CHURN PREDICTION WITH VARIOUS CLASSIFICATION ALGORITHMS AND CLUSTERING CUSTOMERS USING TELECOMMUNICATION DATASET

MSc. BUSINESS ANALYTICS

STATISTICS FOR BUSINESS ANALYTICS II

BURCIN SARAC

2018-2019 / FULL-TIME

F2821825

Table of Contents

1. Introduction

In this report my aim is to understand, analyze and model the churn of customers from the telecommunications company. The data set, which is used in this report, contains the portfolio of customers of the telecommunications company.

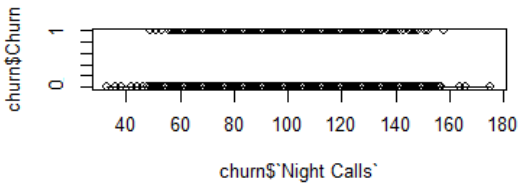
2. Exploratory Data Analysis and Pairwise Comparisons

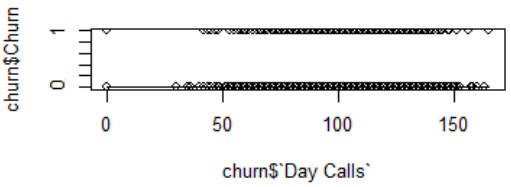
After drilling down to data more, firstly it should be necessary to mention that, the datasets have not any NA(empty) rows, in other words there is no missing data in the dataset.

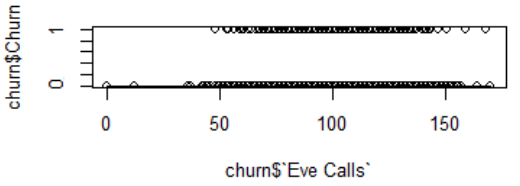
It seems from dataset that, all variables are numeric except “State” and “Gender”, which variables type are characters. However, target variable(Churn) explains if the customer churns(1) or not(0), so I changed data type into factor. I used same strategy to `Int'l Plan` and `VMail Plan`, because both of their type were numeric although they are dummy variables. Moreover, I changed Gender data from "Male" and "Female" to 0 and 1, 0 corresponds to male and 1 to female.

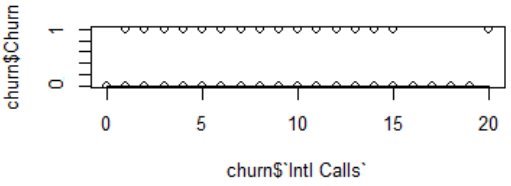
At last, I changed column names into strings without white spaces and convert “Area Code” and “State” variables into categorical to avoid that it may mislead the models.

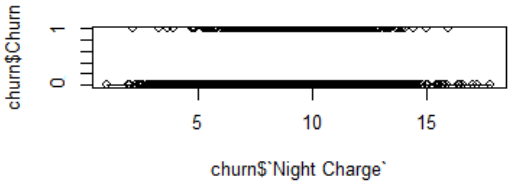
After this cleaning process, I plotted relation of all variables with the target variable separately to aim if a variable has effect on target variable or not. However, from this pairwise comparisons, it cannot be explained with a certainty. But it can basically said that, if generally mins, calls or charges increases, the probability of churn increases. And also all of call variables and some of others distributed seems like Poisson.

