Churn Classifier Final Report

S.S.Ananth Kumar,Eyob Tadele,Munerah AlFayez,Zade Al-Shayeb,Vamshee Deepak

28/11/2021

.libPaths("C:\Users\Ananth\OneDrive\Desktop\MSBA Kent\Fall 2021\Fundamentals of Machine Learning\Assignment\Ass 2")

Group project member's contribution

Group Member	Contribution
Sree Ananth Kumar Seethamraju	discussion, code, report, presentation, editing & review
Eyob Tadele	discussion, code, report, presentation, editing & review
Munerah AlFayez	discussion
Zade Al-Shayeb	discussion
Vamshee Deepak	discussion

Project Objective

The purpose of this project is to help address ABC Wireless Inc.'s customer churn issue. Our goal is to work as part of a team and use the company's historical data to predict, or identify customers who are likely to churn. It is important to realize that churn, also known as loss of customers to a competitor is a major headache for telecom companies. It is more expensive to acquire a new customer, than to keep existing ones. As such, our team is tasked with applying analytics to help reduce churn rate by identifying which particular plan has more impact or service which is influencing churn rate and recommend useful insights that management can apply.

Data Exploration

```
library("dplyr")
library("magrittr")
library("ggplot2")
library("tidyverse")
library("randomForest")
library("randomForestExplainer")
library("DMwR2")
library("tidyr")
library("usmap")
library("ggplot2")
```

Feel of the data

We will first get a feel for our data set by getting a summary of the dataframe `churn_df`.

```
churn_df <- read.csv("data/churn_train.csv", na.strings = c("", "NA"))

summary(churn_df)
```

```
##     state           account_length     area_code         international_plan
##  Length:3333        Min.   :-209.00   Length:3333        Length:3333
##  Class :character   1st Qu.:  72.00   Class :character   Class :character
##  Mode  :character   Median : 100.00   Mode  :character   Mode  :character
##                     Mean   :  97.32
##                     3rd Qu.: 127.00
##                     Max.   : 243.00
##                     NA's   :501
##  voice_mail_plan    number_vmail_messages total_day_minutes total_day_calls
##  Length:3333        Min.   :-10.000       Min.   :   0.0    Min.   :  0.0
##  Class :character   1st Qu.:  0.000       1st Qu.: 149.3    1st Qu.: 87.0
##  Mode  :character   Median :  0.000       Median : 190.5    Median :101.0
##                     Mean   :  7.333       Mean   : 418.9    Mean   :100.3
##                     3rd Qu.: 16.000       3rd Qu.: 237.8    3rd Qu.:114.0
##                     Max.   : 51.000       Max.   :2185.1    Max.   :165.0
##                     NA's   :200           NA's   :200       NA's   :200
##  total_day_charge total_eve_minutes total_eve_calls total_eve_charge
##  Min.   : 0.00    Min.   :   0.0    Min.   :  0.0   Min.   : 0.00
```

```
##  1st Qu.:24.45    1st Qu.: 170.5    1st Qu.: 87.0   1st Qu.:14.14
##  Median :30.65    Median : 209.9    Median :100.0   Median :17.09
##  Mean   :30.63    Mean   : 324.3    Mean   :100.1   Mean   :17.08
##  3rd Qu.:36.84    3rd Qu.: 257.6    3rd Qu.:114.0   3rd Qu.:20.00
##  Max.   :59.64    Max.   :1244.2    Max.   :170.0   Max.   :30.91
##  NA's   :200      NA's   :301       NA's   :200     NA's   :200
##  total_night_minutes total_night_calls total_night_charge total_intl_minutes
##  Min.   : 23.2       Min.   : 33.0     Min.   : 1.040     Min.   : 0.00
##  1st Qu.:167.3       1st Qu.: 87.0     1st Qu.: 7.530     1st Qu.: 8.50
##  Median :201.4       Median :100.0     Median : 9.060     Median :10.30
##  Mean   :201.2       Mean   :100.1     Mean   : 9.054     Mean   :10.23
##  3rd Qu.:235.3       3rd Qu.:113.0     3rd Qu.:10.590     3rd Qu.:12.10
##  Max.   :395.0       Max.   :175.0     Max.   :17.770     Max.   :20.00
##  NA's   :200                           NA's   :200        NA's   :200
##  total_intl_calls total_intl_charge number_customer_service_calls
##  Min.   : 0.00    Min.   :0.000     Min.   :0.000
##  1st Qu.: 3.00    1st Qu.:2.300     1st Qu.:1.000
##  Median : 4.00    Median :2.780     Median :1.000
##  Mean   : 4.47    Mean   :2.762     Mean   :1.561
##  3rd Qu.: 6.00    3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :20.00    Max.   :5.400     Max.   :9.000
##  NA's   :301      NA's   :200       NA's   :200
##     churn
##  Length:3333
##  Class :character
##  Mode  :character
##
##
##
##
```

From the summary we can see a lot of NA values in many of the features except, `state`, `area_code`, `international_plan`, `voice_mail_plan`, `total_night_calls`, and `churn`.

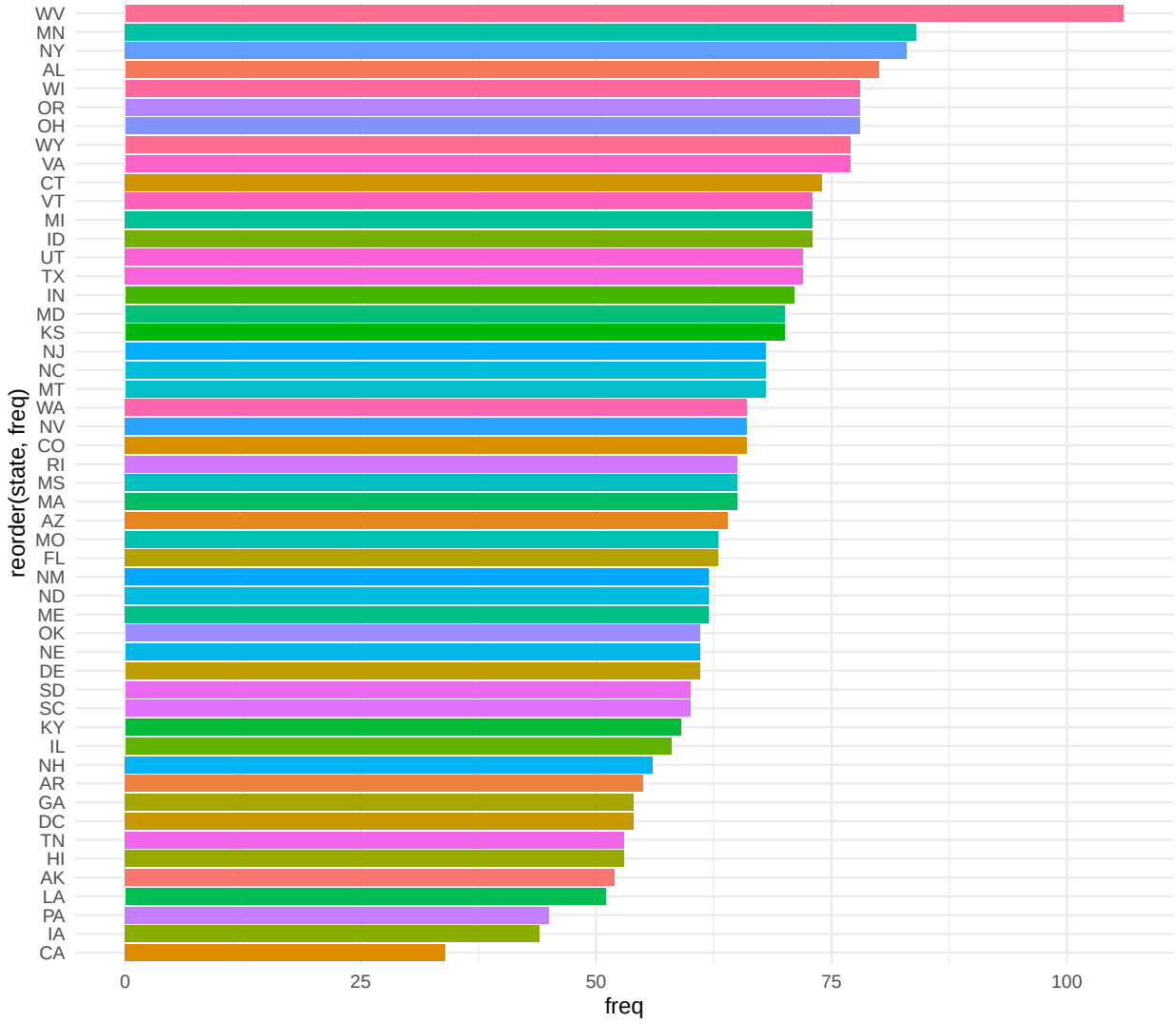
If we take a closer look at the `state` feature, the frequency table and histogram will show where most of the customers observed in the churn dataset come from.

```
state_freq <- churn_df %>%
  select(state) %>%
  group_by(state) %>%
  summarise(freq  = n()) %>%
  arrange(desc(freq))

state_freq %>% head()
```

```
## # A tibble: 6 x 2
##   state  freq
##   <chr> <int>
## 1 WV      106
## 2 MN       84
## 3 NY       83
## 4 AL       80
## 5 OH       78
## 6 OR       78
```

```
ggplot(state_freq, aes(x = reorder(state, freq), y = freq, fill = state)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  theme_minimal() +
  guides(fill = F)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



From the table and the histogram above we can see that *West Virginia* represents the most, with a total of *106* customers.

Negative value observation

looking at the summary data, we can see that some features have negative values, here is a look at it:

```
churn_df %>%
  select(account_length, number_vmail_messages) %>%
  summary()
```

```
##  account_length    number_vmail_messages
##  Min.   :-209.00   Min.   :-10.000
##  1st Qu.:  72.00   1st Qu.:  0.000
##  Median : 100.00   Median :  0.000
##  Mean   :  97.32   Mean   :  7.333
##  3rd Qu.: 127.00   3rd Qu.: 16.000
##  Max.   : 243.00   Max.   : 51.000
##  NA's   :501       NA's   :200
```

`account_length` has values ranging from -209 and 243. The variable account_length is not immediately obvious what it represents. But in the domain of this data set, `account_length` represents how long a customer has had an account in terms of months (we are assuming `account_length` is in months). With that being said, `account_length` should not contain any negative values.

`number_vmail_messages` has values ranging from -10 to 51, this variable represents the number of voice mail messages a customer has had. Clearly such a variable should not have negative values in it.

Other Missing Values in data

16 out of the 20 variables (columns) have NA values; where NA refers to missing values. Further analysis of the summary of our dataframe reveals that 10 variables have about *200* `NA` values while 2 have *301* and 1 has *501*.

For a better understanding of the presence of NAs in our dataframe, a look at the percentage of NAs across all the variables in the dataframe is important.

```
# A function to compute the percentage of NAs accross all columns.
na_percentage <- function(df, fmt = F) {
  return (df %>%
            is.na() %>%
            colMeans() %>%
            sapply(function(x) {
              if (fmt) {
                return(sprintf("%.5f%%", x * 100))
              }

              return (x)
            })
          )
}

na_percent_df <- na_percentage(churn_df) %>%
  data_frame(Columns = names(.), `NA %` = .) %>%
  mutate_at(
    vars(`NA %`),
    funs(round(. * 100, 2))
  ) %>%
  mutate(label = sprintf("%g%%", `NA %`)) %>%
  arrange(desc(`NA %`))
```

```
## Warning: ‘data_frame()‘ was deprecated in tibble 1.1.0.
## Please use ‘tibble()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```

```
## Warning: ‘funs()‘ was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with ‘tibble::lst()‘:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```

