BigDataTeam: RMD File Final

Owen Bell, Myisha Chaudhry, Kayleigh Habib, Maheep Jain, Marcus Rilling

2023-12-06


```
# import libraries
library(ggplot2)
library(dplyr)
library(GGally)
library(ggcorrplot)
library(ROSE)
library(caret)
library(randomForest)
library(scales)
library(rpart)
library(pROC)
```

Telecom Churn Data

1. Introduction

Business Problem:

Customers from a Telecommunication company are no longer using the company's services. This proves to be a business problem since the company is losing revenue for every customer that leaves. Aside from lost business, customer churn has a negative impact as it gives the company a bad reputation. This could deter future customers from wanting to use the company's services/buy the products as they know that their past customers were not satisfied and chose to discontinue the use of services/purchase products. Customers leaving could also be the result of an underlying problem, such as having a bad product or being overpriced. The reason it's worth looking into this business problem is that there are multiple factors at play that could be hurting a company with a high churn rate.

Analytic Problem:

The analytical problem in the case of analyzing churn rate would be determining which factors are common between customers that are choosing to discontinue the use of products or services offered by the company. The factors in this analysis can be used to predict whether a customer is likely to leave, which will allow the company

the opportunity to take mitigating action.

Goal:

The main goal here is to deal with the issue of churn rate as it can have a negative impact on the company in several regards. Machine learning algorithms will be used to predict if certain customers will churn and the ways in which it can be prevented.

2. Data Description

We used sample data from IBM for our customer churn case. The dataset contains over 7,000 records and 33 columns and represents customer data from a telecommunication company based in California, USA.

```
#import data
initial_data<- read.csv("Telco_customer_churn.csv")
head(initial_data)
```

	Customer… <chr>	Co… <int>	Country <chr>	State <chr>	City <chr>	Zip.Code <int>	Lat.Long <chr>
1	3668-QPYBK	1	United States	California	Los Angeles	90003	33.964131, -118.272783
2	9237-HQITU	1	United States	California	Los Angeles	90005	34.059281, -118.30742
3	9305-CDSKC	1	United States	California	Los Angeles	90006	34.048013, -118.293953
4	7892-POOKP	1	United States	California	Los Angeles	90010	34.062125, -118.315709
5	0280-XJGEX	1	United States	California	Los Angeles	90015	34.039224, -118.266293
6	4190-MFLUW	1	United States	California	Los Angeles	90020	34.066367, -118.309868

6 rows | 1-9 of 34 columns

```
# Look at the makeup of the data
str(initial_data)
```

```
## 'data.frame':    7043 obs. of  33 variables:
##  $ CustomerID       : chr  "3668-QPYBK" "9237-HQITU" "9305-CDSKC" "7892-POOKP" ...
##  $ Count            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Country          : chr  "United States" "United States" "United States" "United St
ates" ...
##  $ State            : chr  "California" "California" "California" "California" ...
##  $ City             : chr  "Los Angeles" "Los Angeles" "Los Angeles" "Los Angeles"
...
##  $ Zip.Code         : int  90003 90005 90006 90010 90015 90020 90022 90024 90028 9002
9 ...
##  $ Lat.Long         : chr  "33.964131, -118.272783" "34.059281, -118.30742" "34.04801
3, -118.293953" "34.062125, -118.315709" ...
##  $ Latitude         : num  34 34.1 34 34.1 34 ...
##  $ Longitude        : num  -118 -118 -118 -118 -118 ...
##  $ Gender           : chr  "Male" "Female" "Female" "Female" ...
##  $ Senior.Citizen   : chr  "No" "No" "No" "No" ...
##  $ Partner          : chr  "No" "No" "No" "Yes" ...
##  $ Dependents       : chr  "No" "Yes" "Yes" "Yes" ...
##  $ Tenure.Months    : int  2 2 8 28 49 10 1 1 47 1 ...
##  $ Phone.Service    : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ Multiple.Lines   : chr  "No" "No" "Yes" "Yes" ...
##  $ Internet.Service : chr  "DSL" "Fiber optic" "Fiber optic" "Fiber optic" ...
##  $ Online.Security  : chr  "Yes" "No" "No" "No" ...
##  $ Online.Backup    : chr  "Yes" "No" "No" "No" ...
##  $ Device.Protection: chr  "No" "No" "Yes" "Yes" ...
##  $ Tech.Support     : chr  "No" "No" "No" "Yes" ...
##  $ Streaming.TV     : chr  "No" "No" "Yes" "Yes" ...
##  $ Streaming.Movies : chr  "No" "No" "Yes" "Yes" ...
##  $ Contract         : chr  "Month-to-month" "Month-to-month" "Month-to-month" "Month-
to-month" ...
##  $ Paperless.Billing: chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ Payment.Method   : chr  "Mailed check" "Electronic check" "Electronic check" "Elec
tronic check" ...
##  $ Monthly.Charges  : num  53.9 70.7 99.7 104.8 103.7 ...
##  $ Total.Charges    : num  108 152 820 3046 5036 ...
##  $ Churn.Label      : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ Churn.Value      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Churn.Score      : int  86 67 86 84 89 78 100 92 77 97 ...
##  $ CLTV             : int  3239 2701 5372 5003 5340 5925 5433 4832 5789 2915 ...
##  $ Churn.Reason     : chr  "Competitor made better offer" "Moved" "Moved" "Moved" ...
```

```
# 7043 rows and 33 columns
# target variable: Churn Label
```

Dealing with Missing Values

One of the first things we need to do is determine the type and quality of the data. We want to ensure that any data fields used in model-building are also available on a consistent basis when the model is deployed. We should also ensure that the data does not include any personal data that could be used to identify individuals. We noticed that there are several columns which do not provide new or useful information for instance, all customers

in this data reside in California, so columns for State and Country provide no value. Next, we would look for any missing values, and determine whether these records should be dropped, or if values can be imputed from existing information.

```
# check for missing values
colSums(is.na(initial_data))
```

```
##        CustomerID             Count           Country             State
##                 0                 0                 0                 0
##              City          Zip.Code         Lat.Long          Latitude
##                 0                 0                 0                 0
##         Longitude            Gender    Senior.Citizen           Partner
##                 0                 0                 0                 0
##        Dependents     Tenure.Months     Phone.Service    Multiple.Lines
##                 0                 0                 0                 0
##  Internet.Service   Online.Security     Online.Backup Device.Protection
##                 0                 0                 0                 0
##      Tech.Support      Streaming.TV  Streaming.Movies          Contract
##                 0                 0                 0                 0
## Paperless.Billing    Payment.Method   Monthly.Charges     Total.Charges
##                 0                 0                 0                11
##       Churn.Label       Churn.Value       Churn.Score              CLTV
##                 0                 0                 0                 0
##      Churn.Reason
##                 0
```

```
#can see that total charges has 11 observations missing
```

```
# shows the rows in which total charges is NA
missing_val<- initial_data[is.na(initial_data$Total.Charges),]
missing_val
```

	Customer… <chr>	Co… <int>	Country <chr>	State <chr>	City <chr>	Zip.Code <int>	Lat.Long <chr>
2235	4472-LVYGI	1	United States	California	San Bernardino	92408	34.084909, -117.25
2439	3115-CZMZD	1	United States	California	Independence	93526	36.869584, -118.18
2569	5709-LVOEQ	1	United States	California	San Mateo	94401	37.590421, -122.30
2668	4367-NUYAO	1	United States	California	Cupertino	95014	37.306612, -122.08
2857	1371-DWPAZ	1	United States	California	Redcrest	95569	40.363446, -123.83
4332	7644-OMVMY	1	United States	California	Los Angeles	90029	34.089953, -118.29

	Customer… <chr>	Co… <int>	Country <chr>	State <chr>	City <chr>	Zip.Code <int>	Lat.Long <chr>
4688	3213-VVOLG	1	United States	California	Sun City	92585	33.739412, -117.17
5105	2520-SGTTA	1	United States	California	Ben Lomond	95005	37.078873, -122.09
5720	2923-ARZLG	1	United States	California	La Verne	91750	34.144703, -117.77
6773	4075-WKNIU	1	United States	California	Bell	90201	33.970343, -118.17

1-10 of 11 rows | 1-8 of 34 columns Previous **1** 2 Next

```
# these missing values are when churn = 0 so they are staying with the company
```

```
# dropping the NA values, would not contribute much in our findings as we have about 73%
of the data that is not churn (churn = 0)
data<- na.omit(initial_data)
colSums(is.na(data))
```

```
##        CustomerID             Count           Country             State
##                 0                 0                 0                 0
##              City          Zip.Code         Lat.Long          Latitude
##                 0                 0                 0                 0
##         Longitude            Gender    Senior.Citizen           Partner
##                 0                 0                 0                 0
##        Dependents     Tenure.Months     Phone.Service    Multiple.Lines
##                 0                 0                 0                 0
##  Internet.Service   Online.Security     Online.Backup Device.Protection
##                 0                 0                 0                 0
##      Tech.Support      Streaming.TV  Streaming.Movies          Contract
##                 0                 0                 0                 0
## Paperless.Billing    Payment.Method   Monthly.Charges     Total.Charges
##                 0                 0                 0                 0
##       Churn.Label       Churn.Value       Churn.Score              CLTV
##                 0                 0                 0                 0
##      Churn.Reason
##                 0
```

```
# removing columns that are not providing useful information
data<- data %>% dplyr::select(-c(State, Country))
```

```
# count the target variable
data %>%
  group_by(Churn.Label) %>%
  summarise(count = n(),
            percentage = paste0(round(count/nrow(initial_data)*100,0),"%"),
            .groups="drop")
```

