# BA Group Project - Group 4

*Setting default values to get a clean output*

```
knitr::opts_chunk$set(message = FALSE)
knitr::opts_chunk$set(warning = FALSE)
```

*Loading the required packages*

```
library("ISLR")
library("caret")
library("class")
library("e1071")
library("dplyr")
library("tidyverse")
library("ggplot2")
library("gmodels")
library("MASS")
library("broom")
library("modelr")
library("Hmisc")
library("missForest")
library("rpart")
library("rattle")
library("pROC")
library("ROCR")
library("cutpointr")
library("ROSE")
```

*Setting the default working directory*

```
setwd("/Users/sampathnikhilkumar/Desktop/Final Group Project - BA")
```

*Loading the data sets*

```
#Training Data set
raw_data <- read.csv("Churn_Train.csv")

#Test Data set
load("~/Desktop/Final Group Project - BA/Customers_To_Predict.RData")
```

*Data Cleaning & Transformation*

```
#Removing Unnecessary Columns
churn_Train <- raw_data[,-c(1:3)]
```

```r
#Re-coding few variables
churn_Train$churn <- ifelse(churn_Train$churn =="yes",1,0)
churn_Train$international_plan <- ifelse(churn_Train$international_plan =="yes",1,0)
churn_Train$voice_mail_plan <- ifelse(churn_Train$voice_mail_plan =="yes",1,0)

#Imputing NA Values
all_column_median <- apply(churn_Train,2,median, na.rm=T)

for(i in colnames(churn_Train))
churn_Train[,i][is.na(churn_Train[,i])] <- all_column_median[i]

#Converting integer to factor
churn_Train$churn <- as.factor(churn_Train$churn)

#Changing the order of the factor levels
churn_Train$churn <-  factor(churn_Train$churn,levels(churn_Train$churn)[c(2,1)])
```

*Partitioning the given churn_data into 75% train and 25% validation*

```r
data_part <- createDataPartition(churn_Train$churn,p=.75,list=F)

Train_Data <- churn_Train[data_part,]
Validation_Data <- churn_Train[-data_part,]
```

*Running a logistic regression model with cross validation (cv) as trainControl on train data*

```r
set.seed(125)
train_control <- trainControl(method = "repeatedcv",number=10,repeats = 3,savePredictions = 'final',cla

lr.model <- train(churn~., data = Train_Data, method = "glm", family="binomial", metric="Accuracy", trC
```

*Running kNN model on train data*

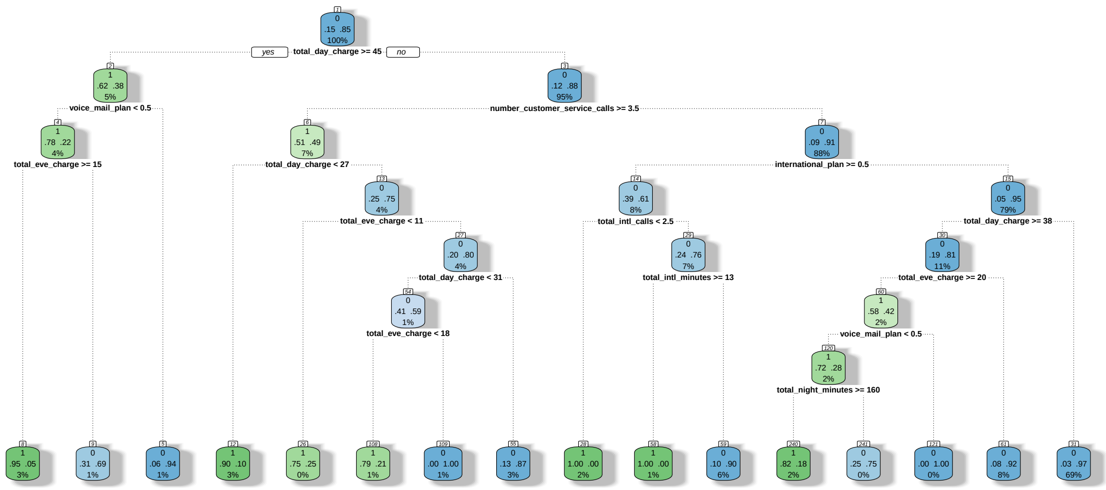
```r
set.seed(125)
train_control <- trainControl(method = "repeatedcv",number=10,repeats = 3,savePredictions = 'final',cla

knn.model <- train(churn~., data = Train_Data, method = "knn", metric="Accuracy", trControl = train_con
```