```
na_percent_df %>% select(-label) %>% head()
```

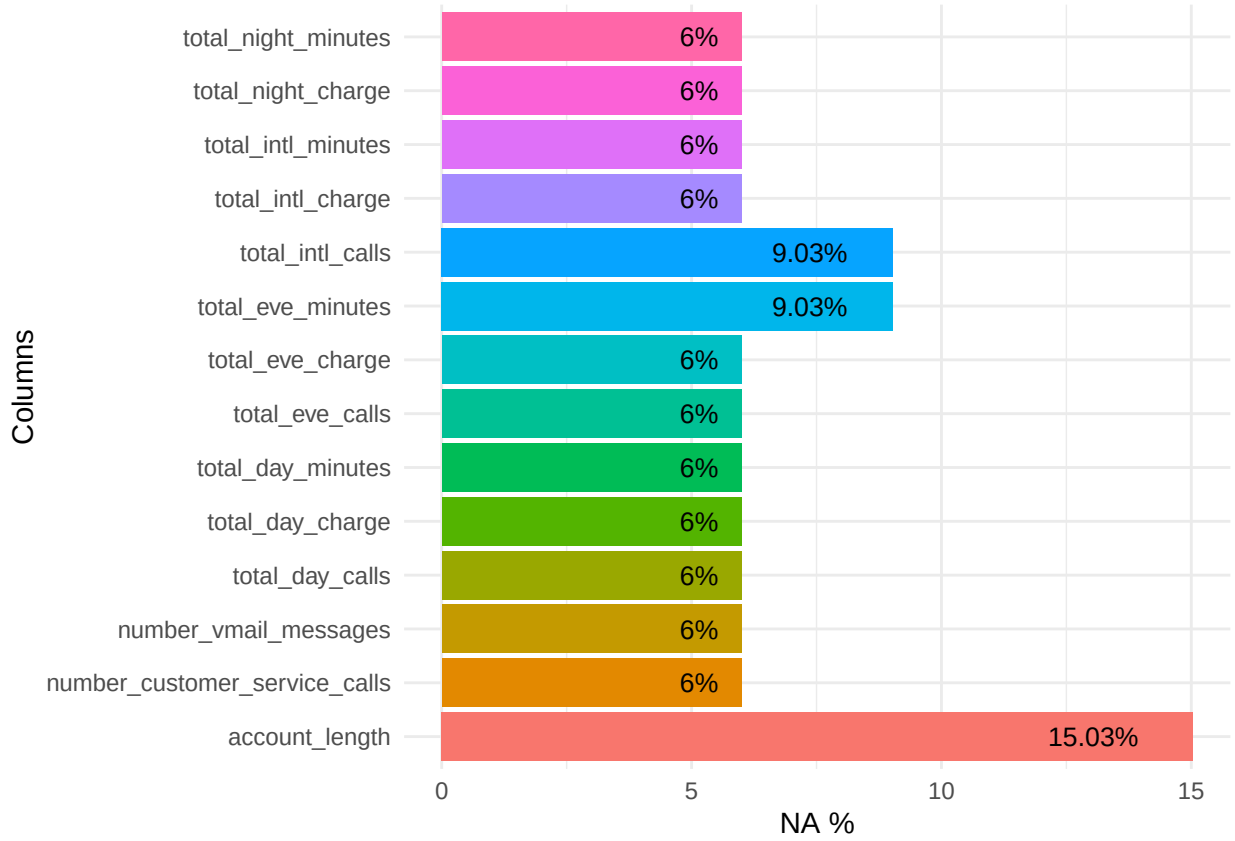
```
## # A tibble: 6 x 2
##   Columns               ‘NA %‘
##   <chr>                  <dbl>
## 1 account_length         15.0
## 2 total_eve_minutes       9.03
## 3 total_intl_calls        9.03
## 4 number_vmail_messages   6
## 5 total_day_minutes       6
## 6 total_day_calls         6
```

The table above lists all the variables (columns) and their respective percentage of `NA`s. We can see that most categorical variables such as `state`, `area_code`, `international_plan`, etc. including `total_night_calls` (numerical variable) have no `NA` values in them.

The bar chart below provides a visual representation of the percentage of `NA` in the dataset. We can see that `account_length`, `total_intl_calls` and `total_intl_charge` contribute the most `NA`s with `account_length` being the top contributor.

```
na_percent_df %>%
  filter(`NA %`> 0) %>%
  ggplot(aes(x = Columns, y = `NA %`, fill = Columns)) +
  geom_bar(stat="identity") +
  guides(fill = F) +
  coord_flip() +
  geom_text(aes(label = label), hjust = 1.6, size = 3.5) +
  theme_minimal()
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```



Further analysis of the `NA` percentage table, we noticed that 11 variables have an NA percentage of *6%*. Such a pattern is interesting and deserves a closer look.

Below is a table that shows only the variables that have `NA` in them. The code chunk removes columns that have an `NA` percentage of *0%* and then only shows rows that have at least 1 `NA` value in them.

```
na_df <- churn_df %>%
  select(-state, -area_code, -international_plan, -voice_mail_plan, -total_night_calls, -churn) %>%
  filter_all(any_vars(is.na(.)))

head(na_df)
```

```
##   account_length number_vmail_messages total_day_minutes total_day_calls
## 1             NA                    30             110.3              71
## 2             89                     0             178.7              81
## 3             65                     0             129.1             137
## 4             NA                    NA                NA              NA
## 5             NA                    32             247.0             109
## 6             NA                     0             150.9              79
##   total_day_charge total_eve_minutes total_eve_calls total_eve_charge
## 1            18.75             182.4             108            15.50
## 2            30.38                NA              74            19.86
## 3            21.95                NA              83            19.42
## 4               NA                NA              NA               NA
## 5            41.99             125.6              91            10.68
## 6            25.65             161.8              87            13.75
```

```
##   total_night_minutes total_night_charge total_intl_minutes total_intl_calls
## 1               183.8               8.27               11.0                8
## 2               131.9               5.94                9.1                4
## 3               208.8               9.40               12.7                6
## 4                  NA                 NA                 NA               NA
## 5               226.5              10.19               10.5                4
## 6               167.7               7.55               11.7                5
##   total_intl_charge number_customer_service_calls
## 1              2.97                             2
## 2              2.46                             1
## 3              3.43                             4
## 4                NA                            NA
## 5              2.84                             3
## 6              3.16                             3
```

We can see that there are many `NA` values present in the 703 row subset of our data set. There are 503 rows that have at least **1** `NA` in one of their columns, while 200 rows have all its elements consisting of completely `NA` values.

Preliminary Data Cleaning

Turning Negatives into Positives

In order to deal with those variables that have negative values in them, the simple strategy is to turn all the numbers for each variable in question to a positive using the `abs` function.

```
churn_df <- churn_df %>%
  mutate_at(.vars = vars(account_length, number_vmail_messages), .funs = funs(abs))

summary(churn_df)
```

```
##     state           account_length   area_code         international_plan
##  Length:3333        Min.   :  1.0   Length:3333        Length:3333
##  Class :character   1st Qu.: 73.0   Class :character   Class :character
##  Mode  :character   Median :101.0   Mode  :character   Mode  :character
##                     Mean   :100.8
##                     3rd Qu.:127.0
##                     Max.   :243.0
##                     NA's   :501
##  voice_mail_plan    number_vmail_messages total_day_minutes total_day_calls
##  Length:3333        Min.   : 0.000        Min.   :   0.0    Min.   :  0.0
##  Class :character   1st Qu.: 0.000        1st Qu.: 149.3    1st Qu.: 87.0
##  Mode  :character   Median : 0.000        Median : 190.5    Median :101.0
##                     Mean   : 8.056        Mean   : 418.9    Mean   :100.3
##                     3rd Qu.:16.000        3rd Qu.: 237.8    3rd Qu.:114.0
##                     Max.   :51.000        Max.   :2185.1    Max.   :165.0
##                     NA's   :200           NA's   :200       NA's   :200
##  total_day_charge total_eve_minutes total_eve_calls total_eve_charge
##  Min.   : 0.00    Min.   :   0.0    Min.   :  0.0   Min.   : 0.00
##  1st Qu.:24.45    1st Qu.: 170.5    1st Qu.: 87.0   1st Qu.:14.14
##  Median :30.65    Median : 209.9    Median :100.0   Median :17.09
##  Mean   :30.63    Mean   : 324.3    Mean   :100.1   Mean   :17.08
```

```
##  3rd Qu.:36.84    3rd Qu.: 257.6    3rd Qu.:114.0   3rd Qu.:20.00
##  Max.   :59.64    Max.   :1244.2    Max.   :170.0   Max.   :30.91
##  NA's   :200      NA's   :301       NA's   :200     NA's   :200
##  total_night_minutes total_night_calls total_night_charge total_intl_minutes
##  Min.   : 23.2       Min.   : 33.0     Min.   : 1.040     Min.   : 0.00
##  1st Qu.:167.3       1st Qu.: 87.0     1st Qu.: 7.530     1st Qu.: 8.50
##  Median :201.4       Median :100.0     Median : 9.060     Median :10.30
##  Mean   :201.2       Mean   :100.1     Mean   : 9.054     Mean   :10.23
##  3rd Qu.:235.3       3rd Qu.:113.0     3rd Qu.:10.590     3rd Qu.:12.10
##  Max.   :395.0       Max.   :175.0     Max.   :17.770     Max.   :20.00
##  NA's   :200                           NA's   :200        NA's   :200
##  total_intl_calls total_intl_charge number_customer_service_calls
##  Min.   : 0.00    Min.   :0.000     Min.   :0.000
##  1st Qu.: 3.00    1st Qu.:2.300     1st Qu.:1.000
##  Median : 4.00    Median :2.780     Median :1.000
##  Mean   : 4.47    Mean   :2.762     Mean   :1.561
##  3rd Qu.: 6.00    3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :20.00    Max.   :5.400     Max.   :9.000
##  NA's   :301      NA's   :200       NA's   :200
##     churn
##  Length:3333
##  Class :character
##  Mode  :character
##
##
##
##
```

From the summary table, we can see that all our variables are positive. `account_length` ranges from 1 to 243 and `number_vmail_messages` ranges from 0 to 51.

No More NAs

Rows which are completely filled with NA values pose a problem, as they do not have enough data in them which can be used for predicting churn. Each row represents a customer and if a row has 14 elements (representing the 14 columns) consisting of NA, then that customer is essentially inconsequential in the training of our model.

Imputing of missing values is an approach to solving this problem. But in this particular case, it would be pointless to do so for rows which have so much of its predictive power missing. There are some rows in which we can impute missing values in. These rows can be salvaged because the percentage of `NA`s in them is not a 100%.

The best course of action is to remove rows that have more than **75%** of its elements that are `NA`. **75%** is an arbitrary threshold that has been chosen based on the consensus of the group, which we believe will keep rows that are salvageable and remove the rows that are unimportant.

```
churn_df_1 <- churn_df[rowMeans(is.na(churn_df)) <= 0.25,]
summary(churn_df_1)
```

```
##     state           account_length   area_code         international_plan
##  Length:3133        Min.   :  1.0   Length:3133        Length:3133
##  Class :character   1st Qu.: 73.0   Class :character   Class :character
##  Mode  :character   Median :101.0   Mode  :character   Mode  :character
```

```
##                     Mean   :100.8
##                     3rd Qu.:127.0
##                     Max.   :243.0
##                     NA's   :301
##  voice_mail_plan    number_vmail_messages total_day_minutes total_day_calls
##  Length:3133        Min.   : 0.000        Min.   :   0.0    Min.   :  0.0
##  Class :character   1st Qu.: 0.000        1st Qu.: 149.3    1st Qu.: 87.0
##  Mode  :character   Median : 0.000        Median : 190.5    Median :101.0
##                     Mean   : 8.056        Mean   : 418.9    Mean   :100.3
##                     3rd Qu.:16.000        3rd Qu.: 237.8    3rd Qu.:114.0
##                     Max.   :51.000        Max.   :2185.1    Max.   :165.0
##
##  total_day_charge total_eve_minutes total_eve_calls total_eve_charge
##  Min.   : 0.00    Min.   :   0.0    Min.   :  0.0   Min.   : 0.00
##  1st Qu.:24.45    1st Qu.: 170.5    1st Qu.: 87.0   1st Qu.:14.14
##  Median :30.65    Median : 209.9    Median :100.0   Median :17.09
##  Mean   :30.63    Mean   : 324.3    Mean   :100.1   Mean   :17.08
##  3rd Qu.:36.84    3rd Qu.: 257.6    3rd Qu.:114.0   3rd Qu.:20.00
##  Max.   :59.64    Max.   :1244.2    Max.   :170.0   Max.   :30.91
##                   NA's   :101
##  total_night_minutes total_night_calls total_night_charge total_intl_minutes
##  Min.   : 23.2       Min.   : 33.0     Min.   : 1.040     Min.   : 0.00
##  1st Qu.:167.3       1st Qu.: 87.0     1st Qu.: 7.530     1st Qu.: 8.50
##  Median :201.4       Median :100.0     Median : 9.060     Median :10.30
##  Mean   :201.2       Mean   :100.1     Mean   : 9.054     Mean   :10.23
##  3rd Qu.:235.3       3rd Qu.:114.0     3rd Qu.:10.590     3rd Qu.:12.10
##  Max.   :395.0       Max.   :175.0     Max.   :17.770     Max.   :20.00
##
##  total_intl_calls total_intl_charge number_customer_service_calls
##  Min.   : 0.00    Min.   :0.000     Min.   :0.000
##  1st Qu.: 3.00    1st Qu.:2.300     1st Qu.:1.000
##  Median : 4.00    Median :2.780     Median :1.000
##  Mean   : 4.47    Mean   :2.762     Mean   :1.561
##  3rd Qu.: 6.00    3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :20.00    Max.   :5.400     Max.   :9.000
##  NA's   :101
##     churn
##  Length:3133
##  Class :character
##  Mode  :character
##
##
##
##
```

Here is look at the percentage of NAs in the data set after removing rows with *75%* of its elements being NA and how its has changed:

```
na_df_1 <- na_percentage(churn_df_1) %>%
  data_frame(Columns = names(.), `NA %` = .) %>%
  mutate_at(
    vars(`NA %`),
    funs(round(. * 100, 2))
  ) %>%
```

