Dongyu Zhang Final Project

Dongyu Zhang

2017/04/30

```
library(C50)
library(ggplot2)
library(fiftystater)
library(gridExtra)
library(caret)
library(pROC)
library(corrplot)
library(rpart)
library(ipred)
library(randomForest)
library(xgboost)
```

Exploring the data

What the rows and columns of the data represent

```
data(churn)
colnames(churnTrain)
```

```
##  [1] "state"                          "account_length"
##  [3] "area_code"                      "international_plan"
##  [5] "voice_mail_plan"                "number_vmail_messages"
##  [7] "total_day_minutes"              "total_day_calls"
##  [9] "total_day_charge"               "total_eve_minutes"
## [11] "total_eve_calls"                "total_eve_charge"
## [13] "total_night_minutes"            "total_night_calls"
## [15] "total_night_charge"             "total_intl_minutes"
## [17] "total_intl_calls"               "total_intl_charge"
## [19] "number_customer_service_calls" "churn"
```

Each row represents the account message of a customer account. Each column represents the predictors and outcome.
state: the customer account state.
account_length: the length of the account.
area_code: the customer account area code.
international plan: if the customer has internation plan.
voice mail plan: if the customer has voice plan.
number vmail message: number of voice message.
total day minutes: total time of calls in the daytime.
total day calls: total number of calls in the daytime.
total day charge: total charge of calls in the daytime.
total eve minutes: total time of calls in the evening.
total eve calls: total number of calls in the evening.
total eve charge: total charge of calls in the evening.
total night minutes: total time of calls at night.
total night calls: total number of calls at night.

total night charge: total charge of calls at night.
total intl minutes: total time of international calls.
total intl calls: total number of international calls.
total intl charge: total charge of international calls.
number customer service calls: number of customer service calls.
churn: if the customer churned.

overall churn rate

```
(table(churnTrain$churn)[1]+table(churnTest$churn)[1])/
  (sum(table(churnTrain$churn))+sum(table(churnTest$churn)))
```

```
##    yes
## 0.1414
```

The overall churn rate is 0.1414.