*Running Naive Bayes Model on train data*

```r
set.seed(876)
naive.bayes <- naiveBayes(churn~.,data=Train_Data)
```

*Running the Decision Tree Model on train data*

```r
set.seed(765)
Dec_Tree.model <- rpart(churn~.,data=Train_Data,method="class")
fancyRpartPlot(Dec_Tree.model)
```




Rattle 2022−Dec−11 16:32:16 sampathnikhilkumar

*Testing the models over validation set*

```
#Predicting the logistic regression model built over the validation data to check the accuracy
lr_validate <- predict(lr.model,Validation_Data,type ="prob")
churn.lr.validate <- cbind(Validation_Data,lr_validate)

#Predicting the kNN model built over the validation data to check the accuracy
knn.validate <- predict(knn.model,Validation_Data,type="prob")
knn.validate.df <- cbind(Validation_Data,knn.validate)

#Predicting the naive bayes model built over the validation data to check the accuracy
bayes.validate <- predict(naive.bayes,Validation_Data,type="raw")
bayes.validate.df <- cbind(Validation_Data,bayes.validate)

#Predicting the decision tree model built over the validation data to check the accuracy
dec_validate <- predict(Dec_Tree.model,Validation_Data,type ="prob")
churn.dec.validate <- cbind(Validation_Data,dec_validate)
```

*Optimal Threshold - Cut Off Point*

```
#Logistic Regression
ROC_pred_lr_test <- prediction(lr_validate[,1],churn.lr.validate$churn)

ROCR_perf_lr_test <- performance(ROC_pred_lr_test,'tpr','fpr')

acc_lr_perf <- performance(ROC_pred_lr_test,"acc")
```

```r
ROC_pred_lr_test@cutoffs[[1]][which.max(acc_lr_perf@y.values[[1]])]
```

```
## [1] 0.6705911
```

```r
#AUC Value
roc.curve(churn.lr.validate$churn, lr_validate[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.822
```

```r
#k-NN
ROC_pred_knn_test <- prediction(knn.validate[,1],knn.validate.df$churn)

ROCR_perf_knn_test <- performance(ROC_pred_knn_test,'tpr','fpr')

acc_knn_perf <- performance(ROC_pred_knn_test,"acc")

ROC_pred_knn_test@cutoffs[[1]][which.max(acc_knn_perf@y.values[[1]])]
```

```
## [1] 0.5555556
```

```r
#AUC Value
roc.curve(knn.validate.df$churn,knn.validate[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.654
```

```r
#Naive Bayes
ROC_pred_bayes_test <- prediction(bayes.validate[,1],bayes.validate.df$churn)

ROCR_perf_bayes_test <- performance(ROC_pred_bayes_test,'tpr','fpr')

acc_bayes_perf <- performance(ROC_pred_bayes_test,"acc")

ROC_pred_bayes_test@cutoffs[[1]][which.max(acc_bayes_perf@y.values[[1]])]
```

```
## [1] 0.3042114
```

```r
#AUC Value
roc.curve(bayes.validate.df$churn,bayes.validate[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.854
```

```r
#Decision Tree
ROC_pred_dec_test <- prediction(dec_validate[,1],churn.dec.validate$churn)

ROCR_perf_dec_test <- performance(ROC_pred_dec_test,'tpr','fpr')

acc_dec_perf <- performance(ROC_pred_dec_test,"acc")

ROC_pred_dec_test@cutoffs[[1]][which.max(acc_dec_perf@y.values[[1]])]
```

```
##      3304
## 0.3076923
```

```r
#AUC Value
roc.curve(churn.dec.validate$churn,dec_validate[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.899
```