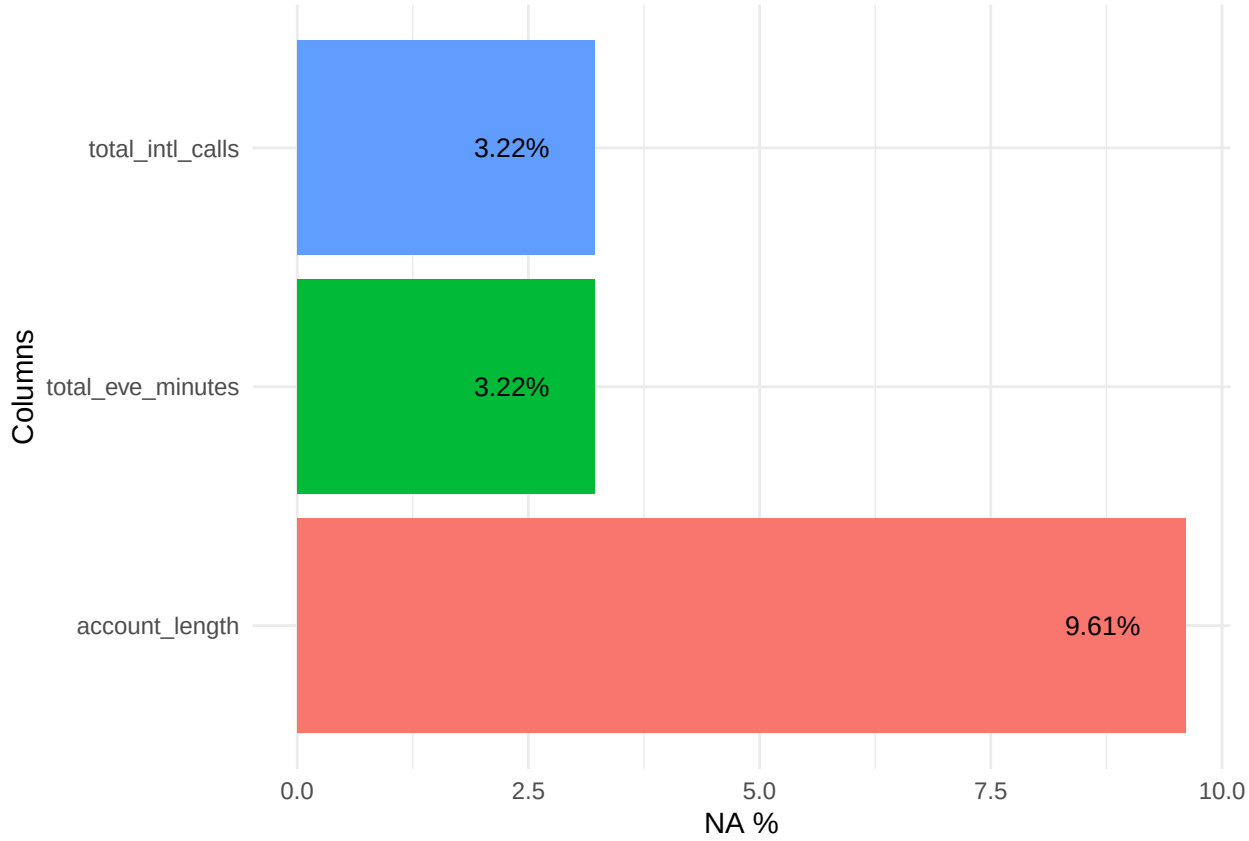
```
  mutate(label = sprintf("%g%%", `NA %`)) %>%
  arrange(desc(`NA %`))

head(na_df_1)
```

```
## # A tibble: 6 x 3
##   Columns            `NA %` label
##   <chr>               <dbl> <chr>
## 1 account_length       9.61 9.61%
## 2 total_eve_minutes    3.22 3.22%
## 3 total_intl_calls     3.22 3.22%
## 4 state                0    0%
## 5 area_code            0    0%
## 6 international_plan   0    0%
```

A look at a visual presentation of how the NA values have changed is shown below:

```
na_df_1 %>%
  filter(`NA %`> 0) %>%
  ggplot(aes(x = Columns, y = `NA %`, fill = Columns)) +
  geom_bar(stat="identity") +
  guides(fill = F) +
  coord_flip() +
  geom_text(aes(label = label), hjust = 1.6, size = 3.5) +
  theme_minimal()
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



Sub-setting the dataset and looking at the columns we initially identified as having at least 1 NA in them:

```
churn_df_1 %>%
  select(-state, -area_code, -international_plan, -voice_mail_plan, -total_night_calls, -churn) %>%
  filter_all(any_vars(is.na(.))) %>%
  head()
```

```
##   account_length number_vmail_messages total_day_minutes total_day_calls
## 1             NA                    30             110.3              71
## 2             89                     0             178.7              81
## 3             65                     0             129.1             137
## 4             NA                    32             247.0             109
## 5             NA                     0             150.9              79
## 6             89                     0             134.9              59
##   total_day_charge total_eve_minutes total_eve_calls total_eve_charge
## 1            18.75             182.4             108            15.50
## 2            30.38                NA              74            19.86
## 3            21.95                NA              83            19.42
## 4            41.99             125.6              91            10.68
## 5            25.65             161.8              87            13.75
## 6            22.93                NA             152            13.26
##   total_night_minutes total_night_charge total_intl_minutes total_intl_calls
## 1               183.8               8.27               11.0               8
## 2               131.9               5.94                9.1               4
## 3               208.8               9.40               12.7               6
## 4               226.5              10.19               10.5               4
```

```
## 5               167.7               7.55               11.7                5
## 6               197.5               8.89               10.2                5
##   total_intl_charge number_customer_service_calls
## 1              2.97                             2
## 2              2.46                             1
## 3              3.43                             4
## 4              2.84                             3
## 5              3.16                             3
## 6              2.75                             1
```

The number of rows initially was 703 and 503 now. We have removed the **200** rows where the population of `NA`s were 75%, reducing our overall dataset from **3333** to **3133**.

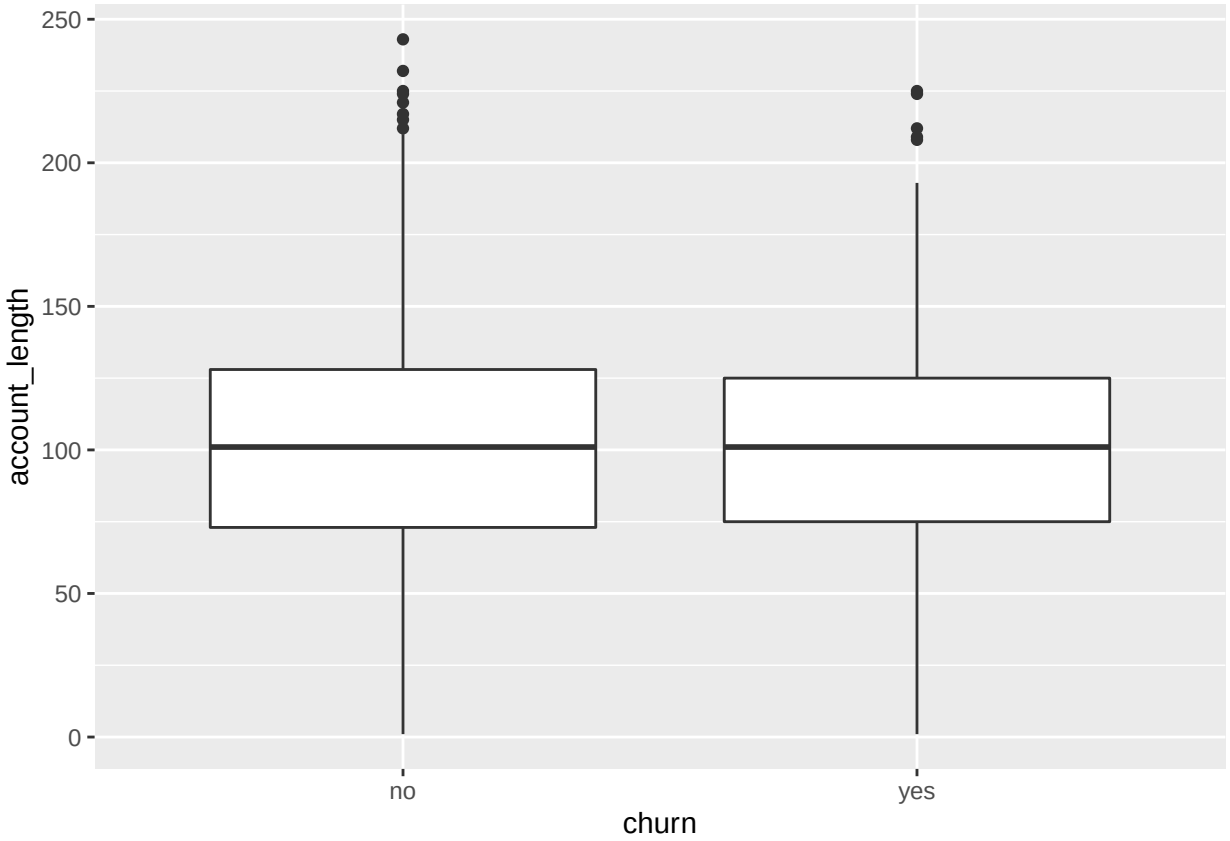
Now that we have removed rows which had very little information in them, we can focus on figuring out a strategy on filling in the missing values of our remaining 503 rows.

The Remaining NAs

We have about 503 `NA` values in total in our dataset. Of which, `account_length` owns 9.61% of it, while `total_eve_minutes` and `total_intl_class` both own 3.22%.

Looking at `account_length` we were not convinced that it was statistically significant or that it had any predictive power when it came to predicting churn. In order to investigate our hypothesis, we compared the correlation of `churn` and `account_length` using a box plot as seen below. The aim is to see whether `account_length` can clearly differentiate between **no** and **yes**.

```
ggplot(churn_df_1, aes(churn, account_length)) +
  geom_boxplot()
```



The boxplot tells us that `account_length` can not differentiate between **no** and **yes**. As `account_length` increases, the number of **no** and **yes** varies very little. Meaning, `account_length` is not a good predictor of `churn`.

Upon validating our hypothesis, we decided on moving forward with the descision to omit `account_length` from the dataset. The benefit of this is that we get rid of the **301** `NA` values in our dataset, which we would have had to spend time imputing for, if we had kept `account_length`.

The decision of omitting `account_length` from our data set benefits us in that:

1. It helps us avoid spending unnecessary effort in imputing a large percentage of missing values contributed by `account_length`
2. We remove a variable that has neither predictive power, nor is statistically significant.

Concluding The Exploration

We will omit `account_length` from our data set and then view the summary.

```
churn_df_2 <- churn_df_1 %>% select(-account_length)

summary(churn_df_2)
```

```
##     state            area_code         international_plan voice_mail_plan
##  Length:3133        Length:3133        Length:3133        Length:3133
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##
##  number_vmail_messages total_day_minutes total_day_calls total_day_charge
##  Min.   : 0.000        Min.   :   0.0    Min.   :  0.0   Min.   : 0.00
##  1st Qu.: 0.000        1st Qu.: 149.3    1st Qu.: 87.0   1st Qu.:24.45
##  Median : 0.000        Median : 190.5    Median :101.0   Median :30.65
##  Mean   : 8.056        Mean   : 418.9    Mean   :100.3   Mean   :30.63
##  3rd Qu.:16.000        3rd Qu.: 237.8    3rd Qu.:114.0   3rd Qu.:36.84
##  Max.   :51.000        Max.   :2185.1    Max.   :165.0   Max.   :59.64
##
##  total_eve_minutes total_eve_calls total_eve_charge total_night_minutes
##  Min.   :   0.0    Min.   :  0.0   Min.   : 0.00    Min.   : 23.2
##  1st Qu.: 170.5    1st Qu.: 87.0   1st Qu.:14.14    1st Qu.:167.3
##  Median : 209.9    Median :100.0   Median :17.09    Median :201.4
##  Mean   : 324.3    Mean   :100.1   Mean   :17.08    Mean   :201.2
##  3rd Qu.: 257.6    3rd Qu.:114.0   3rd Qu.:20.00    3rd Qu.:235.3
##  Max.   :1244.2    Max.   :170.0   Max.   :30.91    Max.   :395.0
##  NA's   :101
##  total_night_calls total_night_charge total_intl_minutes total_intl_calls
##  Min.   : 33.0     Min.   : 1.040     Min.   : 0.00      Min.   : 0.00
##  1st Qu.: 87.0     1st Qu.: 7.530     1st Qu.: 8.50      1st Qu.: 3.00
##  Median :100.0     Median : 9.060     Median :10.30      Median : 4.00
##  Mean   :100.1     Mean   : 9.054     Mean   :10.23      Mean   : 4.47
##  3rd Qu.:114.0     3rd Qu.:10.590     3rd Qu.:12.10      3rd Qu.: 6.00
##  Max.   :175.0     Max.   :17.770     Max.   :20.00      Max.   :20.00
##                                                          NA's   :101
##  total_intl_charge number_customer_service_calls    churn
##  Min.   :0.000     Min.   :0.000                 Length:3133
##  1st Qu.:2.300     1st Qu.:1.000                 Class :character
##  Median :2.780     Median :1.000                 Mode  :character
##  Mean   :2.762     Mean   :1.561
##  3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :5.400     Max.   :9.000
##
```

Currently we have only **202** `NA` values. Essentially removing **301** `NA` values from `account_length`. We now need to impute values for **202** `NA`s

We will save the data frame, `churn_df_2`, that has gone through the preliminary data cleaning phase for data imputation.

Data Preparation

```
library("randomForest")
library("DMwR2") # for kNN imputation
```

Data Imputation

Data Imputation using RandomForest

The proximity matrix from the randomForest is used to update the imputation of the NAs. For continuous predictors, the imputed value is the weighted average of the non-missing observations, where the weights are the proximities. For categorical predictors, the imputed value is the category with the largest average proximity. This process is iterated n times.