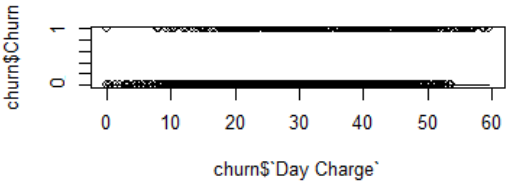

churn$Churn
0
1
40
60
80
100
120
140
160
180
churn$`Night Calls`

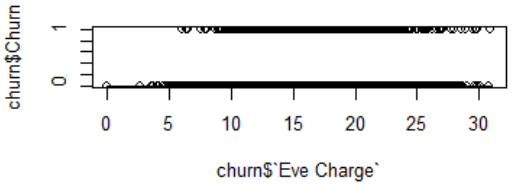

churn$Churn
0
1
0
50
100
150
churn$`Day Calls`

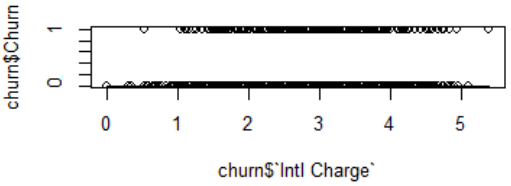

churn$Churn
0
1
0
50
100
150
churn$`Eve Calls`

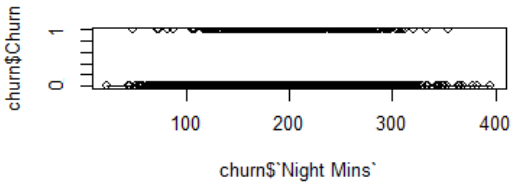

churn$Churn
0
1
0
5
10
15
20
churn$`Intl Calls`

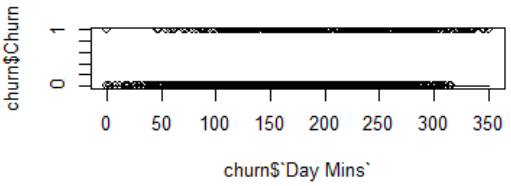

churn$Churn
0
1
5
10
15
churn$`Night Charge`

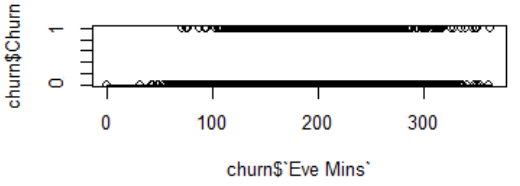

churn$Churn
0
1
0
10
20
30
40
50
60
churn$`Day Charge`

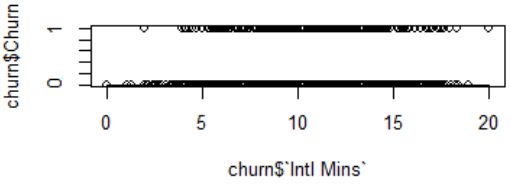

churn$Churn
0
1
0
5
10
15
20
25
30
churn$`Eve Charge`


churn$Churn
0
1
0
1
2
3
4
5
churn$`Intl Charge`

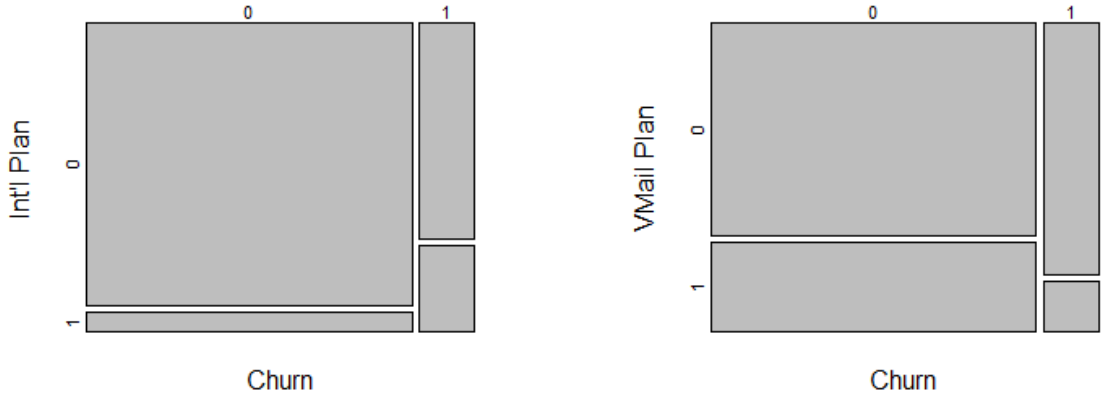

churn$Churn
0
1
100
200
300
400
churn$`Night Mins`


churn$Churn
0
1
0
50
100
150
200
250
300
350
churn$`Day Mins`


churn$Churn
0
1
0
100
200
300
churn$`Eve Mins`


churn$Churn
0
1
0
5
10
15
20
churn$`Intl Mins`

Moreover, if Intl plan is 1, it seems that it is more possible to churn rather that 0. And opposite to this, in VMail Plan it is more possible to churn if it equals to 0 rather than 1. Other variables also checked via graphs, but there is not such a meaningful explanation generated, so I decided not to add these graphs to the report.



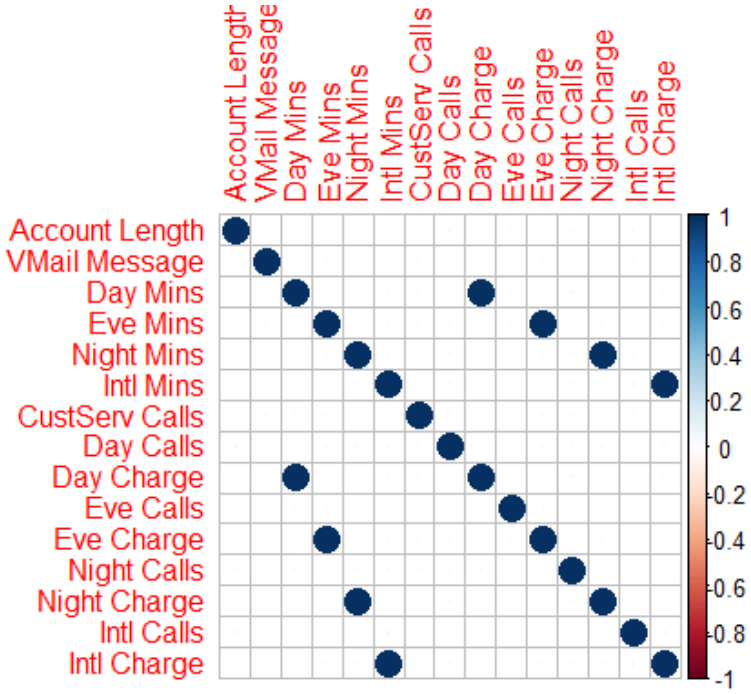
Addition to this, after checking the probabilities I saw that, roughly 14% of customers churned.

Firstly I checked correlation between variables and the target variable. It seem from table below that, both “Day Mins”, “CustServ Calls” and “Day Charge” variables has higher correlation with the target variable.

```
> round(cor(churn[sapply(churn, is.numeric)], y=as.numeric(churn$Churn)),2)
                 [,1]
Account Length   0.02
VMail Message   -0.09
Day Mins         0.21
Eve Mins         0.09
Night Mins       0.04
Intl Mins        0.07
CustServ Calls   0.21
Day Calls        0.02
Day Charge       0.21
Eve Calls        0.01
Eve Charge       0.09
Night Calls      0.01
Night Charge     0.04
Intl Calls      -0.05
Intl Charge      0.07
```

And then, I checked the correlation between all variables to create a clear view about these possible relationships. During this process, I saw that some variables have perfect correlation with some other predictors. It can be seen from the graph below that; "Day Charge" variable has close to perfect correlation with “Day Mins”, which is also same with,"Eve Charge" and “Eve Mins”,

"Night Charge" and “Night Mins”,"Intl Charge" and “Intl Mins”. In other words, both two of each group variables carry nearly same data for model.



