Churn Prediction Project Report - HarvardX PH125.9x

BenoitAnger

2022-12-29

1 Introduction

Churn, or the loss of customers, is a significant concern for companies that have a subscription of pay as you go model, which generally spend significant money to acquire customers hoping they will be retain for a long time. This is especially important in the telecommunications industry. In order to address this issue, it is important for telecommunications companies 'telcos') to understand the factors that contribute to churn and to be able to predict which customers are at risk of leaving. Indeed, a telco which understand customers at risk of churning could be targeted by marketing efforts to limit the chances for them to leave. This report aims to explore churn in a telecommunications company using data science techniques to build a predictive model of churn and understand the main factors influencing the risk of churn.

The dataset for this study consists of over 7000 observations of customers, with 20 features that may potentially influence churn. The dataset is provided by Kaggle.com and is available to download under the name "Telco Customer Churn" (link in the Reference section).

The goal of this analysis is to provide insights that can inform retention strategies for the telecommunications company. As mentioned, a telco would be interested in identifying high risk customers and try to retain them by offering discounts or other incentives. Sensitivity (how well the model will identify customers at risk) is therefore an important measure of success of the model. But specificity (how well the model identifies customer at low risk) is also an important criteria as the telco would like to avoid wasting marketing incentives for customers who are unlikely to churn. Balanced accuracy is therefore an important indicator of the quality of the model.

The objectives of the study are to (1) build a predictive model which achieves both a high level of accuracy (2) identify the most important predictors of churn.

The report is structured into four main parts. In the first part, we will prepare the data for analysis and perform an exploratory data analysis to understand the characteristics of the dataset. In the second part, we will build and evaluate multiple predictive models to determine the best one for our objectives. In the third part, we will analyze the results of the models and discuss the implications. Finally, we will provide a conclusion and list the references used in the study.

To help the reader, we have kept clear definitions of each variables used in the project:

1.1 Data dictionary

1.1.1 Dataframes

Data frame	Description
data_original	Original dataset downloaded as csv and imported into R
data_clean	Dataset after data cleaning operations
train_set	Partition of the whole data (data_clean) allocated to train the models

Data frame	Description
test_set	Partition of the whole data (data_clean) allocated to test the models

There are a few additional dataframes, derived from the main datasets shown here, which are created for the purpose of the data visualization and the modeling which are not listed here.

1.1.2 Columns (predictors or features) in original dataframes

Variable Name	Description	Values	Type
customerID	Customer ID	String	Feature/predictor
gender	Whether the customer is a male or a female	Male, Female	Feature/predictor
SeniorCitizen	Whether the customer is a senior citizen or not	1, 0	Feature/predictor
Partner	Whether the customer has a partner or not	Yes, No	Feature/predictor
Dependents	Whether the customer has dependents or not	Yes, No	Feature/predictor
tenure	Number of months the customer has stayed with the company	Numeric	Feature/predictor
PhoneService	Whether the customer has a phone service or not	Yes, No	Feature/predictor
MultipleLines	Whether the customer has multiple lines or not	Yes, No, No phone service	Feature/predictor
InternetService	Customer's internet service provider	DSL, Fiber optic, No	Feature/predictor
OnlineSecurity	Whether the customer has online security or not	Yes, No, No internet service	Feature/predictor
OnlineBackup	Whether the customer has online backup or not	Yes, No, No internet service	Feature/predictor
DeviceProtection	Whether the customer has device protection or not	Yes, No, No internet service	Feature/predictor
TechSupport	Whether the customer has tech support or not	Yes, No, No internet service	Feature/predictor
StreamingTV	Whether the customer has streaming TV or not	Yes, No, No internet service	Feature/predictor
StreamingMovies	Whether the customer has streaming movies or not	Yes, No, No internet service	Feature/predictor
Contract	The contract term of the customer	Month-to-month, One year, Two year	Feature/predictor
PaperlessBilling	Whether the customer has paperless billing or not	Yes, No	Feature/predictor
PaymentMethod	The customer's payment method	Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))	Feature/predictor

Variable Name	Description	Values	Type
MonthlyCharges	The amount charged to the customer monthly	Numeric	Feature/predictor
TotalCharges	The total amount charged to the customer	Numeric	Feature/predictor
Churn	Whether the customer churned or not	Yes or No	Outcome to predict

2. Methodology & Modeling Process

2.1 Methodology

The methodology applied is the following:

1. Exploration of the data sets in order to have an overview of the available data and variables and visualize them to gain insights
2. Data cleaning to prepare the datasets for the modeling step
3. Modeling, building several algorithms that are potential candidates for our situation and evaluate them based on accuracy and balanced accuracy to retain the best possible model. The detailed approach is describe in the first paragraph (2.5.1) of modeling section.

2.2 Datasets overview

2.2.1 Overview of the original dataset

The dataset contains 7043 customer observations and 21 columns.

Here is an overview of the columns:

```
## '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 year" ...
##  $ PaperlessBilling: chr  "Yes" "No" "Yes" "No" ...
##  $ PaymentMethod   : chr  "Electronic check" "Mailed check" "Mailed check" "Bank transfer (automatic)
##  $ 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" ...
```

The outcome to predict (Churn) is represented as follows:

```
## # A tibble: 2 x 3
##   Churn Count percentage
##   <chr> <int>      <dbl>
## 1 No     5174      0.735
## 2 Yes    1869      0.265
```

Insight(s):

(1) There are many predictors

(2) Most of the predictors are categories rather than numeric (continuous)

(3) The first column is a customer ID hence has no predicted power and can be deleted