Churn.Label	count	percentage
<chr>	<int>	<chr>
No	5163	73%
Yes	1869	27%

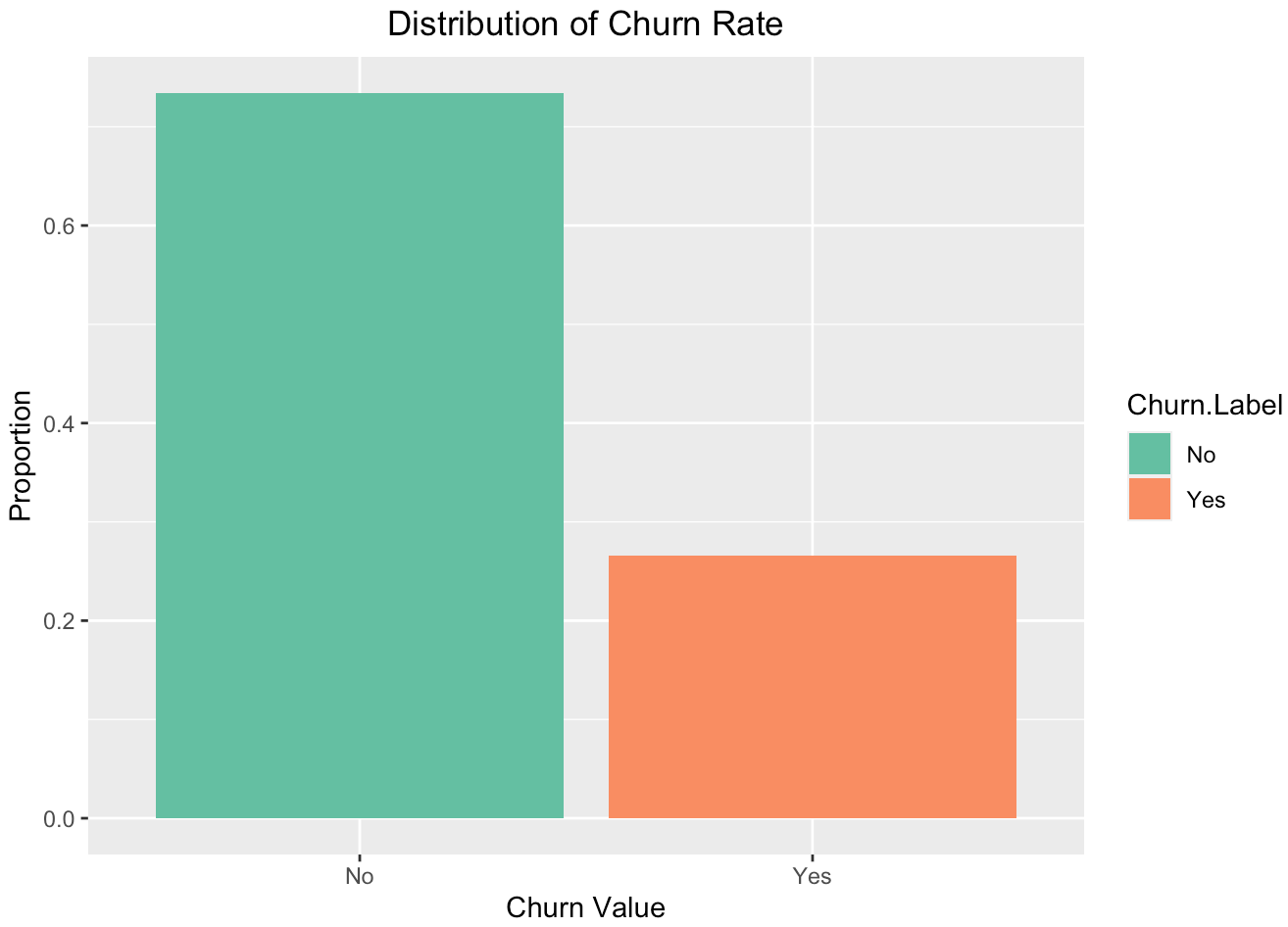
2 rows

The target variable found in the “Churn Label” column indicates whether or not a customer stays or leaves the company. On reviewing the data, we noticed that the dataset is imbalanced with about 73% staying with the company compared to 27% who leave. We will have to take this into account when doing our analysis.

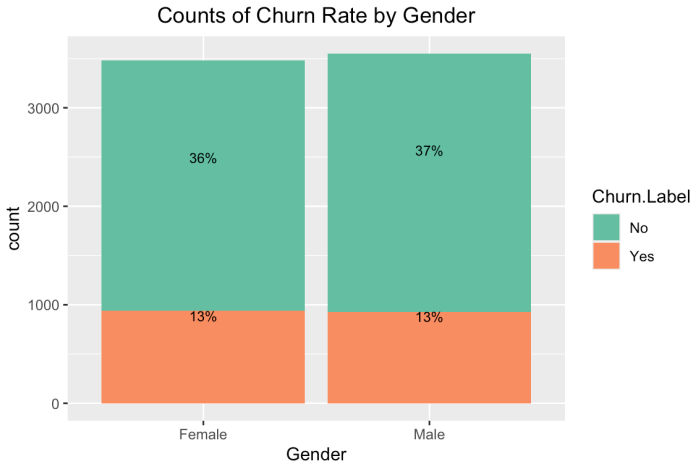
3. Exploratory Data Analysis

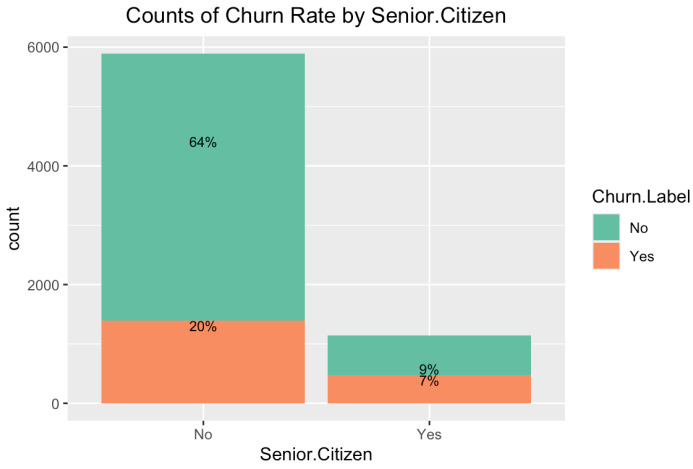
There are several different data fields that are available. We have done some preliminary exploratory data analysis, to see how these fields relate to the target, and whether there are any variable correlations. For all customers we have information on characteristics such as gender, whether they have a partner or any dependents, and how long they have been with the company. We also have information about the different types of services that the company provides such as phone, internet, streaming or a combination of these.

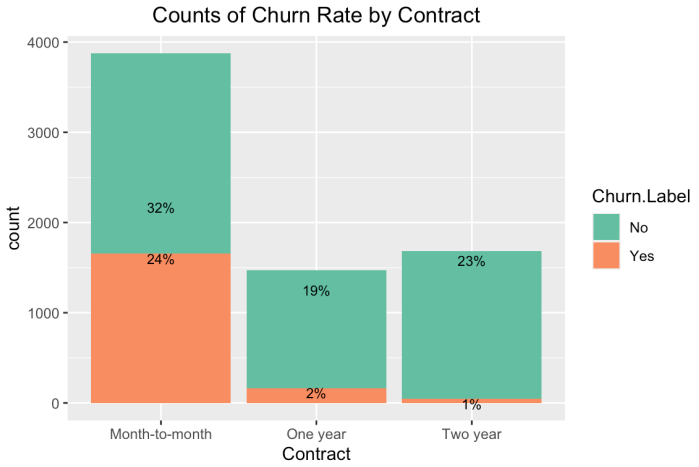
```
# distribution of target variable
data %>%
  ggplot(aes(x = Churn.Label, fill = Churn.Label))+
  geom_bar(aes(y = after_stat(count)/sum(after_stat(count))))+
  ylab("Proportion")+
  xlab("Churn Value")+
  ggtitle("Distribution of Churn Rate")+
  theme(plot.title = element_text(hjust = 0.5))+ # centers the title
  scale_fill_brewer(palette = "Set2")
```

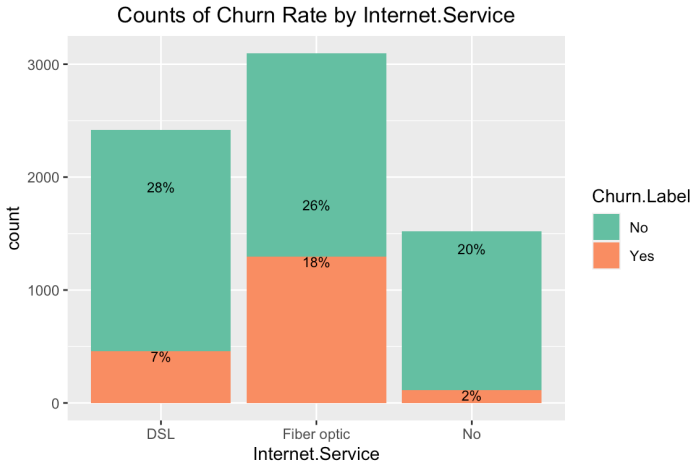


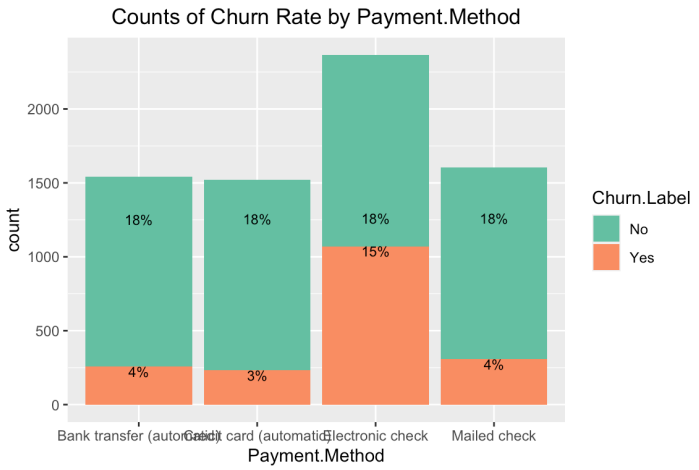

```
categFields <- c("Gender", "Senior.Citizen", "Contract", "Internet.Service", "Payment.Me
thod")

p<- list()
for (i in 1:length(categFields)){
  tab <- data %>%
  group_by(!!sym(categFields[[i]]), Churn.Label) %>%
  summarise(count = n(),
            .groups = "drop")  %>%
  mutate(prop = paste0(round(count/sum(count)*100, 0), '%'))
  p[[i]] <- ggplot(data=tab, aes(x = !!sym(categFields[[i]]), y = count, fill = Churn.La
bel)) +
  geom_col(position = "stack")  +
  geom_text(aes(label = prop), vjust = 1, size = 3) +
  ggtitle(paste0("Counts of Churn Rate by ", categFields[i]))+
  theme(plot.title = element_text(hjust = 0.5))+ # centers the title
  scale_fill_brewer(palette = "Set2")
}

library(gridExtra)

do.call(grid.arrange, p)
```