However, I won’t drop any of the predictors, because I will use classification methods to predict customers churn results. Since feature selection does not necessarily improve the performance of modern classifier systems, and quite frequently makes performance worse, it is better to keep all predictors on board. Since the goal is not to find out which features are the most important, it is better not to use regularization to avoid over-fitting as well.

3. Predictions

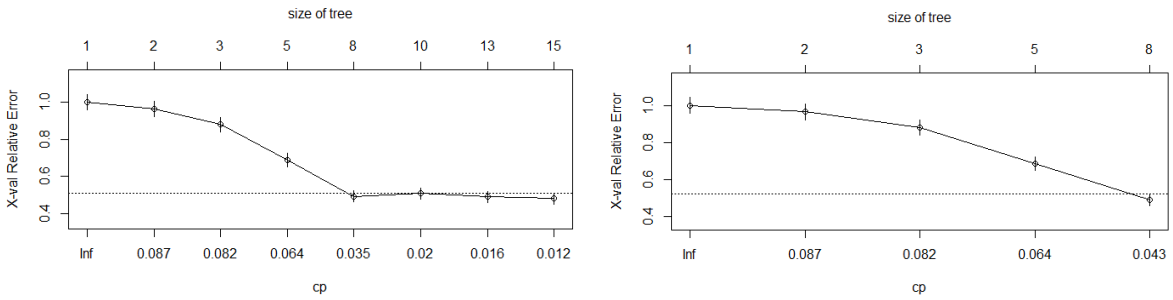
In this report I tried to deal with 2 different tasks. First one is to develop a classification model to predict that customer will churn or not. Second one is about clustering customers using user behaviors and in this task it is expected to ignore customer churn.

3.1 Prediction of Churn through various Classification Methods

3.1.1 Tree Based Models

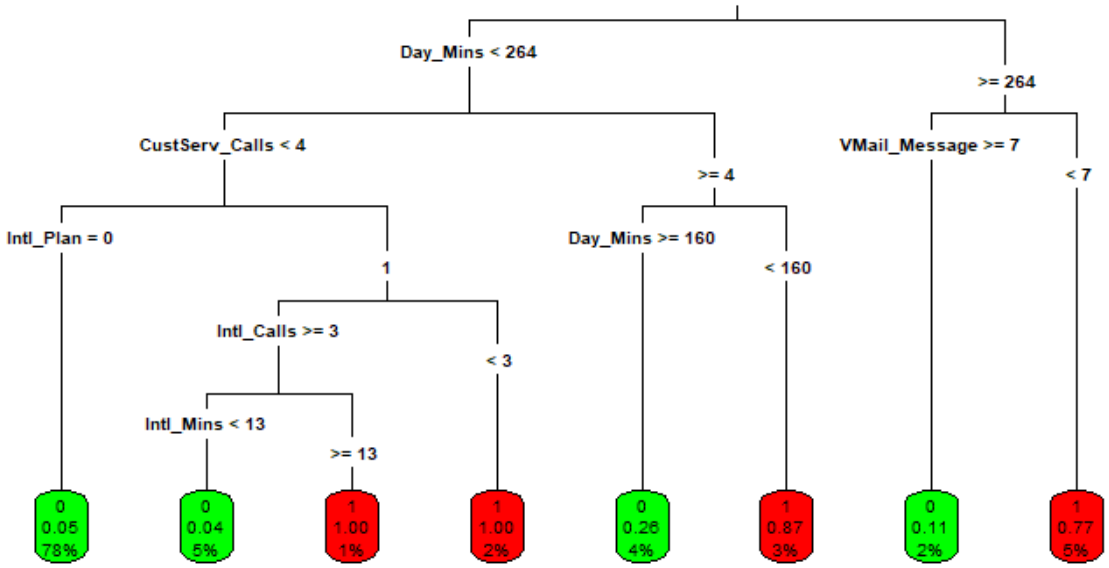
In this classification model selection section for prediction, I tested Naïve Bayes, Decision Tree, Random Forest, SVM and KNN. But since the data is not normally distributed, I avoid to use LDA or QDA models.

At first I started with tree based models without split data into train and test. It is obvious not to believe a model trained only with the whole dataset and only for one time, but I just wanted to see how fit it is. I trained a Random Forest using ranger package with set classification into TRUE, which will assign the prediction probabilities into a group with highest probability and I want it trains quickly so I set number of trees to 200, the default was 500. After that I also trained Decision(Classification) Tree using rpart package with determining tree method as class. Decision tree uses all predictors and observations and try to split all observations into classes which may lead overfitting, where Random forest picks randomly some of observations and creates many small trees which is 200 in our example. And regarding to the decision tree, train tree with all model and prune that tree from some specific point may give us better prediction results to avoid overfitting. In this example, I pruned the tree according to its results after training, which provides me to classify observations with only 6 variables with the accuracy of 93%. And with this accuracy it can predict 2800 of not churn and 304 of churn data correctly and 50 of not churns and 179 of churns predicted wrong.



And the selected variables were, Intl_Plan", "VMail_Message", "Day_Mins", "CustServ_Calls", "Intl_Calls","Intl_Mins". I also tried if it gives the same result with dropping other variables and train a decision tree again. I split 70% of dataset as train data randomly using sample() function and it was proved my result.

And the created tree can be seen below. In the graph red color refers to churn and green refers to not churn and if the path created by tree followed, for example if "Day_Mins" data is less than 264, "CustServ_Calls" is less than 4 and there is an "Intl Plan" which means if it is 1 and lastly if the "Intl Calls" is less than 3, than that customer is expected to churn according to the model.