*Re-Coding Variables - To run the CrossTable()*

```r
#Setting the optimal cutoffs for all the models
#Logistic Regression Model
churn.lr.validate$prob <- as.factor(ifelse(churn.lr.validate$`1`>0.6705911,"yes","no"))
#kNN Model
knn.validate.df$prob <- as.factor(ifelse(knn.validate.df$`1`>0.5555556,"yes","no"))
#Naive Bayes Model
bayes.validate.df$prob <- as.factor(ifelse(bayes.validate.df$`1`>0.3042114,"yes","no"))
#Decision Tree Model
churn.dec.validate$prob <- as.factor(ifelse(churn.dec.validate$`1`>0.3076923,"yes","no"))

#Converting the churn column back to yes and no
churn.lr.validate$churn <- as.factor(ifelse(churn.lr.validate$churn==1,"yes","no"))
knn.validate.df$churn <- as.factor(ifelse(knn.validate.df$churn==1,"yes","no"))
bayes.validate.df$churn <- as.factor(ifelse(bayes.validate.df$churn==1,"yes","no"))
churn.dec.validate$churn <- as.factor(ifelse(churn.dec.validate$churn==1,"yes","no"))
```

*Using CrossTable() to look at the performance metrics and miscalculations for all the models*

```r
#Logistic Regression Model
CrossTable(x=churn.lr.validate$churn,y=churn.lr.validate$prob,prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  832
##
##
##                         | churn.lr.validate$prob
## churn.lr.validate$churn |        no |       yes | Row Total |
## ------------------------|-----------|-----------|-----------|
##                      no |       712 |         0 |       712 |
##                         |     1.000 |     0.000 |     0.856 |
##                         |     0.869 |     0.000 |           |
##                         |     0.856 |     0.000 |           |
```

```
## ------------------------|-----------|-----------|-----------|
##                     yes |       107 |        13 |       120 |
##                         |     0.892 |     0.108 |     0.144 |
##                         |     0.131 |     1.000 |           |
##                         |     0.129 |     0.016 |           |
## ------------------------|-----------|-----------|-----------|
##            Column Total |       819 |        13 |       832 |
##                         |     0.984 |     0.016 |           |
## ------------------------|-----------|-----------|-----------|
##
##
```

***Performance Metrics - Logistic Regression Model***
*True Positive (TP) - 13*
*True Negative (TN) - 712*
*False Positive (FP) - 0*
*False Negative (FN) - 107*
*Miscalculations - 107*
*Accuracy = TP+TN/TP+TN+FP+FN = 13+712/832 = 87.13 %*
*Specificity (TNR) = TN/TN+FP = 712/712+0 = 100 %*
*Sensitivity (TPR) = TP/TP+FN = 13/13+107 = 10.83 %*

```
#kNN Model
CrossTable(x=knn.validate.df$churn,y=knn.validate.df$prob,prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  832
##
##
##                       | knn.validate.df$prob
## knn.validate.df$churn |        no |       yes | Row Total |
## ----------------------|-----------|-----------|-----------|
##                    no |       711 |         1 |       712 |
##                       |     0.999 |     0.001 |     0.856 |
##                       |     0.876 |     0.050 |           |
##                       |     0.855 |     0.001 |           |
## ----------------------|-----------|-----------|-----------|
##                   yes |       101 |        19 |       120 |
##                       |     0.842 |     0.158 |     0.144 |
##                       |     0.124 |     0.950 |           |
##                       |     0.121 |     0.023 |           |
```

```
## ----------------------|-----------|-----------|-----------|
##          Column Total |       812 |        20 |       832 |
##                       |     0.976 |     0.024 |           |
## ----------------------|-----------|-----------|-----------|
##
##
```

***Performance Metrics - kNN Model***

*True Positive (TP) - 19*
*True Negative (TN) - 711*
*False Positive (FP) - 1*
*False Negative (FN) - 101*
*Miscalculations - 102*

