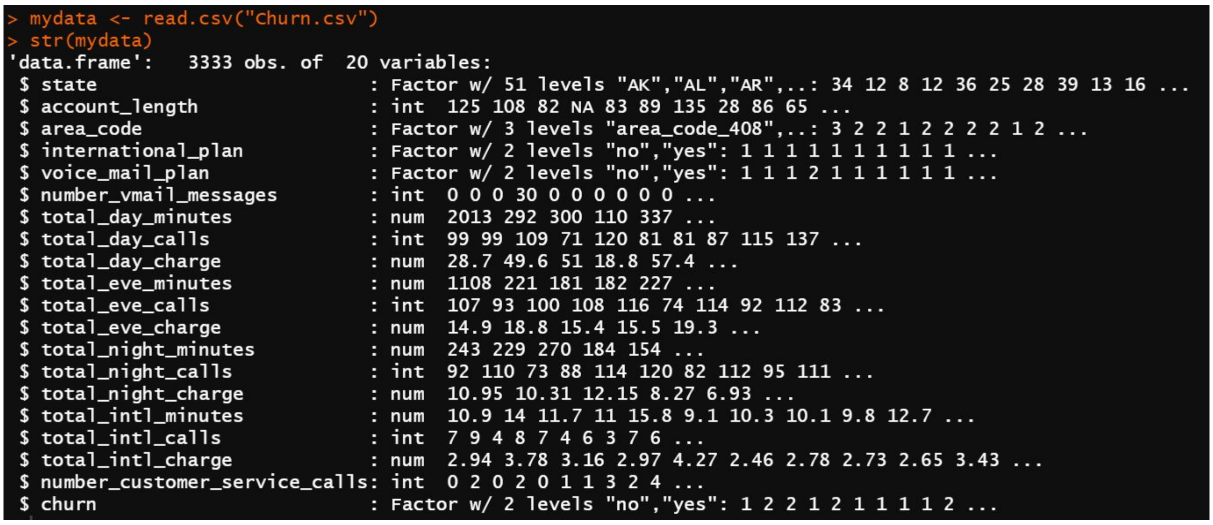
Group 1		
Members	**Model Selection**	**Others**
Casey Greenwald	Neural Networks	Preprocessing data
Jinping Guo	Logistic Regression	Preprocessing data
Molly Friedl	Random Forests	

For the preprocessing data stage, Casey Greenwald and Jinping Guo discussed and create tidy data for the group. Then, each member start to build our own model, Casey Greenwald chose the neural networks model, Jinping Guo worked on the logistic regression model, and Molly Friedl built the random forest model.

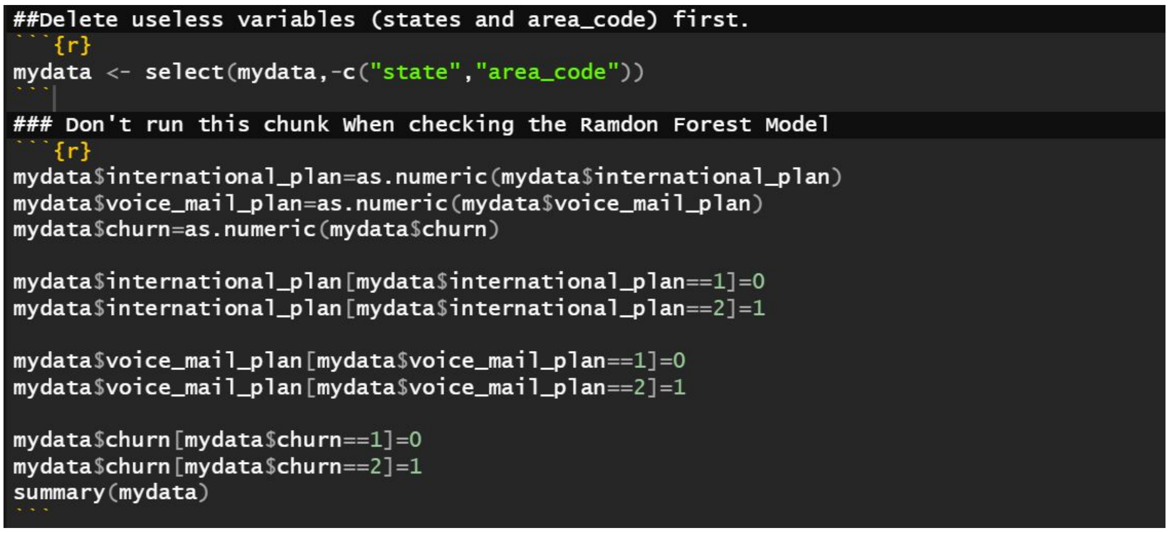
The Objective of our project was to try and accurately make a prediction about customer turnover (churn rate) for a telecommunications company. We did this by taking actual data from the company, processing and scaling it and producing a data set that trained our models to accurately predict the outcome of any given customer.

