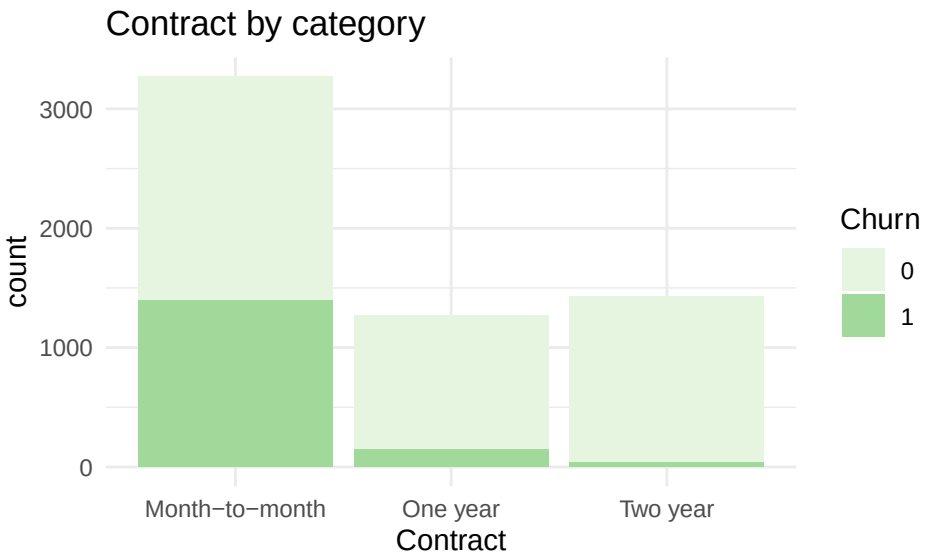


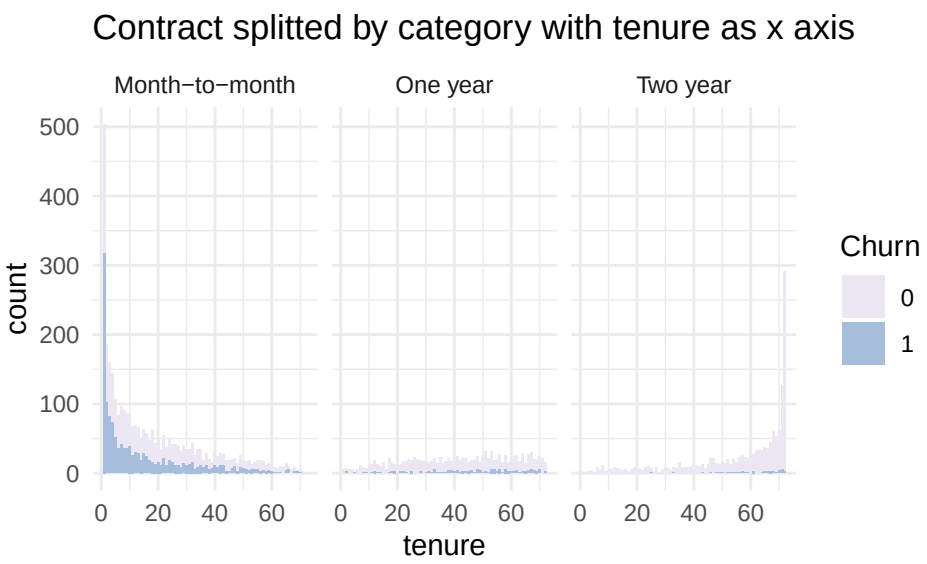
How about streaming tv and streaming movies?

Interesting, no internet service has the lowest churn share for streaming tv and movies.

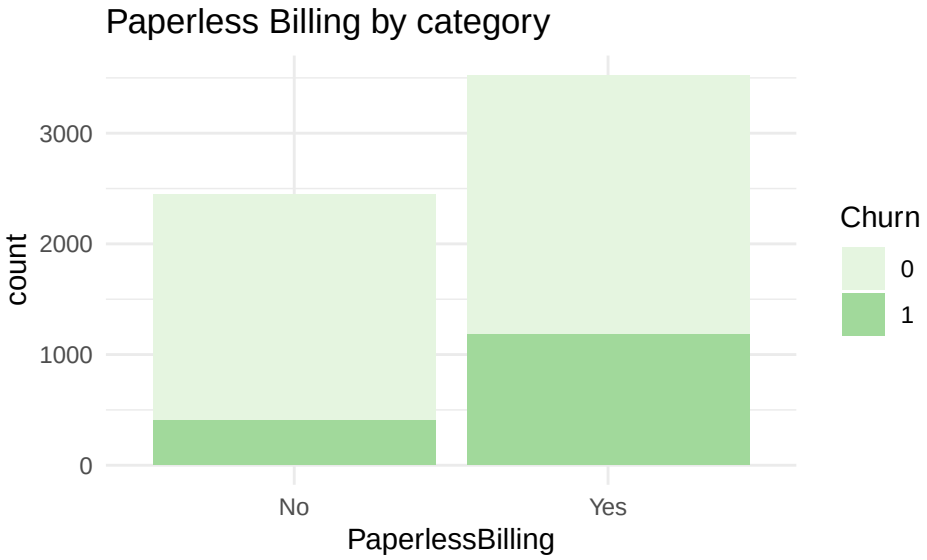


The longer the contract the less the churn. This might be interesting for our random forest later, as it will take each category as own branch. Recognizing that a Month-to-month contract has a quite high churn rate in comparison to the two other contract lengths, it might be even reasonable to combine this variable with another variable to create a feature that could be more meaningful.





One can assume that paperless billing may increase the churn rate, as this would be related to online contract management. It should be easier to cancel contracts this way.

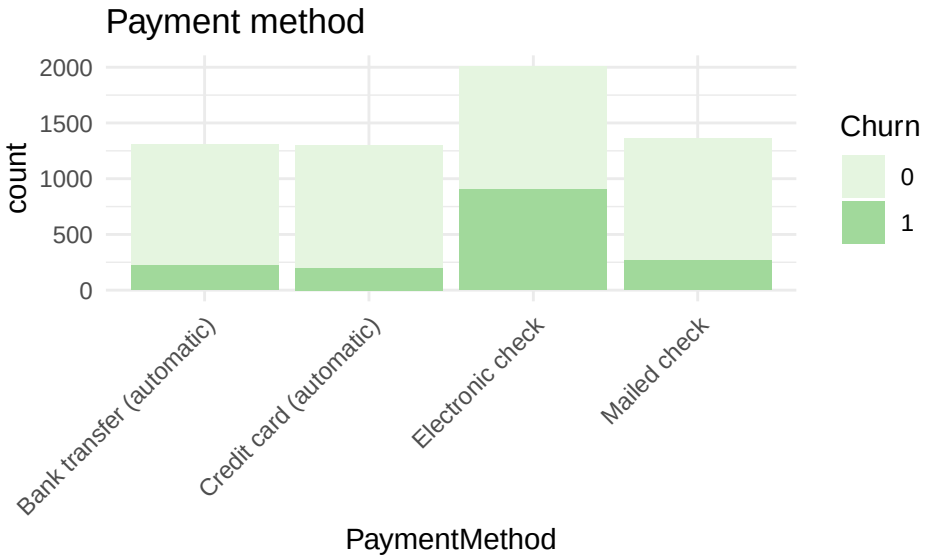


Indeed, almost twice as many customers cancel their phone service when they have paperless billing than no paperless billing. Canceling contracts when billing is managed digital might me more convenient.

Table 4: Percentage of Churn by Paperless Billing

PaperlessBilling	percentage
No	0.16565
Yes	0.33504

The same scenario can be found when looking at the payment method: Even though Bank transfer and credit cards can also be managed online, the electronic check seems the most convenient way to manage ones telecom subscription. Unfortunately, it is also the most convenient to cancel the subscription.