But in spite of their high accuracy levels, both random forest and decision tree models are over-fitted models, because I ran these models with the whole dataset and only with one iteration. So I moved on with cross validation method from now on.

3.1.2 Naïve Bayes using Cross Validation Method

After decision tree trials, I used train() function from "caret" package, which helps me to use cross validation method when training models. When using this function I set trControl parameter to 3 times repeated cross validation with 10 folds. Which means it randomly splits data into 10 folds, reserves one fold to test and uses all others to train data, tests the model with reserved test fold and records the error and uses this method 10 times until all folds are used as test fold. After this process ends, it will repeat it 2 more times. And calculates the average of recorded errors as performance metric. With this method I aimed to avoid over fitting.

However, after train Naïve Bayes model with this method, I observed that it predicts all observations as non-churners. This case mostly observed if positive responders are less than or

equal to 10%, because with predicting every observation as non-responders algorithm can still get more than or equal to 90% accuracy rather than predicting every observation. In this case churners are 14% of dataset but it still avoid to make proper predictions.

```
> confusionMatrix(table(churn$churn, prednb))
Confusion Matrix and Statistics

   prednb
       0    1
  0 2850    0
  1  483    0

               Accuracy : 0.8551
                 95% CI : (0.8427, 0.8669)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8551
            Specificity :     NA
         Pos Pred Value :     NA
         Neg Pred Value :     NA
             Prevalence : 1.0000
         Detection Rate : 0.8551
   Detection Prevalence : 0.8551
      Balanced Accuracy :     NA

       'Positive' Class : 0
```

3.1.2.2 Naïve Bayes using Balanced Sampling & Separate Sampling Methods

For dealing this problem, I decided to use balanced sampling and separate sampling methods in Naïve Bayes to force it to make predictions. Balanced sampling method normally used if the dataset has only 5% or less responders, but I wanted to see how it resulted with this method as well as separate sampling. Regarding to these methods, in balanced sampling I created a sub sample by bringing all responders(churner in this case) and same number of non-responders randomly. And I do it 10 times with a for loop to avoid to get weighted data or not represented customers into the subsample. Separate sampling method uses same strategy but this time subsample has 30% percent responders and 70% non-responders.

After train Naïve Bayes in a sample created by using these methods I used generated model to predict from the whole dataset and there was a remarkable change occurred, the accuracy drops 88% in balanced sampling and 84% in separate sampling. With the model created with the helps of balanced sampling it predicts 2646 of non-churners and 297 of churners correctly, which was 2444 and 349 accordingly with the other model trained via separate sampling.

```
> confusionMatrix(table(churn$Churn, prednb2))
Confusion Matrix and Statistics

   prednb2
      0    1
  0 2646  204
  1  186  297

               Accuracy : 0.883
                 95% CI : (0.8716, 0.8937)
    No Information Rate : 0.8497
    P-Value [Acc > NIR] : 1.6e-08

                  Kappa : 0.535

 Mcnemar's Test P-Value : 0.3893

            Sensitivity : 0.9343
            Specificity : 0.5928
         Pos Pred Value : 0.9284
         Neg Pred Value : 0.6149
             Prevalence : 0.8497
         Detection Rate : 0.7939
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.7636

       'Positive' Class : 0
```

```
> confusionMatrix(table(churn$Churn, prednb3))
Confusion Matrix and Statistics

   prednb3
      0    1
  0 2444  406
  1  134  349

               Accuracy : 0.838
                 95% CI : (0.825, 0.8503)
    No Information Rate : 0.7735
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4702

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9480
            Specificity : 0.4623
         Pos Pred Value : 0.8575
         Neg Pred Value : 0.7226
             Prevalence : 0.7735
         Detection Rate : 0.7333
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.7051

       'Positive' Class : 0
```

3.1.3 SVM using Cross Validation Method

After train Support Vector Machine (SVM) algorithm with using same train() function with only changing method to “svmLinearWeights”, I got same result as Naïve Bayes served. In other words this was also did all predictions as non-churners.

```
> confusionMatrix(table(churn$Churn, predsvm))
Confusion Matrix and Statistics

   predsvm
      0    1
  0 2850    0
  1  483    0

               Accuracy : 0.8551
                 95% CI : (0.8427, 0.8669)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8551
            Specificity :     NA
         Pos Pred Value :     NA
         Neg Pred Value :     NA
             Prevalence : 1.0000
         Detection Rate : 0.8551
   Detection Prevalence : 0.8551
      Balanced Accuracy :     NA

       'Positive' Class : 0
```

3.1.3.2 SVM using Balanced Sampling & Separate Sampling Methods

So since I got positive results using balanced sampling and separate sampling methods I used same method with SVM too.

```
> confusionMatrix(table(churn$Churn, predsvm2))
Confusion Matrix and Statistics

   predsvm2
       0    1
  0 2162  688
  1  107  376

               Accuracy : 0.7615
                 95% CI : (0.7466, 0.7759)
    No Information Rate : 0.6808
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3582

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9528
            Specificity : 0.3534
         Pos Pred Value : 0.7586
         Neg Pred Value : 0.7785
             Prevalence : 0.6808
         Detection Rate : 0.6487
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.6531

       'Positive' Class : 0
```

```
> confusionMatrix(table(churn$Churn, predsvm3))
Confusion Matrix and Statistics

   predsvm3
       0    1
  0 2164  686
  1  110  373

               Accuracy : 0.7612
                 95% CI : (0.7463, 0.7756)
    No Information Rate : 0.6823
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3555

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9516
            Specificity : 0.3522
         Pos Pred Value : 0.7593
         Neg Pred Value : 0.7723
             Prevalence : 0.6823
         Detection Rate : 0.6493
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.6519

       'Positive' Class : 0
```

It was again gave better results, the one trained with balanced sampling data has 76% accuracy and the other trained with separate sampling data provided nearly same accuracy as well.