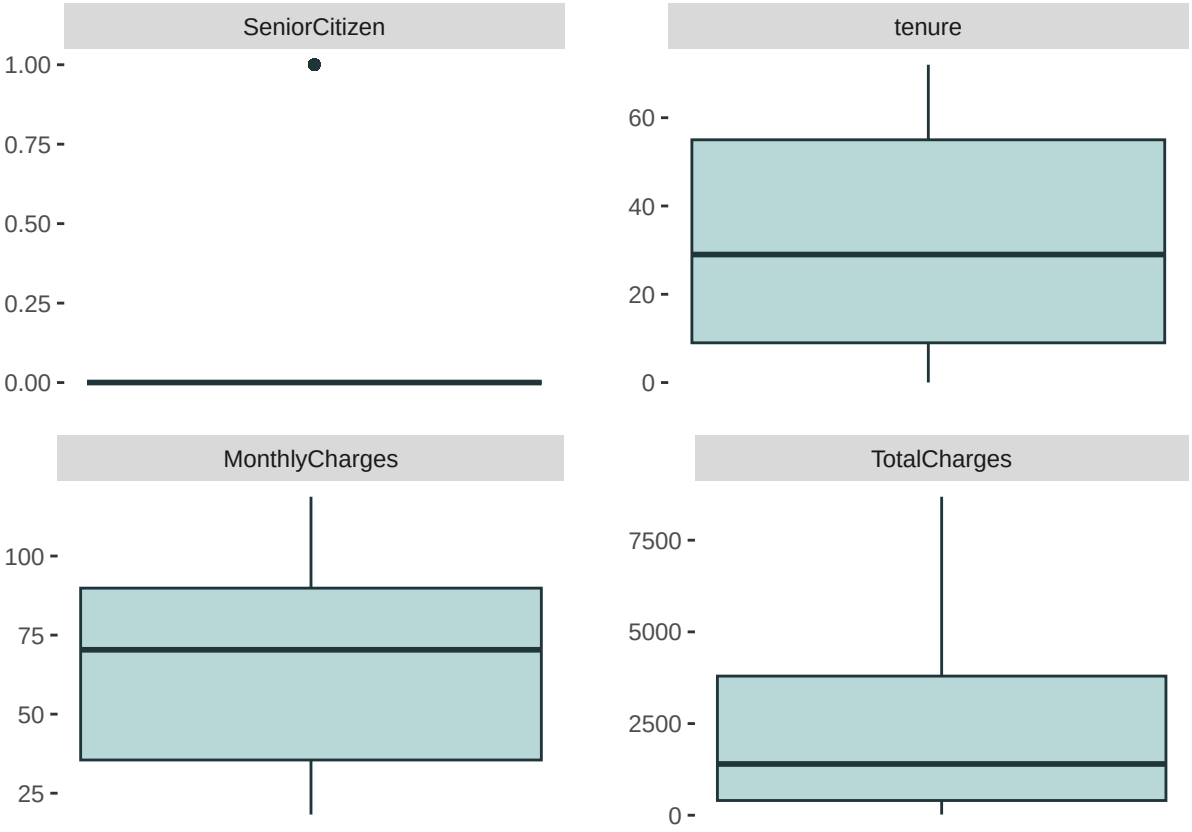
(4) The churn rate is 26.54 %

2.2.2 Overview of the continuous predictors

Here is an overview of the numeric (continuous) predictors:

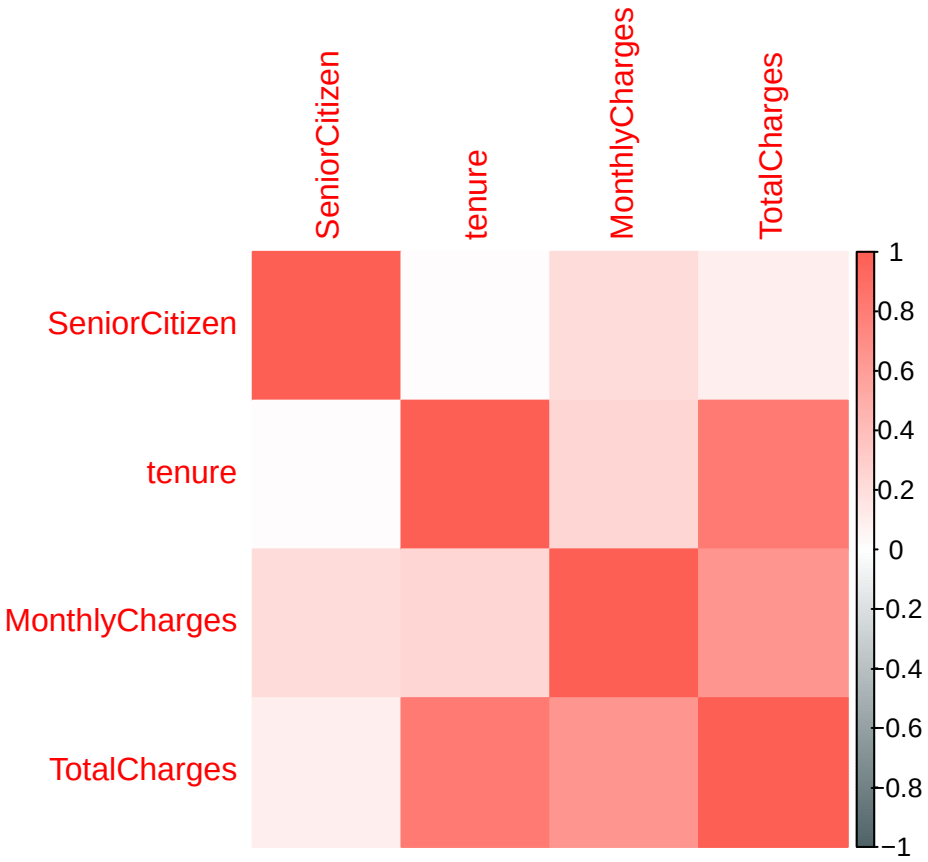
```
##  SeniorCitizen       tenure      MonthlyCharges    TotalCharges
##  Min.   :0.0000   Min.   : 0.00   Min.   : 18.25   Min.   :  18.8
##  1st Qu.:0.0000   1st Qu.: 9.00   1st Qu.: 35.50   1st Qu.: 401.4
##  Median :0.0000   Median :29.00   Median : 70.35   Median :1397.5
##  Mean   :0.1621   Mean   :32.37   Mean   : 64.76   Mean   :2283.3
##  3rd Qu.:0.0000   3rd Qu.:55.00   3rd Qu.: 89.85   3rd Qu.:3794.7
##  Max.   :1.0000   Max.   :72.00   Max.   :118.75   Max.   :8684.8
##                                                    NA's   :11
```

We can visualize the data here:



We can see that those variables are quite widely distributed. This should make them good predictors. However, one might suspect a high degree of correlation between them, which would decrease significantly the predictive power that we could expect.