For our pre-processing methods, the first thing we did was load our data into R. We use the str() function to check the structure of each variables. There are 20 variables in this data, 16 are numeric variables and 4 are factor variables.



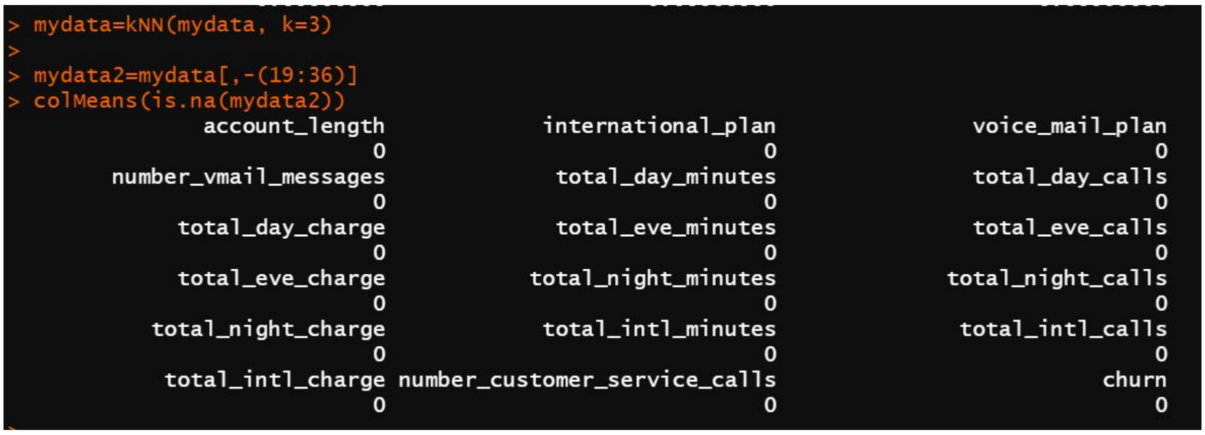
```
> mydata <- read.csv("Churn.csv")
> str(mydata)
'data.frame':   3333 obs. of  20 variables:
 $ state                        : Factor w/ 51 levels "AK","AL","AR",..: 34 12 8 12 36 25 28 39 13 16 ...
 $ account_length               : int  125 108 82 NA 83 89 135 28 86 65 ...
 $ area_code                    : Factor w/ 3 levels "area_code_408",..: 3 2 2 1 2 2 2 2 1 2 ...
 $ international_plan           : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ voice_mail_plan              : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
 $ number_vmail_messages        : int  0 0 0 30 0 0 0 0 0 0 ...
 $ total_day_minutes            : num  2013 292 300 110 337 ...
 $ total_day_calls              : int  99 99 109 71 120 81 81 87 115 137 ...
 $ total_day_charge             : num  28.7 49.6 51 18.8 57.4 ...
 $ total_eve_minutes            : num  1108 221 181 182 227 ...
 $ total_eve_calls              : int  107 93 100 108 116 74 114 92 112 83 ...
 $ total_eve_charge             : num  14.9 18.8 15.4 15.5 19.3 ...
 $ total_night_minutes          : num  243 229 270 184 154 ...
 $ total_night_calls            : int  92 110 73 88 114 120 82 112 95 111 ...
 $ total_night_charge           : num  10.95 10.31 12.15 8.27 6.93 ...
 $ total_intl_minutes           : num  10.9 14 11.7 11 15.8 9.1 10.3 10.1 9.8 12.7 ...
 $ total_intl_calls             : int  7 9 4 8 7 4 6 3 7 6 ...
 $ total_intl_charge            : num  2.94 3.78 3.16 2.97 4.27 2.46 2.78 2.73 2.65 3.43 ...
 $ number_customer_service_calls: int  0 2 0 2 0 1 1 3 2 4 ...
 $ churn                        : Factor w/ 2 levels "no","yes": 1 2 2 1 2 1 1 1 1 2 ...
```