3.1.4 Random Forest using Cross Validation Method

This time I trained Random Forest algorithm with using same train() function with only changing method to “ranger”. This is same algorithm as I used before but this time only difference is using Cross Validation instead of using all data by once. But as this algorithm tends to overfit, with the helps of cross validation it was totally overfitted with the accuracy of 1(100%).

By using predictors() command from caret package again, the picked variables can be seen, I put screenshot below as well.

```
> confusionMatrix(table(churn$Churn, predrf))
Confusion Matrix and Statistics

   predrf
      0    1
  0 2850    0
  1    0  483

               Accuracy : 1
                 95% CI : (0.9989, 1)
    No Information Rate : 0.8551
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.8551
         Detection Rate : 0.8551
   Detection Prevalence : 0.8551
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

```
> predictors(modelrf)
 [1] "Account_Length" "VMail_Message"  "Day_Mins"       "Eve_Mins"
 [5] "Night_Mins"     "Intl_Mins"      "CustServ_Calls" "Intl_Plan1"
 [9] "VMail_Plan1"    "Day_Calls"      "Day_Charge"     "Eve_Calls"
[13] "Eve_Charge"     "Night_Calls"    "Night_Charge"   "Intl_Calls"
[17] "Intl_Charge"    "StateAL"        "StateAR"        "StateAZ"
[21] "StateCA"        "StateCO"        "StateCT"        "StateDC"
[25] "StateDE"        "StateFL"        "StateGA"        "StateHI"
[29] "StateIA"        "StateID"        "StateIL"        "StateIN"
[33] "StateKS"        "StateKY"        "StateLA"        "StateMA"
[37] "StateMD"        "StateME"        "StateMI"        "StateMN"
[41] "StateMO"        "StateMS"        "StateMT"        "StateNC"
[45] "StateND"        "StateNE"        "StateNH"        "StateNJ"
[49] "StateNM"        "StateNV"        "StateNY"        "StateOH"
[53] "StateOK"        "StateOR"        "StatePA"        "StateRI"
[57] "StateSC"        "StateSD"        "StateTN"        "StateTX"
[61] "StateUT"        "StateVA"        "StateVT"        "StateWA"
[65] "StateWI"        "StateWV"        "StateWY"        "Area_Code415"
[69] "Area_Code510"
>
```

Accroding to the predictors picked by model, I thought that it may overfit possibly because of using State information, but I tried it after dropping “State” column and it was still overfitted with using different predictors as it can be observed below.

```
   predrf2
      0    1
  0 2850    0
  1    0  483

               Accuracy : 1
                 95% CI : (0.9989, 1)
    No Information Rate : 0.8551
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.8551
         Detection Rate : 0.8551
   Detection Prevalence : 0.8551
      Balanced Accuracy : 1.0000

       'Positive' Class : 0

> #      0    1
> # 0 2850    0
> # 1    0  483
> accuracy(churn$Churn, predrf2)
[1] 1
> #1
> predictors(modelrf2)
 [1] "Account_Length" "VMail_Message"  "Day_Mins"       "Eve_Mins"
 [5] "Night_Mins"     "Intl_Mins"      "CustServ_Calls" "Intl_Plan1"
 [9] "VMail_Plan1"    "Day_Calls"      "Day_Charge"     "Eve_Calls"
[13] "Eve_Charge"     "Night_Calls"    "Night_Charge"   "Intl_Calls"
[17] "Intl_Charge"    "Area_Code415"   "Area_Code510"
>
```

3.1.5 K-Nearest Neighbors using Cross Validation Method

This time I trained Random Forest algorithm with using same train() function with only changing method to "knn". But this time I also used expand.grid() function from caret package again, to try different k levels and I tried (1,2,3,4,5,10,20,50,100) k values all together with cross validation method. The models gives best result with k=10. And the accuracy was 89%, which seems better than other models.

```
> confusionMatrix(table(churn$Churn, predknn))
Confusion Matrix and Statistics

   predknn
      0    1
  0 2826   24
  1  339  144

               Accuracy : 0.8911
                 95% CI : (0.88, 0.9015)
    No Information Rate : 0.9496
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3973

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8929
            Specificity : 0.8571
         Pos Pred Value : 0.9916
         Neg Pred Value : 0.2981
             Prevalence : 0.9496
         Detection Rate : 0.8479
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.8750

       'Positive' Class : 0
```

As determined from model, "k=10" means algorithm calculates distance via Euclidian method between variables and it takes into account closest 10 points in terms of minimum distance between observations to predict target values. Since Knn requires more data for better predictions, I did not used balanced sampling method or separate sampling method in this algorithm.

3.1.6 Final Model Decision

Although random forest model gives sharp 1 accuracy, it is obviously overfitted to the dataset and it won't work in prediction of unseen data at all. On the other hand both Knn and Naïve Bayes trained with Balanced Sampling Models provide around 89% accuracy, so I planned to decide one of them as final model for later predictions. But to make decision I also wanted to check precision and recall differences between two models.