Useful or interesting findings

The relation between state and if the customer churn

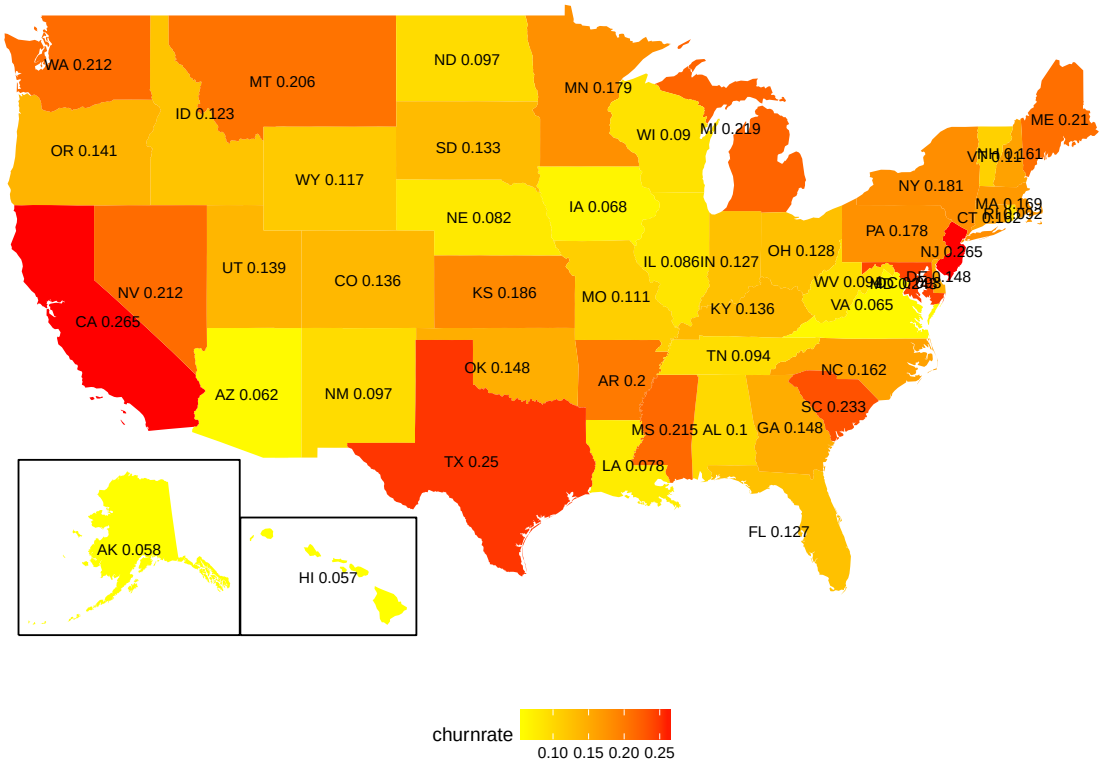
I want to know the churn rate in each state. First, I compute the churn rate in each state and store the result in a dataframe.

```
getchurnrate <- function(){
  stateChurnRate <- data.frame(state = c(), churnrate =c())
  for(i in 1:length(levels(churnTrain$state))){
    cr <- table(churnTrain$churn[which(churnTrain$state == levels(churnTrain$state)[i])])[1]/
      sum(table(churnTrain$churn[which(churnTrain$state == levels(churnTrain$state)[i])]))
    newrow <- data.frame(state = levels(churnTrain$state)[i], churnrate = cr)
    stateChurnRate <- rbind(stateChurnRate, newrow)
  }
  return(stateChurnRate)
}

stateChurnRate <- getchurnrate()

rownames(stateChurnRate) <-NULL
stateChurnRate$statename<-sapply(stateChurnRate$state,
                                 function(x) tolower(state.name[grep(x, state.abb)]))
stateChurnRate$statename[which(stateChurnRate$state == 'DC')] <- 'district of columbia'
colnames(stateChurnRate) <- c("state_Abbr", "churnrate", "state")
stateChurnRate$state <- unlist(stateChurnRate$state)

snames <- aggregate(cbind(long, lat) ~ id, data=fifty_states,
                    FUN=function(x) median(range(x)))
colnames(snames)[1] <- 'state'
stateChurnRate <- merge(stateChurnRate, snames, by = 'state')
```

Then, I used the dataframe I just created to plot a bar chart.

```
State_Churn_Rate <- ggplot(stateChurnRate, aes(map_id = state)) +
  geom_map(aes(fill = churnrate), map = fifty_states)+
  expand_limits(x = fifty_states$long, y = fifty_states$lat) +
  coord_map() +
```