Looking at correlation, we can indeed see that they are very correlated:



Especially:

- Correlation between Tenure and Total Charges: 0.83
- Correlation between Monthly Charges and Total Charges: 0.65
- Correlation between Total Charges and the Tenure multiplied by the Monthly Charges: 0.9996

Insight(s):

(5) Very few Seniors in the dataset

(6) Tenure between Min. : 0.00 and 3rd Qu.:55.00 months with a Median :29.00

(7) Monthly charges between Min. : 18.25 and 3rd Qu.: 89.85 dollars with a Median : 70.35

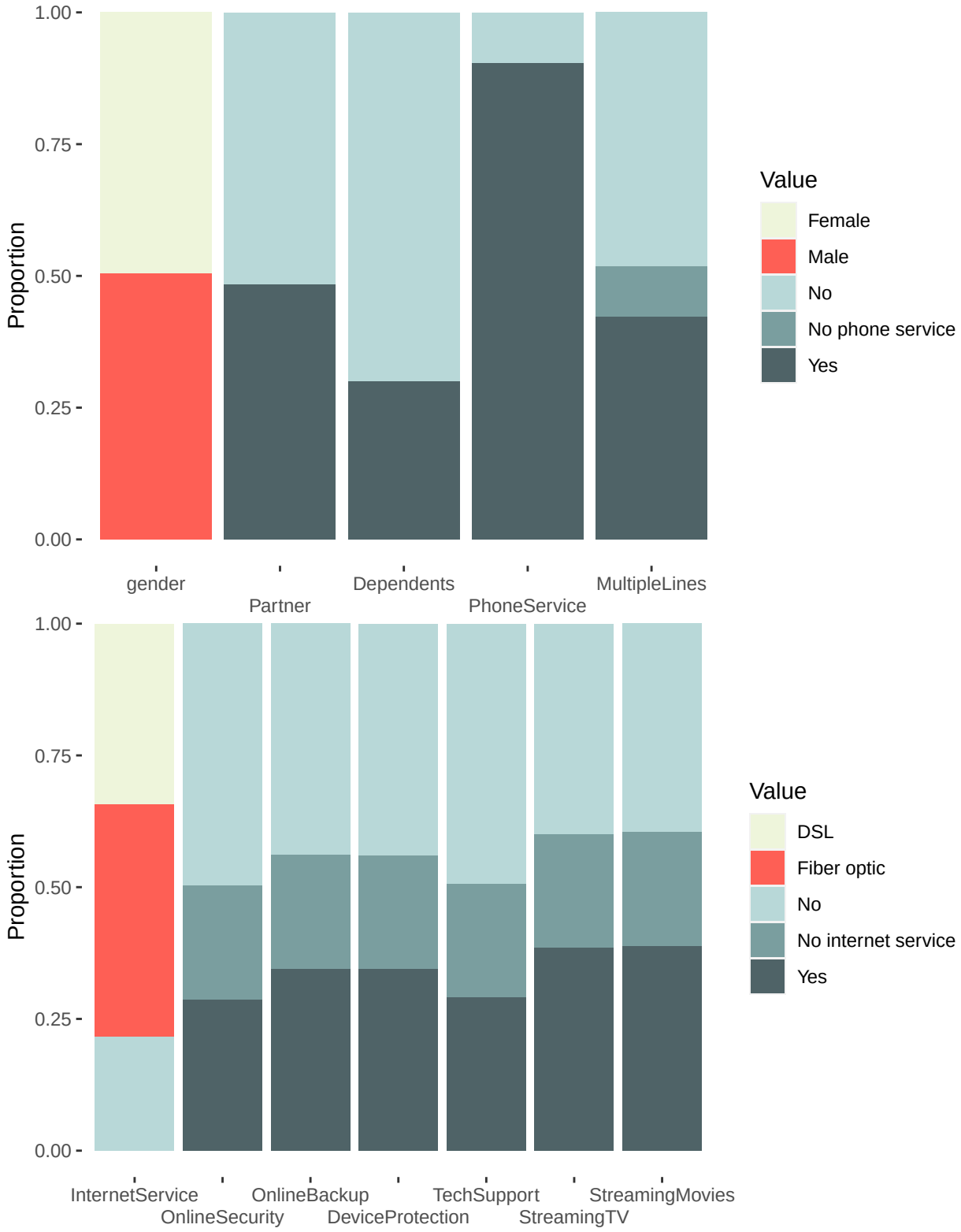
(8) Total charges Min. : 18.8 and 3rd Qu.:3794.7 months with a Median :1397.5

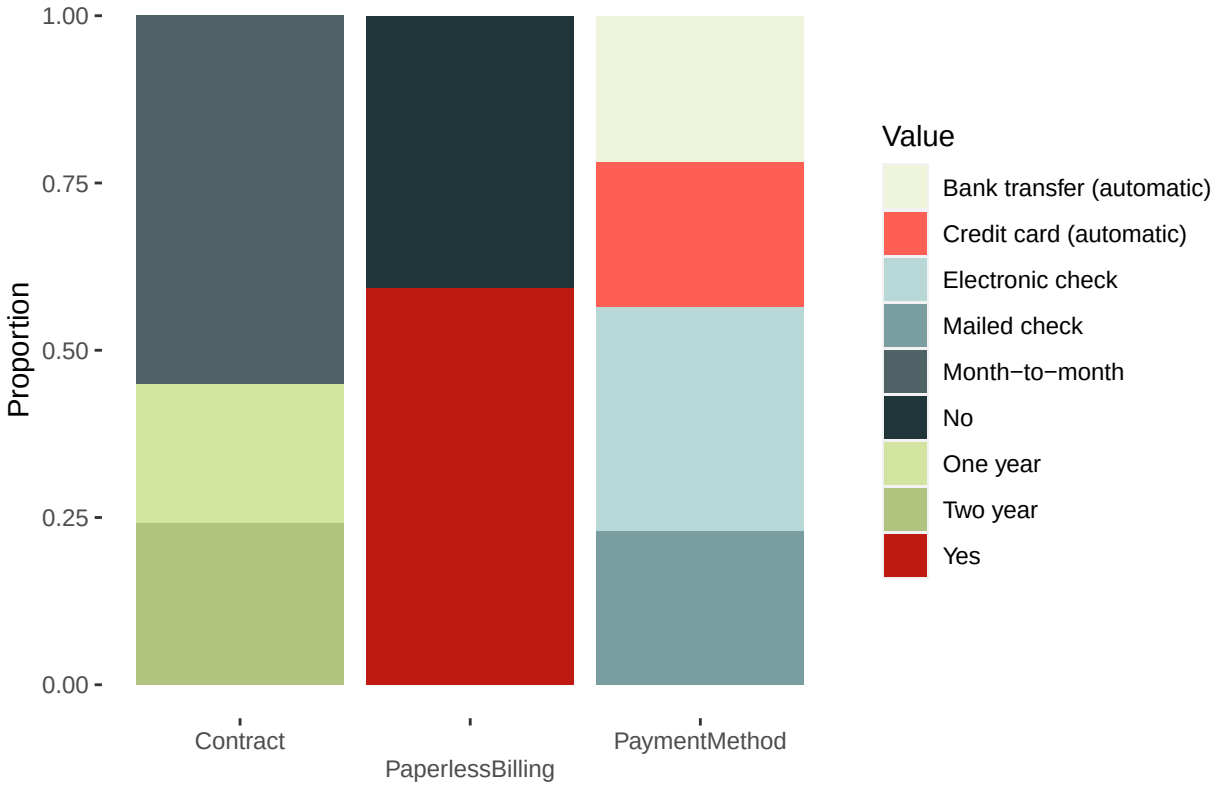
(9) Some NA's in the Total charges

(10) Though Tenure, Monthly Charges and Total Charges seem widely distributed and potential good predictors, they are highly correlated; Total Charges is especially almost perfectly correlated with the Tenure multiplied by the Monthly Charges