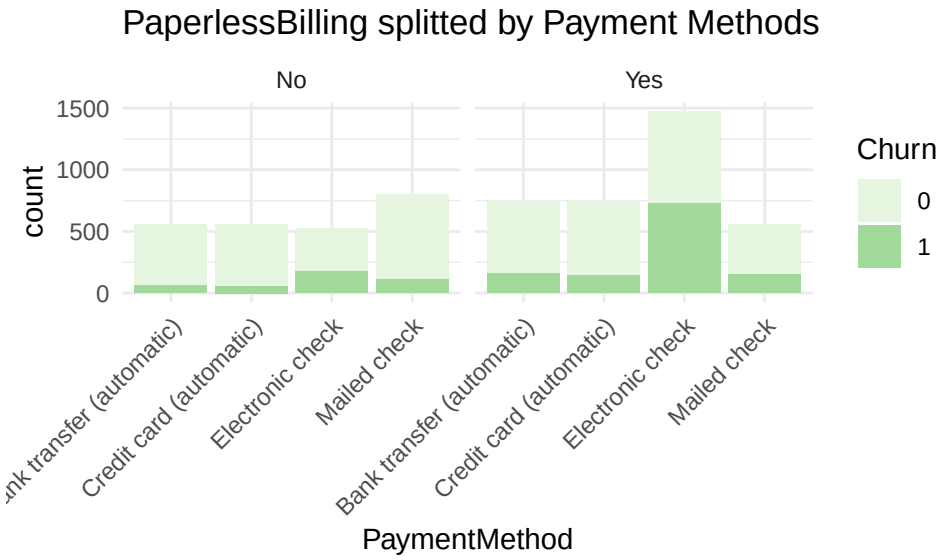


Calculating the churn rates by method, we see that 45% of people churn! That is a lot. Surprisingly, the churn rate for mailed check is higher than for automatic bank transfer and automatic credit card payments.

Table 5: Churn rate by Payment Method

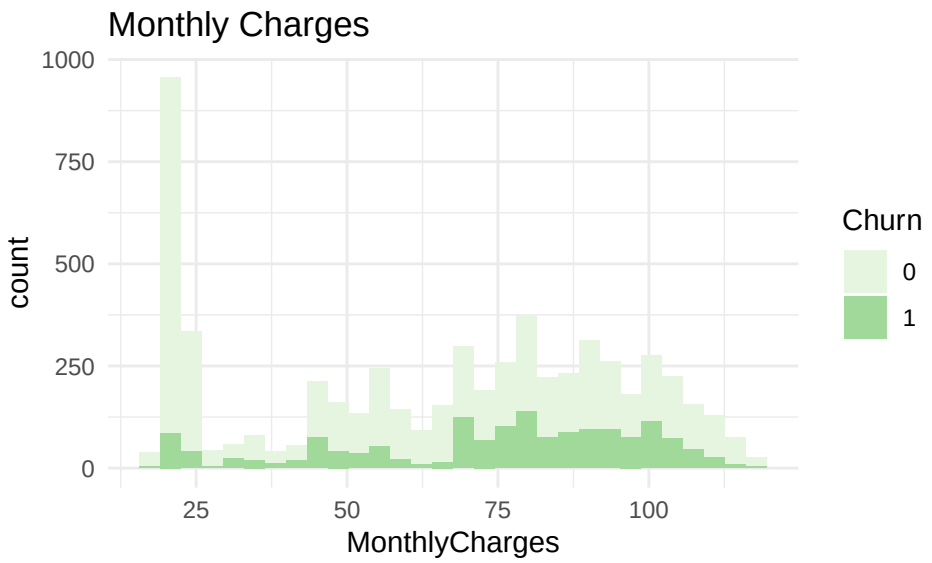
PaymentMethod	percentage
Bank transfer (automatic)	0.17152
Credit card (automatic)	0.15207
Electronic check	0.44965
Mailed check	0.19310

I hoped to generate a group of variables where the majority is made out of churned customers. However, putting payment method and paperless billing doesn't give us more knowledge than before. Nevertheless it might be interesting to connect these two variables into a new feature.

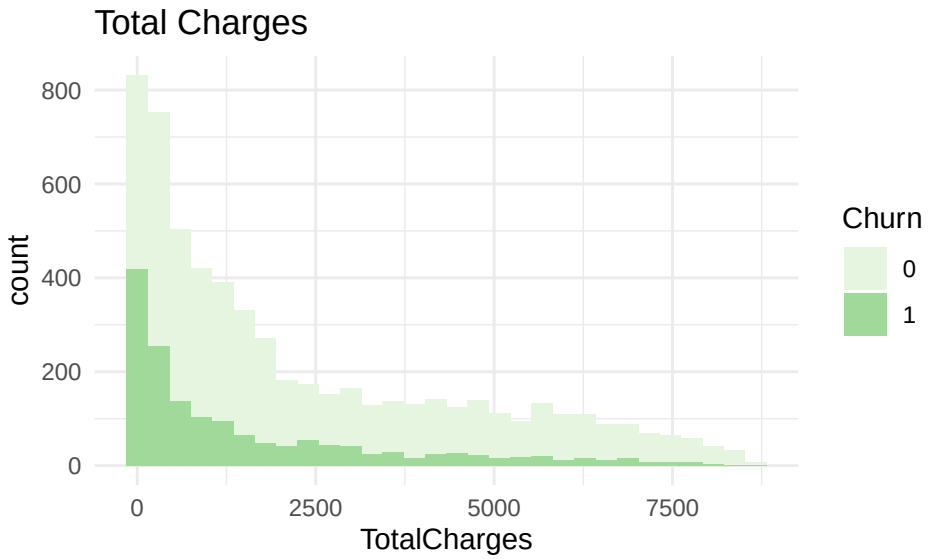


When looking the monthly charges, it seems that there is a zone at above 60, that leads to an increase of the

churn share.



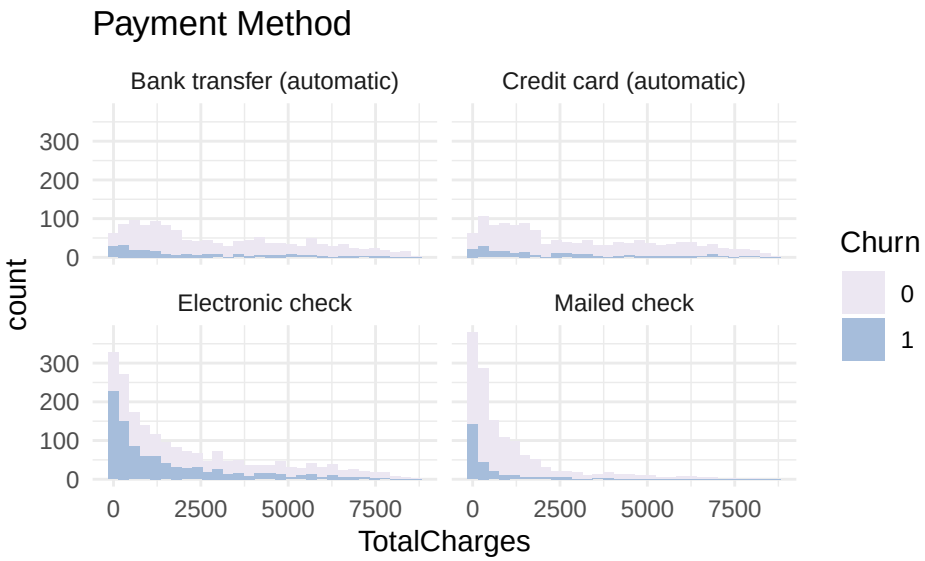
I am a fan of numerical variables, as they will definitely help our models later, especially something like total charges.



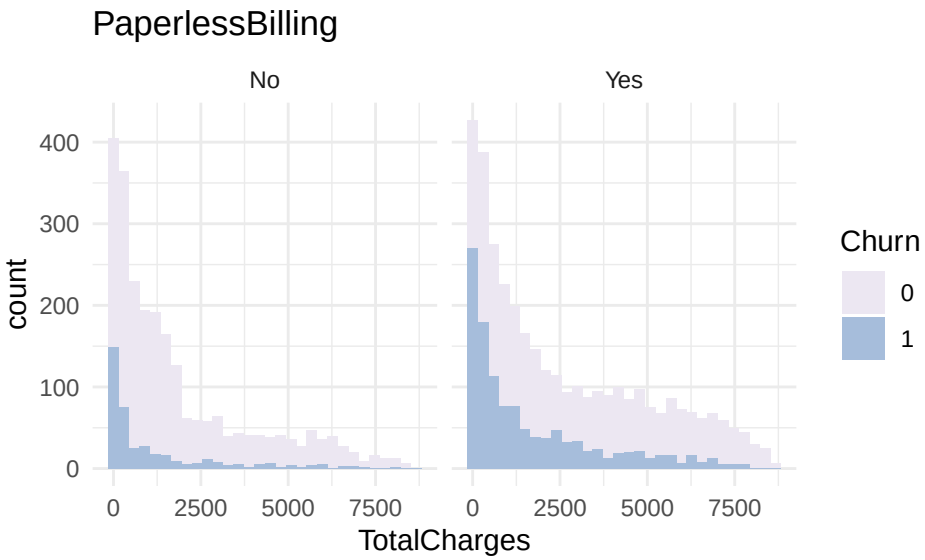
Let's also investigate interrelationships between other promising variables

Starting with the relationship between total charges and payment method.