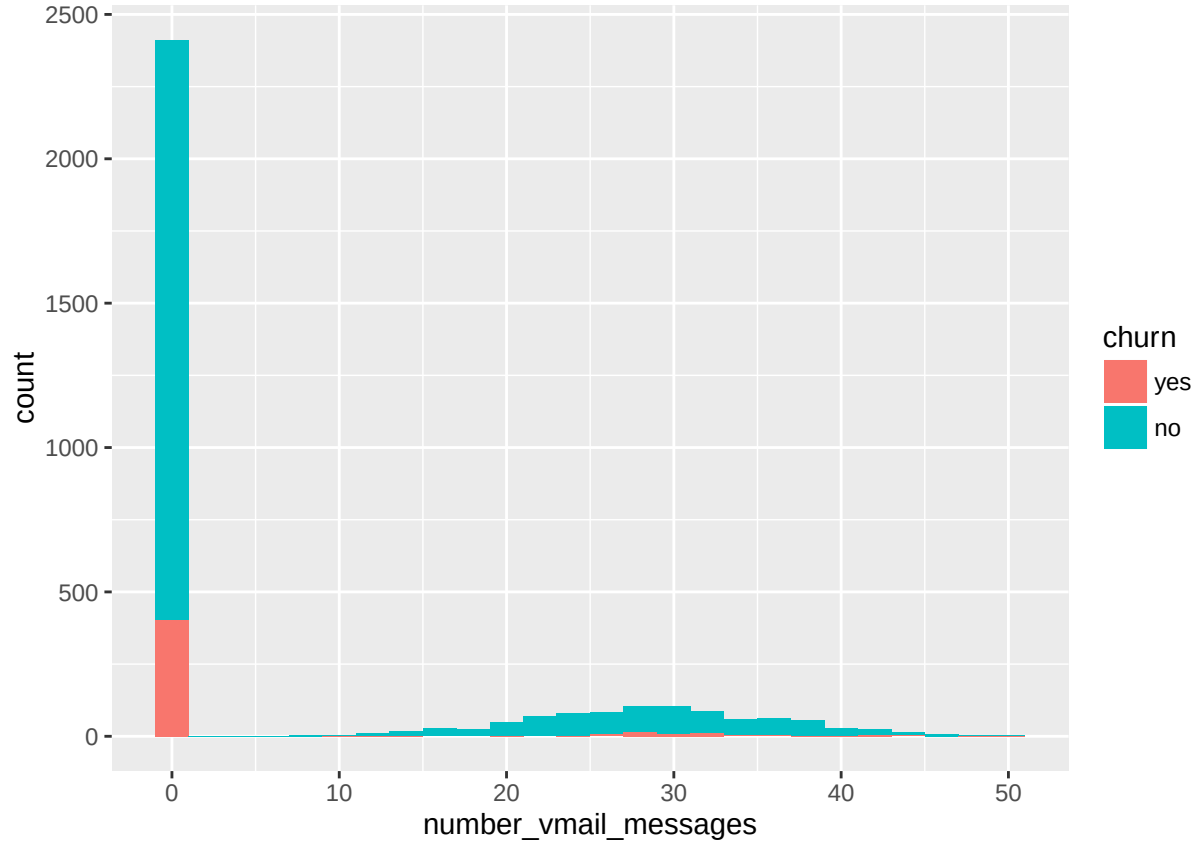
```
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  labs(x = "", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())+
  fifty_states_inset_boxes()+
  geom_text(data=stateChurnRate,
            aes(long, lat, label =paste(state_Abbr,round(churnrate,3))), size=3)+
  scale_fill_continuous(low = "yellow", high = 'red')
State_Churn_Rate
```



As the plot shows, the customers in California and New Jersey are the most likely to churn. While customer in Hawaii has the lowest probability to churn. The maximum churn rate is about 5 times of the minimum churn rate.