2.2.3 Overview of the categorical predictors

We can visualize the data here:


1.00
0.75
0.50
0.25
0.00
Proportion
gender
Partner
Dependents
PhoneService
MultipleLines
Value
Female
Male
No
No phone service
Yes
1.00
0.75
0.50
0.25
0.00
Proportion
InternetService
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
Value
DSL
Fiber optic
No
No internet service
Yes

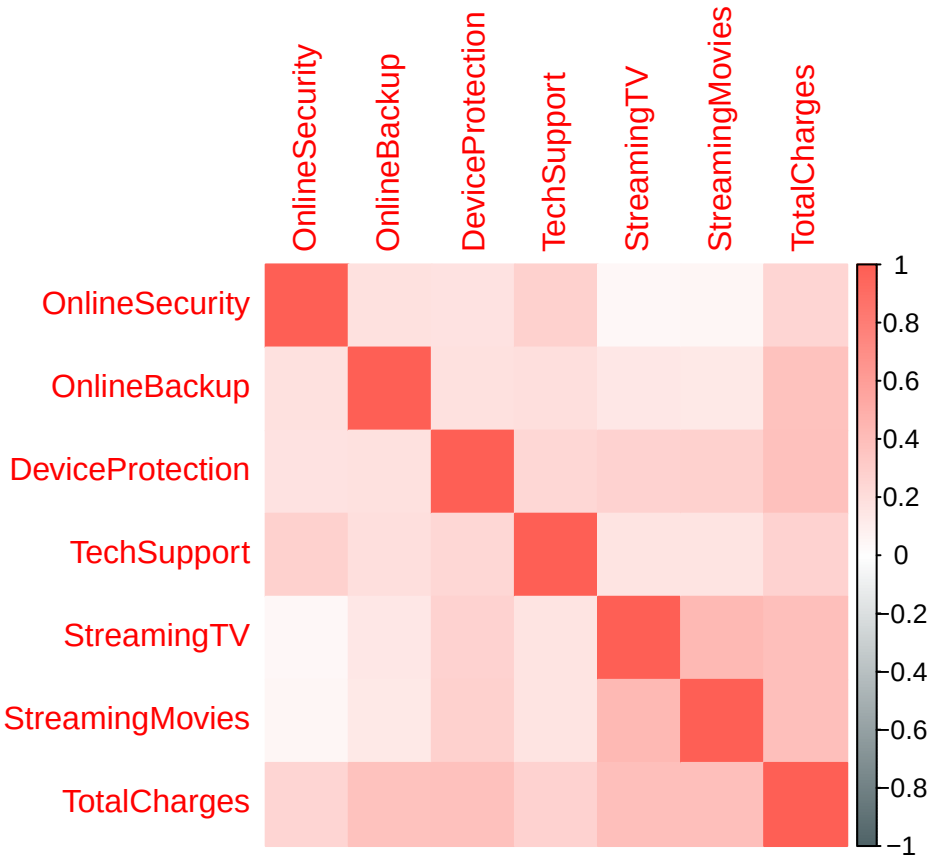


Those variables have binary outcome or at best a few categories which would probably not make them very insightful predictors.

Moreover, some variables are probably highly correlated:

- First, we can reasonably assume that all optional services (Multiple lines, Internet services, and the six additional services to Internet: Online security, Online Backup, Device Protection, Tech support, Streaming TV, Streaming movies) are highly correlated with Monthly Charges
- Second, the six additional services to Internet (Online security, Online Backup, Device Protection, Tech support, Streaming TV, Streaming movies) have all the same category "No internet" with the same customers

This can be visualized here:



Which confirms to some extent the expectations.

Insight(s):

(11) Many variables, rather widely distributed

(12) But probably a limited predicted power as the variables are binary or categorical with a few categories, correlated among the range of categorical variables and correlated with some continuous ones (Charges)

2.4 Datasets preparation

In order to prepare the data for the machine learning algorithm, we will perform a few tasks:

1. We have seen in the exploration that there are some NA's in the dataset, more precisely there are 'r length(which(is.na(data_original))) missing data points. As this is a very low number, we can safely delete the rows with missing data without affecting our results. We do this and store the data in a new dataframe (data_clean) that we will then use for the modeling.
2. The customerID has no predictive power as this is only an ID, hence we remove it to have cleaner data (that R will treat more quickly).
3. We create two partitions of the data to have one dataset to train the models and another one to test the models. We choose to allocate 80% to the train dataset and 20% to the test dataset. We call those sets respectively "train_set" and "test_set"
4. Machine learning algorithms need factored datasets, so we convert "train_set" and "test_set" as factored data sets.

2.5 Modeling

2.5.1 Approach

Thanks to the data exploration, we understand that we have to predict a binary outcome with many predictors, but with several ones that seem correlated and a lot which are categorical.

Based on the course and the research (see references in the reference section), we believe that we should try several types of models adapted to the prediction of binary outcomes. In order to be comprehensive, we will try all the models presented in the course (some are not applicable such as LDA and QDA due to the binary type of outcome to predict and the large number of predictors). We will also try additional methods found in the research as good model for predicting binary outcome. Finally, we will try to ensemble several machine learning algorithm, applying the technique presented in the course.