*Accuracy = TP+TN/TP+TN+FP+FN = 19+711/832 = 87.74 %*
*Specificity (TNR) = TN/TN+FP = 711/711+1 = 99.85 %*
*Sensitivity (TPR) = TP/TP+FN = 19/19+101 = 15.83 %*

```
#Naive Bayes Model
CrossTable(x=bayes.validate.df$churn,y=bayes.validate.df$prob,prop.chisq=F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  832
##
##
##                         | bayes.validate.df$prob
## bayes.validate.df$churn |        no |       yes | Row Total |
## ------------------------|-----------|-----------|-----------|
##                      no |       645 |        67 |       712 |
##                         |     0.906 |     0.094 |     0.856 |
##                         |     0.942 |     0.456 |           |
##                         |     0.775 |     0.081 |           |
## ------------------------|-----------|-----------|-----------|
##                     yes |        40 |        80 |       120 |
##                         |     0.333 |     0.667 |     0.144 |
##                         |     0.058 |     0.544 |           |
##                         |     0.048 |     0.096 |           |
## ------------------------|-----------|-----------|-----------|
##            Column Total |       685 |       147 |       832 |
##                         |     0.823 |     0.177 |           |
## ------------------------|-----------|-----------|-----------|
##
##
```

***Performance Metrics - Naive Bayes Model***
*True Positive (TP) - 80*
*True Negative (TN) - 645*
*False Positive (FP) - 67*
*False Negative (FN) - 40*
*Miscalculations - 107*

*Accuracy = TP+TN/TP+TN+FP+FN = 80+645/832 = 87.13 %*
*Specificity (TNR) = TN/TN+FP = 645/645+67 = 90.58 %*
*Sensitivity (TPR) = TP/TP+FN = 80/80+40 = 66.66 %*

```
#Decision Tree Model
CrossTable(x=churn.dec.validate$churn,y=churn.dec.validate$prob,prop.chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  832
##
##
##                          | churn.dec.validate$prob
## churn.dec.validate$churn |        no |       yes | Row Total |
## -------------------------|-----------|-----------|-----------|
##                       no |       690 |        22 |       712 |
##                          |     0.969 |     0.031 |     0.856 |
##                          |     0.961 |     0.193 |           |
##                          |     0.829 |     0.026 |           |
## -------------------------|-----------|-----------|-----------|
##                      yes |        28 |        92 |       120 |
##                          |     0.233 |     0.767 |     0.144 |
##                          |     0.039 |     0.807 |           |
##                          |     0.034 |     0.111 |           |
## -------------------------|-----------|-----------|-----------|
##             Column Total |       718 |       114 |       832 |
##                          |     0.863 |     0.137 |           |
## -------------------------|-----------|-----------|-----------|
##
##
```

***Performance Metrics - Decision Tree***
*True Positive (TP) - 92*
*True Negative (TN) - 690*
*False Positive (FP) - 22*
*False Negative (FN) - 28*

*Miscalculations - 50*

*Accuracy = TP+TN/TP+TN+FP+FN = 92+690/832 = 93.99 %*
*Specificity (TNR) = TN/TN+FP = 690/690+22 = 96.91 %*
*Sensitivity (TPR) = TP/TP+FN = 92/92+28 = 76.66 %*

***Eventually, we can see that the decision tree model is working quite good on the validation set when compared to that with the other models. Accuracy, Sensitivity and Specificity is comparatively high so we are proceeding with the decision tree model to be implemented on the "test set".***