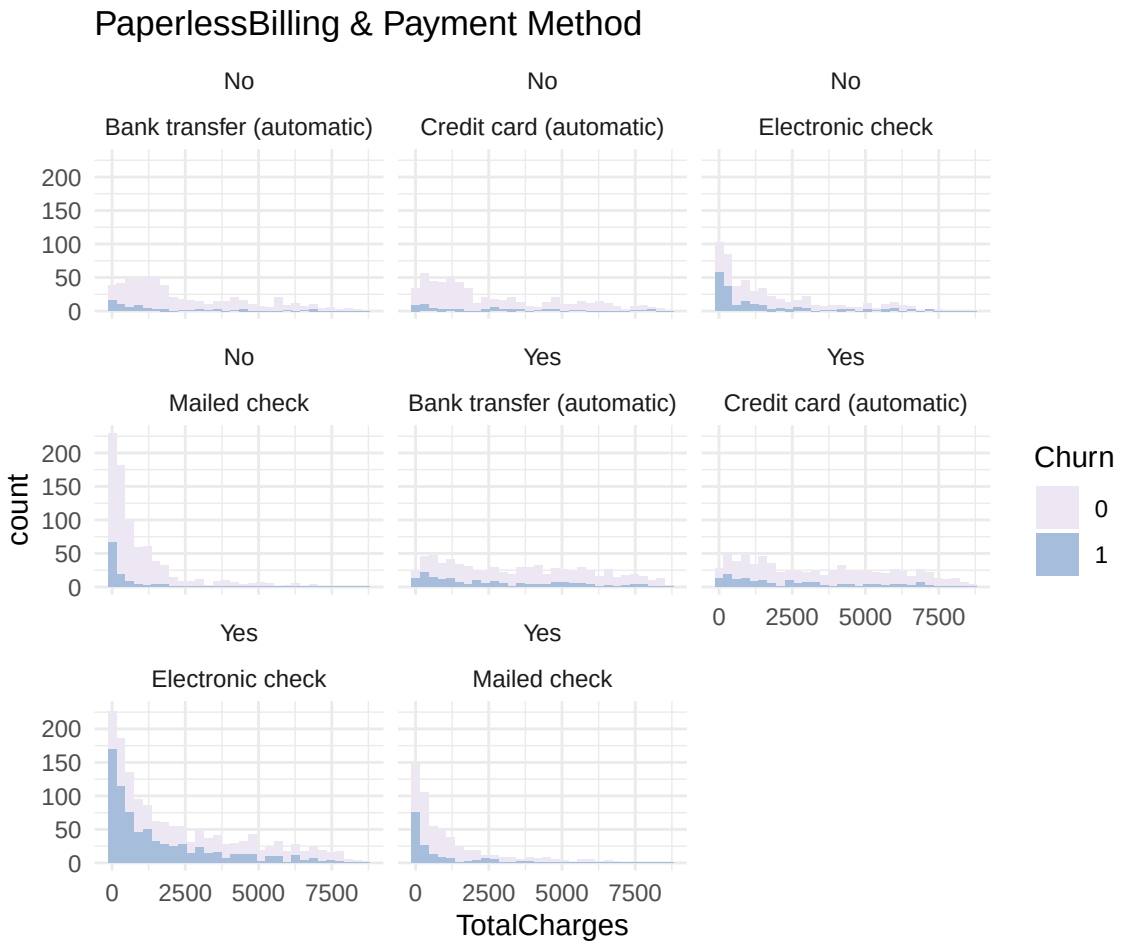
There is a very high share in electronic check as payment method as we saw earlier. It seems that this churn share is distributed through all total charges.



Plotting paperless billing against total charges, we can perceive the distribution of churn rates on the total charges. If a customer has paperless billing, the churn rate decreases with increasing total charges. However, without paperless billing the churn rate stays approximately the same after total charges of approximately 1200.

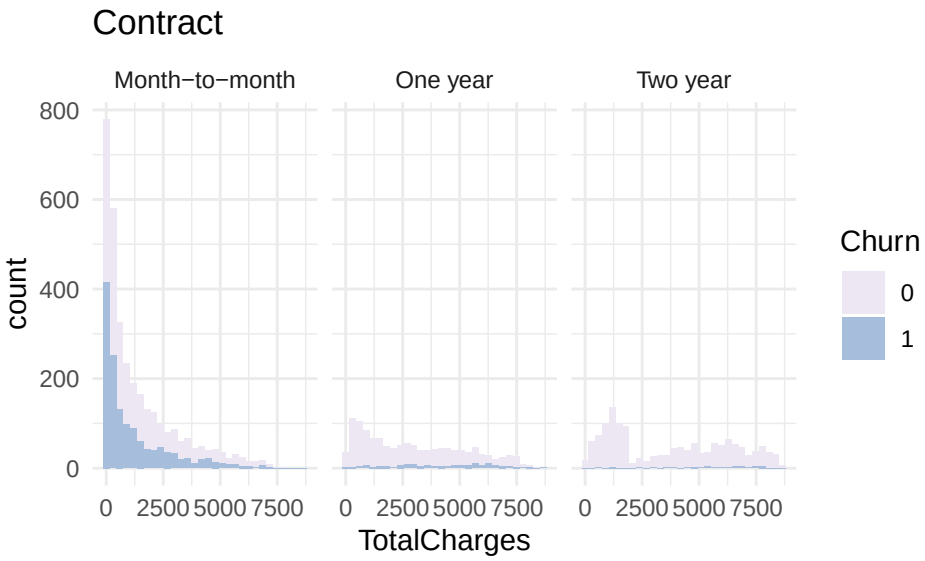


Now we plot paperless billing and payment methods across total charges. When we see that somebody goes fully digital, meaning having an electronic check as payment method and paperless billing, we see that the churn share is pretty high.



2.2 Feature engineering

With our gained insights, we now are going to create some new features that might help our models to perform better. The algorithms won't take into account combinations of variables by itself. Therefore, we have to think of smart combinations which might boost the balanced accuracy of our models later on.



We create a feature that accounts for month-to-month contract with the respective charges in order to give that relationship more weight.

```
# create new feature
feat1 <- df %>%
  filter(Contract == "Month-to-month") %>%
  mutate(Contract_Charges_feature = TotalCharges) %>%
  select(customerID, Contract_Charges_feature)

# join with dataset
df <- left_join(df, feat1, by ="customerID")

# transform all NA's
df$Contract_Charges_feature <- ifelse(is.na(df$Contract_Charges_feature), 0,
                                      df$Contract_Charges_feature)
```

We saw earlier that monthly charges at above 60 has a higher churn share. Let's also create a variable for that.

```
# create new feature
feat2 <- df %>%
  filter(MonthlyCharges > 60) %>%
  mutate(MonthlyCharges_feature = 1) %>%
  select(customerID, MonthlyCharges_feature)

# join with dataset
df <- left_join(df, feat2, by ="customerID")

# transform NA's
df$MonthlyCharges_feature <- ifelse(is.na(df$MonthlyCharges_feature), 0,
                                    df$MonthlyCharges_feature)
```

We will also account for the fully digital people, that is customers with paperless billing and electronic check as payment method.

```
# create feature
feat3 <- df %>%
  filter(PaperlessBilling=="Yes", PaymentMethod == "Electronic check") %>%
  mutate(tenure_PB_PM_feature = tenure) %>%
  select(customerID, tenure_PB_PM_feature)

# join with dataset
df <- left_join(df, feat3, by ="customerID")

# transform NA's
df$tenure_PB_PM_feature <- ifelse(is.na(df$tenure_PB_PM_feature), 0,
                                  df$tenure_PB_PM_feature)
```

Next, we create a monthly charges per tenure and total charges per tenure variable.

```
# Total charges per tenure
feat4 <- df %>%
  mutate(TCperTenure_feature = TotalCharges/tenure) %>%
```

```r
  select(customerID, TCperTenure_feature)

#join dataset
df <- left_join(df, feat4, by="customerID")

# Monthly charges per tenur
feat5 <- df %>%
  mutate(MCperTenure_feature = MonthlyCharges/tenure) %>%
  select(customerID, MCperTenure_feature)

#join datset
df <- left_join(df, feat5, by="customerID")
```