Then we remove variables that were not relevant or useful to our predictions, State and Area Code, these only have to do with the geography of the customer. The next thing we did was transform some of our categorical variables into numerical ones as the neural networks and logistic regression models require this however for our random forest models we skip this step. Next, we use Colemans(is.na) function to detect missing value, then, handheld missing



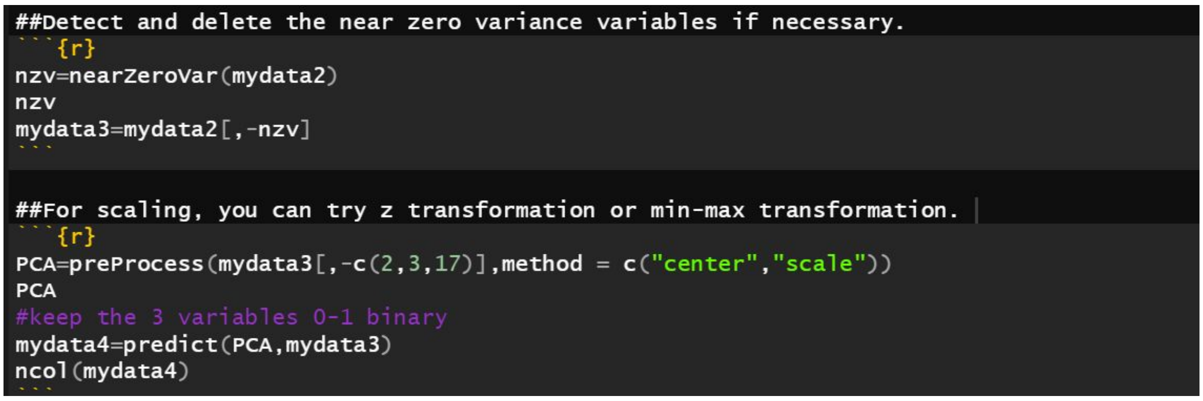
~~~
##Delete useless variables (states and area_code) first.
```{r}
mydata <- select(mydata,-c("state","area_code"))
```
### Don't run this chunk when checking the Ramdon Forest Model
```{r}
mydata$international_plan=as.numeric(mydata$international_plan)
mydata$voice_mail_plan=as.numeric(mydata$voice_mail_plan)
mydata$churn=as.numeric(mydata$churn)

mydata$international_plan[mydata$international_plan==1]=0
mydata$international_plan[mydata$international_plan==2]=1

mydata$voice_mail_plan[mydata$voice_mail_plan==1]=0
mydata$voice_mail_plan[mydata$voice_mail_plan==2]=1

mydata$churn[mydata$churn==1]=0
mydata$churn[mydata$churn==2]=1
summary(mydata)
```
~~~

value imputation. Originally we tried using a few different imputation methods however we ended up using KNN imputation as it was the most user friendly and accurate. After using KNN imputation, we deleted the column from 19 to 36 because this method generated extra useless 18 columns automatically and it is clear to see there are no missing values now. We started tuning with k=5 but we changed it to k=3 for higher accuracy. After this, we deleted the near zero variance variables (NZV) and used Z transformation to scale our data. After our