```
str(churn_df_2)
churn_df_2$churn =as.factor(churn_df_2$churn)
churn_df_2$state = as.factor(churn_df_2$state)
churn_df_2$international_plan = as.factor(churn_df_2$international_plan)
churn_df_2$voice_mail_plan =as.factor(churn_df_2$voice_mail_plan)
churn_df_2$area_code = as.factor(churn_df_2$area_code)
churn_df_2 = select(churn_df_2,-c(area_code))

cdf_rf.imputed <- rfImpute(churn ~ ., data = churn_df_2)
```

```
summary(cdf_rf.imputed)
```

```
##  churn          state      international_plan voice_mail_plan
##  no :2676   WV     : 100   no :2832           no :2259
##  yes: 457   AL     :  79   yes: 301           yes: 874
##             MN     :  78
##             NY     :  77
##             OH     :  76
##             OR     :  75
##             (Other):2648
##  number_vmail_messages total_day_minutes total_day_calls total_day_charge
##  Min.   : 0.000        Min.   :   0.0    Min.   :  0.0   Min.   : 0.00
##  1st Qu.: 0.000        1st Qu.: 149.3    1st Qu.: 87.0   1st Qu.:24.45
##  Median : 0.000        Median : 190.5    Median :101.0   Median :30.65
##  Mean   : 8.056        Mean   : 418.9    Mean   :100.3   Mean   :30.63
##  3rd Qu.:16.000        3rd Qu.: 237.8    3rd Qu.:114.0   3rd Qu.:36.84
##  Max.   :51.000        Max.   :2185.1    Max.   :165.0   Max.   :59.64
##
##  total_eve_minutes total_eve_calls total_eve_charge total_night_minutes
##  Min.   :   0.0    Min.   :  0.0   Min.   : 0.00    Min.   : 23.2
##  1st Qu.: 172.1    1st Qu.: 87.0   1st Qu.:14.14    1st Qu.:167.3
##  Median : 211.7    Median :100.0   Median :17.09    Median :201.4
##  Mean   : 323.9    Mean   :100.1   Mean   :17.08    Mean   :201.2
##  3rd Qu.: 263.0    3rd Qu.:114.0   3rd Qu.:20.00    3rd Qu.:235.3
##  Max.   :1244.2    Max.   :170.0   Max.   :30.91    Max.   :395.0
##
##  total_night_calls total_night_charge total_intl_minutes total_intl_calls
##  Min.   : 33.0     Min.   : 1.040     Min.   : 0.00      Min.   : 0.000
##  1st Qu.: 87.0     1st Qu.: 7.530     1st Qu.: 8.50      1st Qu.: 3.000
##  Median :100.0     Median : 9.060     Median :10.30      Median : 4.000
##  Mean   :100.1     Mean   : 9.054     Mean   :10.23      Mean   : 4.474
##  3rd Qu.:114.0     3rd Qu.:10.590     3rd Qu.:12.10      3rd Qu.: 6.000
##  Max.   :175.0     Max.   :17.770     Max.   :20.00      Max.   :20.000
##
```

```
##  total_intl_charge number_customer_service_calls
##  Min.   :0.000     Min.   :0.000
##  1st Qu.:2.300     1st Qu.:1.000
##  Median :2.780     Median :1.000
##  Mean   :2.762     Mean   :1.561
##  3rd Qu.:3.270     3rd Qu.:2.000
##  Max.   :5.400     Max.   :9.000
##
```

Data Imputation using kNN

kNN is useful for matching a point with its closest k neighbors in a multi-dimensional space and can be used for data that are continuous, discrete, ordinal and categorical. This makes it particularly useful for dealing with most kinds of missing data.The assumption behind using KNN for missing values is that a point value can be approximated by the values of the points that are closest to it, based on other variables. When using KNN, you have to take many parameters into consideration:

- The number of neighbors to look for. Taking a low k will increase the influence of noise and the results are going to be less generalizable while taking a high k will tend to blur local effects which are exactly what we are looking for.
- The aggregation method to use. Here we allow for arithmetic mean, median and mode for numeric variables and mode for categorical ones.
- Normalizing the data is a method that allows to give every attribute the same influence in identifying neighbors when computing certain type of distances like the Euclidean one.The algorithm automatically normalizes the data when both numeric and categorical variable are provided.
- Numeric attribute distances: among the various distance metrics available, we will focus on the main ones, Euclidean and Manhattan. Euclidean is a good distance measure to use if the input variables are similar in type (e.g. all measured widths and heights). Manhattan distance is a good measure to use if the input variables are not similar in type (such as age, height, etc.).
- Categorical attribute distances: without prior transformation, applicable distances are related to frequency and similarity. Here we allow the use of two distances: Hamming distance and the Weighted Hamming distance.

1. Hamming distance: take all the categorical attributes and for each, count one if the value is not the same between two points. The Hamming distance is then the number of attributes for which the value was different.
2. Weighted Hamming distance: also return one if the value is different, but returns the frequency of the value in the attribute if they are matching, increasing the distance when the value is more frequent. When more than one attribute is categorical, the harmonic mean is applied. The result remain between zero and one but the mean value is shifted toward the lower values compared to the arithmetic mean.

- Binary attribute distances: those attributes are generally obtained via categorical variables transformed into dummies.

```
cdf_knn.imputed <- knnImputation(churn_df_2)
```

```
summary(cdf_knn.imputed)
```

```
##      state      international_plan voice_mail_plan number_vmail_messages
##  WV     : 100   no :2832           no :2259        Min.   : 0.000
##  AL     :  79   yes: 301           yes: 874        1st Qu.: 0.000
##  MN     :  78                                      Median : 0.000
##  NY     :  77                                      Mean   : 8.056
##  OH     :  76                                      3rd Qu.:16.000
##  OR     :  75                                      Max.   :51.000
##  (Other):2648
##  total_day_minutes total_day_calls total_day_charge total_eve_minutes
##  Min.   :   0.0    Min.   :  0.0   Min.   : 0.00    Min.   :   0.0
##  1st Qu.: 149.3    1st Qu.: 87.0   1st Qu.:24.45    1st Qu.: 170.9
##  Median : 190.5    Median :101.0   Median :30.65    Median : 209.5
##  Mean   : 418.9    Mean   :100.3   Mean   :30.63    Mean   : 320.3
##  3rd Qu.: 237.8    3rd Qu.:114.0   3rd Qu.:36.84    3rd Qu.: 256.1
##  Max.   :2185.1    Max.   :165.0   Max.   :59.64    Max.   :1244.2
##
##  total_eve_calls total_eve_charge total_night_minutes total_night_calls
##  Min.   :  0.0   Min.   : 0.00    Min.   : 23.2       Min.   : 33.0
##  1st Qu.: 87.0   1st Qu.:14.14    1st Qu.:167.3       1st Qu.: 87.0
##  Median :100.0   Median :17.09    Median :201.4       Median :100.0
##  Mean   :100.1   Mean   :17.08    Mean   :201.2       Mean   :100.1
##  3rd Qu.:114.0   3rd Qu.:20.00    3rd Qu.:235.3       3rd Qu.:114.0
##  Max.   :170.0   Max.   :30.91    Max.   :395.0       Max.   :175.0
##
##  total_night_charge total_intl_minutes total_intl_calls total_intl_charge
##  Min.   : 1.040     Min.   : 0.00      Min.   : 0.000   Min.   :0.000
##  1st Qu.: 7.530     1st Qu.: 8.50      1st Qu.: 3.000   1st Qu.:2.300
##  Median : 9.060     Median :10.30      Median : 4.000   Median :2.780
##  Mean   : 9.054     Mean   :10.23      Mean   : 4.472   Mean   :2.762
##  3rd Qu.:10.590     3rd Qu.:12.10      3rd Qu.: 6.000   3rd Qu.:3.270
##  Max.   :17.770     Max.   :20.00      Max.   :20.000   Max.   :5.400
##
##  number_customer_service_calls churn
##  Min.   :0.000                 no :2676
##  1st Qu.:1.000                 yes: 457
##  Median :1.000
##  Mean   :1.561
##  3rd Qu.:2.000
##  Max.   :9.000
##
```

Stepwise Regression

Stepwise regression is a semi-automated process of building a model by successively adding or removing variables based on the t-statistics of their estimated coefficients. The stepwise option lets you either begin with no variables in the model and proceed forward (i.e., adding one variable at a time) or start with all potential variables in the model and proceed backwards (i.e., removing one variable at a time). At each step, the program performs for each variable currently in the model the t-statistic for its estimated coefficient. For each variable not in the model, it computes the t-statistic that its coefficient would have if it were the next variable added, and squares it. At the next step, the program automatically enters the variable with the highest statistic (forward), or removes the variable with the lowest statistic (backward). In general, as in this case, if you have a modest-sized set of potential variables from which you wish to eliminate a few (i.e., if you're fine-tuning some prior selection of variables), you should generally go backward.

Stepwise Logistic Regression with R Akaike information criterion (AIC), where AIC = 2k - 2 log L = 2k + Deviance, where k = number of parameters. In general, smaller numbers are better. Stepwise Logistic Regression penalizes models with many independent or predictor parameters and with models with poor fit. In general, the lower value of AIC suggests "better" model, but it is a relative measure of model fit. It is used for model selection (i.e. it lets you compare different models estimated on the same dataset. Backwards selection is the default in the Logistic Regression method, although there may be some evidence in the logistic regression literature that backward selection is less successful than forward selection. This may be due to the fact that the full model fit in the first step is the model most likely to result in a complete or quasi-complete separation of response values. However, backward seemed to be successful in this case. As a warning, since the interpretation of coefficients in a model depends on the other terms included, it may seem unwise to let an automatic algorithm determine the questions that we should ask about our data. The decision which variables to include into an analysis should be based on theory. However, there is little theory about these variables, so we need to operate on common business application.

Stepwise Regression using kNN Imputed Data

Using Stepwise Regression, we proceed to identify the variables that may have a significant impact in determining 'churn', using the kNN imputed values. The chunk also identifies states which may be impacted by churn.

```
churn_model_knn <- glm(churn~., data = cdf_knn.imputed, family = "binomial")
summary(churn_model_knn)
```

```
##
## Call:
## glm(formula = churn ~ ., family = "binomial", data = cdf_knn.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8805  -0.5082  -0.3116  -0.1639   3.0810
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -9.4162410  0.9781331  -9.627  < 2e-16 ***
## stateAL                      0.3667726  0.7649865   0.479  0.63162
## stateAR                      0.8178984  0.7629712   1.072  0.28372
## stateAZ                      0.1147094  0.8509463   0.135  0.89277
## stateCA                      1.9563131  0.7912675   2.472  0.01342 *
## stateCO                      0.5131271  0.7738744   0.663  0.50729
## stateCT                      1.1199424  0.7279247   1.539  0.12392
## stateDC                      0.8291810  0.8120997   1.021  0.30724
## stateDE                      0.7016542  0.7529845   0.932  0.35142
## stateFL                      0.4433583  0.7795480   0.569  0.56953
## stateGA                      0.7437439  0.7836642   0.949  0.34259
## stateHI                     -0.1146274  0.9002866  -0.127  0.89868
## stateIA                      0.1738285  0.9076568   0.192  0.84812
## stateID                      0.7566077  0.7616004   0.993  0.32049
## stateIL                     -0.1558754  0.8374189  -0.186  0.85234
## stateIN                      0.4405115  0.7559886   0.583  0.56010
## stateKS                      1.0535719  0.7331931   1.437  0.15073
## stateKY                      0.6872253  0.7835904   0.877  0.38048
## stateLA                      0.6528953  0.8418742   0.776  0.43803
## stateMA                      1.0711402  0.7511926   1.426  0.15389
```

```
## stateMD                        1.1621422  0.7190172   1.616  0.10603
## stateME                        1.2625369  0.7380769   1.711  0.08716 .
## stateMI                        1.4857205  0.7208440   2.061  0.03929 *
## stateMN                        1.1842373  0.7183798   1.648  0.09925 .
## stateMO                        0.5773513  0.7995542   0.722  0.47024
## stateMS                        1.2594384  0.7374249   1.708  0.08766 .
## stateMT                        1.7668169  0.7226617   2.445  0.01449 *
## stateNC                        0.6220340  0.7582923   0.820  0.41204
## stateND                        0.1902519  0.7973475   0.239  0.81141
## stateNE                        0.2131982  0.8408324   0.254  0.79984
## stateNH                        1.1155165  0.7777771   1.434  0.15150
## stateNJ                        1.5534946  0.7163150   2.169  0.03010 *
## stateNM                        0.5397322  0.7964848   0.678  0.49800
## stateNV                        1.2397993  0.7311005   1.696  0.08992 .
## stateNY                        1.0925075  0.7286850   1.499  0.13380
## stateOH                        0.5775613  0.7573945   0.763  0.44572
## stateOK                        0.6913908  0.7701285   0.898  0.36931
## stateOR                        0.6672372  0.7466084   0.894  0.37149
## statePA                        1.2055945  0.7828900   1.540  0.12358
## stateRI                       -0.3446325  0.8513967  -0.405  0.68564
## stateSC                        1.7427931  0.7503179   2.323  0.02019 *
## stateSD                        0.8444380  0.7640027   1.105  0.26904
## stateTN                        0.3807643  0.8254919   0.461  0.64461
## stateTX                        1.6809839  0.7107994   2.365  0.01803 *
## stateUT                        1.2269628  0.7466953   1.643  0.10034
## stateVA                       -0.4142633  0.8578996  -0.483  0.62918
## stateVT                        0.0310499  0.8018079   0.039  0.96911
## stateWA                        1.4901744  0.7315178   2.037  0.04164 *
## stateWI                        0.3400039  0.7838353   0.434  0.66446
## stateWV                        0.5842009  0.7372192   0.792  0.42811
## stateWY                        0.2176439  0.7638271   0.285  0.77569
## international_planyes          2.1981789  0.1588259  13.840  < 2e-16 ***
## voice_mail_planyes            -1.2444991  0.4390683  -2.834  0.00459 **
## number_vmail_messages          0.0078703  0.0145466   0.541  0.58848
## total_day_minutes             -0.0043398  0.0022776  -1.905  0.05673 .
## total_day_calls                0.0032167  0.0029661   1.084  0.27815
## total_day_charge               0.0998499  0.0138183   7.226 4.98e-13 ***
## total_eve_minutes              0.0082540  0.0044861   1.840  0.06578 .
## total_eve_calls                0.0004211  0.0029638   0.142  0.88702
## total_eve_charge               0.0005291  0.0534358   0.010  0.99210
## total_night_minutes           -0.0490610  0.9270213  -0.053  0.95779
## total_night_calls             -0.0005854  0.0030206  -0.194  0.84635
## total_night_charge             1.1803957 20.5995133   0.057  0.95430
## total_intl_minutes            -2.6137646  5.6915749  -0.459  0.64607
## total_intl_calls              -0.0844451  0.0267577  -3.156  0.00160 **
## total_intl_charge              9.9766406 21.0784410   0.473  0.63599
## number_customer_service_calls  0.5412498  0.0421218  12.850  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 1954.4  on 3066  degrees of freedom
```

```
## AIC: 2088.4
##
## Number of Fisher Scoring iterations: 6
```

From the output of the churn model using stepwise regression on kNN imputed data, we see that the variables number_customer_service_calls, total_day_charge, international_planyes, total_intl_calls, and voice_mail_planyes are projected to have a likely impact on 'churn', with the variables total_eve_minutes and total_day_minutes tending towards significance. The output also points towards states that may experience a higher churn rate than others (shown by * as well as by .). We will refine this output in the following steps using direction, backward and both.

Using the 'backward' option, we proceed to identify the variables that have a significant impact in determining 'churn', using the kNN imputed values.