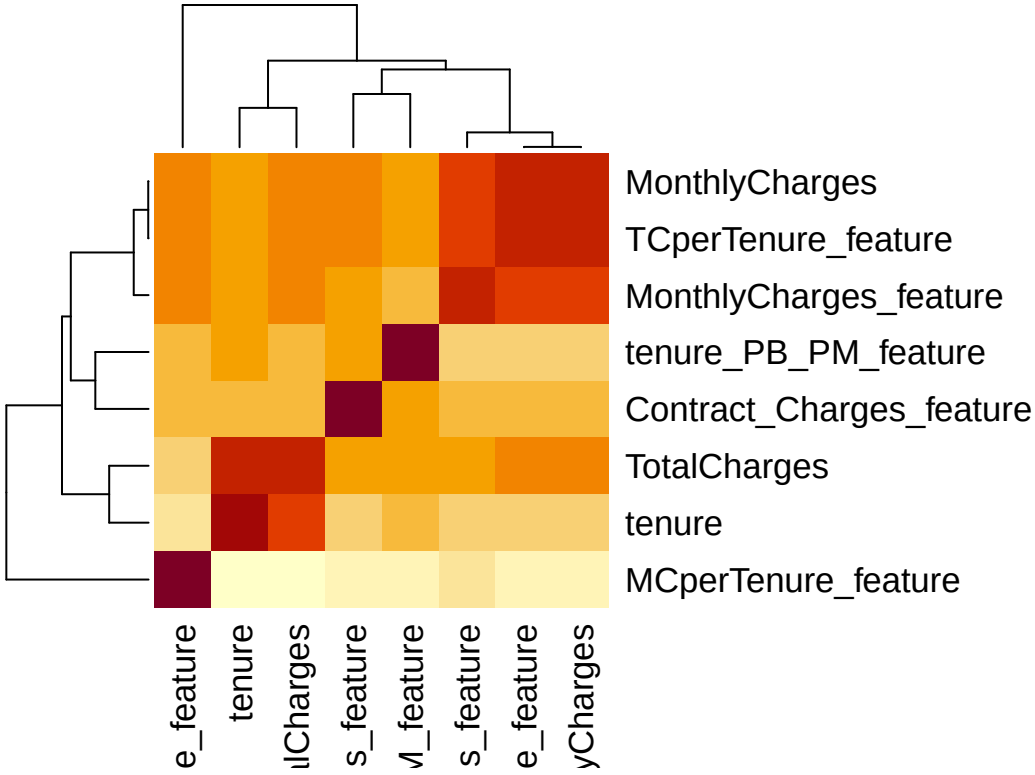
3. Model building and results

In this section, we are going to build our classification models to predict whether the customer will churn or not. We will train each model, try to improve it and discuss the results shortly. Again, our main goal is to incraese balanced accuracy as good as possible.

Checking the correlation matrix for the numeric values we already see which variables are independent and the ones very dependent, which may lead to multicollinearity.

Our feature engineered total charges per tenure and the monthly charges variable are highly correlated. Also the tenure with the total charges. Thus, combining those variables together was reasonable.

```
##                            tenure MonthlyCharges TotalCharges
## tenure                    1.00000       0.255676      0.82744
## MonthlyCharges            0.25568       1.000000      0.65653
## TotalCharges              0.82744       0.656534      1.00000
## Contract_Charges_feature  0.10805       0.355075      0.23325
## MonthlyCharges_feature    0.18970       0.880497      0.52348
## tenure_PB_PM_feature      0.21629       0.291150      0.31823
## TCperTenure_feature       0.25549       0.996222      0.65661
## MCperTenure_feature      -0.53339       0.045675     -0.39167
##                          Contract_Charges_feature MonthlyCharges_feature
## tenure                                    0.10805               0.189697
## MonthlyCharges                            0.35507               0.880497
## TotalCharges                              0.23325               0.523482
## Contract_Charges_feature                  1.00000               0.299865
## MonthlyCharges_feature                    0.29986               1.000000
## tenure_PB_PM_feature                      0.31133               0.211502
## TCperTenure_feature                       0.35398               0.876985
## MCperTenure_feature                      -0.16071               0.086598
##                          tenure_PB_PM_feature TCperTenure_feature
## tenure                                0.21629            0.255493
## MonthlyCharges                        0.29115            0.996222
## TotalCharges                          0.31823            0.656605
## Contract_Charges_feature              0.31133            0.353977
## MonthlyCharges_feature                0.21150            0.876985
## tenure_PB_PM_feature                  1.00000            0.290683
## TCperTenure_feature                   0.29068            1.000000
## MCperTenure_feature                  -0.13507            0.045155
##                          MCperTenure_feature
## tenure                             -0.533387
## MonthlyCharges                      0.045675
## TotalCharges                       -0.391665
## Contract_Charges_feature           -0.160710
## MonthlyCharges_feature              0.086598
## tenure_PB_PM_feature               -0.135068
## TCperTenure_feature                 0.045155
## MCperTenure_feature                 1.000000
```



Data Split

To check which models are better we create a 80/20 train-test-split. This split seems reasonable as the dataset is not very huge and we need some data to train our model with.

```
set.seed(2021)
test_index <- createDataPartition(y = df$Churn, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- df[-test_index,]
test_set <- df[test_index,]

train_set$customerID <- NULL
test_set$customerID <- NULL
```

3.1 Logistic Regression 1

Our first classification model will be standard logistic regression. We will train our first model by just the variables provided with the dataset, meaning without our featured variables.

We see that only 6 variables are statistically siginificant at a 5%-significance level. Some of the p-values for the variables are quite high that let us believe there is multicollinearity. Some of our factor variables show NA's. This is due to the dummy variable trap

```
## logistic regression 1
lr <- glm(as.factor(Churn) ~ gender + SeniorCitizen + Partner + Dependents +
            PhoneService + MultipleLines + InternetService + OnlineSecurity +
            OnlineBackup + DeviceProtection + TechSupport + StreamingTV +
            StreamingMovies + Contract + tenure + MonthlyCharges + TotalCharges,
```

```
          data=train_set, family=binomial(link="logit"))
```

Our McFaddenR is acceptable. A McFaddenR over 40% is considered as good, while under 20% as bad.

```
McFaddenR <- 1- lr$deviance/lr$null.deviance
McFaddenR
```

```
## [1] 0.28155
```

Our very first model achieves an accuracy of 0.789 which is not bad already. However, we are interested in the balanced accuracy as we want to predict the churn rate as good as the non-churned customers.

```
#prediction
pred <- predict(lr, newdata=test_set, type="response")
pred <- if_else(pred >0.5, 1, 0)

# Balanced Accuracy
CM <- confusionMatrix(as.factor(pred), as.factor(test_set$Churn))
```

Table 6: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.7893	0.70907

3.2 Logistic Regression 2

In this model, we want to take into account our created features as well.

```
## logistic regression 2
lr <- glm(as.factor(Churn) ~ ., data=train_set, family=binomial(link="logit"))
```

Our McFaddenR increased!

```
McFaddenR <- 1- lr$deviance/lr$null.deviance
McFaddenR
```

```
## [1] 0.30614
```

We were able to increase our BA by 0.01%. Our Contract Charges feature and monthly charge per tenure feature are significant at a 0.01%-significance level, while the others are not significant at a 10%-signifcance level.

```
#prediction
pred <- predict(lr, newdata=test_set, type="response")
pred <- if_else(pred >0.5, 1, 0)

CM <- confusionMatrix(as.factor(pred), as.factor(test_set$Churn))
```

Table 7: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962

3.3 Logistic regression 3

As there are a lot more non-churn customers than churn customers, we balance the train set, such that our model has the chance to have a larger train share for the churned customers. Balancing the dataset is a common procedure, as having too less data on one label makes it difficult for the algorithms to learn. Thus the models might overfit on the label that has the bigger share in the train set.

```
## logistic regression 3
# balancing the training set
train_Churn <- train_set %>% filter(Churn == 1)
n <- train_set %>% filter(Churn == 1) %>% nrow()

train_noChurn <- train_set %>% filter(Churn == 0)

set.seed(2021)
train_noChurn <- train_noChurn[sample(seq(1:n),
                                      size = n,
                                      replace = FALSE), ]

train_new <- rbind(train_Churn, train_noChurn)

set.seed(2021)
train_set <- train_new[sample(seq(1:nrow(train_new)),
                              size = nrow(train_new),
                              replace =FALSE), ]


# train model
lr <- glm(as.factor(Churn) ~ ., data=train_set, family=binomial(link="logit"))
```

We could increase our McFaddenR by 2% which is not bad!

```
McFaddenR <- 1- lr$deviance/lr$null.deviance
McFaddenR
```

```
## [1] 0.32821
```

Wow! Our BA jumped to 73.5%. However, we lost accuracy. This means our model got better at predicting Churn than non-Churn.