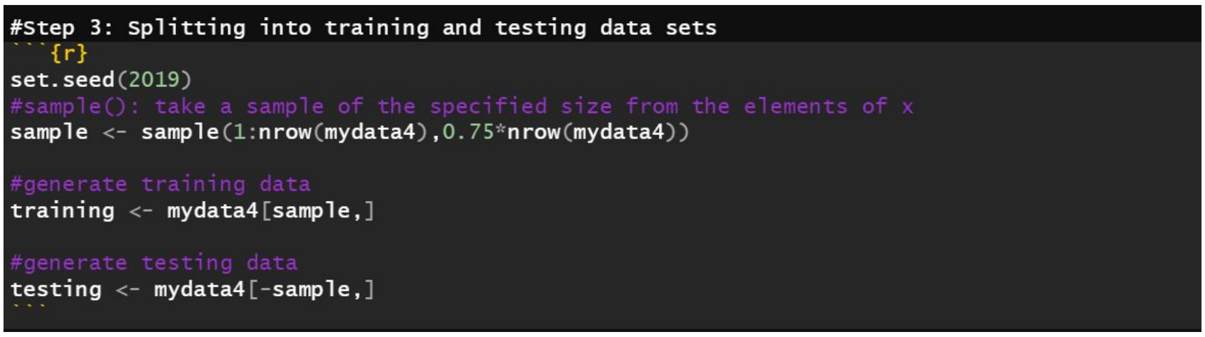


```
> mydata=kNN(mydata, k=3)
>
> mydata2=mydata[,-(19:36)]
> colMeans(is.na(mydata2))
             account_length            international_plan               voice_mail_plan
                          0                             0                             0
       number_vmail_messages             total_day_minutes               total_day_calls
                          0                             0                             0
            total_day_charge             total_eve_minutes               total_eve_calls
                          0                             0                             0
            total_eve_charge           total_night_minutes             total_night_calls
                          0                             0                             0
          total_night_charge             total_intl_minutes              total_intl_calls
                          0                             0                             0
           total_intl_charge number_customer_service_calls                         churn
                          0                             0                             0
```

data was cleaned we split it into testing and training data sets and worked separately on our own models sending it back and forth to each other for peer review and revision until we were satisfied.

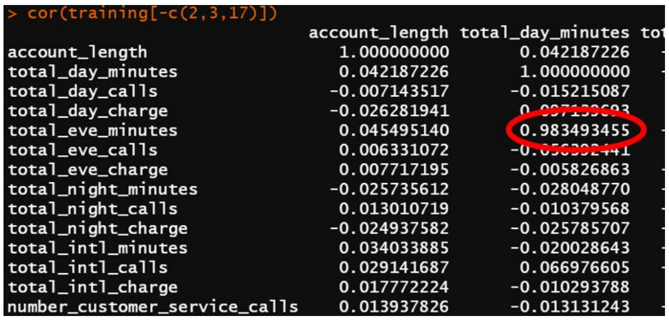


~~~
##Detect and delete the near zero variance variables if necessary.
```{r}
nzv=nearZeroVar(mydata2)
nzv
mydata3=mydata2[,-nzv]
~~~

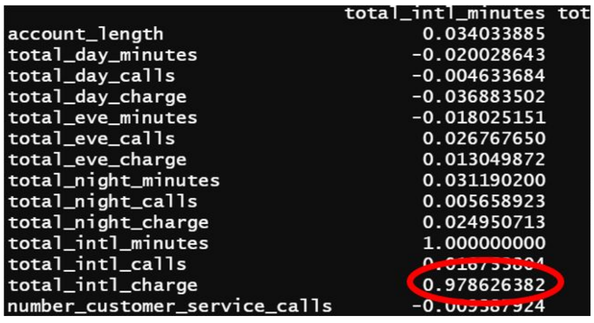
##For scaling, you can try z transformation or min-max transformation.
```{r}
PCA=preProcess(mydata3[,-c(2,3,17)],method = c("center","scale"))
PCA
#keep the 3 variables 0-1 binary
mydata4=predict(PCA,mydata3)
ncol(mydata4)
```
```



```
#Step 3: Splitting into training and testing data sets
```{r}
set.seed(2019)
#sample(): take a sample of the specified size from the elements of x
sample <- sample(1:nrow(mydata4),0.75*nrow(mydata4))

#generate training data
training <- mydata4[sample,]