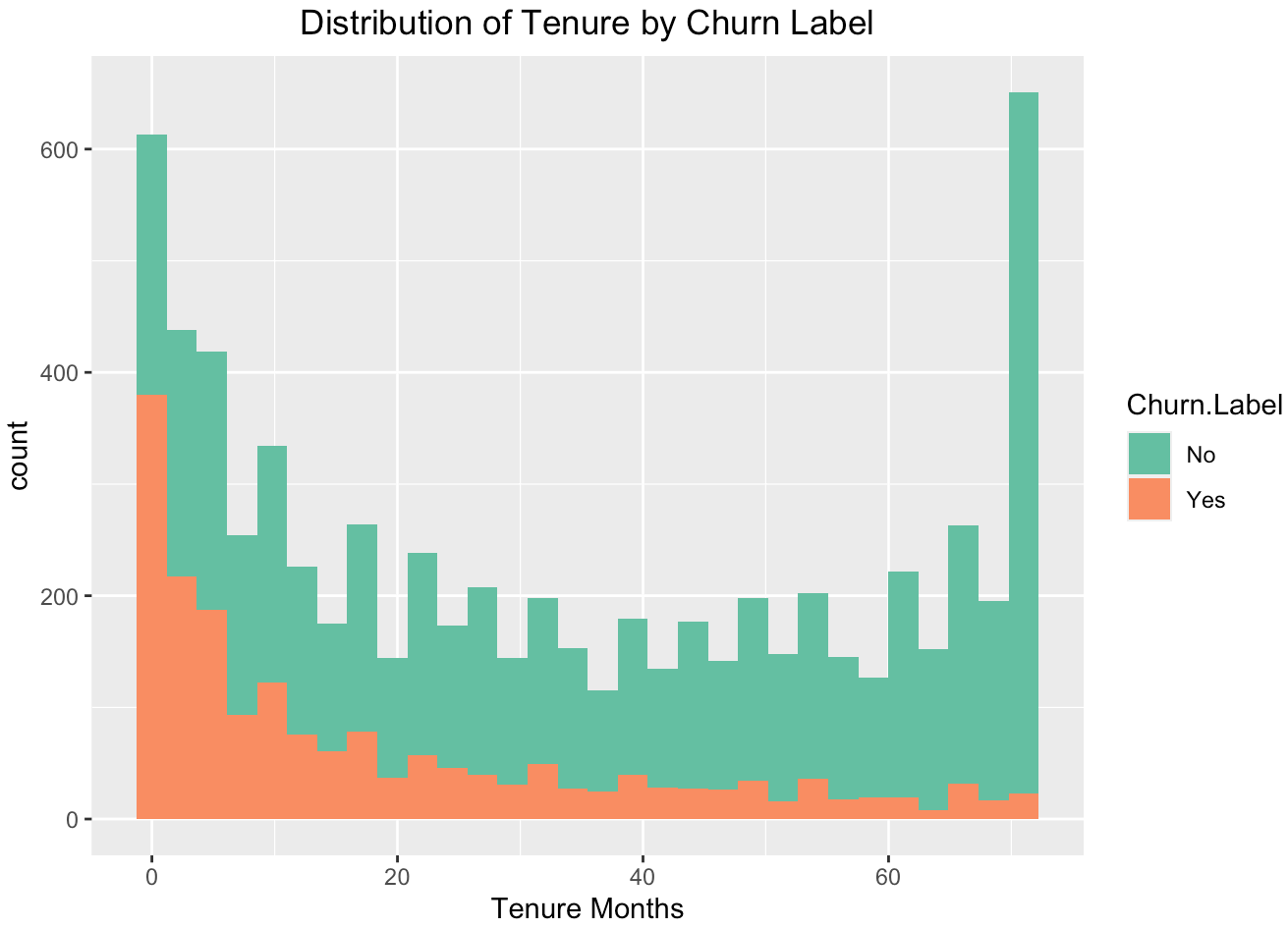




The graph on the top left shows there are similar numbers of customers by gender, with each having similar proportions that leave versus remaining with the company. The graph on the top right shows that majority of customers are not senior citizens, but those that are, have a higher likelihood to leave. The next distribution shows that there are more people leaving the company with a contract that is month-to month versus those that are fixed for one or two years. This makes sense because customers with longer term contracts might face penalties if they terminate early. The fourth graph shows that most customers have Fibre Optic internet services, but this category also has the largest proportion of customers that leave the company. The last graph shows the majority of customers choose to pay by electronic cheque. However, this also leads to higher likelihood of customer loss.

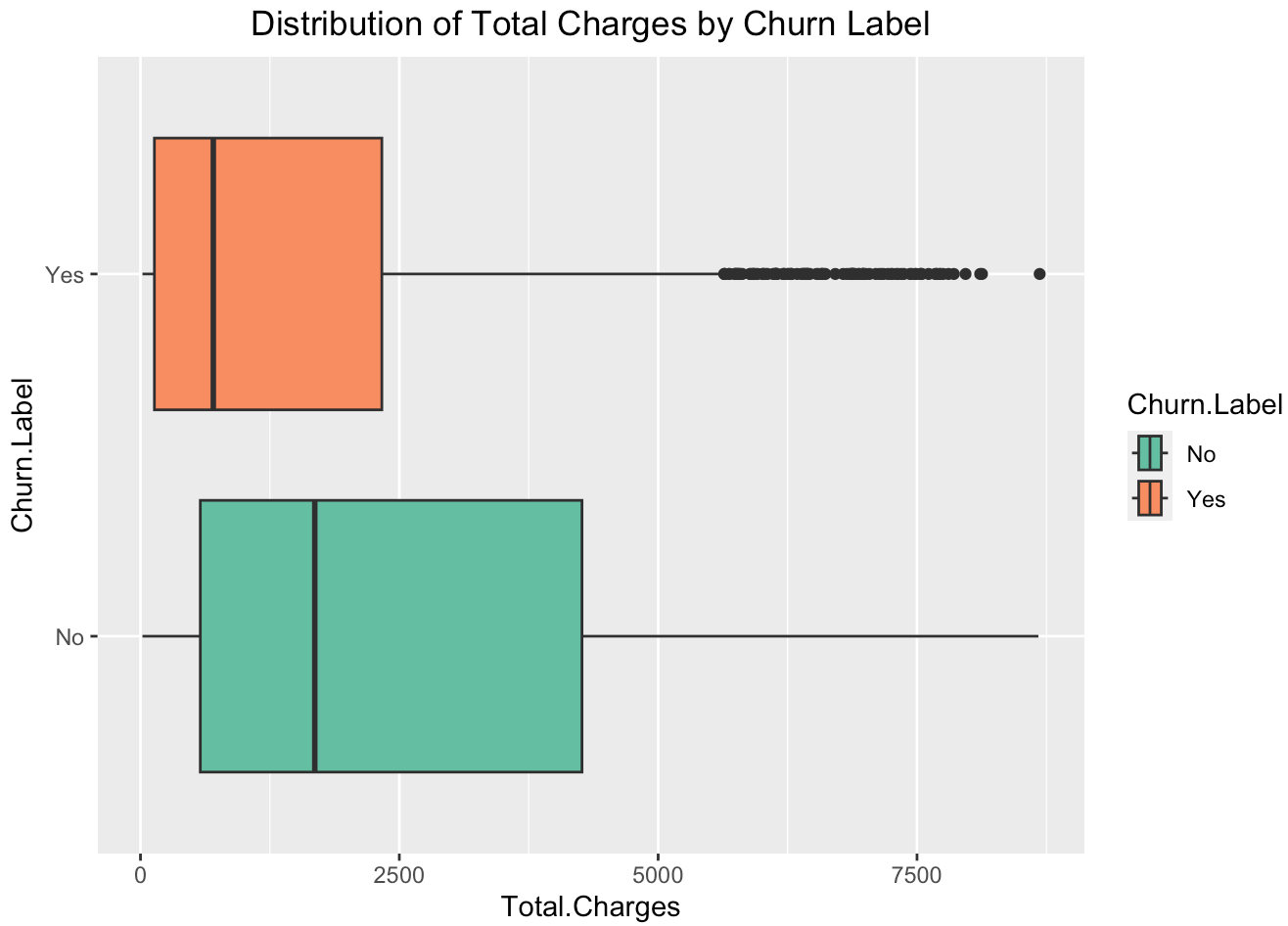
Customers who have been with the company for a shorter period of time are more likely to leave, with a median of 10 months.

```
# plots for continuous variables
data %>%
  ggplot(aes(x = Tenure.Months, fill = Churn.Label))+
  geom_histogram(bins = 30)+
  ggtitle("Distribution of Tenure by Churn Label")+
  theme(plot.title = element_text(hjust = 0.5))+ # centers the title
  scale_fill_brewer(palette = "Set2")+
  xlab("Tenure Months")
```



Although there are more outliers, it appears that customers who left the company had a lower median total charge than those who remained with the company. This may indicate that cost alone may not be the motivating factor that causes customers to leave.

```
data %>%
  ggplot(aes(x = Total.Charges, y = Churn.Label, fill = Churn.Label))+
  geom_boxplot()+
  ggtitle("Distribution of Total Charges by Churn Label")+
  theme(plot.title = element_text(hjust = 0.5))+ # centers the title
  scale_fill_brewer(palette = "Set2")
```