```
stepwise_knn_bkwd = step(churn_model_knn, direction = c("backward"), trace = F)
```

```
summary(stepwise_knn_bkwd)
```

```
##
## Call:
## glm(formula = churn ~ international_plan + voice_mail_plan +
##     total_day_minutes + total_day_charge + total_eve_minutes +
##     total_night_minutes + total_intl_calls + total_intl_charge +
##     number_customer_service_calls, family = "binomial", data = cdf_knn.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1208  -0.5192  -0.3379  -0.1962   3.1188
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -7.9855943  0.5199729 -15.358  < 2e-16 ***
## international_planyes         2.0556485  0.1509174  13.621  < 2e-16 ***
## voice_mail_planyes           -0.9762708  0.1498811  -6.514 7.34e-11 ***
## total_day_minutes            -0.0040278  0.0005937  -6.785 1.16e-11 ***
## total_day_charge              0.0973125  0.0075047  12.967  < 2e-16 ***
## total_eve_minutes             0.0075873  0.0011495   6.600 4.10e-11 ***
## total_night_minutes           0.0037548  0.0011446   3.281 0.001036 **
## total_intl_calls             -0.0880784  0.0259629  -3.392 0.000693 ***
## total_intl_charge             0.3109593  0.0771096   4.033 5.51e-05 ***
## number_customer_service_calls  0.5154167  0.0404079  12.755  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 2040.1  on 3123  degrees of freedom
## AIC: 2060.1
##
## Number of Fisher Scoring iterations: 6
```

Continuing to refine the model with stepwise regression on kNN imputed data, but including direction as 'backward', the model outputs the variables international_planyes, voice_mail_planyes,

total_day_minutes, total_day_charge, total_eve_minutes, total_night_minutes, total_intl_calls, total_intl_charge and number_customer_service_calls as being significant towards prediction of churn.

Continuing with using Stepwise Regression, but amending direction to 'both', we again proceed to identify the variables that may have a significant impact in determining 'churn', using the kNN imputed values. The outputs from both chunks - direction backward & both - is seen to be identical in all respects.

```
stepwise_knn_bth = step(churn_model_knn, direction = c("both"), trace = F)
```

```
summary(stepwise_knn_bth)
```

```
##
## Call:
## glm(formula = churn ~ international_plan + voice_mail_plan +
##     total_day_minutes + total_day_charge + total_eve_minutes +
##     total_night_minutes + total_intl_calls + total_intl_charge +
##     number_customer_service_calls, family = "binomial", data = cdf_knn.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1208  -0.5192  -0.3379  -0.1962   3.1188
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -7.9855943  0.5199729 -15.358  < 2e-16 ***
## international_planyes          2.0556485  0.1509174  13.621  < 2e-16 ***
## voice_mail_planyes            -0.9762708  0.1498811  -6.514 7.34e-11 ***
## total_day_minutes             -0.0040278  0.0005937  -6.785 1.16e-11 ***
## total_day_charge               0.0973125  0.0075047  12.967  < 2e-16 ***
## total_eve_minutes              0.0075873  0.0011495   6.600 4.10e-11 ***
## total_night_minutes            0.0037548  0.0011446   3.281 0.001036 **
## total_intl_calls              -0.0880784  0.0259629  -3.392 0.000693 ***
## total_intl_charge              0.3109593  0.0771096   4.033 5.51e-05 ***
## number_customer_service_calls  0.5154167  0.0404079  12.755  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 2040.1  on 3123  degrees of freedom
## AIC: 2060.1
##
## Number of Fisher Scoring iterations: 6
```

Continuing with stepwise regression on kNN imputed data but amending direction to 'both', the model outputs the variables international_planyes, voice_mail_planyes, total_day_minutes, total_day_charge, total_eve_minutes, total_night_minutes, total_intl_calls, total_intl_charge and number_customer_service_calls as being significant. We notice that these variables are the same as those identified in the 'backward' direction with AIC at 2060.2

Stepwise Regression using Randomforest Imputed Data

Using Stepwise Regression, we proceed to identify the variables that may have a significant impact in determining 'churn', using the RandomForest imputed values. The chunk also identifies states which may be impacted by churn.

```
churn_model_rf <- glm(churn~., data = cdf_rf.imputed, family = "binomial")
summary(churn_model_rf)
```

```
##
## Call:
## glm(formula = churn ~ ., family = "binomial", data = cdf_rf.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8820  -0.5078  -0.3117  -0.1657   3.0912
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -9.4554990  0.9786318  -9.662  < 2e-16 ***
## stateAL                        0.4134481  0.7643986   0.541  0.58859
## stateAR                        0.8341915  0.7631474   1.093  0.27435
## stateAZ                        0.1506813  0.8457557   0.178  0.85860
## stateCA                        1.9704300  0.7924840   2.486  0.01290 *
## stateCO                        0.5078103  0.7741309   0.656  0.51184
## stateCT                        1.0993060  0.7285668   1.509  0.13133
## stateDC                        0.8276082  0.8120796   1.019  0.30814
## stateDE                        0.7135016  0.7526869   0.948  0.34316
## stateFL                        0.4418847  0.7801966   0.566  0.57114
## stateGA                        0.7314209  0.7842448   0.933  0.35100
## stateHI                       -0.0791913  0.8986164  -0.088  0.92978
## stateIA                        0.1858423  0.9078355   0.205  0.83780
## stateID                        0.7641269  0.7611869   1.004  0.31544
## stateIL                       -0.1316705  0.8376467  -0.157  0.87509
## stateIN                        0.4671395  0.7563332   0.618  0.53681
## stateKS                        1.0784311  0.7330825   1.471  0.14127
## stateKY                        0.6942564  0.7837861   0.886  0.37574
## stateLA                        0.6619519  0.8415862   0.787  0.43154
## stateMA                        1.1181572  0.7500629   1.491  0.13603
## stateMD                        1.1703010  0.7193613   1.627  0.10377
## stateME                        1.2860667  0.7375708   1.744  0.08122 .
## stateMI                        1.5019456  0.7207923   2.084  0.03718 *
## stateMN                        1.2068186  0.7182933   1.680  0.09293 .
## stateMO                        0.5955881  0.7999537   0.745  0.45656
## stateMS                        1.2730609  0.7370657   1.727  0.08413 .
## stateMT                        1.7769402  0.7223915   2.460  0.01390 *
## stateNC                        0.6597943  0.7582153   0.870  0.38419
## stateND                        0.2001678  0.7984354   0.251  0.80205
## stateNE                        0.2294421  0.8406522   0.273  0.78490
## stateNH                        1.1185598  0.7771973   1.439  0.15009
## stateNJ                        1.5616437  0.7161463   2.181  0.02921 *
## stateNM                        0.5852898  0.7948466   0.736  0.46151
## stateNV                        1.3025374  0.7288160   1.787  0.07391 .
```

```
## stateNY                        1.1076335  0.7288270   1.520  0.12857
## stateOH                        0.5685214  0.7575902   0.750  0.45299
## stateOK                        0.7320202  0.7701576   0.950  0.34187
## stateOR                        0.6657258  0.7469576   0.891  0.37280
## statePA                        1.2125568  0.7832385   1.548  0.12159
## stateRI                       -0.2726158  0.8450475  -0.323  0.74700
## stateSC                        1.7401388  0.7506757   2.318  0.02044 *
## stateSD                        0.8596627  0.7642860   1.125  0.26068
## stateTN                        0.3835842  0.8274887   0.464  0.64297
## stateTX                        1.6826654  0.7111923   2.366  0.01798 *
## stateUT                        1.2227268  0.7462758   1.638  0.10133
## stateVA                       -0.3779475  0.8579123  -0.441  0.65954
## stateVT                        0.0893644  0.8007418   0.112  0.91114
## stateWA                        1.5119752  0.7316205   2.067  0.03877 *
## stateWI                        0.3295182  0.7861261   0.419  0.67509
## stateWV                        0.6125206  0.7375328   0.830  0.40626
## stateWY                        0.2447496  0.7642269   0.320  0.74877
## international_planyes          2.1927719  0.1586119  13.825  < 2e-16 ***
## voice_mail_planyes            -1.2177279  0.4384717  -2.777  0.00548 **
## number_vmail_messages          0.0070180  0.0145525   0.482  0.62962
## total_day_minutes             -0.0008638  0.0009782  -0.883  0.37718
## total_day_calls                0.0032504  0.0029652   1.096  0.27299
## total_day_charge               0.0812844  0.0081935   9.921  < 2e-16 ***
## total_eve_minutes              0.0013895  0.0019371   0.717  0.47319
## total_eve_calls                0.0004174  0.0029697   0.141  0.88822
## total_eve_charge               0.0798747  0.0259101   3.083  0.00205 **
## total_night_minutes            0.0479957  0.9273398   0.052  0.95872
## total_night_calls             -0.0008555  0.0030128  -0.284  0.77645
## total_night_charge            -0.9762981 20.6065569  -0.047  0.96221
## total_intl_minutes            -2.7388946  5.6905901  -0.481  0.63030
## total_intl_calls              -0.0837608  0.0267772  -3.128  0.00176 **
## total_intl_charge             10.4416753 21.0749019   0.495  0.62028
## number_customer_service_calls  0.5392867  0.0420255  12.832  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 1957.5  on 3066  degrees of freedom
## AIC: 2091.5
##
## Number of Fisher Scoring iterations: 6
```

From the output of the churn model using stepwise regression on RandomForest imputed data, we see that the 6 variables number_customer_service_calls, total_day_charge, international_planyes, total_intl_calls, voice_mail_planyes and total_eve_charge are projected to have a likely impact on 'churn'. The output also points towards states that may experience a higher churn rate than others (shown by * as well as by .). We will refine this output in the following steps using direction, backward and both.

Using the 'backward' option, we proceed to identify the variables that have a significant impact in determining 'churn', using the RandomForest imputed values.

```
stepwise_rf_bkwd = step(churn_model_rf, direction = c("backward"), trace = F)
```

```
summary(stepwise_rf_bkwd)
```

```
##
## Call:
## glm(formula = churn ~ international_plan + voice_mail_plan +
##     total_day_minutes + total_day_charge + total_eve_charge +
##     total_night_minutes + total_intl_calls + total_intl_charge +
##     number_customer_service_calls, family = "binomial", data = cdf_rf.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1122  -0.5194  -0.3408  -0.1982   3.1013
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -8.0604780  0.5295858 -15.220  < 2e-16 ***
## international_planyes          2.0565757  0.1506266  13.653  < 2e-16 ***
## voice_mail_planyes            -0.9704065  0.1496478  -6.485 8.90e-11 ***
## total_day_minutes             -0.0001865  0.0001001  -1.863 0.062460 .
## total_day_charge               0.0770958  0.0066029  11.676  < 2e-16 ***
## total_eve_charge               0.0887513  0.0137488   6.455 1.08e-10 ***
## total_night_minutes            0.0037442  0.0011436   3.274 0.001060 **
## total_intl_calls              -0.0871718  0.0259913  -3.354 0.000797 ***
## total_intl_charge              0.3126105  0.0770563   4.057 4.97e-05 ***
## number_customer_service_calls  0.5140789  0.0403362  12.745  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 2042.3  on 3123  degrees of freedom
## AIC: 2062.3
##
## Number of Fisher Scoring iterations: 6
```

Continuing with using Stepwise Regression and the RandomForest imputed values, but amending direction to 'both', we again proceed to identify the variables that may have a significant impact in determining 'churn'.

On comparison of the outputs from both chunks - direction backward & both - it is seen to be identical in all respects.