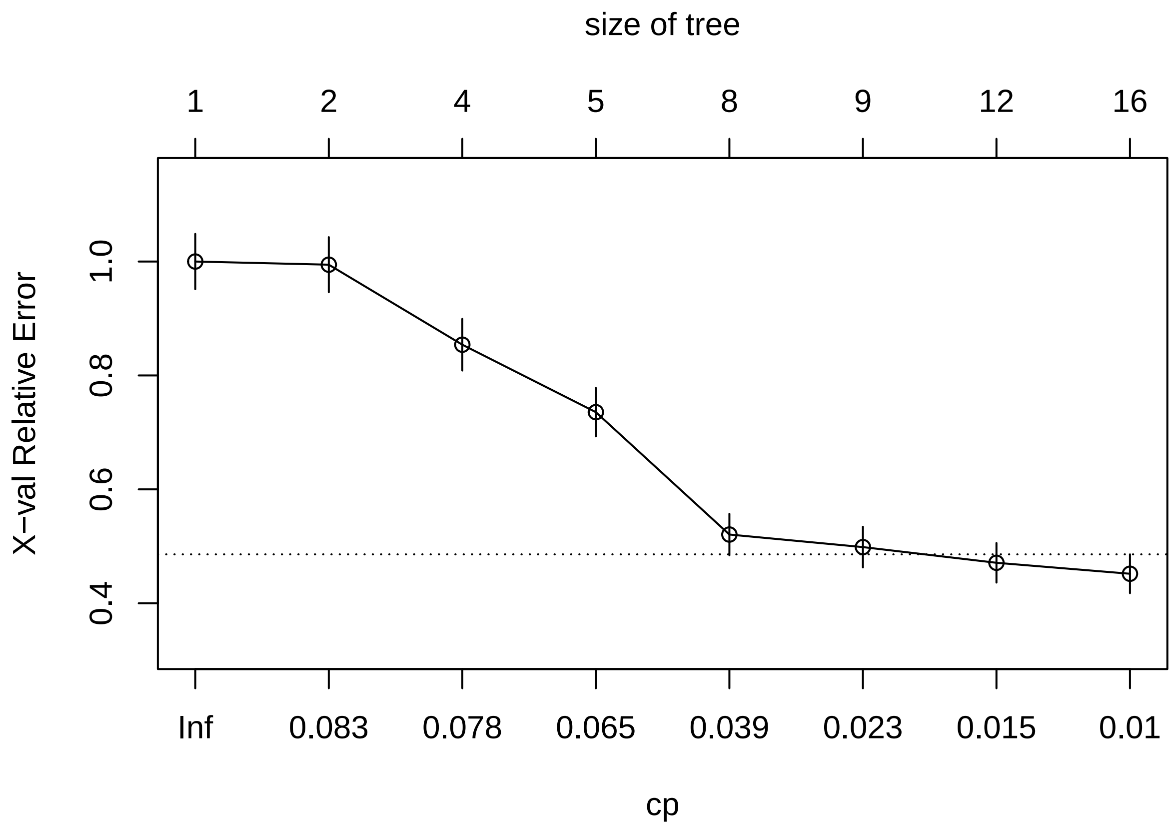
*In order to use an effective model on the test set we did try to use pruning as well to check if there's any rise in the accuracy*