From now, we will use the balanced train set for all following models.

```
# prediction
# increased BA by a 4% when balancing -> boosting might help
pred_lr <- predict(lr, newdata=test_set, type="response")
pred_lr <- as.factor(if_else(pred_lr >0.5, 1, 0))

CM <- confusionMatrix(as.factor(pred_lr), as.factor(test_set$Churn))
```

Table 8: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907

method	Accuracy	BalancedAccuracy
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458

3.4 Logistic Regression 4

How will our model change, if we take out all the insignificant variables and the dummy variable trap?

```
## logistic regression 4
# eliminating all NA's bc of dummy variable trap -> multicollinearity
# interestingly the drop is not huge, evidence of multicollinearity
lr <- glm(as.factor(Churn) ~ gender + SeniorCitizen + Partner + Dependents +
            PhoneService + InternetService + Contract + tenure + MonthlyCharges +
            TotalCharges + Contract_Charges_feature + MCperTenure_feature ,
          data=train_set, family=binomial(link="logit"))
```

Our McFaddenR decreased by 3%.

```
McFaddenR <- 1- lr$deviance/lr$null.deviance
McFaddenR
```

```
## [1] 0.29865
```

However, we could increase our BA to 74%. We are on the right track!

```
#prediction
pred <- predict(lr, newdata=test_set, type="response")
pred <- if_else(pred >0.5, 1, 0)

# Balanced Accuracy
CM <- confusionMatrix(as.factor(pred), as.factor(test_set$Churn))
```

Table 9: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973

3.5 Random Forest 1

Let's check how random forest will do on our dataset.

```
## Random Forest 1

rf_train <- train_set %>% select(-Churn)
rf_label <- as.factor(train_set$Churn)
rf_test <- test_set %>% select(-Churn)

# train model
```

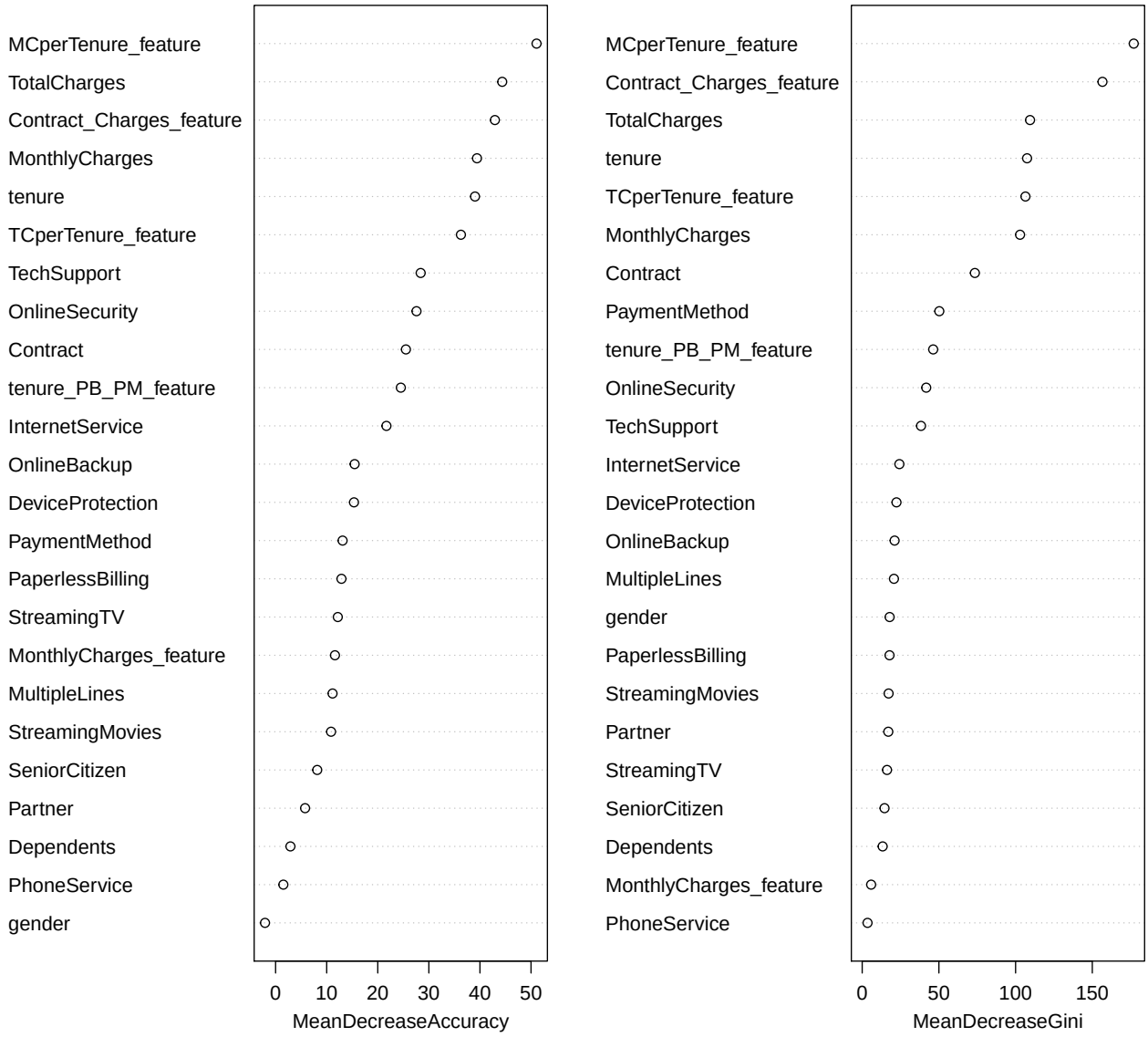
```r
set.seed(2021)
rf <- randomForest(x= rf_train, y = rf_label, importance = TRUE, ntree =1000)
rf
```

```
##
## Call:
##  randomForest(x = rf_train, y = rf_label, ntree = 1000, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24.67%
## Confusion matrix:
##     0   1 class.error
## 0 929 340     0.26793
## 1 286 983     0.22537
```

Our feature engineering was worth it. They are listed high in the Mean ecrease Accuary and the Mean Decrease Gini. Thus, the created variables are important for the random forest in order to decide whether a customer will churn or not.

The mean decrease accuracy tells us the loss in accuracy when the variable excluded. MCperTenure, TotalCharges as well as Contract_Charges are important variables to classify correctly.

On the other hand the mean decrease gini tells us how much each variable contributes to homogeneity. The higher the rank the better the model can use that variable to decide whether the customer will churn or not. In this case it's the same variables in the top three.



However, our balanced accuracy doesn't look promising as it is lower than the logistic regression.

```
#prediction
pred_rf <- predict(rf, newdata = rf_test, type="response")

CM <- confusionMatrix(as.factor(pred_rf), as.factor(test_set$Churn))
```

Table 10: Model results

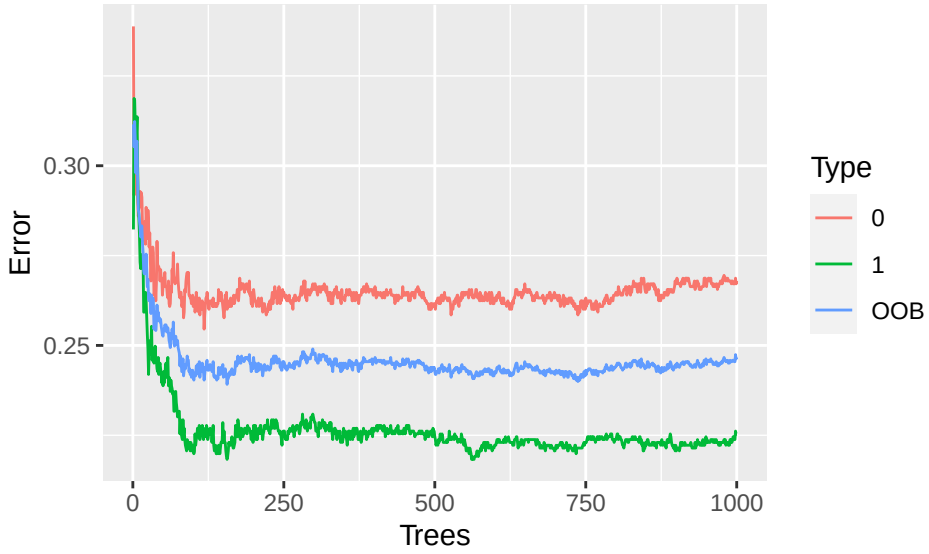
method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458

method	Accuracy	BalancedAccuracy
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911

3.6 Random Forest with fine tuning

We plot the out of bag error rate in order to see whether we have enough number of trees. We know, when we have enough trees, when the error rate is converging.