```
> confusionMatrix(table(churn$Churn, prednb2), mode="prec_recall")
Confusion Matrix and Statistics

   prednb2
      0    1
  0 2646  204
  1  186  297

               Accuracy : 0.883
                 95% CI : (0.8716, 0.8937)
    No Information Rate : 0.8497
    P-Value [Acc > NIR] : 1.6e-08

                  Kappa : 0.535

 Mcnemar's Test P-Value : 0.3893

              Precision : 0.9284
                 Recall : 0.9343
                     F1 : 0.9314
             Prevalence : 0.8497
         Detection Rate : 0.7939
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.7636

       'Positive' Class : 0
```

```
> confusionMatrix(table(churn$Churn, predknn), mode="prec_recall")
Confusion Matrix and Statistics

   predknn
      0    1
  0 2826   24
  1  339  144

               Accuracy : 0.8911
                 95% CI : (0.88, 0.9015)
    No Information Rate : 0.9496
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3973

 Mcnemar's Test P-Value : <2e-16

              Precision : 0.9916
                 Recall : 0.8929
                     F1 : 0.9397
             Prevalence : 0.9496
         Detection Rate : 0.8479
   Detection Prevalence : 0.8551
      Balanced Accuracy : 0.8750

       'Positive' Class : 0
```

		Predicted	
		1	**0**
Actual	**1**	A	C
	0	B	D

To explain Precision and Recall, I would like to use the sheet above,

Precision = A/(A+B) (number of true positives over the number of true positives plus the number of false positives)

Recall = A/(A+C) (number of true positives over the number of true positives plus the number of false negatives)

Which ratios can be used to evaluate output quality. Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

In this case recall is more important metric to take into account rather than precision. Because precision ratio interests false positives, which means regarding to this case is predicted as churner although it is non-churner. However recall interests false negatives, which means predicted non-churner but actually churner.

So I decided to use “modelnb2” which was the Naïve Bayes Model trained with a sub sample picked via balanced sampling method, as final model for future predictions, because since predicting correct churners is more important, higher recall ratio is better and its recall ratio is 93% where knn’s was 89%.

3.2 Clustering Customers via Unsupervised Learning Algorithms

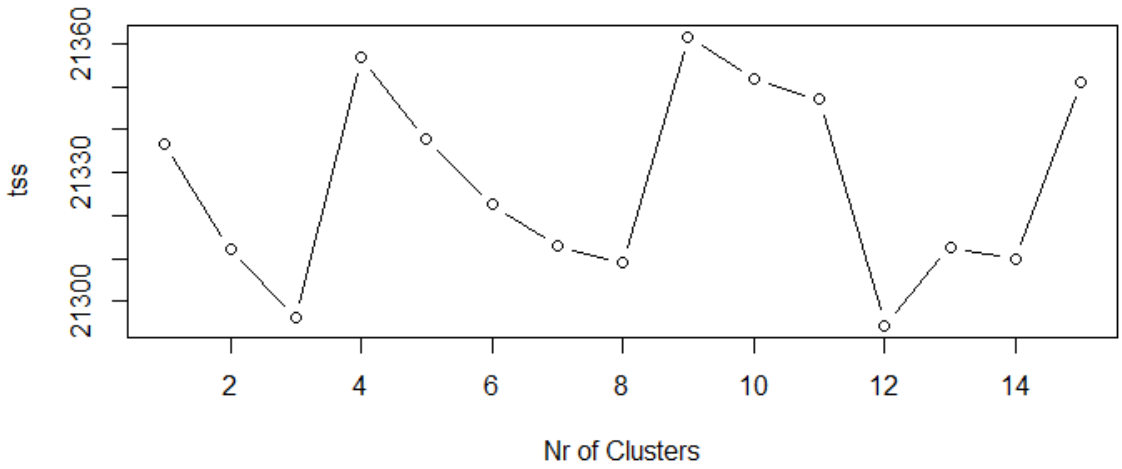
In this topic, I tried to cluster customers using some of all predictors without taking into account if they were churner or not. This type of predictions called Unsupervised Learning and in this methods there is not such a correct answer, but clusters created due to business decisions. In this case my goal is to assign each customer into some clusters for marketing purposes. Thanks to this, marketing campaigns or targets may have specialized related to the characteristics of customers in each cluster.