#generate testing data
testing <- mydata4[-sample,]
```
```
```

For the neural network sections I (Casey) worked on it. For me the hardest part was getting all the codes written and getting them to work properly with one another. I had a couple packages conflicting with each other but eventually we got it sorted out. I did many tests with various numbers of hidden layers and with nodes however being on my laptop not alot of processing power was available to me and I was limited to two layers. Originally I had wanted to do three layers with something lik 5,9,3 however I found working with two layers it was more beneficial to go high then low so I started with 8,3 then we settled on 6,3. although the accuracy changes each time you run the chunk it was hovering around 94% with the highest time coming in at 95.2%.

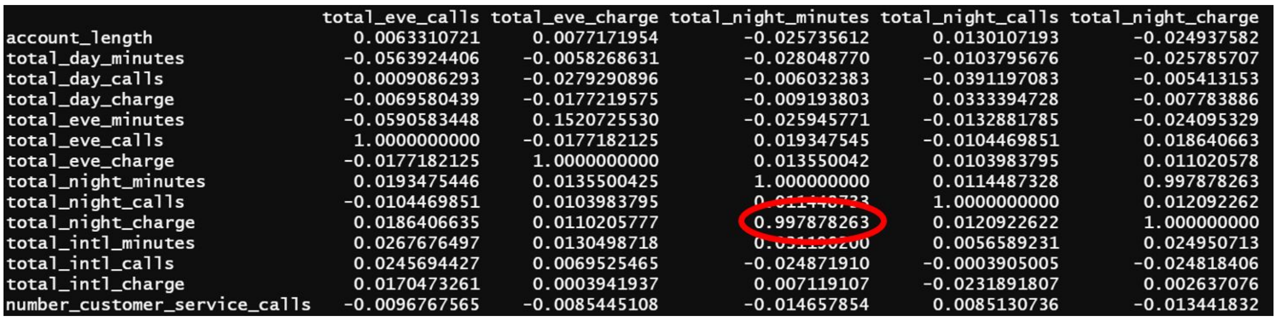
For logistic regression, I checked the multicollinearity of explanatory variables first since it could exist multicollinearity if there are two variables correlation coefficients is above 0.9. If there is multicollinearity in the data that will influence the accuracy of model. We can



```
> cor(training[-c(2,3,17)])
                             account_length total_day_minutes tot
account_length                  1.000000000       0.042187226
total_day_minutes               0.042187226       1.000000000
total_day_calls                -0.007143517      -0.015215087
total_day_charge               -0.026281941       [illegible]
total_eve_minutes               0.045495140       0.983493455
total_eve_calls                 0.006331072       [illegible]
total_eve_charge                0.007717195      -0.005826863
total_night_minutes            -0.025735612      -0.028048770
total_night_calls               0.013010719      -0.010379568
total_night_charge             -0.024937582      -0.025785707
total_intl_minutes              0.034033885      -0.020028643
total_intl_calls                0.029141687       0.066976605
total_intl_charge               0.017772224      -0.010293788
number_customer_service_calls   0.013937826      -0.013131243
```

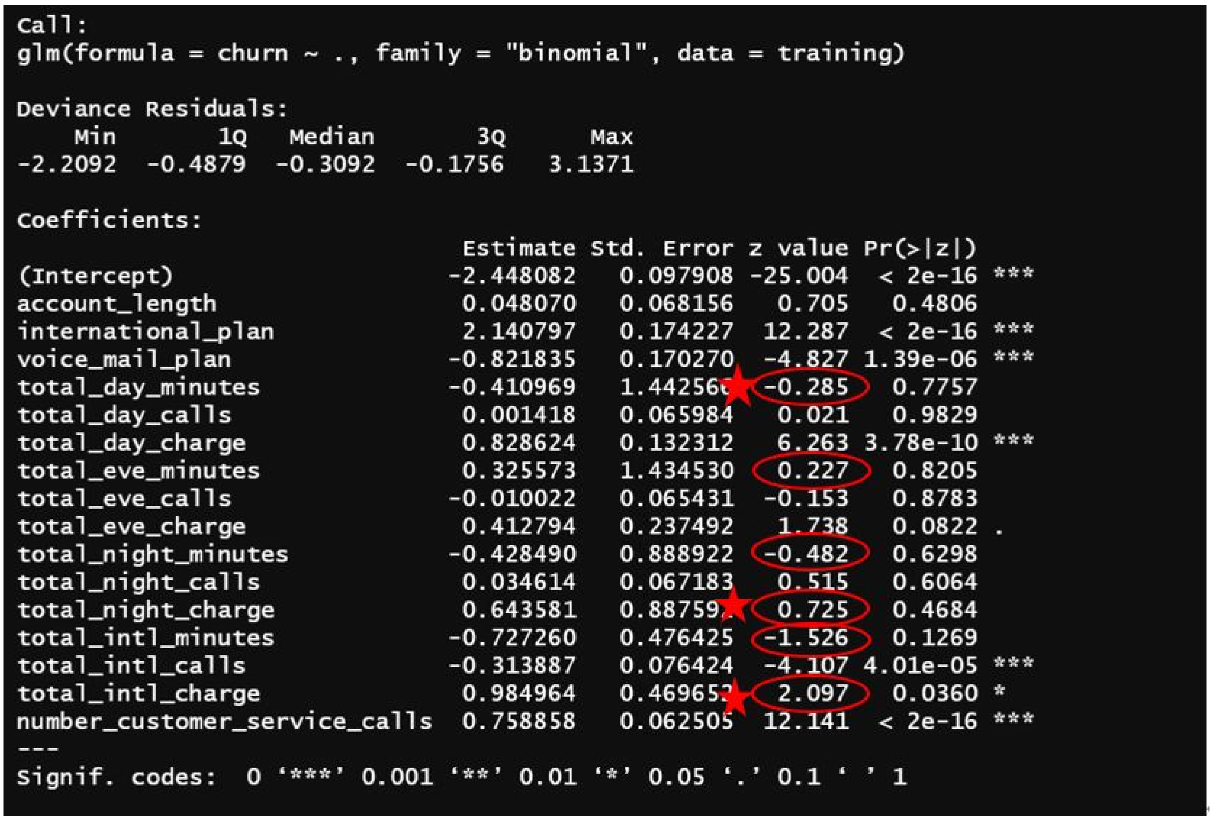