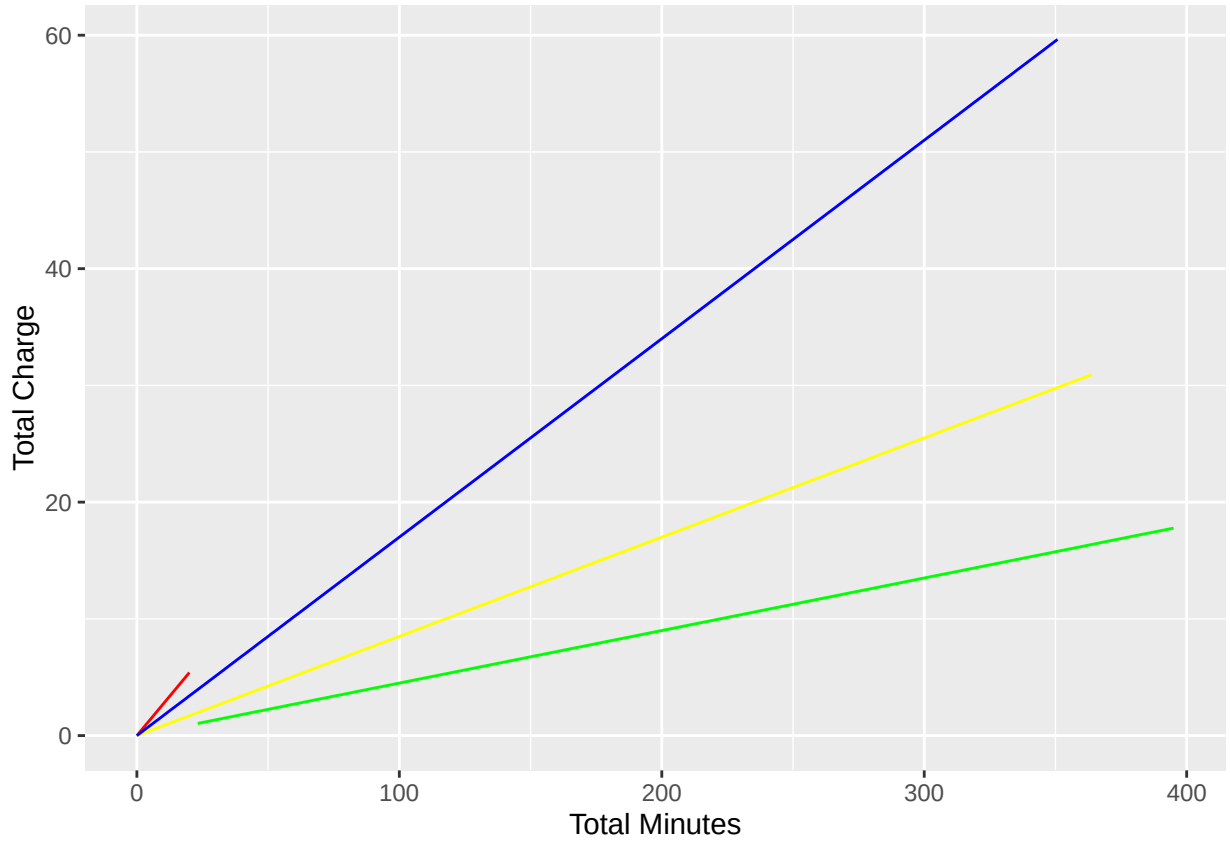
The distribution of number of voicemail messages

```
vmail_churn <- ggplot(churnTrain, aes(x = number_vmail_messages, fill = churn))+
  geom_histogram(binwidth = 2)
vmail_churn
```



As we can see, most people does not receive voicemail messages, only small proportion of people receive voicemail messages.