```
stepwise_rf_bth = step(churn_model_rf, direction = c("both"), trace = F)
```

```
summary(stepwise_rf_bth)
```

```
##
## Call:
## glm(formula = churn ~ international_plan + voice_mail_plan +
```

```
##     total_day_minutes + total_day_charge + total_eve_charge +
##     total_night_minutes + total_intl_calls + total_intl_charge +
##     number_customer_service_calls, family = "binomial", data = cdf_rf.imputed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1122  -0.5194  -0.3408  -0.1982   3.1013
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -8.0604780  0.5295858 -15.220  < 2e-16 ***
## international_planyes         2.0565757  0.1506266  13.653  < 2e-16 ***
## voice_mail_planyes           -0.9704065  0.1496478  -6.485 8.90e-11 ***
## total_day_minutes            -0.0001865  0.0001001  -1.863 0.062460 .
## total_day_charge              0.0770958  0.0066029  11.676  < 2e-16 ***
## total_eve_charge              0.0887513  0.0137488   6.455 1.08e-10 ***
## total_night_minutes           0.0037442  0.0011436   3.274 0.001060 **
## total_intl_calls             -0.0871718  0.0259913  -3.354 0.000797 ***
## total_intl_charge             0.3126105  0.0770563   4.057 4.97e-05 ***
## number_customer_service_calls  0.5140789  0.0403362  12.745  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.3  on 3132  degrees of freedom
## Residual deviance: 2042.3  on 3123  degrees of freedom
## AIC: 2062.3
##
## Number of Fisher Scoring iterations: 6
```

In general, the best model is the one with the lowest AIC possible in the logistic regression model with churn as the dependent variable. The final model with the most important variables in predicting churn were, in ascending order:

1. `total_day_minutes`
2. `total_night_minutes`
3. `total_intl_charge`
4. `total_eve_charge`
5. `voice_mail_plan`
6. `total_day_charge`
7. `number_customer_service_calls`
8. `international_plan`

In other words, `international_plan` was considered the best predictor of churn followed by `number_customer_service_call` etc. In backward, starting out with the full model, the single best predictor was `international_plan`. This procedure was used to help in the creation of a best predicted model for churn as the dependent variable.

The common variables in each of the four stepwise regressions are:

1. `international_plan`
2. `voice_mail_plan`
3. `total_day_minutes`
4. `total_day_charge`

5. total_night_minutes
6. total_intl_calls
7. total_intl_charge
8. number_customer_service_calls

The variables that are not common to the four models are:

1. total_eve_minutes
2. total_eve_charge

We suggest that these 10 variables be used in the next stage of the model building process.

Random forest significant attributes

```
international_planyes          2.1972515  0.1589164  13.826  < 2e-16 ***
voice_mail_planyes            -1.0275099  0.3676354  -2.795 0.005191 **
total_day_charge               0.0805890  0.0080508  10.010  < 2e-16 ***
total_eve_charge               0.0830480  0.0249265   3.332 0.000863 ***
total_intl_calls              -0.0837351  0.0267472  -3.131 0.001744 **
number_customer_service_calls  0.5395426  0.0420437  12.833  < 2e-16 ***
```

```
states (12)

stateCA                        1.9772622  0.7938041   2.491 0.012743 *
stateME                        1.2932594  0.7391097   1.750 0.080161 .
stateMI                        1.4919750  0.7224169   2.065 0.038899 *
stateMN                        1.2142000  0.7194116   1.688 0.091456 .
stateMS                        1.2761173  0.7383695   1.728 0.083936 .
stateMT                        1.7768485  0.7233884   2.456 0.014038 *
stateNJ                        1.5639891  0.7174847   2.180 0.029271 *
stateNV                        1.3044053  0.7301848   1.786 0.074034 .
stateSC                        1.7432299  0.7519276   2.318 0.020430 *
stateTX                        1.6774593  0.7127652   2.353 0.018600 *
stateUT                        1.2332761  0.7465346   1.652 0.098534 .
stateWA                        1.5031936  0.7340795   2.048 0.040587 *
```

Only RandomForest had an additional state MN as compared to KNN

KNN significant attributes

```
international_plan             2.205e+00  1.594e-01  13.835  < 2e-16 ***
voice_mail_plan               -1.252e+00  4.395e-01  -2.849  0.00438 **
total_day_minutes             -5.762e-03  2.240e-03  -2.572  0.01010 *
total_day_charge               1.074e-01  1.375e-02   7.813 5.58e-15 ***
total_eve_minutes              1.106e-02  4.403e-03   2.511  0.01204 *
total_intl_calls              -8.585e-02  2.679e-02  -3.205  0.00135 **
number_customer_service_calls  5.432e-01  4.221e-02  12.868  < 2e-16 ***
```

```
states(11)
stateCA                      1.956e+00  7.908e-01   2.473  0.01339 *
stateME                      1.256e+00  7.389e-01   1.700  0.08904 .
stateMI                      1.465e+00  7.217e-01   2.030  0.04234 *
stateMS                      1.253e+00  7.375e-01   1.699  0.08926 .
stateMT                      1.754e+00  7.235e-01   2.424  0.01535 *
stateNJ                      1.539e+00  7.166e-01   2.148  0.03173 *
stateNV                      1.217e+00  7.316e-01   1.663  0.09632 .
stateSC                      1.735e+00  7.498e-01   2.314  0.02064 *
stateTX                      1.674e+00  7.108e-01   2.355  0.01852 *
stateUT                      1.229e+00  7.463e-01   1.646  0.09968 .
stateWA                      1.471e+00  7.321e-01   2.009  0.04455 *
```

We note the following states for being important to churn, but not a good predictor of churn and therefore we chose to omit it from the model building phase.

```
stateCA
stateME
stateMI
stateMN
stateMS
stateMT
stateNJ
stateNV
stateSC
stateTX
stateUT
stateWA
```

Model Building

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Spliting Dataset into Training and Test

The dataset was split between two subgroups, training and test, to avoid any sense of biases and obtain better results. The approach gives us a chance to test the accuracy of the result before committing. For this train and test split, our group agreed on a seed (123) value. The major advantage of setting a seed is that it gives the same sequence of random numbers whenever you supply the same seed in the random number generator. It also improves reproducibility of our model training, and creates a constancy of results among the AUC.

```
set.seed(123)

rf_train_index <- create_data_partition(cdf_rf.imputed)
knn_train_index <- create_data_partition(cdf_knn.imputed)

train_df_knn <- cdf_knn.imputed[knn_train_index,]
test_df_knn <- cdf_knn.imputed[-knn_train_index,]
train_df_rf <- cdf_rf.imputed[rf_train_index,]
test_df_rf <- cdf_rf.imputed[-rf_train_index,]
```

Resampling

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
OSKNN <- upSample(x=train_df_knn[, -ncol(train_df_knn)],y = train_df_knn$churn)

OSRFT <- upSample(x=train_df_rf[, -ncol(train_df_rf)],y = train_df_rf$churn)
```

Building The Models

In this stage, we focus on the model building aspect of our report. The building of the model is used to to generate predictions. These predictions include the international_planyes, voice_mail_planyes, etc. To find out which model is better, we will compare the coefficients for both models (ModelRF and ModelKNN) listed below. When comparing against the Churn, as the dependent variable, the ModelRf and ModelKNN demostrate similar findings. It shows the international_planyes has a positive correlation in both models; The only parameters that show a negative attributes for both ModelRf and ModelKNN are voice_mail_planyes, total_eve_minutes and total_intl_calls. A possible;e exlanation for this development is that the more these variables increase, the more churn decreases. However, totl_eve_minutes is a poor predictor of churn. It is beneficial because we are trying to keep churn as small as possible in comparison to the others variables (i.e., international_planyes,total_day_charge,total_day_calls,total_eve_charge,total_night_minutes,total_intl_charge,number_custo
. These variables illustrate that an increase in these section would also cause an increase in churn.

```
modelRF <- glm(
  churn~international_plan +
    voice_mail_plan +
    total_day_charge +
    total_day_calls +
```

```
    total_eve_charge +
    total_eve_minutes +
    total_night_minutes +
    total_intl_charge +
    total_intl_calls ,
  data = OSRFT ,
  family = "binomial"
)

summary(modelRF)
```

```
##
## Call:
## glm(formula = churn ~ international_plan + voice_mail_plan +
##     total_day_charge + total_day_calls + total_eve_charge + total_eve_minutes +
##     total_night_minutes + total_intl_charge + total_intl_calls,
##     family = "binomial", data = OSRFT)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32144  -0.97651   0.00026   0.93185   2.20997
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -3.3706564  0.3405081  -9.899  < 2e-16 ***
## international_planyes  1.9292704  0.1107140  17.426  < 2e-16 ***
## voice_mail_planyes    -1.0653847  0.0925744 -11.508  < 2e-16 ***
## total_day_charge       0.0502866  0.0036885  13.633  < 2e-16 ***
## total_day_calls       -0.0025448  0.0018257  -1.394   0.1634
## total_eve_charge       0.0876824  0.0091660   9.566  < 2e-16 ***
## total_eve_minutes      0.0001142  0.0001210   0.944   0.3451
## total_night_minutes    0.0014703  0.0007714   1.906   0.0567 .
## total_intl_charge      0.1077659  0.0478760   2.251   0.0244 *
## total_intl_calls      -0.0650308  0.0152090  -4.276  1.9e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5198.6  on 3749  degrees of freedom
## Residual deviance: 4324.8  on 3740  degrees of freedom
## AIC: 4344.8
##
## Number of Fisher Scoring iterations: 4
```

```
modelKNN <- glm(
  Class~international_plan +
    voice_mail_plan +
    total_day_charge +
    total_day_calls +
    total_eve_charge +
    total_eve_minutes +
    total_night_minutes +
```

```
    total_intl_charge +
    total_intl_calls +
    number_customer_service_calls,
  data = OSKNN,
  family = "binomial"
)

summary(modelKNN)
```

```
##
## Call:
## glm(formula = Class ~ international_plan + voice_mail_plan +
##     total_day_charge + total_day_calls + total_eve_charge + total_eve_minutes +
##     total_night_minutes + total_intl_charge + total_intl_calls +
##     number_customer_service_calls, family = "binomial", data = OSKNN)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.98519  -0.75522  -0.02353   0.78085   2.48007
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -6.4322926  0.4144986 -15.518  < 2e-16 ***
## international_planyes          2.6540093  0.1264125  20.995  < 2e-16 ***
## voice_mail_planyes            -0.9611393  0.1037567  -9.263  < 2e-16 ***
## total_day_charge               0.0856523  0.0044931  19.063  < 2e-16 ***
## total_day_calls               -0.0024540  0.0020043  -1.224 0.220822
## total_eve_charge               0.0790083  0.0099767   7.919 2.39e-15 ***
## total_eve_minutes             -0.0005012  0.0001406  -3.566 0.000362 ***
## total_night_minutes            0.0015403  0.0008446   1.824 0.068187 .
## total_intl_charge              0.2958341  0.0551486   5.364 8.13e-08 ***
## total_intl_calls              -0.0102500  0.0158058  -0.648 0.516663
## number_customer_service_calls  0.7031343  0.0306411  22.947  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5140.4  on 3707  degrees of freedom
## Residual deviance: 3609.5  on 3697  degrees of freedom
## AIC: 3631.5
##
## Number of Fisher Scoring iterations: 5
```

##Predicting & Evaluating Accuracy

We are comparing the prediction and accuracy of our two models, "ModelRF" and "ModelKNN", which are listed below. However, before we move on to this analysis, we need to establish the meaning of the AUC. The AUC is the area under the curve, is a measure of accuracy fit of our model, which is calculated into a single variable to determine the better accuracy results.

Listed below, it is observed that the "ModelRF" has an 82% accuracy rate in comparison to the "ModelKNN", which has an 80% accuracy rate. This means that the "ModelRF" would be the ideal choice between both models. One can argue that both models deliver acceptable AUC metrics.

#Evaluating The Accuracy of `modelRF` The "ModelRF" has a AUC of 82% of accuracy.

```
pred_churn_rf <- predict(modelRF, newdata = test_df_rf, type = "response")
roc_out_rf <- roc(test_df_rf$churn, pred_churn_rf)
```

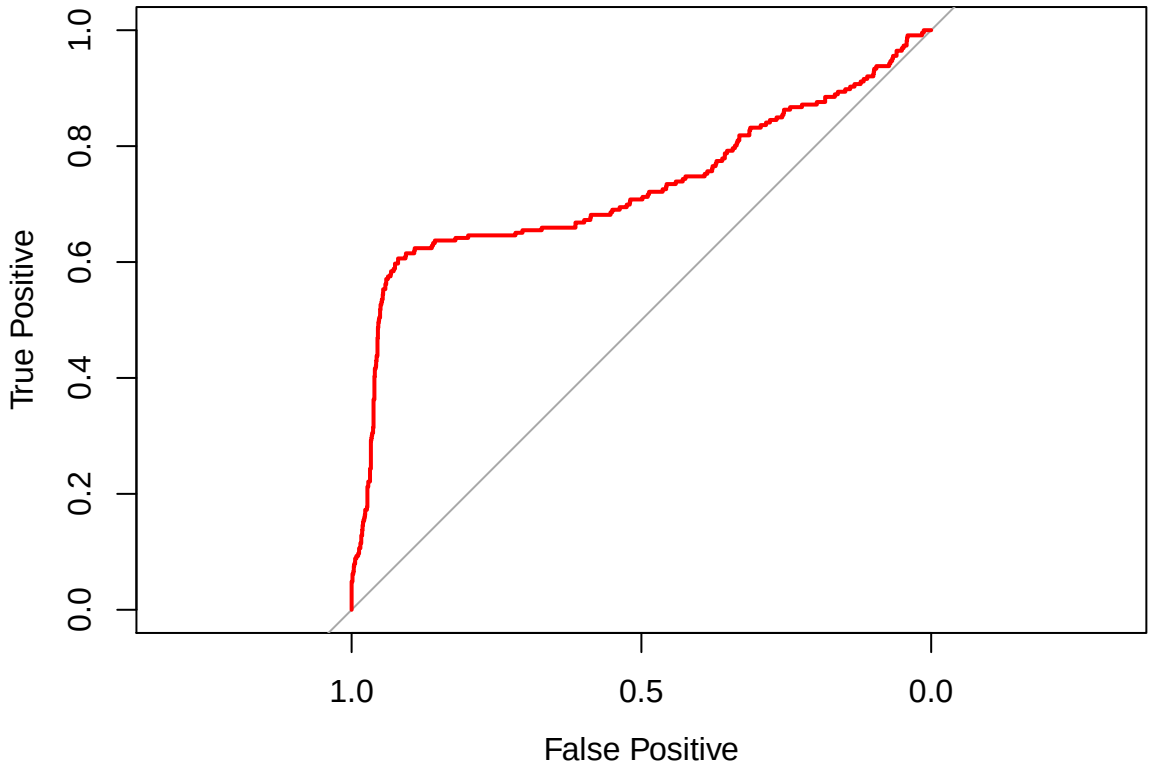
```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
roc_out_rf
```

```
##
## Call:
## roc.default(response = test_df_rf$churn, predictor = pred_churn_rf)
##
## Data: pred_churn_rf in 1323 controls (test_df_rf$churn no) < 226 cases (test_df_rf$churn yes).
## Area under the curve: 0.7277
```

```
plot(roc_out_rf, col = "red", xlab = "False Positive", ylab = "True Positive")
```



The ROC curve is an evaluation method we used to assess the efficacy of binary characteristic algorithm, as well as choose the optimal threshold based on our tolerance for false negatives and desire for true positives. Here, we have a curve that shows a relatively good result based on its usefulness as predictor. As displayed on the graph, the x axis shows the False Positive and the y axis shows the True Positive. The area under the curve is used as a singular measure for assessing the usefulness of a classifier. For a perfect classifier the area under the ROC curve would be 1. Therefore, the higher the AUC we have greater confidence in the predictive nature of our model.