Although all variables might have some affects, I picked "State, Intl_Plan, Area_Code, Gender, Account_Length, Day_Mins, Eve_Mins, Intl_Mins, VMail_Plan, Eve_Calls, Day_Calls, Intl_Calls" features only based on subjective decision.

After filtering data by letting it only includes these variables, I first re-scale all numeric variables to be with 0 mean. I used scale() function for this command, but it would be also easy to write a function with implementing this { [variable-mean(variable)] / [max(variable)-min(variable)] } formula to all observations. After this process, I also needed to deal with categorical variables. The easiest way to handle this is using dummyVars() function from "caret" package again. This function automatically select categorical variables and converts them into dummy variables. Dummy variables mean all categories become separate columns and if an observation belongs to for example first category then the value on that category's column is become "1" otherwise "0". This is 2 step process, in first step you should show the data to the function dummyVars() and how to separate categories inside a column, to explain this if we determine to separate columns with "." and there is a X categorical column with categories a,b,c , then function convert these column into 3 different columns by using "." , then the columns become X.a, X.b, X.c . And in the second step predict() function should be used to get dummy variables from the main dataset.

3.2.1 Clustering using K-means algorithm

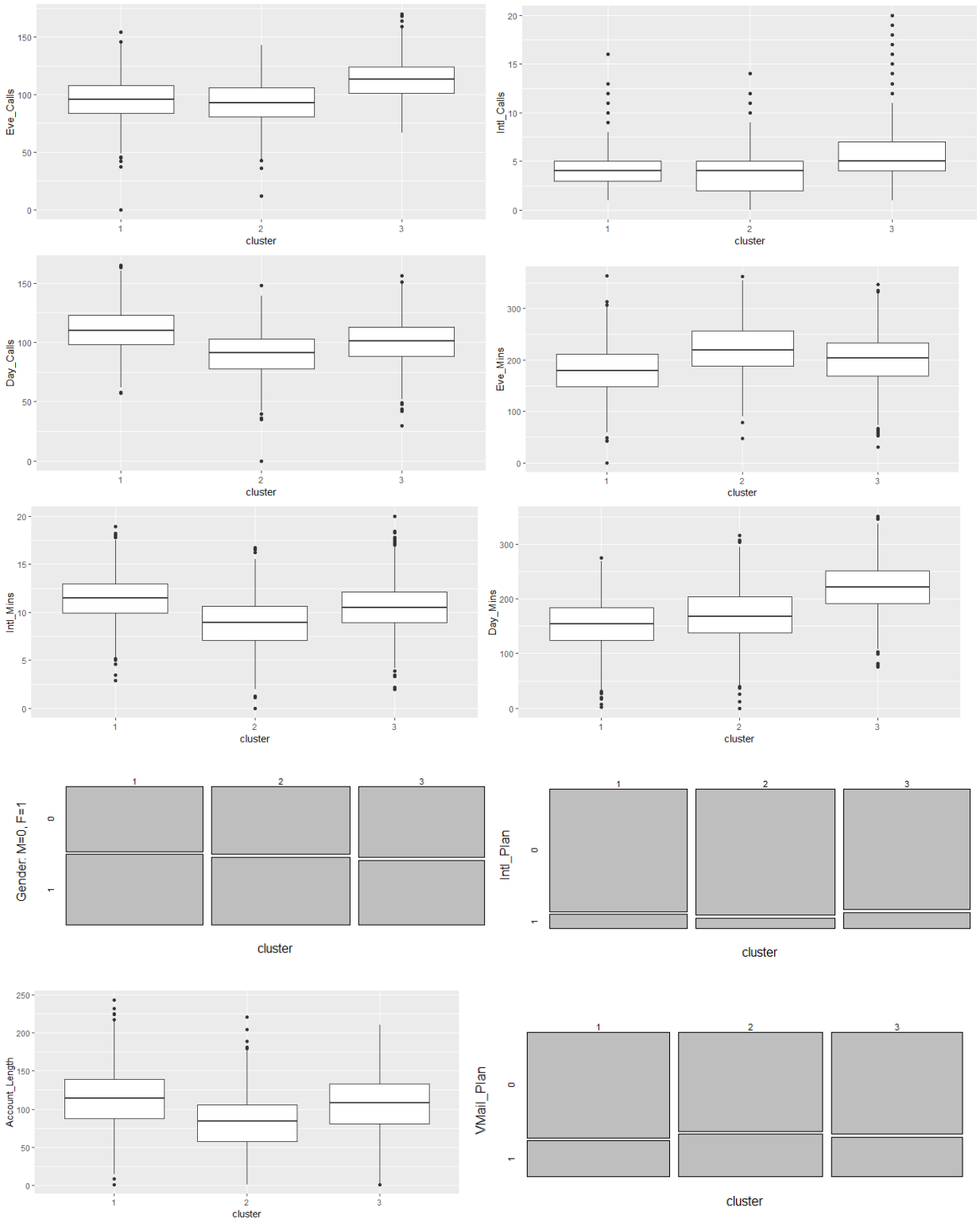
In this process, I used kmeans() function from "stats" package. Inside function I set centers = 15, nstart = 20, iter.max = 20 regarding to parameters. Centers used for the number of clusters, as k,, iter.max is used for the maximum number of iterations allowed and nstart corresponds to how many random sets should be chosen. And I put it into a for loop with 15 iterations to avoid the affect of randomization.



After iterations the graph generated by using total sum of squares(tss) and number of clusters. With this graph I can determine how many clusters should be in my case due to lower tss levels. The first clear cluster number seems 3 from the graph, after that elbow the second cluster number would be 8 or 12 for example. These should be determined regarding to business needs and decisions. In this case I will use 3 clusters.

After determining my cluster numbers, I fit another kmeans model with 3 clusters(k=3). I my clusters created with the numbers below. It is a balanced distribution and acceptable to use.

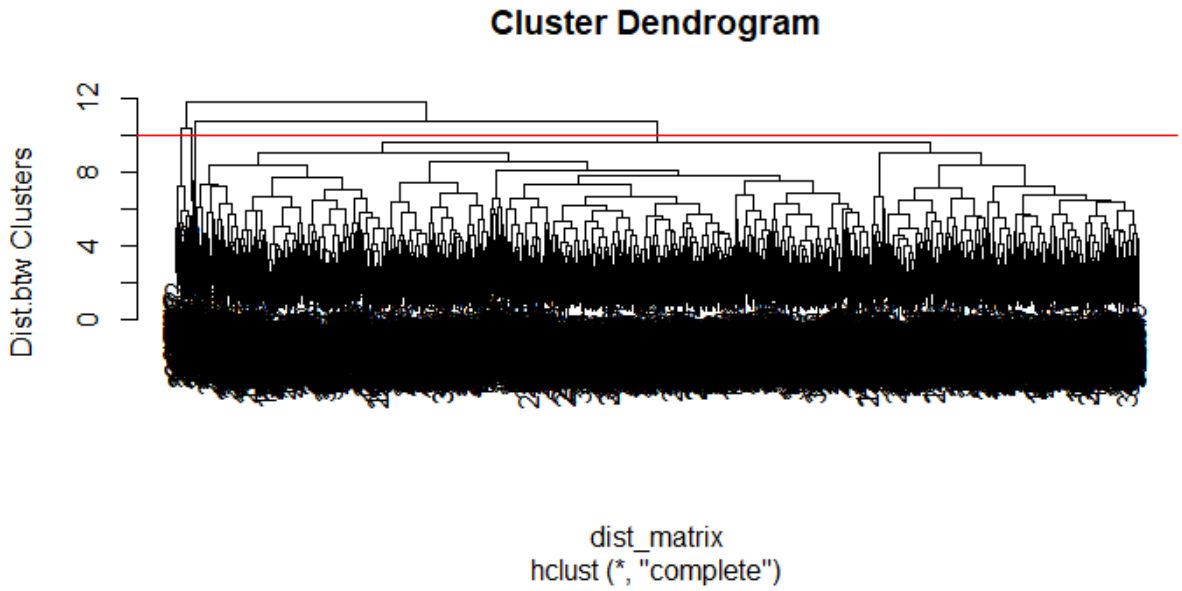
```
> table(modelkmeans2$cluster)