```
                              total_intl_minutes tot
account_length                       0.034033885
total_day_minutes                   -0.020028643
total_day_calls                     -0.004633684
total_day_charge                    -0.036883502
total_eve_minutes                   -0.018025151
total_eve_calls                      0.026767650
total_eve_charge                     0.013049872
total_night_minutes                  0.031190200
total_night_calls                    0.005658923
total_night_charge                   0.024950713
total_intl_minutes                   1.000000000
total_intl_calls                     0.016733804
total_intl_charge                    0.978626382
number_customer_service_calls       -0.009387924
```



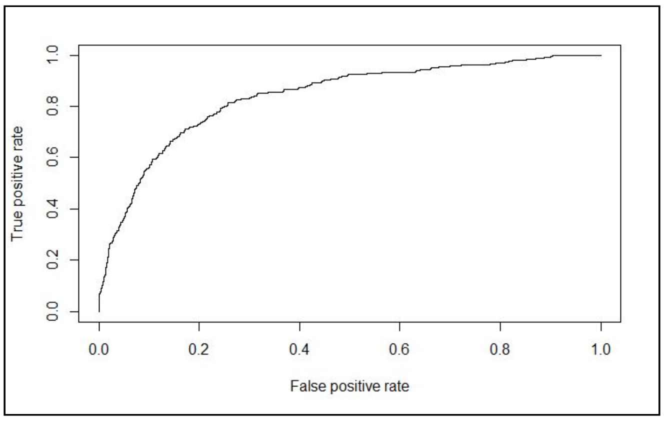
```
                              total_eve_calls total_eve_charge total_night_minutes total_night_calls total_night_charge
account_length                   0.0063310721     0.0077171954        -0.025735612      0.0130107193       -0.024937582
total_day_minutes               -0.0563924406    -0.0058268631        -0.028048770     -0.0103795676       -0.025785707
total_day_calls                  0.0009086293    -0.0279290896        -0.006032383     -0.0391197083       -0.005413153
total_day_charge                -0.0069580439    -0.0177219575        -0.009193803      0.0333394728       -0.007783886
total_eve_minutes               -0.0590583448     0.1520725530        -0.025945771     -0.0132881785       -0.024095329
total_eve_calls                  1.0000000000    -0.0177182125         0.019347545     -0.0104469851        0.018640663
total_eve_charge                -0.0177182125     1.0000000000         0.013550042      0.0103983795        0.011020578
total_night_minutes              0.0193475446     0.0135500425         1.000000000      0.0114487328        0.997878263
total_night_calls               -0.0104469851     0.0103983795         [illegible]      1.0000000000        0.012092262
total_night_charge               0.0186406635     0.0110205777         0.997878263      0.0120922622        1.000000000
total_intl_minutes               0.0267676497     0.0130498718         [illegible]      0.0056589231        0.024950713
total_intl_calls                 0.0245694427     0.0069525465        -0.024871910     -0.0003905005       -0.024818406
total_intl_charge                0.0170473261     0.0003941937         0.007119107     -0.0231891807        0.002637076
number_customer_service_calls   -0.0096767565    -0.0085445108        -0.014657854      0.0085130736       -0.013441832
```

see the results from below pictures, there are 3 pairs of variables have multicollinearity, so we need to delete one of each paris. The important thing is which variable should be deleted, then, I build the first model to find out the importance of each variable. In the logistical regression model, the absolute value of z value represents the importance of each variable, which means greater the absolute value of z value, the more important. The below pictures show the result of the first model, I marked three variables with a star symbol, total_day_minutes, total_night_charge, and total_intl_charge, which need to keep it in the model because absolute value of z value of those three variables greater than other three variables. After selecting variables, I built a new model by training data.

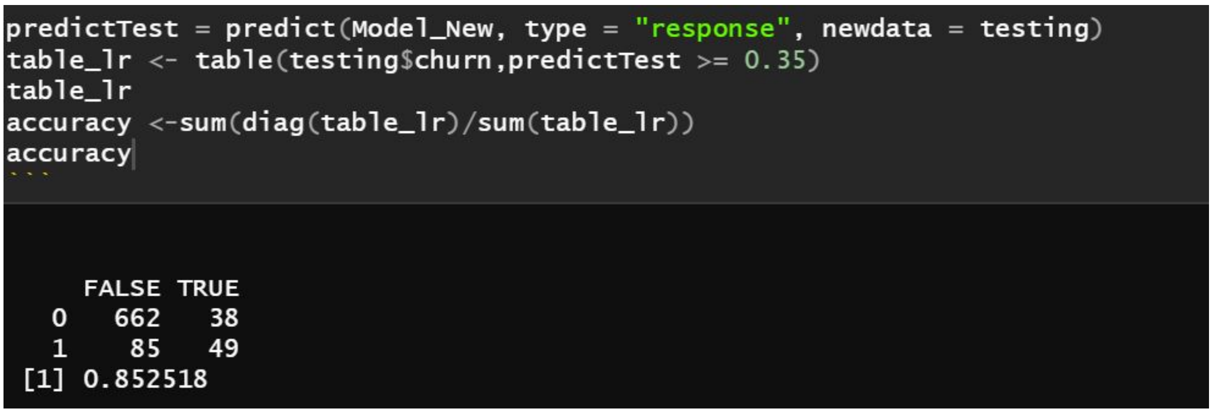