1000 trees seem the right amount alredy.



To further fine tune our random forest we will check for the right mtry. That is the variables random forest considers to build each tree.

We do that by testing our train set by the out of bag set which can be seen as "new" test or validation set.

```
# check how many random features (mtry) is the best
oob.values <- vector(length=10)
for(i in 1:10){
  temp.model <- randomForest(x = rf_train, y = rf_label,
                             mtry = i,
                             importance = TRUE,
                             ntree = 1000)
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}

which.min(oob.values)
```

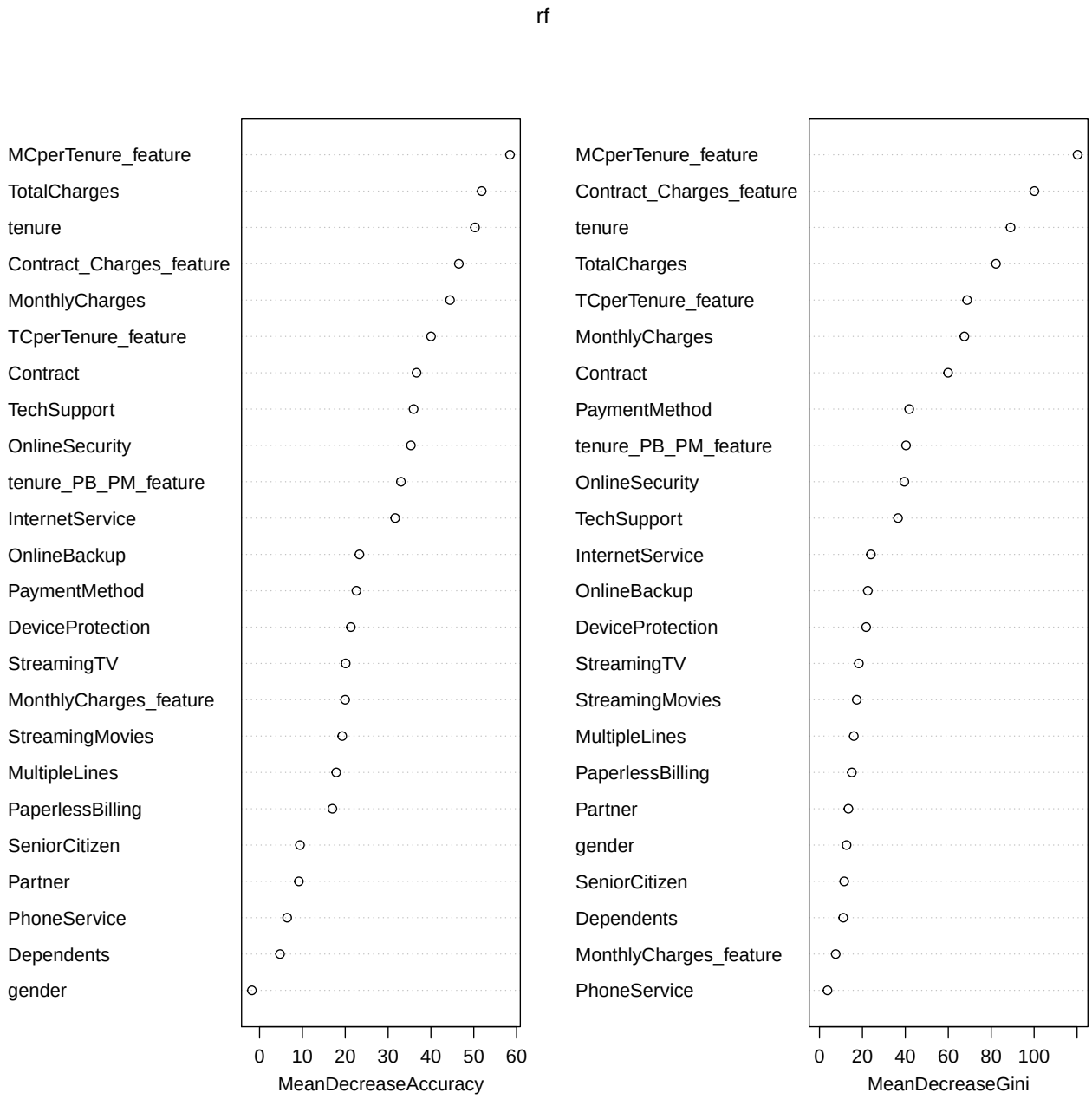
```
## [1] 2
```

Training the model after hyperparameter tuning.

```
##
## Call:
##  randomForest(x = rf_train, y = rf_label, ntree = 2000, mtry = which.min(oob.values),      importanc
##                Type of random forest: classification
```

```
##                       Number of trees: 2000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 23.56%
## Confusion matrix:
##     0    1 class.error
## 0 934  335     0.26399
## 1 263 1006     0.20725
```

Tenure got more important in both rankings. However, we see the same variables in the upper ranks as in our previous random forest.



Luckily, we were able to increase our BA!

Table 11: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586

3.7 Gradient Boosting

Our next ensemble method is gradient boosting. In particular we will use gradient boosting in order to combine weak learners to create a strong learner. In boosting the models are build on top of each other, meaning that after a model is trained more weight is put on the instances in the sample that were missclassified. Then, a new model is trained on the weighted sample set such that the model becomes an expert in the missclassified instances and so on. However, this model can quickly overfit.

```
# gradient boosting combines weak learners to create a strong learner
# predictions are sequential, while each subsequent predictor learns from
# the errors of the previous predictors

# to control nuances of train function
set.seed(2021)
gb_ctrl_specs <- trainControl(method = "cv", number = 10)

#train the gradient boosting with the "auto" function
set.seed(2021)
gb <- train(as.factor(Churn)~., data=train_set,
            method="gbm",
            distribution="adaboost",
            trControl=gb_ctrl_specs,
            verbose=FALSE)

# predict and confusion matrix
pred_gb <- predict(gb, newdata=test_set)
CM <- confusionMatrix(as.factor(pred_gb), as.factor(test_set$Churn))
```

Not bad! We achieved both a higher accuracy and BA than random forest.

Table 12: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586

3.8 Gradient Boosting with Fine Tuning

Now we will take one step further and fine tune our gradient boosting model. In other words find the right hyperparameters. This is done by the expand.grid() function. We will see whether this will improve our BA.

```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 3     150                 1       0.1             10
```

I could have set up some vectors for the grid search. However, this part would take a lot of time (about 30 minutes on my laptop). Therefore, I skipped this part here and just show the results that were outputted as the best hyperparameters. And luckily we were able to again increase our BA!

```
set.seed(2021)

grid <- expand.grid(
  n.trees = 3000,
  interaction.depth = 1,
  shrinkage = 0.001,
  n.minobsinnode = 20
)


set.seed(2021)
gb_tuned <- train(as.factor(Churn)~., data=train_set,
                  method="gbm",
                  distribution="adaboost",
                  trControl=gb_ctrl_specs,
                  verbose =FALSE,
                  tuneGrid= grid)

pred <- predict(gb_tuned, newdata=test_set)
CM <- confusionMatrix(as.factor(pred), as.factor(test_set$Churn))
```

Table 13: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586
(8) gradient boosting + tuned	0.72074	0.74461

3.9 Lasso regression

Lasso regression allows for feature selection. Depending on the lambda, coefficients are just set to 0 and thus eliminated from predicting churn.

```
# set control sepcs
set.seed(2021)
ctrl_specs <- trainControl(method ="cv",
```

```
                           savePredictions = "all",
                           number = 10)


# create vector for lambdas
lambda <- 10^seq(-5,5,length =500)

set.seed(2021)
lassoreg <- train(as.factor(Churn) ~., data=train_set,
                  method="glmnet",
                  tuneGrid = expand.grid(alpha=1, lambda=lambda),
                  trControl = ctrl_specs,
                  preProcess=c("center", "scale"))

lassoreg$bestTune
```

```
##     alpha   lambda
## 160     1 0.015359
```