10.2 Evaluating The Accuracy of `modelKNN`

The "ModelKNN" has a AUC of 80%

```
pred_churn_knn <- predict(modelKNN, newdata = test_df_knn, type = "response")
roc_out_knn <- roc(test_df_knn$churn, pred_churn_knn)
```

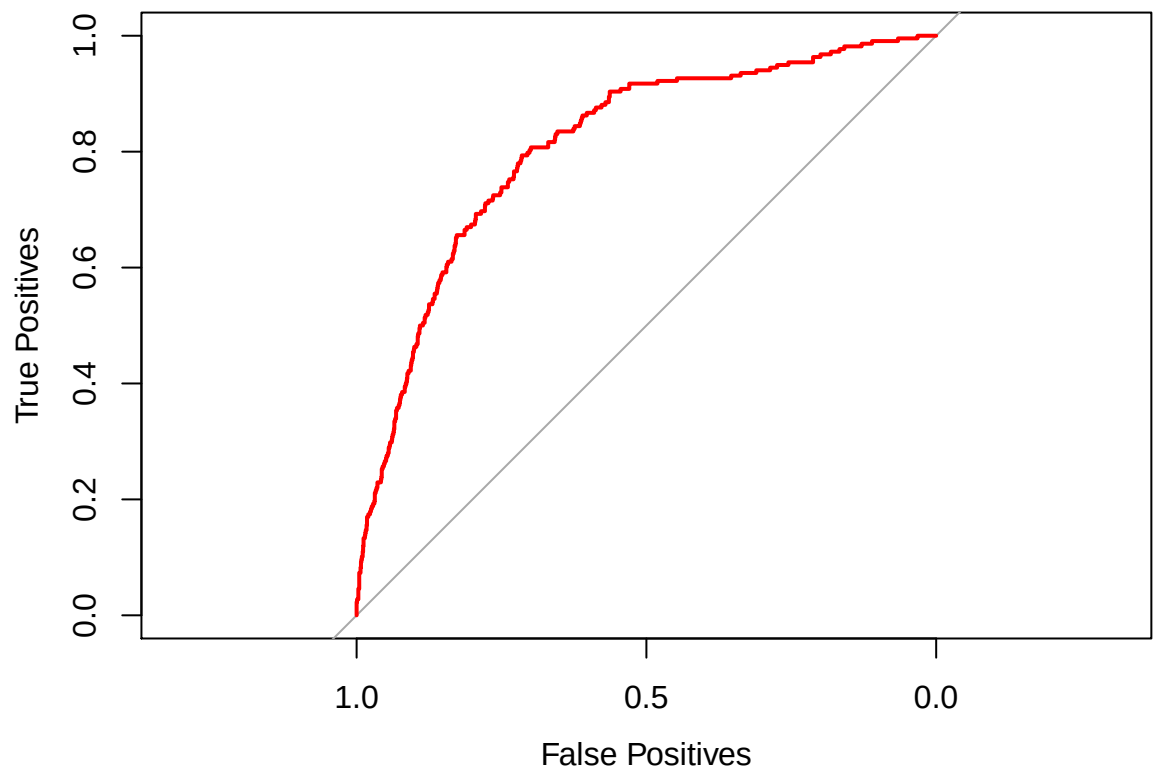
```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
roc_out_knn
```

```
##
## Call:
## roc.default(response = test_df_knn$churn, predictor = pred_churn_knn)
##
## Data: pred_churn_knn in 1337 controls (test_df_knn$churn no) < 218 cases (test_df_knn$churn yes).
## Area under the curve: 0.8097
```

```
plot(roc_out_knn, col = "red", xlab = "False Positives", ylab = "True Positives")
```



11. Evaluating The Winining Model

```
predicted_churn_status <- as.factor(pred_churn_rf > 0.2)
levels(predicted_churn_status)  <- list(no = "FALSE", yes = "TRUE")
confusion_matrix <- table(predicted_churn_status, actual_churn_status = test_df_rf$churn)

confusion_matrix
```

```
##                       actual_churn_status
## predicted_churn_status   no  yes
##                    no   204   24
##                    yes 1119  202
```

The group reached a consensus on the threshold value of 0.2 to use for our model. **0.2** provides the best confusion matrix. Looking at our prediction churn and our actual churn status findings, we found out our misclassification rate to be:

(186 + 81)/1547 errors - 1.73% misclassification rate, a relatively low rate.

Model Predicition

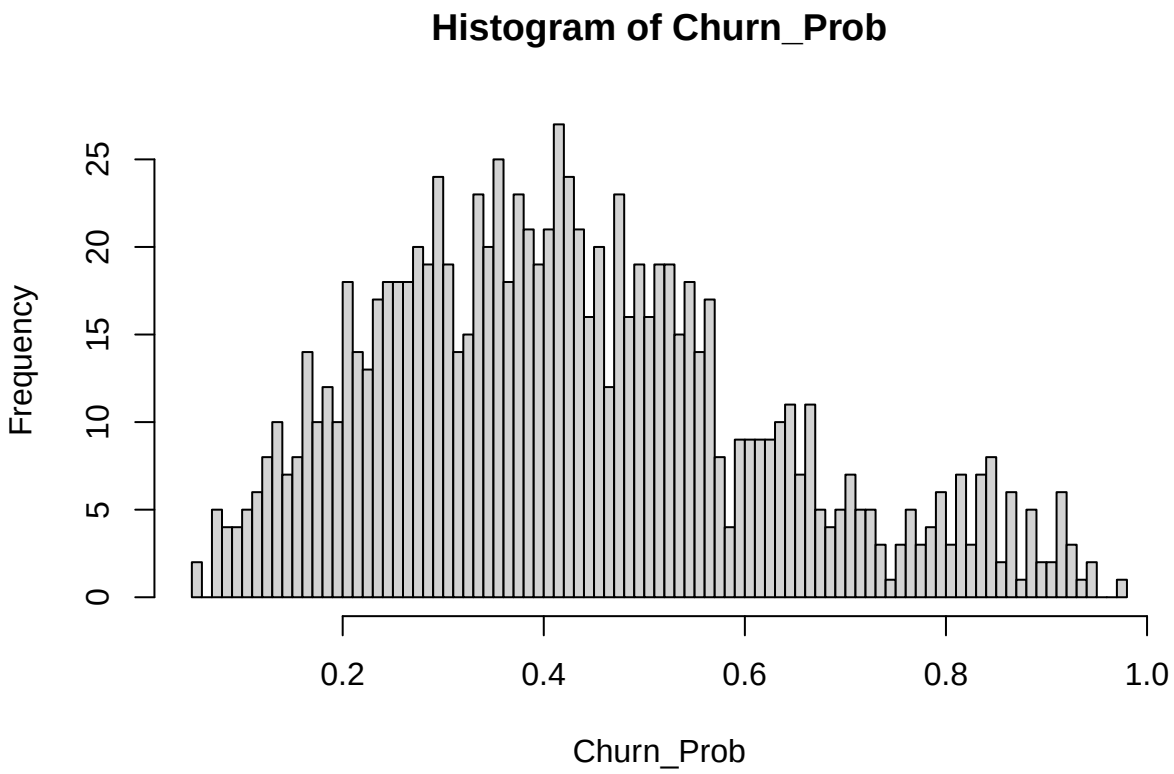
```
library("corrplot")
```

```
## corrplot 0.92 loaded
```

```
load("./data/Customers_To_Predict.Rdata")
```

12. Making The Prediction

```
Churn_Prob <- predict(modelRF, newdata = Customers_To_Predict, type = "response")

hist(Churn_Prob, 100)
```



`Churn_Prob` contains all the probabilities (from 0 to 1) that a customer from a pool of 1000 customers will churn or not. The histogram above reveals the distribution of the probabilities of churn across the 1000 customers we predicted. The histogram tells us that most customers stayed (i.e. they did not churn). Since the frequency of a customer not churning was higher between the probabilities of 0.0 to 0.5, with the larger subset between 0.0 to 0.2 (our group concluded that 0.5 was threshold for a customer churning or not). Previous research [^1] done on churn rate for wireless carriers suggested that the ideal churn rate was between 1.9 to 2.0.

Another researcher further indicates that a good churn rate is between 5% to 7% annual or 0.42% - 0.58% monthly. This means that companies with acceptable churn only loose 1 out of every 200 customers per month [^2].

Using the threshold (cutoff) of **0.2** for churning that was concluded in the model building stage, we obtain the "yes" and "no" churn responses for the `Customers_To_Predict` dataframe.

```
churn <- rep("no", nrow(Customers_To_Predict))
churn[Churn_Prob > 0.2] = "yes"

Customers_To_Predict$churn <- as.factor(churn) # Assign the responses into a variable called churn in t
```