For customers who have left, the data also includes a reason for leaving. While this cannot be used in prediction this can be helpful in determining what incentives or mitigating actions could be taken by the company to promote customer retention. About 10% of the customers who left, did so due to price or charges.

```
# reason for leaving
data %>%
  filter(Churn.Value == 1) %>%
  group_by(Churn.Reason) %>%
  summarise(count = n(),
            .groups = 'drop') %>%
  mutate(perc =paste0(round(count / sum(count)*100, 0),'%')) %>%
  arrange(desc(count))
```

Churn.Reason <chr>	**count** <int>	**perc** <chr>
Attitude of support person	192	10%
Competitor offered higher download speeds	189	10%
Competitor offered more data	162	9%
Don't know	154	8%
Competitor made better offer	140	7%
Attitude of service provider	135	7%

Churn.Reason <chr>	count <int>	perc <chr>
Competitor had better devices	130	7%
Network reliability	103	6%
Product dissatisfaction	102	5%
Price too high	98	5%

1-10 of 20 rows Previous **1** 2 Next

Feature Engineering

The zipCode field contains a high number of unique values, which can be problematic for a categorical variable. To reduce dimensionality of this field, we grouped the values into a new feature, “Region”.

```
#Create a location variable from first 3 digits of zip code
#California can be divided into North and South
#Zip codes < 935** considered South, else considered North

data <- data %>%
  mutate(Region = if_else(Zip.Code <= 93500, "South", "North"))
```

After doing some exploratory data analysis we realized some columns were not providing us additional information. In addition, we converted the categorical variables to factors and assigned the appropriate levels. Finally, we scaled the continuous variables which will help with performance of the predictive models.

```
# need to drop specific columns
# (do we need both churn value and churn label), lat long (we have the latitude and long
itude separated)
data<- data %>%
  dplyr::select(c(Gender: Total.Charges, Region, Churn.Label))
head(data)
```

	Gen… <chr>	Senior.Citizen <chr>	Partner <chr>	Depende… <chr>	Tenure.Months <int>	Phone.Service <chr>	Multiple.Lines <chr>
1	Male	No	No	No	2	Yes	No
2	Female	No	No	Yes	2	Yes	No
3	Female	No	No	Yes	8	Yes	Yes
4	Female	No	Yes	Yes	28	Yes	Yes
5	Male	No	No	Yes	49	Yes	Yes
6	Female	No	Yes	No	10	Yes	No

6 rows | 1-8 of 22 columns

```
# data that won't be available for why they left (can't use for prediction)
```

```
# rename the columns
names(data) <- c("Gender", "Senior",
                    "Partner", "Dependents",
                    "Tenure","PhoneSvc",
                    "MultLines","InternetSvc",
                    "OLSecurity","OLBackup",
                    "DeviceProt","TechSupport",
                    "StreamTV","StreamMovies",
                    "Contract", "Paperless",
                    "PayMethod", "MonthlyCharges",
                    "TotalCharges","Region", "ChurnLabel")

head(data)
```

	Gen…	Senior	Partner	Dependents	Tenure	Phone…	MultLines	InternetSvc	OLSecurity
	<chr>	<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>
1	Male	No	No	No	2	Yes	No	DSL	Yes
2	Female	No	No	Yes	2	Yes	No	Fiber optic	No
3	Female	No	No	Yes	8	Yes	Yes	Fiber optic	No
4	Female	No	Yes	Yes	28	Yes	Yes	Fiber optic	No
5	Male	No	No	Yes	49	Yes	Yes	Fiber optic	No
6	Female	No	Yes	No	10	Yes	No	DSL	No

6 rows | 1-10 of 22 columns

```
#Convert character fields to categorical variables
data <- data %>%
  mutate(MultLines = if_else(PhoneSvc=="No", "No", MultLines),
         OLSecurity = if_else(InternetSvc=="No", "No", OLSecurity),
         OLBackup = if_else(InternetSvc=="No", "No", OLBackup),
         DeviceProt = if_else(InternetSvc=="No", "No", DeviceProt),
         TechSupport = if_else(InternetSvc=="No", "No", TechSupport),
         StreamTV = if_else(InternetSvc=="No", "No", StreamTV),
         StreamMovies = if_else(InternetSvc=="No", "No", StreamMovies))

#Convert character fields to categorical variables
#Set reference level for target variable
data <- data %>%
  mutate_if(~is.character(.),as.factor) %>%
  mutate(ChurnLabel = relevel(ChurnLabel, ref = "No"))
# looking at which variable is our indicator (1: customer is leaving)
contrasts(data$ChurnLabel)
```

```
##     Yes
## No    0
## Yes   1
```

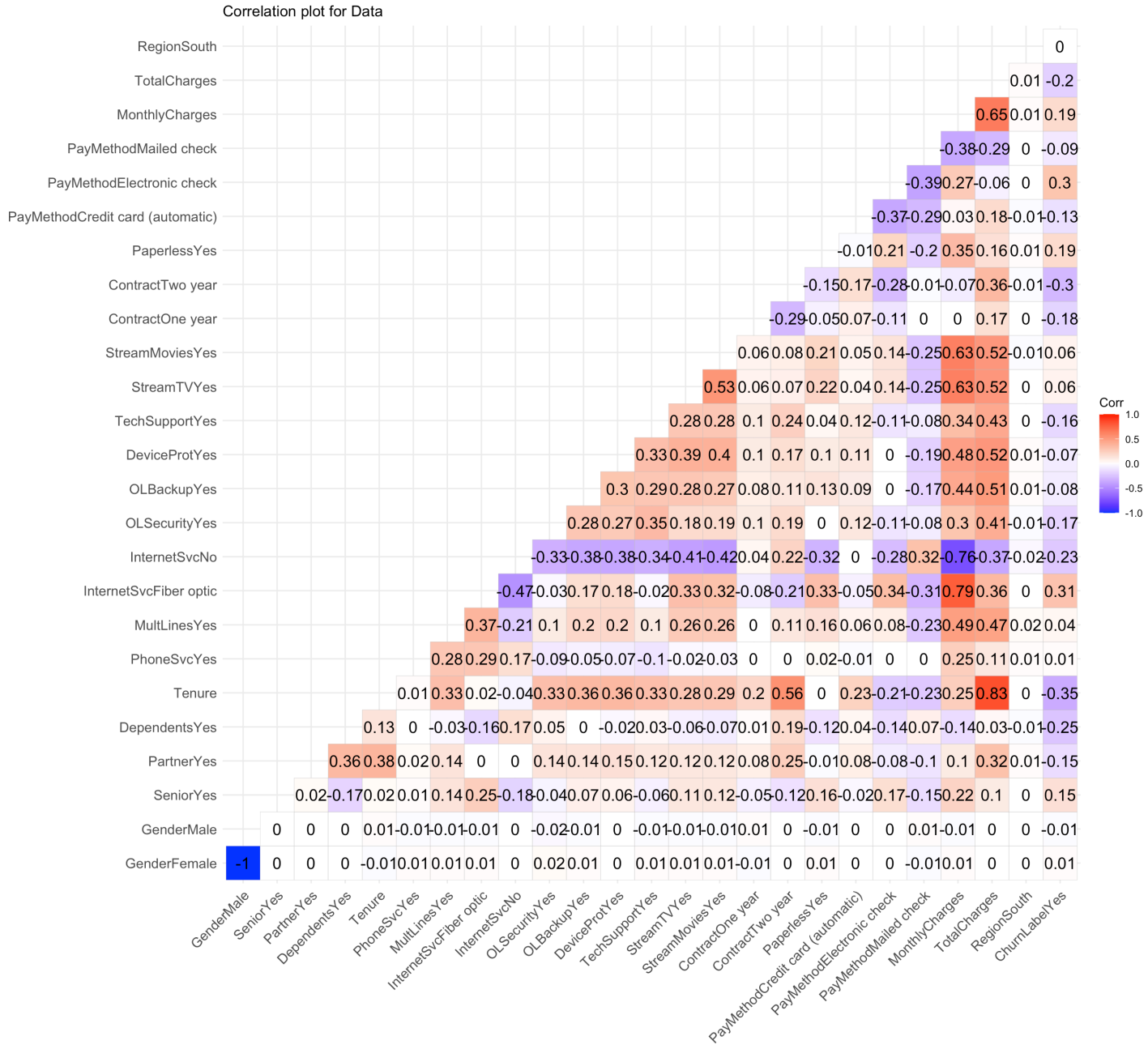
```
str(data)
```

```
## 'data.frame':    7032 obs. of  21 variables:
##  $ Gender        : Factor w/ 2 levels "Female","Male": 2 1 1 1 2 1 2 2 2 2 ...
##  $ Senior        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
##  $ Partner       : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...
##  $ Dependents    : Factor w/ 2 levels "No","Yes": 1 2 2 2 2 1 1 1 2 1 ...
##  $ Tenure        : int  2 2 8 28 49 10 1 1 47 1 ...
##  $ PhoneSvc      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 1 2 2 1 ...
##  $ MultLines     : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 1 1 1 2 1 ...
##  $ InternetSvc   : Factor w/ 3 levels "DSL","Fiber optic",..: 1 2 2 2 2 1 1 3 2 1 ...
##  $ OLSecurity    : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ OLBackup      : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 1 1 1 2 2 ...
##  $ DeviceProt    : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 2 1 1 1 ...
##  $ TechSupport   : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 1 1 ...
##  $ StreamTV      : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 1 1 1 2 1 ...
##  $ StreamMovies  : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 1 2 1 2 1 ...
##  $ Contract      : Factor w/ 3 levels "Month-to-month",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Paperless     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 1 2 1 ...
##  $ PayMethod     : Factor w/ 4 levels "Bank transfer (automatic)",..: 4 3 3 3 1 2 3 4
3 3 ...
##  $ MonthlyCharges: num  53.9 70.7 99.7 104.8 103.7 ...
##  $ TotalCharges  : num  108 152 820 3046 5036 ...
##  $ Region        : Factor w/ 2 levels "North","South": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ChurnLabel    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:11] 2235 2439 2569 2668 2857 4332 4688 5
105 5720 6773 ...
##   ..- attr(*, "names")= chr [1:11] "2235" "2439" "2569" "2668" ...
```

```
#Scale continuous vars
data_S <- data %>%
  mutate_if(~is.numeric(.),rescale)
```

To test for collinearity among variables, we produced a heatmap. Based on this we noted some relationships among variables, but none that were highly correlated.

```
#heatmap
model.matrix(~0+.,data_S) %>%
  cor(use="pairwise.complete.obs") %>%
  ggcorrplot(show.diag=FALSE, type="lower",lab=TRUE, lab_size=5)+
  ggtitle("Correlation plot for Data")
```



4. Split the data into train, test and validation sets

We need to split our data into training, test, and validation sets. Using and 80/10/10 ratio of available data with 80% of the being used for the training data and the remaining being used for the test and validation data. The training data will be what we use to develop and build the model, while the test data will be used for evaluating the model and doing any hyper tuning (to choose the best parameters), and the validation data will evaluate and compare the final models.