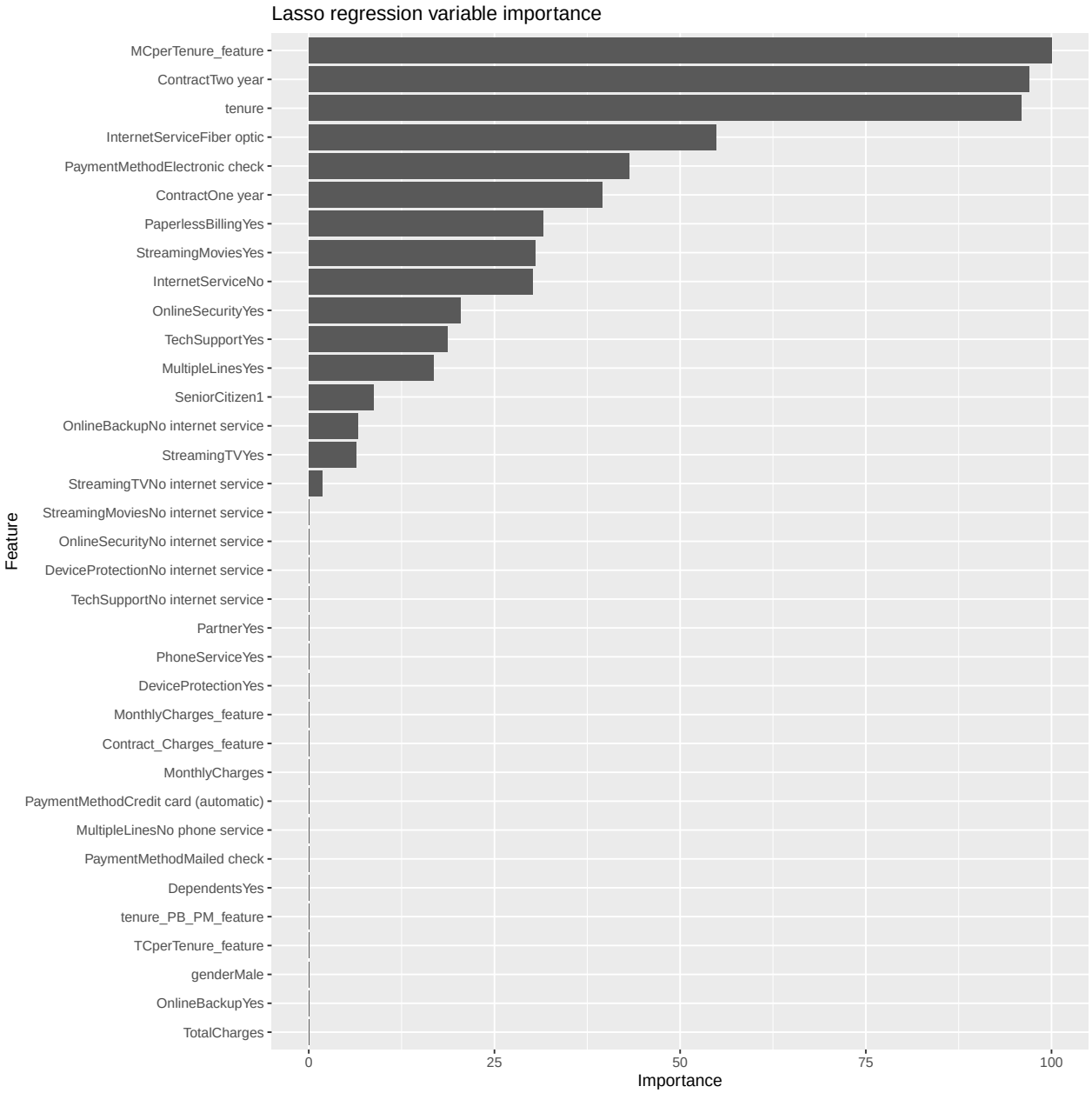
Showing the coefficients in a table we see that some coefficients have no value in the second column. This is because those variables were left out by the lasso regression as this algorithm does feature selection.

```
## 36 x 1 sparse Matrix of class "dgCMatrix"
##                                              1
## (Intercept)                           -0.024
## genderMale                             .
## SeniorCitizen1                         0.042
## PartnerYes                             .
## DependentsYes                          .
## tenure                                -0.460
## PhoneServiceYes                        .
## MultipleLinesNo phone service          .
## MultipleLinesYes                       0.081
## InternetServiceFiber optic             0.264
## InternetServiceNo                     -0.145
## OnlineSecurityNo internet service      0.000
## OnlineSecurityYes                     -0.098
## OnlineBackupNo internet service       -0.032
## OnlineBackupYes                        .
## DeviceProtectionNo internet service    0.000
## DeviceProtectionYes                    .
## TechSupportNo internet service         0.000
## TechSupportYes                        -0.090
## StreamingTVNo internet service        -0.009
## StreamingTVYes                         0.030
## StreamingMoviesNo internet service     0.000
## StreamingMoviesYes                     0.147
## ContractOne year                      -0.189
## ContractTwo year                      -0.466
## PaperlessBillingYes                    0.152
## PaymentMethodCredit card (automatic)   .
## PaymentMethodElectronic check          0.207
## PaymentMethodMailed check              .
## MonthlyCharges                         .
## TotalCharges                           .
```

```
## Contract_Charges_feature              .
## MonthlyCharges_feature                .
## tenure_PB_PM_feature                  .
## TCperTenure_feature                   .
## MCperTenure_feature                   0.480
```

When plotting the variables by importance, we see that the first three (as in random forest) are very important for the lasso regression.

From the plot is it evident that most features got cancelled out. Tenure, ContractTwo year and MCperTenure_feature are the most important variables for the lasso regression.



Sadly, we couldn't improve our BA by lasso regression.

```
#predict
# same BA but random forest is better in predicting churn
pred_lasso <- predict(lassoreg, newdata=test_set)
CM <- confusionMatrix(as.factor(pred_lasso), as.factor(test_set$Churn))
```

Table 14: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586
(8) gradient boosting + tuned	0.72074	0.74461
(9) lasso regression	0.72074	0.73358

3.10 Ridge regression

In comparison to lasso regression, ridge regression doesn't allow for feature selection, thus every feature is again taken into account.

```
set.seed(2021)
# set another control specs but alpha = 0 and not 1
ridge_ctrl_specs <-trainControl(method = "cv",
                                savePredictions = "all",
                                number = 10)

set.seed(2021)
ridgereg <- train(as.factor(Churn) ~., data=train_set,
                  method="glmnet",
                  tuneGrid = expand.grid(alpha=0, lambda=lambda),
                  trControl = ridge_ctrl_specs,
                  preProcess=c("center", "scale"))

ridgereg$bestTune
```