2.5.2 Building algorithms

2.5.2.1 Logistic regression (glm) The idea to use this model comes from both the course and the research.

In order to try to get the best results, we train and immediately optimize the model using the caret package. The code is the following:

```
# configurate the control parameters
fit_control_glm <- trainControl(method = "repeatedcv", number = 10, repeats = 50)

# train the model
fit_glm <- train(Churn ~ ., data = train_set, method = "glm", family = "binomial",
trControl = fit_control_glm)

# build predictions and evaluate them
predictions_glm <- predict(fit_glm, newdata = test_set)
cm_glm<-confusionMatrix(data = predictions_glm, reference = test_set$Churn, positive =
"Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  929 170
##        Yes 104 204
##
##                Accuracy : 0.8053
##                  95% CI : (0.7836, 0.8257)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 2.857e-10
##
##                   Kappa : 0.4713
##
##  Mcnemar's Test P-Value : 8.609e-05
##
##             Sensitivity : 0.5455
##             Specificity : 0.8993
##          Pos Pred Value : 0.6623
##          Neg Pred Value : 0.8453
##              Prevalence : 0.2658
##          Detection Rate : 0.1450
##    Detection Prevalence : 0.2189
```

```
##       Balanced Accuracy : 0.7224
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885

2.5.2.2 Nearest neighbors (knn) The idea to use this model comes from both the course and the research.

In order to try to get the best results, we train and immediately optimize the model using the caret package. The code is the following:

```
# knn k optimization function
accuracy_function_knn <- function(x){
    fit_knn <- fit_knn <- knn3(Churn ~ ., data = train_set, k=x)
    predictions_knn <- predict(fit_knn, newdata = test_set, type = "class")
    confusionMatrix(data = predictions_knn, reference =
    test_set$Churn)$overall["Accuracy"]
}

# finding the k that optimizes the accuracy
knn_ks <- seq(1, 50, 1)
knn_accuracies<-sapply(knn_ks, accuracy_function_knn)
knn_kfinal<- knn_ks[which.max(knn_accuracies)]

# train the model
fit_knn <- knn3(Churn ~ ., data = train_set, k= knn_kfinal)

# build predictions and evaluate them
predictions_knn <- predict(fit_knn, newdata = test_set, type = "class")
cm_knn<-confusionMatrix(data = predictions_knn, reference = test_set$Churn, positive =
"Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  969 221
##        Yes  64 153
##
##                Accuracy : 0.7974
##                  95% CI : (0.7755, 0.8182)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 2.01e-08
##
##                   Kappa : 0.4008
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.4091
##             Specificity : 0.9380
##          Pos Pred Value : 0.7051
##          Neg Pred Value : 0.8143
##              Prevalence : 0.2658
##          Detection Rate : 0.1087
##    Detection Prevalence : 0.1542
##       Balanced Accuracy : 0.6736
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677

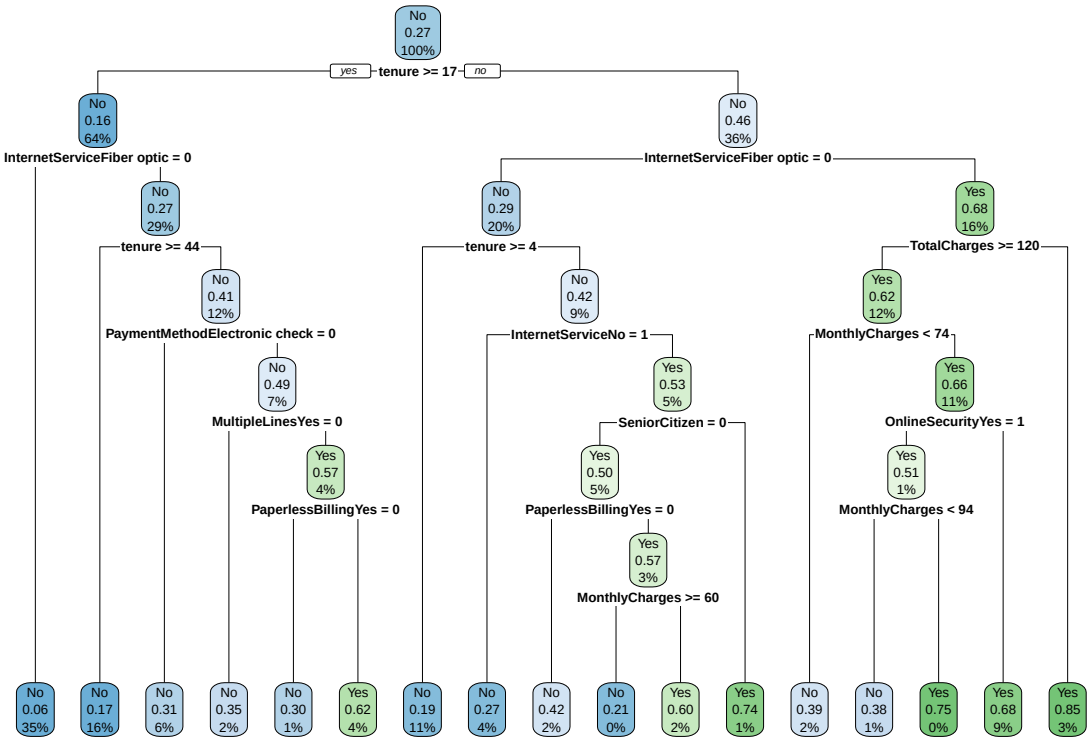
2.5.2.3 Decision tree (rpart) The idea to use this model comes from both the course and the research. The course especially mention that decision tree are well suited to predict categorical outcomes.

In order to try to get the best results, we train and immediately optimize the model using the caret package. The code is the following (we use here the Caret package which after testing gave better performance than the Rpart package):

```
# train the model with optimization embedded in the code
fit_rpart <- train(Churn ~ ., method = "rpart", tuneGrid = data.frame(cp = seq(0, 0.05,
len = 25)), data = train_set)

# build predictions and evaluate them
predictions_rpart <- predict(fit_rpart, newdata = test_set)
cm_rpart<-confusionMatrix(data = predictions_rpart, reference = test_set$Churn, positive
="Yes")
```

The tree built is the following:



The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  953 193
##        Yes  80 181
##
##                Accuracy : 0.806
##                  95% CI : (0.7843, 0.8263)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 1.890e-10
##
##                   Kappa : 0.4499
##
##  Mcnemar's Test P-Value : 1.214e-11
##
##             Sensitivity : 0.4840
##             Specificity : 0.9226
##          Pos Pred Value : 0.6935
##          Neg Pred Value : 0.8316
##              Prevalence : 0.2658
##          Detection Rate : 0.1286
##    Detection Prevalence : 0.1855
##       Balanced Accuracy : 0.7033
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564