One consideration we must make is the fact that the data we are working is skewed about 70/30 to customer staying versus leaving as mentioned previously. We will need to modify how we build the models to make sure they still accurately reflect the general environment. If we do not do this, we run the risk of either training data

being skewed, leading to a highly inaccurate model, or the test set being skewed leading to inaccurate conclusions we can draw from our models.

```
#creating train and test splits
set.seed(0)
splitSample <- sample(1:3, size=nrow(data_S), prob=c(0.8,0.1,0.1), replace=TRUE)

train_data <- data_S[splitSample==1, ]
test_data <- data_S[splitSample==2, ]
val_data <- data_S[splitSample==3, ]

#check dimensions of train and test
dim(train_data)
```

```
## [1] 5557   21
```

```
dim(test_data)
```

```
## [1] 753  21
```

```
dim(val_data)
```

```
## [1] 722  21
```

```
#check proportion + incidence of target in each set
table(train_data$ChurnLabel)
```

```
##
##   No  Yes
## 4070 1487
```

```
table(test_data$ChurnLabel)
```

```
##
##  No Yes
## 554 199
```

```
table(val_data$ChurnLabel)
```

```
##
##  No Yes
## 539 183
```

As our data was imbalanced in favour, we used an oversampling technique to help balance the dataset. This was applied to the training data only.

```
# Oversample on the train only
oversampleTrain<- ovun.sample(ChurnLabel ~ ., data = train_data, method = "over", seed =
0)$data

#checking distribution of target variable
table(oversampleTrain$ChurnLabel)
```

```
##
##   No  Yes
## 4070 4027
```

```
table(train_data$ChurnLabel)
```

```
##
##   No  Yes
## 4070 1487
```

5. Build the models

Due to the nature of Churn being one of two options either a customer stays or leaves we decided it fits best to use a classification models. These models will be used to predict the likelihood that a customer will leave (class 1) or staying (class 0).

Confusion Matrix and Other Metrics

Due to of our models being classification models we plan to use the confusion matrix as our main source of evaluation. The columns being the actual values of what we are trying to predict and the rows being our predictions. This gives us four values: True Positive - TP (We predict the customer leaves, and they do), False Positive - FP (We predict they leave, and they do not), True Negative - TN (We predict they stay, and they do), and finally False Negative - FN (We predict they stay, but they leave). With these four values we gain a lot of information about our different models, and they offer us many statistics we can use, most importantly the model's accuracy. Another benefit of the confusion matrix is it offers us the ability to better fine tune our models to better suit the company's needs.

```
#Functions to be used in model evaluation

#Get class predictions based on specified threshold
getClass <- function(model, tool, data, cutoff=0.5){
  if (tool =="lr"){
    preds <- predict(model, data, type = "response")
    preds <-ifelse(preds > cutoff,"Yes","No")
    preds <- as.factor(preds)
  }
  else {
    preds <- data.frame(predict(model, newdata = data, type="prob"))$Yes
    preds <-ifelse(preds > cutoff,"Yes","No")
    preds <- as.factor(preds)
  }
}

#Get probability predictions
getProbs <- function(model, tool, data){
  if (tool =="lr"){
    preds <- predict(model, data, type = "response")
  }
  else {
    preds <- data.frame(predict(model, newdata = data, type="prob"))$Yes
  }
}

# gets the error values
getError <- function(preds, target){
  error = mean(target != preds)
  return(error)
}

# prints out the confusion matrix
getCM <- function(preds, data){
  cm <- confusionMatrix(preds, data, positive="Yes")
  print(cm)
}

 # prints specific metrics we want displayed
getMetrics <- function(preds, data){
  cm <- confusionMatrix(preds, data, positive="Yes")
  accuracy <- cm$overall[1]
  precision <- cm$byClass[5]
  sensitivity <- cm$byClass[1]
  specificity <- cm$byClass[2]
  f1Score <- 2*(precision*sensitivity)/(precision+sensitivity)
  results <- tibble("Accuracy"=accuracy,
                "Precision"=precision,
                "Sensitivity"=sensitivity,
                "Specificity"=specificity,
                "F1Score"=f1Score)
  return(results)
```

```
}
```

```
#Used in setting parameters for caret model building
trainControl <- trainControl(method = "repeatedcv",
                             number = 5,
                             repeats = 3,
                             summaryFunction = twoClassSummary,
                             classProbs = T)
```

Logistic Regression

Logistic Regression was chosen due to its high interpretability and simplicity. Using the predictors, the model will produce a probability between 0 and 1 and from there we can determine if the customer is closer to leaving (1) or staying (0). It is a model that we all have experience with and feel comfortable with using and explaining the results to management and other stakeholders of the telecom company. The model is easy to improve on through regularization techniques such as Lasso and Ridge as well as StepAIC.

```
#build Logistic Regression
modelLR <- glm(ChurnLabel ~. ,data = oversampleTrain, family = "binomial")
summary(modelLR)
```

```
##
## Call:
## glm(formula = ChurnLabel ~ ., family = "binomial", data = oversampleTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.29937  -0.69396  -0.06076   0.71112   3.00401
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       1.442632   0.246683   5.848 4.97e-09 ***
## GenderMale                       -0.028642   0.057560  -0.498  0.61876
## SeniorYes                         0.223673   0.075428   2.965  0.00302 **
## PartnerYes                        0.268179   0.066023   4.062 4.87e-05 ***
## DependentsYes                    -1.574332   0.093908 -16.765  < 2e-16 ***
## Tenure                           -4.221649   0.344531 -12.253  < 2e-16 ***
## PhoneSvcYes                       0.138846   0.573871   0.242  0.80882
## MultLinesYes                      0.375760   0.158267   2.374  0.01759 *
## InternetSvcFiber optic            1.282489   0.703531   1.823  0.06831 .
## InternetSvcNo                    -1.543561   0.712221  -2.167  0.03022 *
## OLSecurityYes                    -0.221392   0.158000  -1.401  0.16115
## OLBackupYes                       0.053303   0.157191   0.339  0.73454
## DeviceProtYes                    -0.003793   0.156500  -0.024  0.98066
## TechSupportYes                   -0.228881   0.158514  -1.444  0.14876
## StreamTVYes                       0.507009   0.288950   1.755  0.07932 .
## StreamMoviesYes                   0.486428   0.287746   1.690  0.09094 .
## ContractOne year                 -0.754888   0.088631  -8.517  < 2e-16 ***
## ContractTwo year                 -1.353970   0.135873  -9.965  < 2e-16 ***
## PaperlessYes                      0.231934   0.065006   3.568  0.00036 ***
## PayMethodCredit card (automatic) -0.130855   0.097359  -1.344  0.17893
## PayMethodElectronic check         0.366149   0.082408   4.443 8.87e-06 ***
## PayMethodMailed check            -0.007983   0.097953  -0.081  0.93505
## MonthlyCharges                   -3.118212   2.819492  -1.106  0.26875
## TotalCharges                      2.840947   0.485421   5.853 4.84e-09 ***
## RegionSouth                      -0.019824   0.057437  -0.345  0.72999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11224.6  on 8096  degrees of freedom
## Residual deviance:  7491.7  on 8072  degrees of freedom
## AIC: 7541.7
##
## Number of Fisher Scoring iterations: 6
```

Metrics for Logistic Regression

```
# Get train and test errors for logistic regression
sprintf("Training Error for Logistic Regression: %f",getError(getClass(modelLR, "lr", ov
ersampleTrain,0.5), oversampleTrain$ChurnLabel))
```

```
## [1] "Training Error for Logistic Regression: 0.222922"
```

```
sprintf("Testing Error for Logistic Regression: %f",getError(getClass(modelLR, "lr", tes
t_data,0.5), test_data$ChurnLabel))
```

```
## [1] "Testing Error for Logistic Regression: 0.243028"
```

Training error and testing error are similar, indicating that the model is likely not overfitting. Using Logistic Regression we are able to predict customer churn at a 76% accuracy and a sensitivity of about 80%. However, there are some features that appear to be not significant as predictors, so we could refine the logistic regression using regularization techniques.

```
# Get metrics for logistic regression based on test data
getCM(getClass(modelLR, "lr", test_data, 0.5), test_data$ChurnLabel)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  411  40
##        Yes 143 159
##
##                Accuracy : 0.757
##                  95% CI : (0.7247, 0.7872)
##     No Information Rate : 0.7357
##     P-Value [Acc > NIR] : 0.09932
##
##                   Kappa : 0.4639
##
##  Mcnemar's Test P-Value : 4.698e-14
##
##             Sensitivity : 0.7990
##             Specificity : 0.7419
##          Pos Pred Value : 0.5265
##          Neg Pred Value : 0.9113
##              Prevalence : 0.2643
##          Detection Rate : 0.2112
##    Detection Prevalence : 0.4011
##       Balanced Accuracy : 0.7704
##
##        'Positive' Class : Yes
##
```

```
print(getMetrics(getClass(modelLR, "lr", test_data, 0.5), test_data$ChurnLabel))
```

```
## # A tibble: 1 × 5
##   Accuracy Precision Sensitivity Specificity F1Score
##      <dbl>     <dbl>       <dbl>       <dbl>   <dbl>
## 1    0.757     0.526       0.799       0.742   0.635
```