```
Call:
glm(formula = churn ~ ., family = "binomial", data = training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2092  -0.4879  -0.3092  -0.1756   3.1371

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -2.448082   0.097908 -25.004  < 2e-16 ***
account_length                0.048070   0.068156   0.705   0.4806
international_plan            2.140797   0.174227  12.287  < 2e-16 ***
voice_mail_plan              -0.821835   0.170270  -4.827 1.39e-06 ***
total_day_minutes            -0.410969   1.442560  -0.285   0.7757
total_day_calls               0.001418   0.065984   0.021   0.9829
total_day_charge              0.828624   0.132312   6.263 3.78e-10 ***
total_eve_minutes             0.325573   1.434530   0.227   0.8205
total_eve_calls              -0.010022   0.065431  -0.153   0.8783
total_eve_charge              0.412794   0.237492   1.738   0.0822 .
total_night_minutes          -0.428490   0.888922  -0.482   0.6298
total_night_calls             0.034614   0.067183   0.515   0.6064
total_night_charge            0.643581   0.887597   0.725   0.4684
total_intl_minutes           -0.727260   0.476425  -1.526   0.1269
total_intl_calls             -0.313887   0.076424  -4.107 4.01e-05 ***
total_intl_charge             0.984964   0.469652   2.097   0.0360 *
number_customer_service_calls 0.758858   0.062505  12.141  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Then, I used the predict() function to predict the model in training and testing data. But when I predict, I need to set the threshold. ROCR package could help me find out which threshold is best in this model. According to this ROC curve, I need to find the point on the line and it is the closest to the point (0,1). I tried 0.35, 0.4 and 0.5, when threshold equal to 0.35, I got the highest