*Pruning the decision tree model*

```
#Base Model
#The Dec_Tree.model is the base model which was already built at the beginning
printcp(Dec_Tree.model)
```

```
##
## Classification tree:
## rpart(formula = churn ~ ., data = Train_Data, method = "class")
##
## Variables actually used in tree construction:
## [1] international_plan            number_customer_service_calls
## [3] total_day_charge              total_eve_charge
## [5] total_intl_calls              total_intl_minutes
## [7] total_night_minutes           voice_mail_plan
##
## Root node error: 363/2501 = 0.14514
##
## n= 2501
##
##         CP nsplit rel error  xerror     xstd
## 1 0.085399      0   1.00000 1.00000 0.048528
## 2 0.081267      1   0.91460 0.99449 0.048417
## 3 0.074380      3   0.75207 0.85399 0.045398
## 4 0.056474      4   0.67769 0.73554 0.042544
## 5 0.027548      7   0.49587 0.52066 0.036413
## 6 0.019284      8   0.46832 0.49862 0.035696
## 7 0.011019     11   0.41047 0.47107 0.034771
## 8 0.010000     15   0.36639 0.45179 0.034103
```

```
plotcp(Dec_Tree.model)
```




```
#The base model accuracy as seen above is 93.99% (94% approx)

# Pre-Pruning
# Growing a tree with minsplit of 50 and maxdepth of 6
Dec_Tree.model_preprun <- rpart(churn ~ ., data = Train_Data, method = "class", control = rpart.control

# predicting the above pre-pruned tree on the validation set
churn.dec.validate.preprun <- predict(Dec_Tree.model_preprun, Validation_Data, type = "prob")
churn.dec.validate.preprun.df <- cbind(Validation_Data,churn.dec.validate.preprun)

#Optimal K
ROC_pred_dec.pre_test <- prediction(churn.dec.validate.preprun[,1],churn.dec.validate.preprun.df$churn)

ROCR_perf_dec.pre_test <- performance(ROC_pred_dec.pre_test,'tpr','fpr')

acc_dec.pre_perf <- performance(ROC_pred_dec.pre_test,"acc")

ROC_pred_dec.pre_test@cutoffs[[1]][which.max(acc_dec.pre_perf@y.values[[1]])]
```

```
##      3169
## 0.7857143
```

```
#AUC Value
roc.curve(churn.dec.validate.preprun.df$churn,churn.dec.validate.preprun[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.890
```

```
#Calculating Accuracy
churn.dec.validate.preprun.df$prob <- as.factor(ifelse(churn.dec.validate.preprun.df$`1`>0.7857143,1,0))

accuracy_preprun <- mean(churn.dec.validate.preprun.df$churn==churn.dec.validate.preprun.df$prob)
accuracy_preprun
```

```
## [1] 0.921875
```

```
#Post- Pruning
# Pruning the Dec_Tree.model based on the optimal cp value
Dec_tree.model_pruned <- prune(Dec_Tree.model, cp = 0.0100)

#predicting the above pruned tree on the validation set
churn.dec.validate.pruned <- predict(Dec_tree.model_pruned, Validation_Data, type = "prob")
churn.dec.validate.pruned.df <- cbind(Validation_Data,churn.dec.validate.pruned)

#Optimal K
ROC_pred_dec.pos_test <- prediction(churn.dec.validate.pruned[,1],churn.dec.validate.pruned.df$churn)

ROCR_perf_dec.pos_test <- performance(ROC_pred_dec.pos_test,'tpr','fpr')

acc_dec.pos_perf <- performance(ROC_pred_dec.pos_test,"acc")

ROC_pred_dec.pos_test@cutoffs[[1]][which.max(acc_dec.pos_perf@y.values[[1]])]
```

```
##      3304
## 0.3076923
```

```
#AUC Value
roc.curve(churn.dec.validate.pruned.df$churn,churn.dec.validate.pruned[,1], plotit = F)
```

```
## Area under the curve (AUC): 0.899
```

```
#Calculating Accuracy
churn.dec.validate.pruned.df$prob <- as.factor(ifelse(churn.dec.validate.pruned.df$`1`>0.3076923,1,0))

accuracy_postprun <- mean(churn.dec.validate.pruned.df$churn==churn.dec.validate.pruned.df$prob)
accuracy_postprun
```

```
## [1] 0.9399038
```

```
#Comparing the base mode, pre-pruning model and post pruning model's accuracy
#Base model accuracy = 0.9399038
data.frame(accuracy_preprun, accuracy_postprun)
```

```
##   accuracy_preprun accuracy_postprun
## 1         0.921875         0.9399038
```

*Pruning can not have significant impact when the data is imbalanced and this can be a possible reason to not see any change in the accuracy in "post - pruning model". We are thereby affirming to the base model and using the base model (Dec_Tree_Model) to predict the test set.*

*Prediction - Test Set*

```r
#Re-coding the variables as being used in the train set
Customers_To_Predict$international_plan <- ifelse(Customers_To_Predict$international_plan =="yes",1,0)
Customers_To_Predict$voice_mail_plan <- ifelse(Customers_To_Predict$voice_mail_plan =="yes",1,0)

#Predicting the decision tree model built over the unseen data
dec.test <- predict(Dec_Tree.model,Customers_To_Predict,type="prob")
churn.dec.test <- cbind(Customers_To_Predict,dec.test)

#Setting the baseline model cutoff point i.e. 0.3076923 on the test set
churn.dec.test$prob <- as.factor(ifelse(churn.dec.test$`1`>0.3076923,"yes","no"))

#Deleting the probability columns 1 and 0
churn.dec.test <- churn.dec.test[,-c(20:21)]
```

*The final file to look for the churns and no churns is the churn.dec.test.*