2.5.2.4 Random forest (rf) The idea to use this model comes from both the course and the research.

In order to try to get the best results, we train and immediately optimize the model using the caret package. The code is the following (we use here the Random Forest package which after testing gave better performance than the Rborist method of the Caret package):

```
# train the model with optimization embedded in the code
fit_rf <-  randomForest(y=train_set$Churn, x=train_set[,-20], prox=TRUE)

# build predictions and evaluate them
predictions_rf <- predict(fit_rf , newdata = test_set)
cm_rf<-confusionMatrix(data = predictions_rf, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  947 186
##        Yes  86 188
##
##                Accuracy : 0.8067
##                  95% CI : (0.7851, 0.827)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 1.245e-10
##
##                   Kappa : 0.4585
##
##  Mcnemar's Test P-Value : 1.940e-09
##
##             Sensitivity : 0.5027
##             Specificity : 0.9167
##          Pos Pred Value : 0.6861
##          Neg Pred Value : 0.8358
##              Prevalence : 0.2658
##          Detection Rate : 0.1336
##    Detection Prevalence : 0.1947
##       Balanced Accuracy : 0.7097
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677

Model #	Algorithm details	Accuracy	Balanced Accuracy
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106

2.5.2.5 Gradient boosting (gbm) The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```
# train the model
fit_gbm<-train(Churn~.,data=train_set, method ='gbm')

# build predictions and evaluate them
predictions_gbm<-predict(fit_gbm, newdata = test_set)
cm_gbm<-confusionMatrix(data = predictions_gbm, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  949 200
##        Yes  84 174
##
##                Accuracy : 0.7982
##                  95% CI : (0.7762, 0.8188)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 1.396e-08
##
##                   Kappa : 0.4261
##
##  Mcnemar's Test P-Value : 8.854e-12
##
##             Sensitivity : 0.4652
##             Specificity : 0.9187
##          Pos Pred Value : 0.6744
##          Neg Pred Value : 0.8259
##              Prevalence : 0.2658
##          Detection Rate : 0.1237
##    Detection Prevalence : 0.1834
##       Balanced Accuracy : 0.6920
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620

2.5.2.6 Multilayer perceptron (mlp)

The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```
# train the model
fit_mlp<-train(Churn~.,data=train_set, method ='mlp')

# build predictions and evaluate them
predictions_mlp<-predict(fit_mlp, newdata = test_set)
cm_mlp<-confusionMatrix(data = predictions_mlp, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1033  374
##        Yes    0    0
##
##                Accuracy : 0.7342
##                  95% CI : (0.7103, 0.7571)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 0.5139
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.7342
##              Prevalence : 0.2658
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000

2.5.2.7 Naïve bayes (nb)

The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```r
# train the model
fit_nb<-train(Churn~.,data=train_set, method ='naive_bayes')

# build predictions and evaluate them
predictions_nb<-predict(fit_nb, newdata = test_set)
cm_nb<-confusionMatrix(data = predictions_nb, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  635  43
##        Yes 398 331
##
##                Accuracy : 0.6866
##                  95% CI : (0.6616, 0.7108)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3836
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8850
##             Specificity : 0.6147
##          Pos Pred Value : 0.4540
##          Neg Pred Value : 0.9366
##              Prevalence : 0.2658
##          Detection Rate : 0.2353
##    Detection Prevalence : 0.5181
##       Balanced Accuracy : 0.7499
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000
7	Naïve bayes (nb)	0.6865672	0.7498706

2.5.2.8 Bayesian generalized linear model (bglm) The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```r
# train the model
fit_bglm<-train(Churn~.,data=train_set, method ='bayesglm')

# build predictions and evaluate them
predictions_bglm<-predict(fit_bglm, newdata = test_set)
cm_bglm<-confusionMatrix(data = predictions_bglm, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  928 170
##        Yes 105 204
##
##                Accuracy : 0.8045
##                  95% CI : (0.7828, 0.825)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 4.299e-10
##
##                   Kappa : 0.4699
##
##  Mcnemar's Test P-Value : 0.0001137
##
##             Sensitivity : 0.5455
##             Specificity : 0.8984
##          Pos Pred Value : 0.6602
##          Neg Pred Value : 0.8452
##              Prevalence : 0.2658
##          Detection Rate : 0.1450
##    Detection Prevalence : 0.2196
##       Balanced Accuracy : 0.7219
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000
7	Naïve bayes (nb)	0.6865672	0.7498706
8	Bayesian GLM (bglm)	0.8045487	0.7219044

2.5.2.9 Neural network (nn) The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```r
# train the model
fit_nn<-train(Churn~.,data=train_set, method ='pcaNNet')

# build predictions and evaluate them
predictions_nn<-predict(fit_nn, newdata = test_set)
cm_nn<-confusionMatrix(data = predictions_nn, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  925 168
##        Yes 108 206
##
##                Accuracy : 0.8038
##                  95% CI : (0.7821, 0.8243)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 6.44e-10
##
##                   Kappa : 0.4703
##
##  Mcnemar's Test P-Value : 0.0003832
##
##             Sensitivity : 0.5508
##             Specificity : 0.8955
##          Pos Pred Value : 0.6561
##          Neg Pred Value : 0.8463
##              Prevalence : 0.2658
##          Detection Rate : 0.1464
##    Detection Prevalence : 0.2232
##       Balanced Accuracy : 0.7231
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000
7	Naïve bayes (nb)	0.6865672	0.7498706
8	Bayesian GLM (bglm)	0.8045487	0.7219044
9	Neural network (nn)	0.8038380	0.7231261

2.5.2.10 Support vector machine (svm) The idea to use this model comes from both the research.

In order to try to get the best results, we train the model using the caret package. The code is the following:

```r
# train the model
fit_svm<-train(Churn~.,data=train_set, method ='svmLinearWeights')

# build predictions and evaluate them
predictions_svm<-predict(fit_svm, newdata = test_set)
cm_svm<-confusionMatrix(data = predictions_svm, reference = test_set$Churn, positive
="Yes")
```

The confusion matrix for the model is the following:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  924 182
##        Yes 109 192
##
##                Accuracy : 0.7932
##                  95% CI : (0.7711, 0.8141)
##     No Information Rate : 0.7342
##     P-Value [Acc > NIR] : 1.642e-07
##
##                   Kappa : 0.4349
##
##  Mcnemar's Test P-Value : 2.435e-05
##
##             Sensitivity : 0.5134
##             Specificity : 0.8945
##          Pos Pred Value : 0.6379
##          Neg Pred Value : 0.8354
##              Prevalence : 0.2658
##          Detection Rate : 0.1365
##    Detection Prevalence : 0.2139
##       Balanced Accuracy : 0.7039
##
##        'Positive' Class : Yes
##
```

The main results of this algorithm are as follows:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000
7	Naïve bayes (nb)	0.6865672	0.7498706
8	Bayesian GLM (bglm)	0.8045487	0.7219044
9	Neural network (nn)	0.8038380	0.7231261
10	Support vector machine (svm)	0.7931770	0.7039255

2.5.3 Build the ensemble algorithm

Despite testing many model, we are unfortunately topping at around 80% accuracy. In order to try and improve the performance, we will try a last technique described in the course: ensembling different models into one.