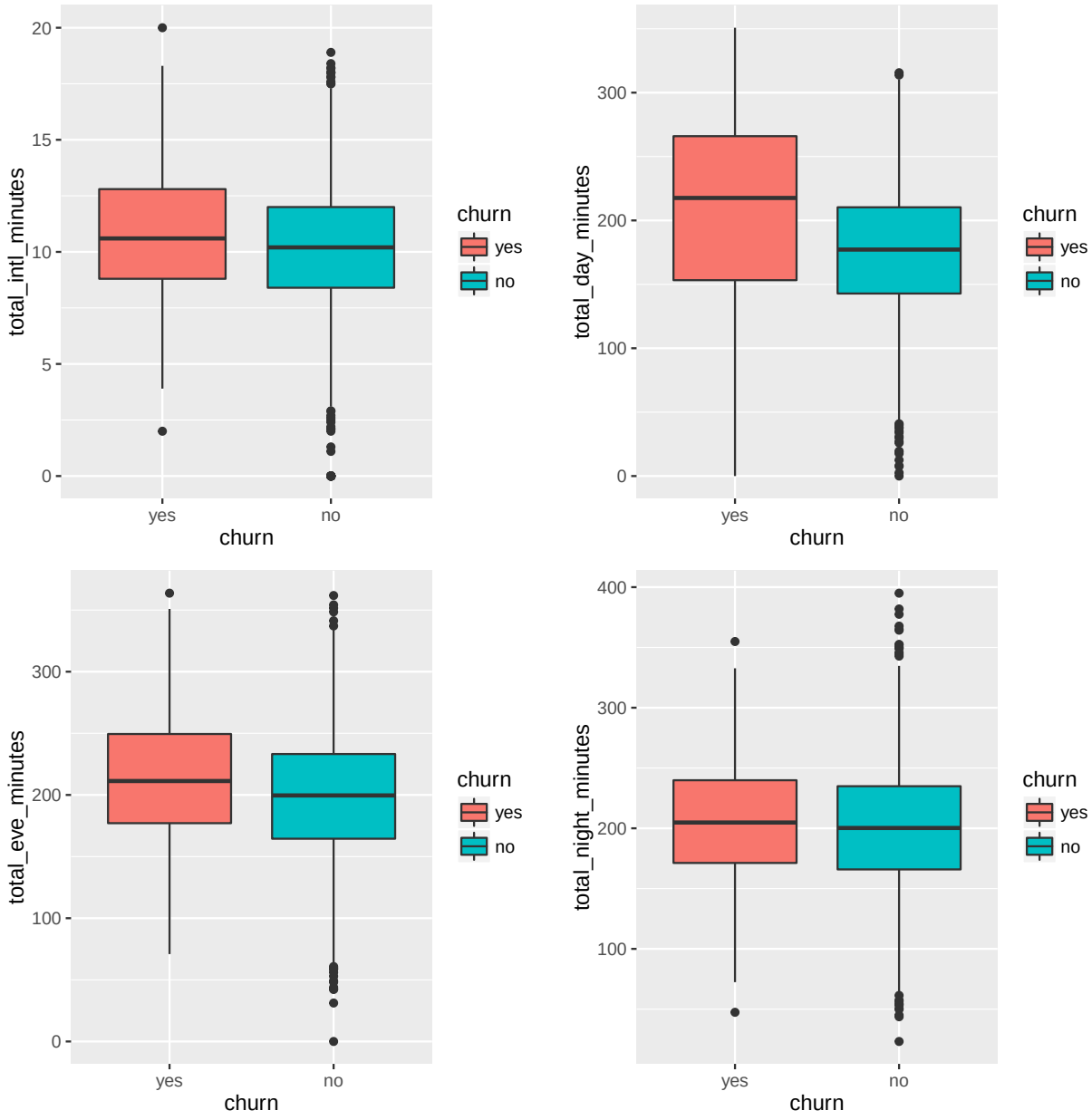
```
Charge_Mins <- ggplot(data = churnTrain)+
  geom_line(aes(x = total_intl_minutes, y = total_intl_charge), colour='red')+
  geom_line(aes(x = total_eve_minutes, y = total_eve_charge), colour='yellow')+
  geom_line(aes(x = total_night_minutes, y = total_night_charge), colour='green')+
  geom_line(aes(x = total_day_minutes, y = total_day_charge), colour ='blue')+
  scale_x_continuous(name = 'Total Minutes')+scale_y_continuous(name = 'Total Charge')
Charge_Mins
```



In the graph above, the red line represents the international call, the blue line represents the day call, the yellow line represents the evening call, the green line represents the night call.

As we can see, for each time period and the international call, the total minutes of call is highly correlated with the total charge.

```
intl_box <- ggplot(churnTrain)+
  geom_boxplot(aes(y = total_intl_minutes, x = churn, fill = churn))
day_box <- ggplot(churnTrain)+
  geom_boxplot(aes(y = total_day_minutes, x = churn, fill = churn))
eve_box <- ggplot(churnTrain)+
  geom_boxplot(aes(y = total_eve_minutes, x = churn, fill = churn))
night_box <- ggplot(churnTrain)+
  geom_boxplot(aes(y = total_night_minutes, x = churn, fill = churn))
grid.arrange(intl_box, day_box, eve_box, night_box)
```



As the graph shows, the customers who churned in average have more call time than the customers who did not churn.