Hypertuning the Logistic Regression

Using cross validation for Lasso and Ridge Regression

```
# trying to tune our model to see if we can improve performance
# alpha = 0 lasso ; alpha = 1 ridge
set.seed(0)
tuneGridReg =expand.grid(alpha = c(0,1),
                         lambda=seq(0.1,3, by = 0.1))

modelReg <- train(ChurnLabel ~ ., data = oversampleTrain,
                  method = "glmnet",
                  trControl = trainControl,
                  tuneGrid = tuneGridReg,
                  metric = "ROC",
                  family="binomial",
                  savePredictions = "all")
modelReg
```

```
## glmnet
##
## 8097 samples
##   20 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 6477, 6477, 6478, 6478, 6478, 6477, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda  ROC        Sens       Spec
##   0      0.1     0.8537189  0.7470925  0.7900030
##   0      0.2     0.8520685  0.7450450  0.7885139
##   0      0.3     0.8510432  0.7447174  0.7885145
##   0      0.4     0.8503560  0.7447174  0.7869421
##   0      0.5     0.8498508  0.7456183  0.7867767
##   0      0.6     0.8494469  0.7457002  0.7857833
##   0      0.7     0.8491184  0.7458640  0.7847901
##   0      0.8     0.8488627  0.7455364  0.7849552
##   0      0.9     0.8486476  0.7460278  0.7842932
##   0      1.0     0.8484524  0.7460278  0.7840446
##   0      1.1     0.8483027  0.7454545  0.7832999
##   0      1.2     0.8481531  0.7454545  0.7829687
##   0      1.3     0.8480297  0.7455364  0.7827204
##   0      1.4     0.8479259  0.7457002  0.7823891
##   0      1.5     0.8478261  0.7457002  0.7820580
##   0      1.6     0.8477527  0.7457821  0.7821408
##   0      1.7     0.8476747  0.7458640  0.7818924
##   0      1.8     0.8476084  0.7459459  0.7818924
##   0      1.9     0.8475497  0.7462735  0.7818096
##   0      2.0     0.8474907  0.7464373  0.7817268
##   0      2.1     0.8474361  0.7466011  0.7816440
##   0      2.2     0.8473801  0.7468468  0.7814786
##   0      2.3     0.8473374  0.7470925  0.7813131
##   0      2.4     0.8472937  0.7473382  0.7813131
##   0      2.5     0.8472587  0.7476658  0.7811477
##   0      2.6     0.8472158  0.7483210  0.7811477
##   0      2.7     0.8471857  0.7488124  0.7811477
##   0      2.8     0.8471573  0.7493038  0.7811477
##   0      2.9     0.8471232  0.7495495  0.7808166
##   0      3.0     0.8470930  0.7497133  0.7808166
##   1      0.1     0.8353936  0.6996724  0.8058971
##   1      0.2     0.7456266  0.7211302  0.6254329
##   1      0.3     0.5000000  1.0000000  0.0000000
##   1      0.4     0.5000000  1.0000000  0.0000000
##   1      0.5     0.5000000  1.0000000  0.0000000
##   1      0.6     0.5000000  1.0000000  0.0000000
##   1      0.7     0.5000000  1.0000000  0.0000000
##   1      0.8     0.5000000  1.0000000  0.0000000
##   1      0.9     0.5000000  1.0000000  0.0000000
##   1      1.0     0.5000000  1.0000000  0.0000000
```

```
##   1      1.1     0.5000000  1.0000000  0.0000000
##   1      1.2     0.5000000  1.0000000  0.0000000
##   1      1.3     0.5000000  1.0000000  0.0000000
##   1      1.4     0.5000000  1.0000000  0.0000000
##   1      1.5     0.5000000  1.0000000  0.0000000
##   1      1.6     0.5000000  1.0000000  0.0000000
##   1      1.7     0.5000000  1.0000000  0.0000000
##   1      1.8     0.5000000  1.0000000  0.0000000
##   1      1.9     0.5000000  1.0000000  0.0000000
##   1      2.0     0.5000000  1.0000000  0.0000000
##   1      2.1     0.5000000  1.0000000  0.0000000
##   1      2.2     0.5000000  1.0000000  0.0000000
##   1      2.3     0.5000000  1.0000000  0.0000000
##   1      2.4     0.5000000  1.0000000  0.0000000
##   1      2.5     0.5000000  1.0000000  0.0000000
##   1      2.6     0.5000000  1.0000000  0.0000000
##   1      2.7     0.5000000  1.0000000  0.0000000
##   1      2.8     0.5000000  1.0000000  0.0000000
##   1      2.9     0.5000000  1.0000000  0.0000000
##   1      3.0     0.5000000  1.0000000  0.0000000
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 0.1.
```

```
#get Metrics for regularized logistic regression using test data
getMetrics(getClass(modelReg, "reg", test_data, 0.5), test_data$ChurnLabel)
```

Accuracy	Precision	Sensitivity	Specificity	F1Score
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.7622842	0.5342466	0.7839196	0.7545126	0.6354379

1 row

The regularization indicates that there is slightly higher accuracy, but lower sensitivity. This means that the regularized model is not as capable of identifying the customers that will leave the company, so it is less useful than the full glm.

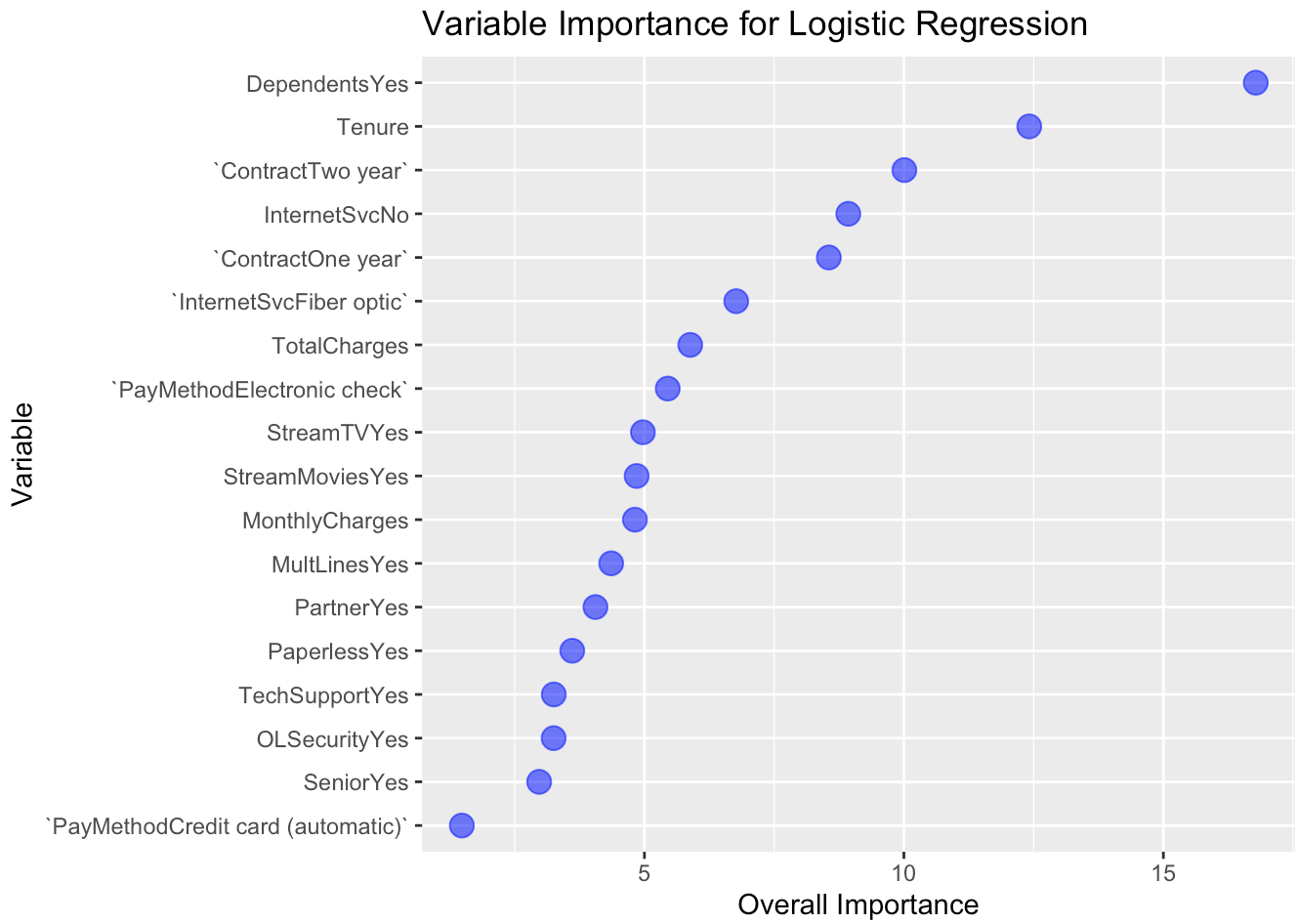
Looking at the output for the full model, there are some features which appear to be insignificant in the model. Removing some of these will help reduce complexity. We will use stepwise AIC to try reducing predictors in the logistic regression model.

```
# Reduce model complexity by applying stepwise logistic regression
set.seed(0)
modelRed <- train(ChurnLabel ~ ., data = oversampleTrain,
                  method = "glmStepAIC",
                  trControl = trainControl,
                  direction ="backward",
                  metric = "ROC",
                  family="binomial", trace=F)

modelRed$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##                         (Intercept)                           SeniorYes
##                              1.3745                              0.2239
##                          PartnerYes                       DependentsYes
##                              0.2671                             -1.5740
##                              Tenure                        MultLinesYes
##                             -4.2048                              0.3444
##           `InternetSvcFiber optic`                       InternetSvcNo
##                              1.1307                             -1.3789
##                      OLSecurityYes                      TechSupportYes
##                             -0.2510                             -0.2611
##                       StreamTVYes                    StreamMoviesYes
##                              0.4382                              0.4200
##                  `ContractOne year`                  `ContractTwo year`
##                             -0.7570                             -1.3576
##                      PaperlessYes  `PayMethodCredit card (automatic)`
##                              0.2342                             -0.1263
##        `PayMethodElectronic check`                      MonthlyCharges
##                              0.3698                             -2.4792
##                      TotalCharges
##                              2.8203
##
## Degrees of Freedom: 8096 Total (i.e. Null);  8078 Residual
## Null Deviance:       11220
## Residual Deviance: 7492  AIC: 7530
```