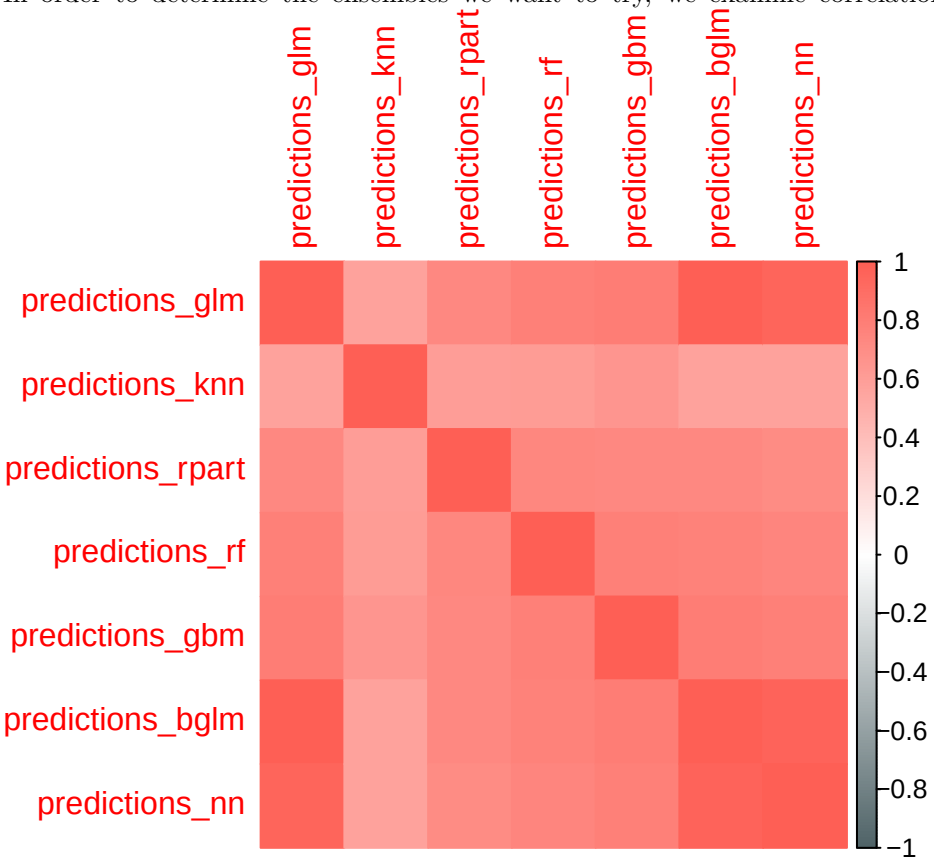
As we have binary results, we need an odd number of models to combine. Since we have built 10 models and two of them models, Naives bayes and Multilayer perceptron, have a very poor performance compared to the other ones, we eliminate those one, hence we will chose 7 models (to keep an even number).

The 7 best ones based on accuracy are the followings:

Model #	Algorithm details	Accuracy	Balanced Accuracy
4	Random forest (rf)	0.8066809	0.7097106
3	Decision tree (rpart)	0.8059701	0.7032564
1	Logistic regression (glm)	0.8052594	0.7223885
8	Bayesian GLM (bglm)	0.8045487	0.7219044
9	Neural network (nn)	0.8038380	0.7231261
5	Gradient boosting (gbm)	0.7981521	0.6919620
2	Nearest neighbors (knn)	0.7974414	0.6735677

So we build a table with the predictions of those 7 models.

In order to determine the ensembles we want to try, we examine correlation between the predictions:



We decide to build 3 ensembles:

- Ensemble 1: made up of the three less correlated predictions: logistic regression (glm), nearest neighbor (knn) and decision tree (rpart)

- Ensemble 2: adding two more models, selecting the least correlated ones: random forest (rf) and gradient boosting (gbm)
- Ensemble 3: adding the last two models: bayesian glm (bglm) and nueral network (nn)

We then assess the performance against the test data set and we get the following results:

Model #	Algorithm details	Accuracy	Balanced Accuracy
1	Logistic regression (glm)	0.8052594	0.7223885
2	Nearest neighbors (knn)	0.7974414	0.6735677
3	Decision tree (rpart)	0.8059701	0.7032564
4	Random forest (rf)	0.8066809	0.7097106
5	Gradient boosting (gbm)	0.7981521	0.6919620
6	Multilayer perceptron (mlp)	0.7341862	0.5000000
7	Naïve bayes (nb)	0.6865672	0.7498706
8	Bayesian GLM (bglm)	0.8045487	0.7219044
9	Neural network (nn)	0.8038380	0.7231261
10	Support vector machine (svm)	0.7931770	0.7039255
E1	Ensemble 1	0.8102345	0.7087192
E2	Ensemble 2	0.8081023	0.7064143
E3	Ensemble 3	0.8073916	0.7178704

3. Results

We first discuss the best model and then examine the importance of the features.