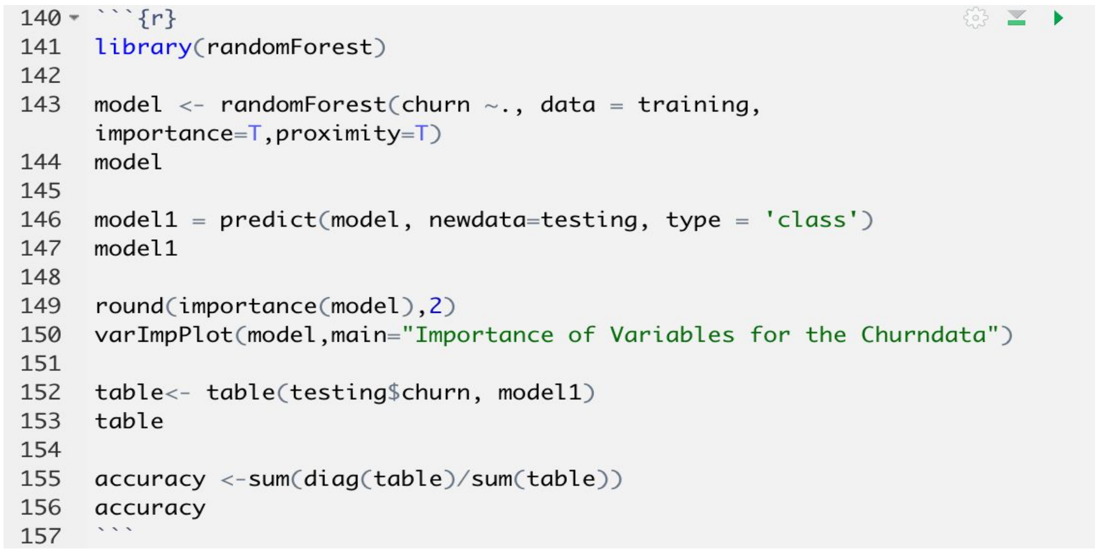
accuracy of 85.25%. Finally, I made a table to show the result of model and calculate the accuracy by using formula. My highest accuracy is 85.25%.



```
predictTest = predict(Model_New, type = "response", newdata = testing)
table_lr <- table(testing$churn,predictTest >= 0.35)
table_lr
accuracy <-sum(diag(table_lr)/sum(table_lr))
accuracy
```

```
    FALSE TRUE
  0   662   38
  1    85   49
[1] 0.852518
```

For the random forests section, it was very difficult to run all of the data at first. I realized that I had too much unnecessary data, with the help of my teammates we were able to figure out what I was doing wrong. I then ran the random forest tool from my library and created a model by training the data and finding the importance and proximity of our churn customers, which is labeled “Importance of Variables for the Churndata”. After that was found, I created another model with the new data found from the first and I tested the new data into class type. I rounded up the data from the first model(model) and the second model (model1). I then created a table testing the churn column of our data and model1. I ran the accuracy of our table and came to the conclusion of the highest accuracy of 94.36%. This section from our class was one of the hardest for me to comprehend, so very thankful for my teammates for helping me out along the way.



~~~
```{r}
library(randomForest)

model <- randomForest(churn ~., data = training,
importance=T,proximity=T)
model

model1 = predict(model, newdata=testing, type = 'class')
model1

round(importance(model),2)
varImpPlot(model,main="Importance of Variables for the Churndata")

table<- table(testing$churn, model1)
table

accuracy <-sum(diag(table)/sum(table))
accuracy
```
~~~

After building three molds, neural network, logistic regression, and random forest. We also calculate the accuracy of each model, there are 95.2%, 85.25% and 94.36% repetitively. Therefore, the neural network is the most appropriate model to predict the churn rate for ABC Wireless Inc. since the neural network model got the highest accuracy among models.

Based on what we find from our analysis, customer service, international plan, and the price is an important factor for customer retention. Therefore, we suggest that the company could make some promotions, like price discounts, and phone plans, for current customers to increase brand loyalty. For the international plan, we suggest the company could develop more international plans, which could attract more new customers and stabilize existing customers. For customers who are likely to churn, the company could give some service calls to investigate the reason and know the customer satisfaction. On the other hand, customers lay emphasis on the service, so the company could improve the phone lines of service to enhance customer loyalty, it will increase the customer lifetime. In conclusion, the company would follow the above information on creating some incentives, product development to increase customer lifetime and encourage the customer to stay.