   1    2    3
1134 1152 1047
>
```



It seems from the graphs that, all selected variables except Gender has some effect on the determination of clusters.

3.2.2 Clustering using Hierarchical Method

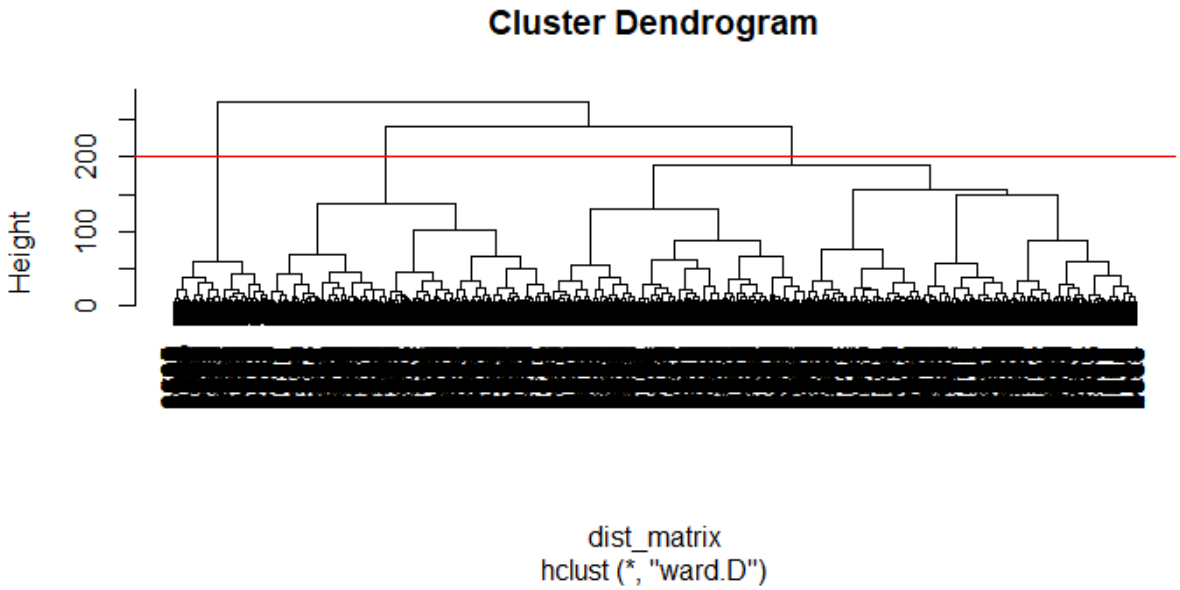
At this step I used hierarchical clustering method with using both “complete” and “ward” methods. For implementing hierarchical clustering method, I used hclust() function from stats package, however in this function it requires distribution matrix between variables instead of putting the data. Because of that I used dist() function with the same dataset used above with kmeans algorithm. And when I used “method=complete” parameter inside hclust() function it did not cluster data as expected. Complete method finds pairwise similarity between all obsservations and separate them into clusters by using largest of similarities. On the other hand, ward method uses the dissimilarities which are squared before cluster updating.



In the cluster dendrogram created by using complete method, it can be seen below that the clusters separated unbalanced. I tried to create 3 and 4 clusters but both of them did not distributed in a balanced way. The numbers of clusters can be seen below.

```
> clustershc <- cutree(modelhc, h=10)
> table(clustershc)
clustershc
   1    2    3    4
3262    9   53    9
> clustershc <- cutree(modelhc, k=3)
> table(clustershc)
clustershc
   1    2    3
3262   62    9
```

After that, I tried ward method, and the cluster dendrogram generated as below;



This time clusters distributed more balanced but still one of the clusters has very low numbers. But for example if 8 clusters would be better for marketing expectations, this algorithm would give better results. The cluster numbers can be observed below in different cluster levels. But still kmeans gave more balanced results with 8 clusters as well.

```
>
> clustershc_wd <- cutree(modelhc_wd, k=3)
> table(clustershc_wd)
clustershc_wd
   1    2    3
2025  979  329
> clustershc_wd <- cutree(modelhc_wd, k=4)
> table(clustershc_wd)
clustershc_wd
   1    2    3    4
 881 1144  979  329
> clustershc_wd <- cutree(modelhc_wd, k=8)
> table(clustershc_wd)
clustershc_wd
  1   2   3   4   5   6   7   8
586 438 566 413 295 329 401 305
>
```

```
> table(modelkmeans3$cluster)

  1   2   3   4   5   6   7   8
423 437 383 321 441 449 403 476
>
```

In this case I decided not to continue analyzing this hierarchical clustering algorithm and to use kmeans model as my final prediction model.