Important Variables for Logistic Regression

```
#plot the Variable Importance
ggplot(varImp(modelRed$finalModel), aes(x=reorder(rownames(varImp(modelRed$finalModel)),
Overall), y=Overall)) +
geom_point( color="blue", size=4, alpha=0.6)+
xlab('Variable')+
ylab('Overall Importance')+ggtitle("Variable Importance for Logistic Regression")+
coord_flip()
```



Final Logistic Regression Model

```
# using the reduced number of predictors from AIC
finalLR <- glm(ChurnLabel ~ Senior + Partner + Dependents + Tenure +
                 MultLines + InternetSvc + OLSecurity + TechSupport + StreamTV + StreamM
ovies +
                 Contract + Paperless + PayMethod + MonthlyCharges + TotalCharges,
               family = "binomial", data = oversampleTrain)

summary(finalLR)
```

```
##
## Call:
## glm(formula = ChurnLabel ~ Senior + Partner + Dependents + Tenure +
##     MultLines + InternetSvc + OLSecurity + TechSupport + StreamTV +
##     StreamMovies + Contract + Paperless + PayMethod + MonthlyCharges +
##     TotalCharges, family = "binomial", data = oversampleTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.29071  -0.69467  -0.06062   0.71338   2.99631
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      1.381019   0.171616   8.047 8.47e-16 ***
## SeniorYes                        0.223882   0.075356   2.971 0.002968 **
## PartnerYes                       0.266851   0.065916   4.048 5.16e-05 ***
## DependentsYes                   -1.573907   0.093815 -16.777  < 2e-16 ***
## Tenure                          -4.210023   0.343690 -12.249  < 2e-16 ***
## MultLinesYes                     0.344397   0.079003   4.359 1.30e-05 ***
## InternetSvcFiber optic           1.130099   0.167170   6.760 1.38e-11 ***
## InternetSvcNo                   -1.378235   0.154648  -8.912  < 2e-16 ***
## OLSecurityYes                   -0.250951   0.077189  -3.251 0.001149 **
## TechSupportYes                  -0.261063   0.080224  -3.254 0.001137 **
## StreamTVYes                      0.438041   0.088180   4.968 6.78e-07 ***
## StreamMoviesYes                  0.420135   0.086608   4.851 1.23e-06 ***
## ContractOne year                -0.756754   0.088526  -8.548  < 2e-16 ***
## ContractTwo year                -1.357731   0.135653 -10.009  < 2e-16 ***
## PaperlessYes                     0.233995   0.064908   3.605 0.000312 ***
## PayMethodCredit card (automatic) -0.130515   0.097303  -1.341 0.179816
## PayMethodElectronic check        0.365604   0.082331   4.441 8.97e-06 ***
## PayMethodMailed check           -0.008771   0.097929  -0.090 0.928631
## MonthlyCharges                  -2.481245   0.515156  -4.816 1.46e-06 ***
## TotalCharges                     2.825577   0.482993   5.850 4.91e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11224.6  on 8096  degrees of freedom
## Residual deviance:  7492.5  on 8077  degrees of freedom
## AIC: 7532.5
##
## Number of Fisher Scoring iterations: 6
```

```
#Eval of the model
print(paste0("Training Error for final logistic regression is ", getError(getClass(final
LR, "lr", oversampleTrain, 0.5), oversampleTrain$ChurnLabel)))
```

```
## [1] "Training Error for final logistic regression is 0.223045572434235"
```

```
print(paste0("Testing Error for final logistic regression is ", getError(getClass(finalL
R, "lr", test_data, 0.5), test_data$ChurnLabel)))
```

```
## [1] "Testing Error for final logistic regression is 0.243027888446215"
```

```
#get ROC with test data and optimal threshold for rf
roc_score_lr <- roc(test_data$ChurnLabel, getProbs(finalLR, "lr", test_data))
lr_cutoff<- coords(roc_score_lr, "best", ret="threshold")[[1]]

getCM(getClass(finalLR, "lr", test_data, lr_cutoff), test_data$ChurnLabel)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  439  48
##        Yes 115 151
##
##                Accuracy : 0.7835
##                  95% CI : (0.7524, 0.8125)
##     No Information Rate : 0.7357
##     P-Value [Acc > NIR] : 0.0014
##
##                   Kappa : 0.4975
##
##  Mcnemar's Test P-Value : 2.347e-07
##
##             Sensitivity : 0.7588
##             Specificity : 0.7924
##          Pos Pred Value : 0.5677
##          Neg Pred Value : 0.9014
##              Prevalence : 0.2643
##          Detection Rate : 0.2005
##    Detection Prevalence : 0.3533
##       Balanced Accuracy : 0.7756
##
##        'Positive' Class : Yes
##
```

```
print(getMetrics(getClass(finalLR, "lr", test_data, lr_cutoff), test_data$ChurnLabel))
```

```
## # A tibble: 1 × 5
##   Accuracy Precision Sensitivity Specificity F1Score
##      <dbl>     <dbl>       <dbl>       <dbl>   <dbl>
## 1    0.784     0.568       0.759       0.792   0.649
```

The logistic regression after using step-wise feature selection produces similar results to the full model. However, it is less complex due to fewer features used and is easier to communicate to management. Therefore we will use this model.

Random Forest

Random Forest is more complicated but produces accurate results especially on imbalanced datasets. The model will tell us if a customer decides to leave or stay based on aggregating predictions from multiple decision trees. The model also does an excellent job of reducing variance and can be scaled easily for future projects.

```
# build Random Forest
tunegrid_rf <- expand.grid(mtry = seq(3,9,by=2))
```

```
# build Random Forest
# TAKES A WHILE TO RUN
for(i in c(500, 1000)){
  set.seed=0
  modelRF <- train(ChurnLabel~. ,
                   data = oversampleTrain,
                   method = "rf",
                   trControl = trainControl,
                   ntrees = i,
                   metric = "ROC",
                   tuneGrid = tunegrid_rf)

  print(modelRF)
}
```

```
## Random Forest
##
## 8097 samples
##   20 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 6477, 6478, 6478, 6478, 6477, 6477, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   3     0.9471199  0.7980344  0.9446271
##   5     0.9683118  0.8302211  0.9600233
##   7     0.9716027  0.8375102  0.9600233
##   9     0.9719538  0.8378378  0.9582851
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 9.
## Random Forest
##
## 8097 samples
##   20 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 6478, 6478, 6477, 6478, 6477, 6477, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   3     0.9475752  0.7968059  0.9458651
##   5     0.9685579  0.8290745  0.9596884
##   7     0.9720078  0.8392301  0.9602678
##   9     0.9728852  0.8378378  0.9596882
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 9.
```

The highest ROC occurs when mtry=9, so we will fit a random forest model using these values.

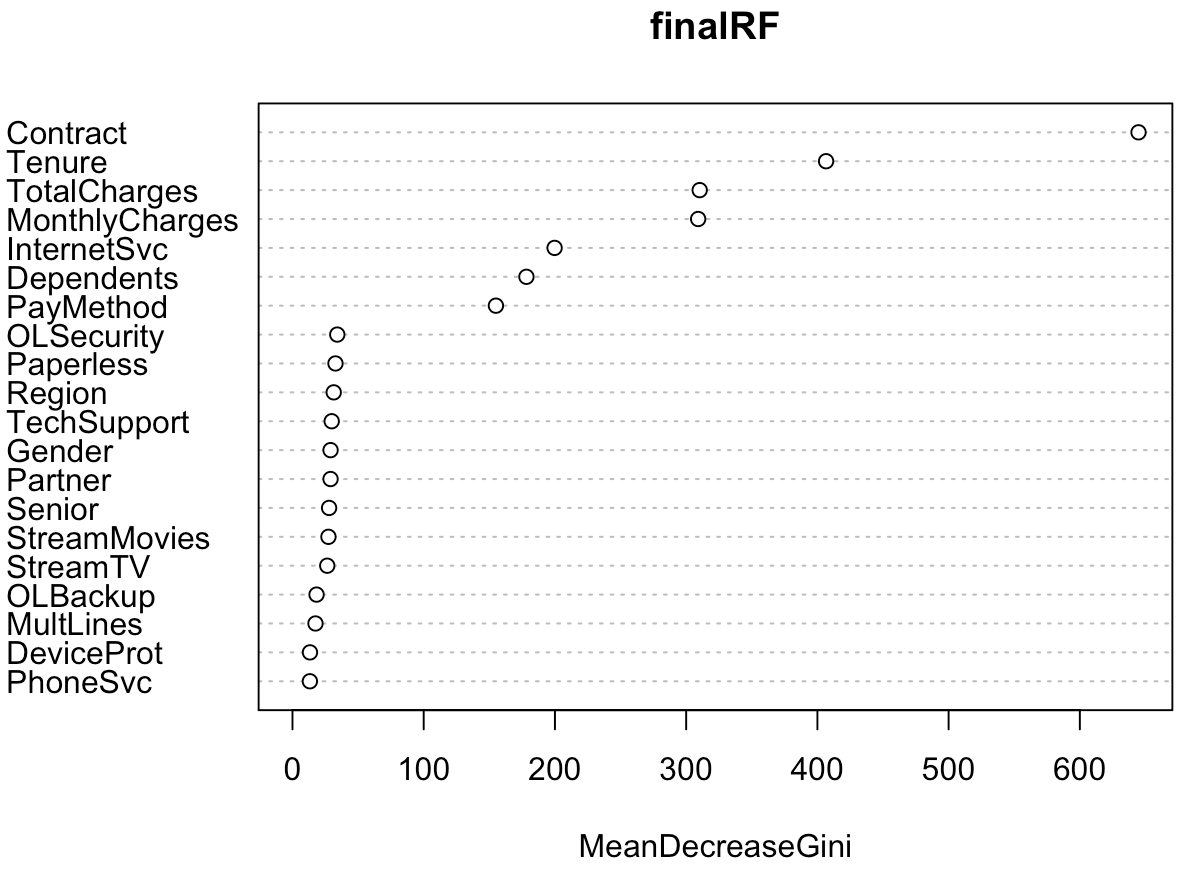
Hypertuning the Random Forest

```
# Final rf model trained on all train data
#1000 and 9
set.seed(0)
finalRF <- randomForest(ChurnLabel ~., data = oversampleTrain, mtry = 9, ntree = 1000, n
odesize = 30)
print(finalRF)
```

```
##
## Call:
##  randomForest(formula = ChurnLabel ~ ., data = oversampleTrain,      mtry = 9, ntree
= 1000, nodesize = 30)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 15.73%
## Confusion matrix:
##       No  Yes class.error
## No  3215  855   0.2100737
## Yes  419 3608   0.1040477
```