3.1 Best model

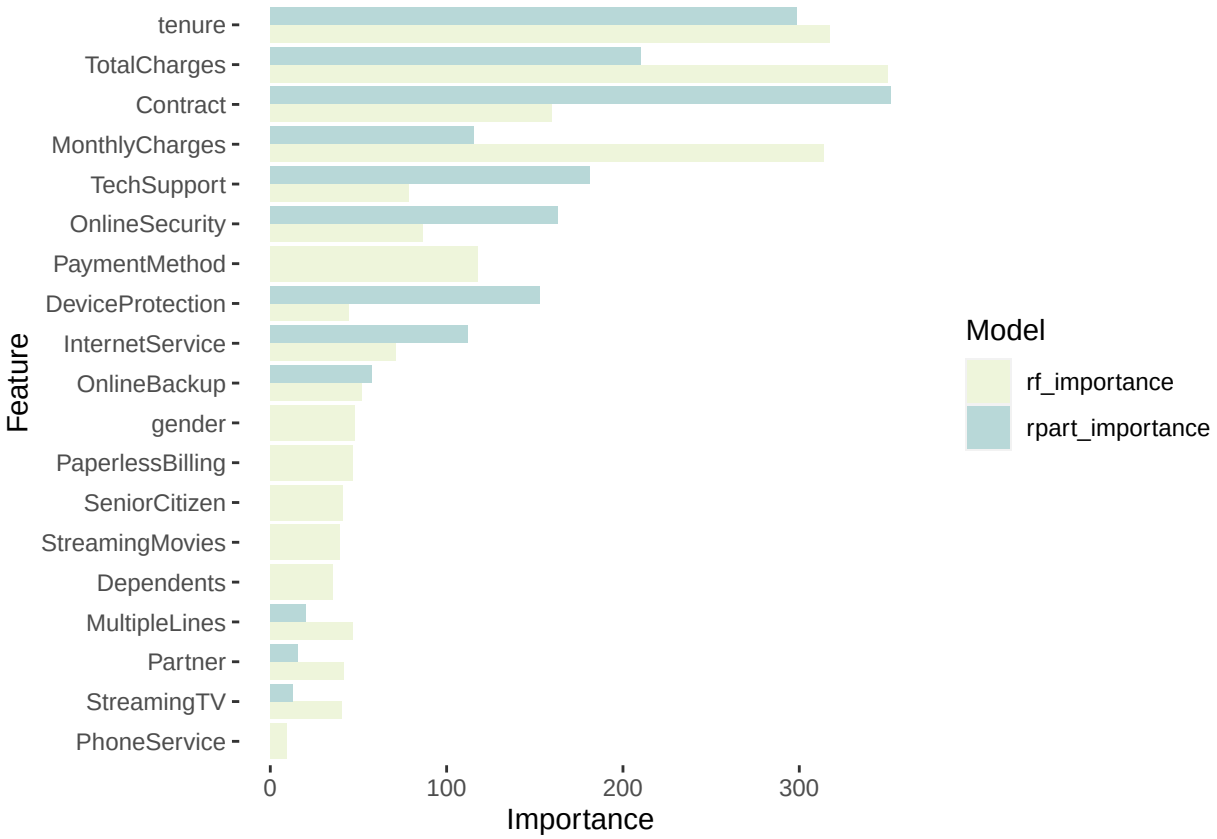
We can draw the following conclusions from the result table displayed above:

- Most of the models have similar performance (Accuracy around 80% and Balanced accuracy slightly above 70%). Only two models have a much poorer performance
- This performance can be improved significantly by ensembling several model. The first ensemble of models (ensemble 1 which is combining 3 models) give the best performance with an Accuracy of 81.02 % and a Balanced accuracy of 70.87 %. The fact that this ensemble performs better seems logical as the 3 single models ensembled seemed the least correlated.
- While the performance of the best model (Ensemble 1) is relatively good, we could have expected better results given the initial data set made up of more than 7 thousands observations and 20 features. The below expected performance is due to the fact that some features are correlated, many are categorical (and a lot even binary) and that the outcome is binary

3.2 Feature importance

For most of the models, it is not possible to understand the importance of features. For instance for logistic regression, it is difficult due to correlation of features. Two models can help us to understand feature importance: decision tree and random forest.

We can visualize the importance here:



As expected in the exploration part, we can observe that:

- Only a few features have high importance: Tenure, Contract, Total Charges and Monthly Charges (bearing in mind that Total Charges is extremely highly correlated with Tenure multiplied by Monthly Charges)
- Around half of the features have very low importance
- In general, continuous features have much more importance than categorical or binary ones

4. Conclusion

In this report, we analyzed a dataset of customer churn data and built 10 single models and 3 ensembles to predict churn rate. Our best model (Ensemble 1) achieved an accuracy of over 80%, which is highly useful for a company looking to reduce churn. With this level of accuracy, the company can run targeted marketing retention programs to improve customer retention.

Through our analysis, we also identified the key features that predict churn: Tenure, Contract type, and Charges paid. Understanding these factors can provide valuable insights for the company's retention efforts.

While the performance of our models could be improved by building and combining additional models, the current results are still promising. Additionally, the models can be optimized to prioritize either sensitivity or specificity based on the company's marketing campaign results and cost-benefit analysis.

5. References

Here is the list of resources used to build this report

Dataset

- Original Data:

Telco Customer Churn, https://www.kaggle.com/datasets/blastchar/telco-customer-churn, accessed on Dec. 29th, 2022

- Data copied on github: https://github.com/BN33/TelcoChurnProject-HarvardX-PH125.9x-Data-Science-Capstone-/blob/main/WA_Fn-UseC_-Telco-Customer-Churn.csv

Data visualisation

- Grouped, stacked and percent stacked barplot in ggplot2 https://r-graph-gallery.com/48-grouped-barplot-with-ggplot2.html, accessed on Dec. 29th, 2022
- Reorder a variable with ggplot2 https://r-graph-gallery.com/267-reorder-a-variable-in-ggplot2.html , accessed on Dec. 29th, 2022
- Plotting binary outcomes, https://casual-inference.com/post/plotting-binary-outcomes/, accessed on Dec. 29th, 2022
- ggplot2 colors : How to change colors automatically and manually? http://www.sthda.com/english/wiki/ggplot2-colors-how-to-change-colors-automatically-and-manually , accessed on Dec. 29th, 2022
- ggplot2 - Easy Way to Mix Multiple Graphs on The Same Page http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/, accessed on Dec. 29th, 2022
- Combine (rbind) data frames and create column with name of original data frames https://stackoverflow.com/questions/15162197/combine-rbind-data-frames-and-create-column-with-name-of-original-data-frames, accessed on Dec. 29th, 2022
- Chapter 4 Bivariate Graphs https://rkabacoff.github.io/datavis/Bivariate.html , accessed on Dec. 29th, 2022
- HarvardX Data Science Certificate Courses

Machine learning for binary outcome predictions

- Predicting Customer Churn using Machine Learning Models https://www.vshsolutions.com/blogs/predicting-customer-churn-using-machine-learning-models/, accessed on Dec. 29th, 2022
- Which machine learning method should choose to predict binary outcome based on several binary predictors? https://community.rstudio.com/t/which-machine-learning-method-should-choose-to-predict-binary-outcome-based-on-several-binary-predictors/154335/4, accessed on Dec. 29th, 2022
- Machine Learning: Trying to classify your data https://srnghn.medium.com/machine-learning-trying-to-predict-a-categorical-outcome-6ba542b854f5, accessed on Dec. 29th, 2022
- CHAPTER 1 Choosing the Right Classification Model https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html, accessed on Dec. 29th, 2022
- 6-Available Models https://topepo.github.io/caret/available-models.html, accessed on Dec. 29th, 2022
- The best machine learning model for binary classification https://ruslanmv.com/blog/The-best-binary-Machine-Learning-Model, accessed on Dec. 29th, 2022
- HarvardX "Data Science: Machine Learning" Course

R

- https://stackoverflow.com, accessed on Dec. 29th, 2022