Insights from Prediction

Churn Rate per State

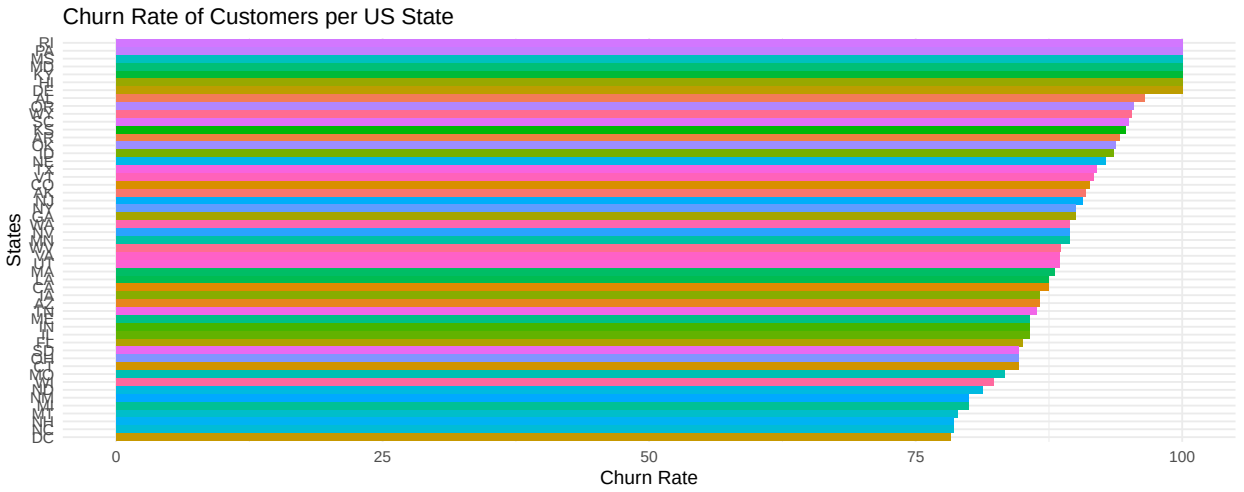
Given that we know the customers that churned or not, it would be advantageous to visualize the churn rate of customers in each state represented in `Customers_To_Predict` dataframe.

```
calc.churn_rate <- function(churn) {
  count_churn <- function(value) {
    return (churn %>%
              subset(churn == value) %>%
              length())
  }

  num_yes <- count_churn("yes")

  return(num_yes/length(churn) * 100)
}

state_churn_rate <- Customers_To_Predict %>%
  select(state, churn) %>%
  group_by(state) %>%
  summarise(churn_rate = calc.churn_rate(churn))

head(state_churn_rate)
```

```
## # A tibble: 6 x 2
##   state churn_rate
##   <fct>      <dbl>
## 1 AK          90.9
## 2 AL          96.4
## 3 AR          94.1
## 4 AZ          86.7
## 5 CA          87.5
## 6 CO          91.3
```

```
ggplot(state_churn_rate, aes(x = reorder(state, churn_rate),  y = churn_rate, fill = state)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  theme_minimal() +
  guides(fill = F) +
  labs(x = "States", y = "Churn Rate", title = "Churn Rate of Customers per US State")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



In terms of population, the states with the highest number were CA, TX, NY. The states with the next highest included OH, PA, IL, AZ, and WA.

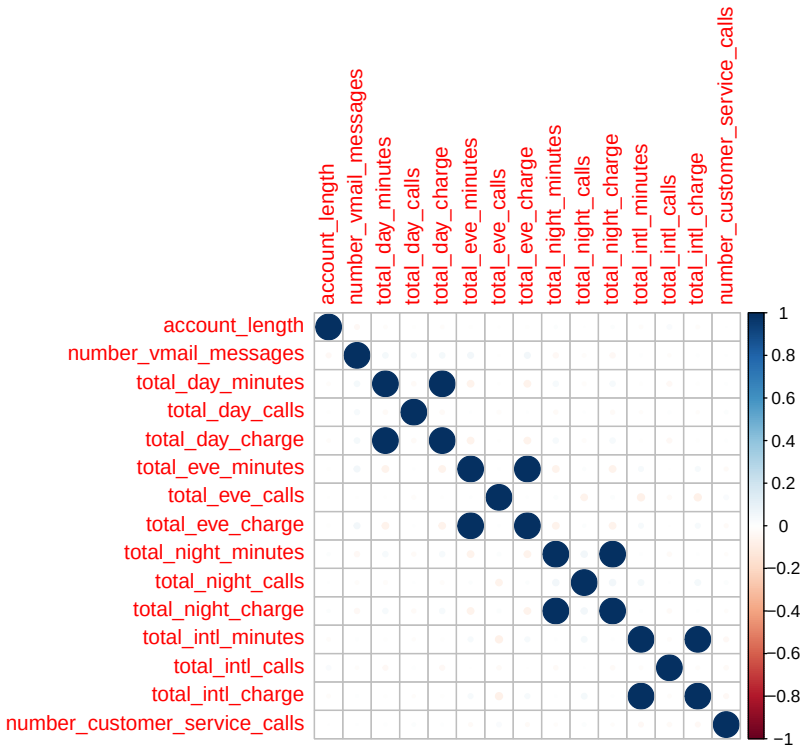
There seems to be a direct, but not perfect, positive correlation between churn rates and population density. For example, the states typically with the highest churn rates where WA, UT, IL, OK, NJ, and NY. Those states with moderate churn included TX, MI, IA, IN, NV, MS, OK, and ME.

From of the stepwise classification analysis, with churn as the dependent variable, we found that the following states from the initial model building were of interest (i.e., significant contribution in predicting churn at the .05 level or less): CA, ME, MI, MN, MS, MT, NJ, NV, SC, TX, UT, and WA. Note that the states ME, MS, NV, and UT were theoretically, not statistically, significant at the 0.05 but were approaching .05. In comparing the states with relatively higher population and churn rates (i.e., comparing the lists of moderate to highest for both), included CA, TX, and NY. Other states of interest include OH, PA, IL, AZ, IN, OK, ME, IL, and WA.

#How Predictors Affect Churn

#Correlation of Numeric Predictors

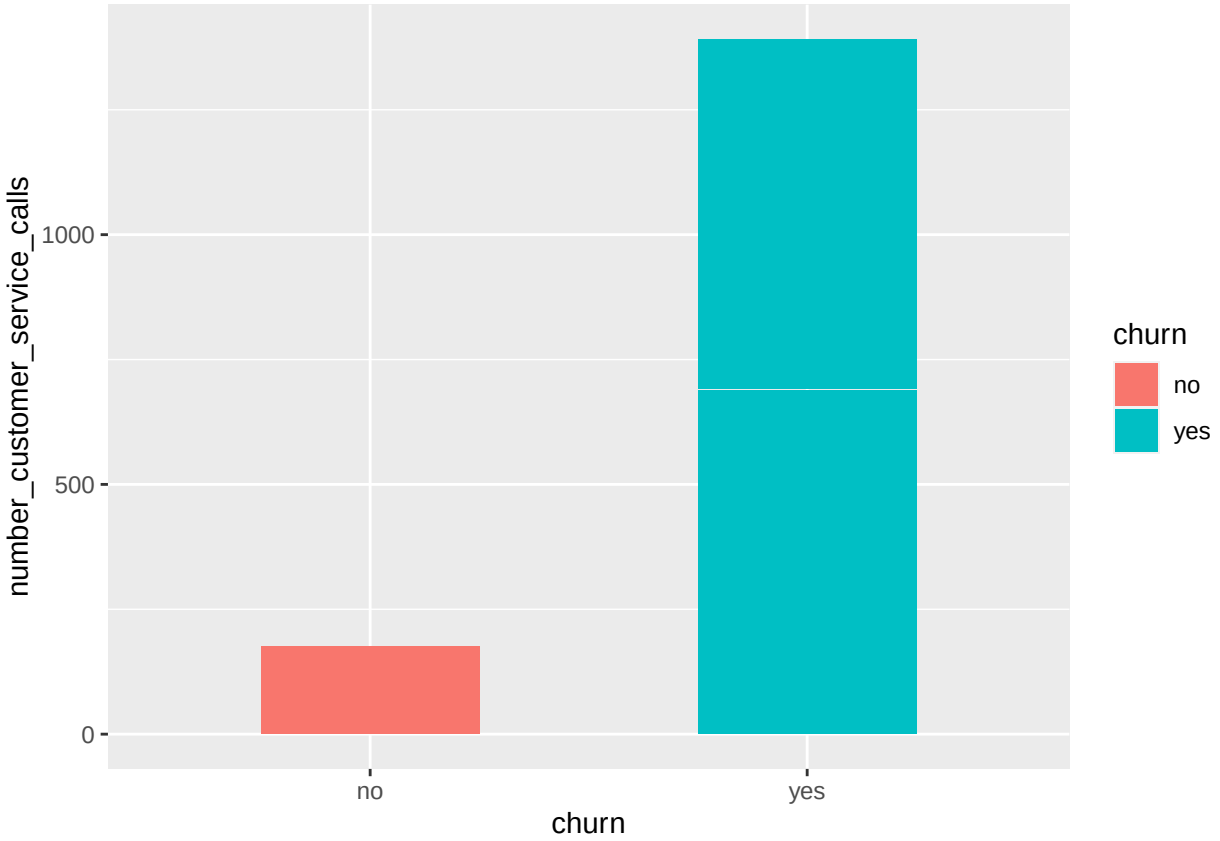
```
Customers_To_Predict %>%
  select_if(.predicate = function(x) !is.factor(x)) %>%
  cor() %>%
  corrplot()
```



This correlation plot simply tells us that the more time spent calling (day, evening, or night), the more the charge. It is positive correlation between minutes spent in a call (day, evening, or night) and charges incurred (day, evening, or night). The negative correlations are very minuscule due to faded red regions present.

How does Number of Service Calls Affect Churn

```
ggplot(Customers_To_Predict, aes(x = churn, y = number_customer_service_calls)) +
  geom_bar(stat='identity', aes(fill = churn), width=.5)
```



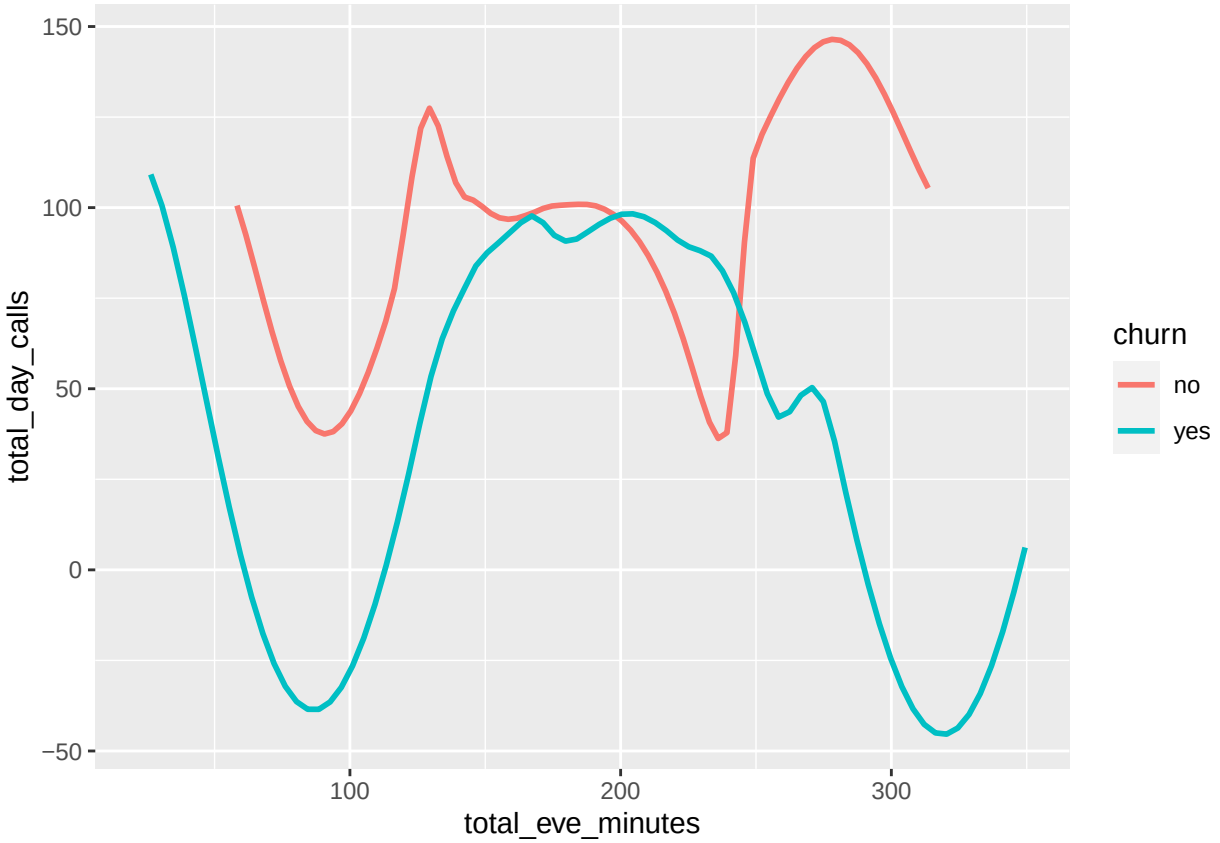
The number of service calls has a direct correlation with churn. It can be said that customers with more number of service calls are satisfied with the customer service provided by ABC wireless and choose not to churn.

On the other hand, customers with lower number of service calls are not happy with the resolutions of their issues and become more likely to churn.

We recommend that ABC Wireless improves on their customer service call center and keep in regular contact with their customers in order to improve their retention rate of existing customers.

#How does Total Day Calls and Total Day Charges Affect Churn

```
ggplot(data = Customers_To_Predict, aes(total_eve_minutes, total_day_calls, color = churn)) +
  geom_smooth(method = "loess", se = FALSE, formula = y ~ poly(x, 2))
```

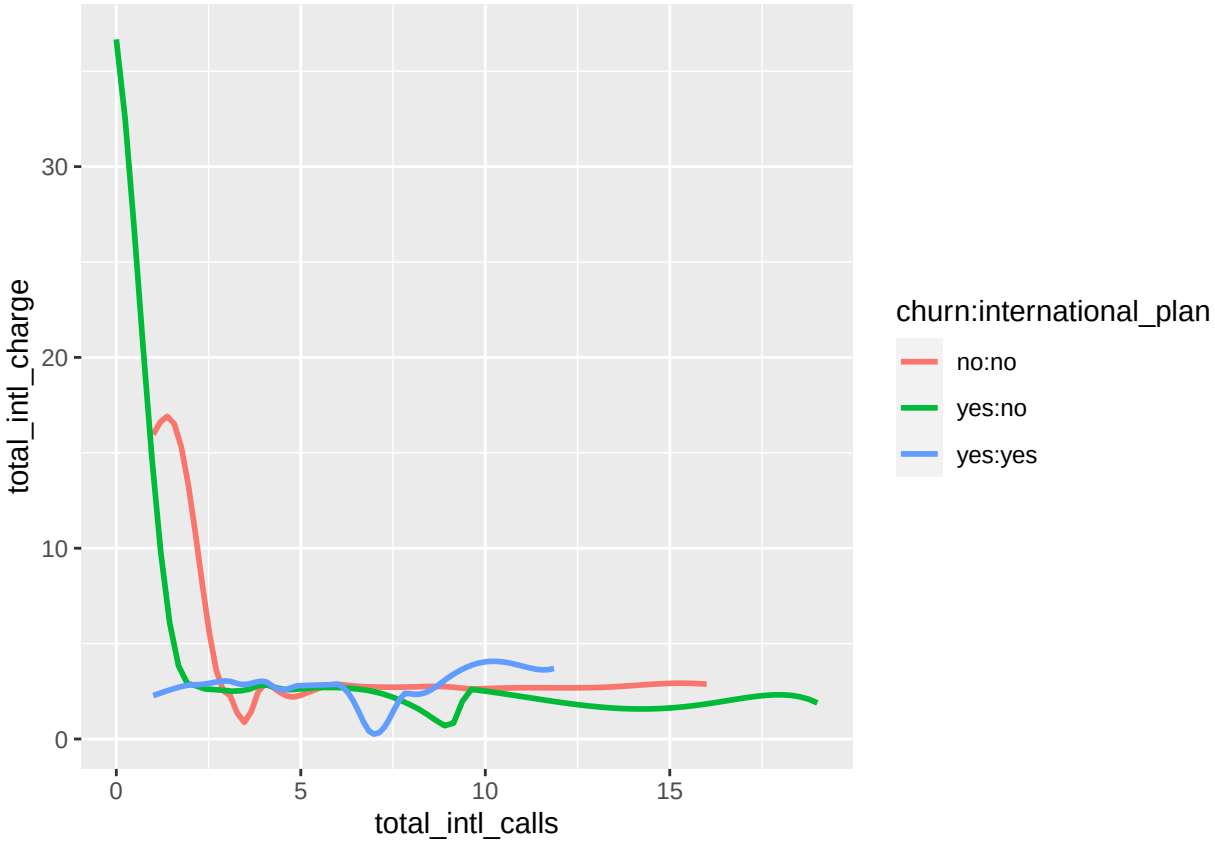


As shown in the graph there is a dipolar relationship; The people who did not churn were categorized by low total evening minutes and lower total day calls, resulting in higher totals and evening minutes and day calls.

For those who churned the opposite was true. We recommend that ABC wireless offers more competitive packages that take advantage of these dipolar relationships

Total International Calls and Total International Charges Affect Churn

```
ggplot(data = Customers_To_Predict, aes(total_intl_calls, total_intl_charge, color = churn:internationa
  geom_smooth(method = "loess", se = FALSE, formula = y ~ poly(x, 2))
```



For those who do not churn and do not have international plan, they still make the most total international calls with initially the highest total international charges (we suspect these customers to be new ones). There appears to be a middle range between 3 and 8 on the x-axis that is the same for all users, for those that did churn and did not have an international plan (green curve) had higher total international charges.

We recommend that ABC wireless should promote their international plans more competitively in order to reduce churn.

We focused on only the numerical variables `total_intl_charge`, `total_intl_calls`, `total_eve_minutes`, `total_eve_charge` and categorical variable `international_plan` when discussing their affect on `churn` due to time constraint and their sign and size of their coefficients and statistical significance.