Important Variables for Random Forest

```
varImpPlot(finalRF)
```



Confusion Matrix and Other Metrics for the Random Forest Model

```
# class predictions with test data
#get ROC with test data and optimal threshold for rf
roc_score_rf <- roc(test_data$ChurnLabel, as.data.frame(predict(finalRF, test_data, type
="prob"))$Yes)
rf_cutoff <- coords(roc_score_rf, "best", ret="threshold")[[1]]

getCM(getClass(finalRF, "rf",test_data, rf_cutoff), test_data$ChurnLabel)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  401  34
##        Yes 153 165
##
##                Accuracy : 0.7517
##                  95% CI : (0.7192, 0.7821)
##     No Information Rate : 0.7357
##     P-Value [Acc > NIR] : 0.1711
##
##                   Kappa : 0.4641
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8291
##             Specificity : 0.7238
##          Pos Pred Value : 0.5189
##          Neg Pred Value : 0.9218
##              Prevalence : 0.2643
##          Detection Rate : 0.2191
##    Detection Prevalence : 0.4223
##       Balanced Accuracy : 0.7765
##
##        'Positive' Class : Yes
##
```

```
print(paste0("Testing Error for random forest is ", getError(getClass(finalRF, "rf", tes
t_data, rf_cutoff), test_data$ChurnLabel)))
```

```
## [1] "Testing Error for random forest is 0.248339973439575"
```

6. Evaluate models

Using the validation dataset we can get the metrics using data that has not been seen. Finally, we can also use an ROC (receiving operating characteristic) curve that provides us with a visual analysis of the model's results to better compare our different model's performances.

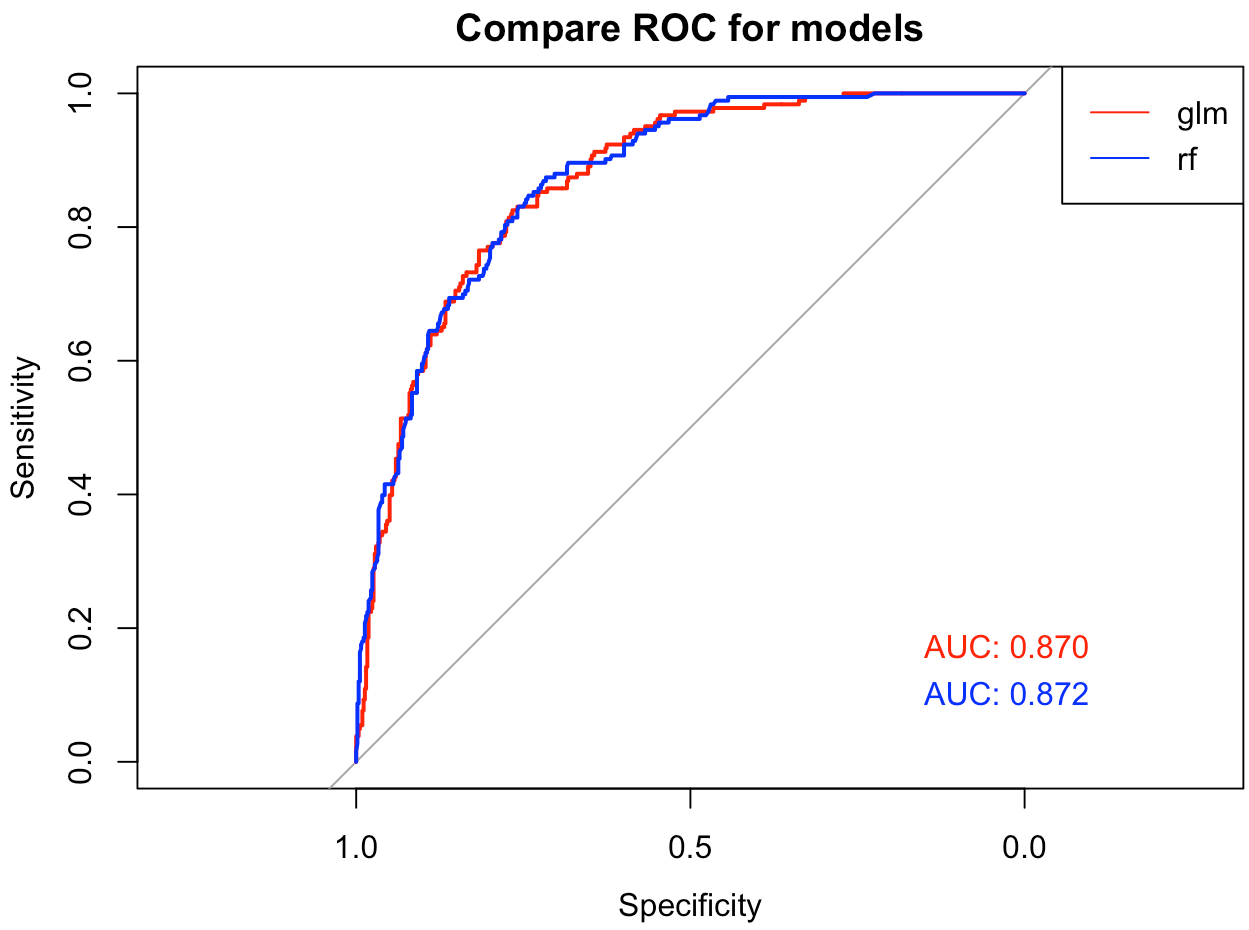
```
# Create confusion matrix to look at values

#Logistic Regression
lrResults <- getMetrics(getClass(finalLR, "lr", val_data, lr_cutoff), val_data$ChurnLabe
l)

#Random Forest
rfResults <- getMetrics(getClass(finalRF, "rf", val_data, rf_cutoff), val_data$ChurnLabe
l)

#join them together
results <- data.frame(Model = c("Logistic Regression", "Random Forest"))
results <- cbind(results, rbind(lrResults, rfResults))
results
```

Model <chr>	Accuracy <dbl>	Precision <dbl>	Sensitivity <dbl>	Specificity <dbl>	F1Score <dbl>
Logistic Regression	0.7867036	0.5568627	0.7759563	0.7903525	0.6484018
Random Forest	0.7548476	0.5096154	0.8688525	0.7161410	0.6424242

2 rows

From the results, we can conclude that our logistic regression model is a more accurate model in comparison to our random forest model. However, the random forest has a much higher sensitivity score in comparison to the logistic regression model, which means the random forest doesn't predict a customer staying when they end up leaving at the rate of which the logistic regression does. For this business case, a model like this would be better in a business sense, depending on the costs associated with each outcome of the model.

```
#ROC curve of models
roc_score_lr <- roc(val_data$ChurnLabel, getProbs(finalLR, "lr",val_data))
roc_score_rf <- roc(val_data$ChurnLabel, getProbs(finalRF, "rf",val_data))

plot(roc_score_lr, col="red", print.auc=TRUE, print.auc.x =0.15, print.auc.y=0.19,main
="Compare ROC for models")
plot(roc_score_rf, add=TRUE, col="blue", print.auc=TRUE, print.auc.x =0.15, print.auc.y=
0.12)
legend("topright",legend = c("glm","rf"),lty = c(1,1), col = c("red","blue"))
```




```
# Final test on validation data set

#Logistic Regression
print(paste0("Testing Error for final logistic regression is ", getError(getClass(finalL
R, "lr", test_data, lr_cutoff), test_data$ChurnLabel)))
```

```
## [1] "Testing Error for final logistic regression is 0.216467463479416"
```

```
print(paste0("Validation Error for final logistic regression is ", getError(getClass(fin
alLR, "lr", val_data, lr_cutoff), val_data$ChurnLabel)))
```

```
## [1] "Validation Error for final logistic regression is 0.213296398891967"
```

```
#Random Forest
print(paste0("Testing Error for final random forest is ", getError(getClass(finalRF, "r
f", test_data, rf_cutoff), test_data$ChurnLabel)))
```

```
## [1] "Testing Error for final random forest is 0.248339973439575"
```

```
print(paste0("Validation Error for final random forest is ", getError(getClass(finalRF,
"rf", val_data, rf_cutoff), val_data$ChurnLabel)))
```

```
## [1] "Validation Error for final random forest is 0.245152354570637"
```

After testing both of our final models on the validation dataset, we can see that the error for both models is marginally lower compared to the testing set. Since the difference is so minor, we can conclude that our models and their respective errors are consistent across multiple datasets.

7. Conclusion

Both models indicate some of the important features in predicting customer churn include the Contract Length, Payment Method, and whether the customer has any Dependents. Using this information, the company can better tailor certain promotions to target these important features to limit customer churn and thus improve their profitability.

To finalize the results of this analysis it is recommended to use the machine learning models to enhance the telecom company's churn prediction to help in targeting preventative actions.