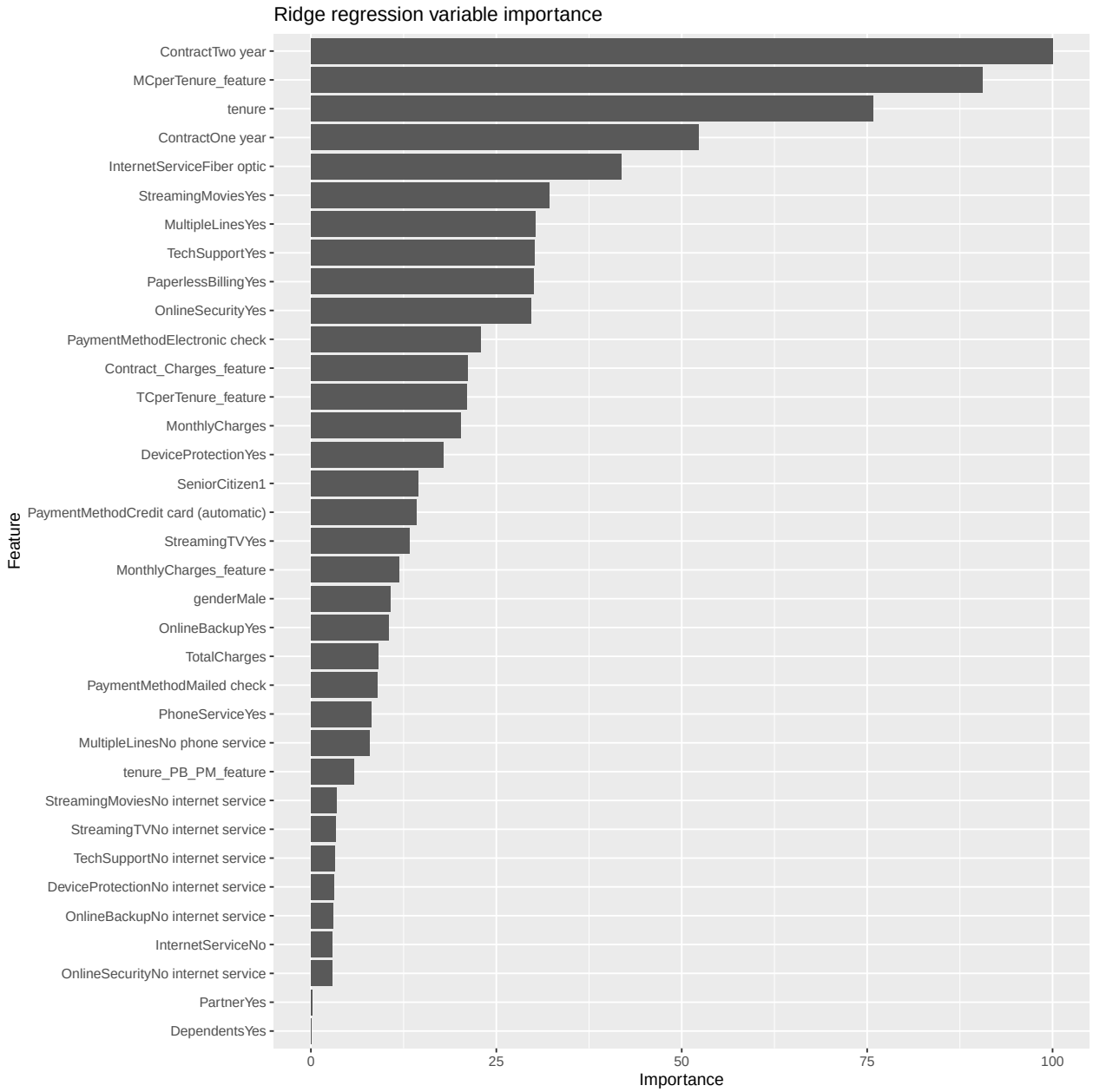
```
##     alpha   lambda
## 167     0 0.021216
```

The next table shows all variables are taken into account with differents weights.

```
## 36 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                         -0.029
## genderMale                           0.069
## SeniorCitizen1                       0.089
## PartnerYes                          -0.015
## DependentsYes                       -0.014
## tenure                              -0.406
## PhoneServiceYes                     -0.056
## MultipleLinesNo phone service        0.055
```

```
## MultipleLinesYes                       0.170
## InternetServiceFiber optic             0.230
## InternetServiceNo                     -0.029
## OnlineSecurityNo internet service     -0.029
## OnlineSecurityYes                     -0.167
## OnlineBackupNo internet service       -0.029
## OnlineBackupYes                       -0.068
## DeviceProtectionNo internet service   -0.030
## DeviceProtectionYes                   -0.106
## TechSupportNo internet service        -0.031
## TechSupportYes                        -0.170
## StreamingTVNo internet service        -0.031
## StreamingTVYes                         0.083
## StreamingMoviesNo internet service    -0.032
## StreamingMoviesYes                     0.180
## ContractOne year                      -0.284
## ContractTwo year                      -0.531
## PaperlessBillingYes                    0.169
## PaymentMethodCredit card (automatic) -0.088
## PaymentMethodElectronic check          0.132
## PaymentMethodMailed check             -0.060
## MonthlyCharges                         0.119
## TotalCharges                          -0.061
## Contract_Charges_feature              -0.123
## MonthlyCharges_feature                -0.075
## tenure_PB_PM_feature                   0.044
## TCperTenure_feature                    0.123
## MCperTenure_feature                    0.482
```

Again plotting the variables and ordering by importance.



Ridge regresion performs a bit better than lasso regression.

```
pred_ridge <- predict(ridgereg, newdata=test_set)
CM <- confusionMatrix(as.factor(pred_ridge), as.factor(test_set$Churn))
```

Table 15: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586

method	Accuracy	BalancedAccuracy
(8) gradient boosting + tuned	0.72074	0.74461
(9) lasso regression	0.72074	0.73358
(10) ridge regression	0.72408	0.73485

3.11 Naive Bayes

Finally, we use Naive Bayes in order to classify our customers.

Surprisingly, we achieved a BA as high as our best model so far, which is gradient boosting with parameter tuning.

```
set.seed(2021)
nb <- naiveBayes(as.factor(Churn) ~., data=train_set, laplace=1)

pred_nb <- predict(nb, newdata=test_set)
CM <- confusionMatrix(as.factor(pred), as.factor(test_set$Churn))
```

Table 16: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586
(8) gradient boosting + tuned	0.72074	0.74461
(9) lasso regression	0.72074	0.73358
(10) ridge regression	0.72408	0.73485
(11) naive bayes	0.72074	0.74461

3.12 Ensemble

I also wanted to include all our best models in one model, which is the ultimate ensemble model. However, this hasn't helped us to increase the BA.

```
# ensemble
ensemble <- cbind(logreg = pred_lr == 1, rf = pred_rf == 1, adaboost = pred_gb == 1,
                  lasso = pred_lasso == 1, ridge = pred_ridge == 1, nb = pred_nb == 1)

ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, 1, 0)
CM <- confusionMatrix(as.factor(ensemble_preds), as.factor(test_set$Churn))
```

Table 17: Model results

method	Accuracy	BalancedAccuracy
(1) logistic regression	0.78930	0.70907
(2) logistic regression with feature engineering	0.79599	0.70962
(3) logistic regression + balanced train_set	0.72074	0.73458

method	Accuracy	BalancedAccuracy
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(5) random forest	0.72742	0.72911
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586
(8) gradient boosting + tuned	0.72074	0.74461
(9) lasso regression	0.72074	0.73358
(10) ridge regression	0.72408	0.73485
(11) naive bayes	0.72074	0.74461
(12) ensemble	0.72575	0.72998

4. Results

As a result, after modifying and training 11 models, the best balanced accuracy was achieved by gradient boosting with tuning and naive bayes. Firstly, by learning from earlier missclassifications, gradient boosting becomes better after every iteration. However, it has the danger of overfitting. Secondly, even though naive bayes is way less computational expensive, it performs as good as gradient boosting. The third place goes to logistic regression, yet another rather "simple" but highly effective algorithm for classification. Especially if there was feature engineering involved, logistic regression can be a powerful method.

method	Accuracy	BalancedAccuracy
(8) gradient boosting + tuned	0.72074	0.74461
(11) naive bayes	0.72074	0.74461
(4) logistic regression + balanced train_set + w/o NA variables	0.72241	0.73973
(6) random forest + mtry = 2	0.72408	0.73586
(7) gradient boosting	0.72408	0.73586
(10) ridge regression	0.72408	0.73485
(3) logistic regression + balanced train_set	0.72074	0.73458
(9) lasso regression	0.72074	0.73358
(12) ensemble	0.72575	0.72998
(5) random forest	0.72742	0.72911
(2) logistic regression with feature engineering	0.79599	0.70962
(1) logistic regression	0.78930	0.70907

5. Conclusion

In conclusion, I had a lot of fun exploring, understanding and building models on the telecom dataset as well as interpreting the results and thinking about ways about how to improve the models.

In general it can be said that numerical variables are highly important for the algorithms. Especially if its about a recorded time or monetary value. We saw that tenure and total charges correlated whether a customer churned or not pretty well. However, having data on how the customer is paying and the contract lengths were also important predictors for the churn rate. Surprisingly, gender didn't play a big role in churn prediction.

For future research, these models could be improved even further and different models can be applied such as bagging or artificial neural network. Also more features can be combined to support the models. As the dataset wasn't too big to begin with, I purposely didn't create a validation set, even though it would have been nice to see which models would have overfitted the underlying train data.

Churned customers are a crucial topic for every business and busines should pay close attention to it. With the models, I hope I was able to give a deeper understanding which variables are particular important in order to predict but most importantly to prevent customers from canceling their contract.

However, I am very happy that I was able to apply all my knowledge that I learned throughout this course and I am looking forward to my next data science journeys!