Build an interpretable model and measure its performance

logistic regression

First, create dummy variable for each area code, then combine the dummy variable with the origin dataset.

```
churn<- rbind(churnTest, churnTrain)
dummy_state <- class2ind(churn$state)[,-1]
dummy_area_code <- class2ind(churn$area_code)[,-1]
dummy_int <- class2ind(churn$international_plan)[,-1]
dummy_voice <- class2ind(churn$voice_mail_plan)[,-1]
```

```
churn$churn <- relevel(churn$churn, "no")
combined <- cbind(churn, dummy_state, dummy_area_code, dummy_int, dummy_voice)
combined_test <- combined[1:1667,]
combined_train <- combined[1668:5000,]
```

Make the formula for the logistic regression model

```
input_features <- colnames(combined_train)[c(-1,-3, -4, -5, -20)]
make_formula <- function(input_features){
  input_features_string <- paste(input_features, collapse = ' + ')
  formula_string <- paste('churn ~ ', input_features_string)
  formula <- as.formula(formula_string)
  return(formula)
}
```

At first, I inclued all the variable in the model. Because there might be colinearity and degenerate variable in the model. I use stepwise regression to filter out the meaningless variable by choosing the model with lower AIC.

```
firstmodel <- glm(make_formula(input_features),
                  data = combined_train, family = binomial())
logstep <- step(firstmodel, trace = 0, direction = "backward")
summary(logstep)
```

```
##
## Call:
## glm(formula = churn ~ number_vmail_messages + total_day_calls +
##     total_day_charge + total_eve_minutes + total_night_charge +
##     total_intl_calls + total_intl_charge + number_customer_service_calls +
##     CA + HI + IL + MD + ME + MI + MN + MS + MT + NJ + NV + NY +
##     RI + SC + TX + VA + WA + dummy_int + dummy_voice, family = binomial(),
##     data = combined_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1243  -0.5019  -0.3220  -0.1732   3.0277
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -8.855054   0.608829 -14.544  < 2e-16 ***
## number_vmail_messages          0.035218   0.018430   1.911 0.056017 .
## total_day_calls                0.004102   0.002806   1.462 0.143759
## total_day_charge               0.077003   0.006472  11.898  < 2e-16 ***
## total_eve_minutes              0.007609   0.001168   6.515 7.26e-11 ***
## total_night_charge             0.088096   0.025294   3.483 0.000496 ***
## total_intl_calls              -0.092686   0.025450  -3.642 0.000271 ***
## total_intl_charge              0.318081   0.077484   4.105 4.04e-05 ***
## number_customer_service_calls  0.526673   0.040085  13.139  < 2e-16 ***
## CA                             1.164243   0.461348   2.524 0.011617 *
## HI                            -0.846926   0.632706  -1.339 0.180709
## IL                            -0.856886   0.540813  -1.584 0.113094
## MD                             0.499485   0.336583   1.484 0.137811
## ME                             0.691218   0.359070   1.925 0.054226 .
## MI                             0.737346   0.329444   2.238 0.025211 *
## MN                             0.507301   0.335072   1.514 0.130025
```

```
## MS                            0.703499   0.360069   1.954 0.050726 .
## MT                            1.200440   0.336866   3.564 0.000366 ***
## NJ                            0.937736   0.321311   2.918 0.003518 **
## NV                            0.594511   0.355010   1.675 0.094007 .
## NY                            0.523256   0.343928   1.521 0.128157
## RI                           -0.767867   0.521448  -1.473 0.140868
## SC                            1.123577   0.375922   2.989 0.002800 **
## TX                            1.003340   0.316927   3.166 0.001546 **
## VA                           -1.074627   0.524251  -2.050 0.040381 *
## WA                            0.775803   0.351421   2.208 0.027271 *
## dummy_int                     2.172863   0.150503  14.437  < 2e-16 ***
## dummy_voice                  -2.028208   0.586542  -3.458 0.000544 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2758.3  on 3332  degrees of freedom
## Residual deviance: 2091.2  on 3305  degrees of freedom
## AIC: 2147.2
##
## Number of Fisher Scoring iterations: 6
```

```
logpred <- predict(firstmodel, combined_test, type = "response")
steppred <- predict(logstep, combined_test, type = "response")
```

I got the final model and compare its performance with the raw model.

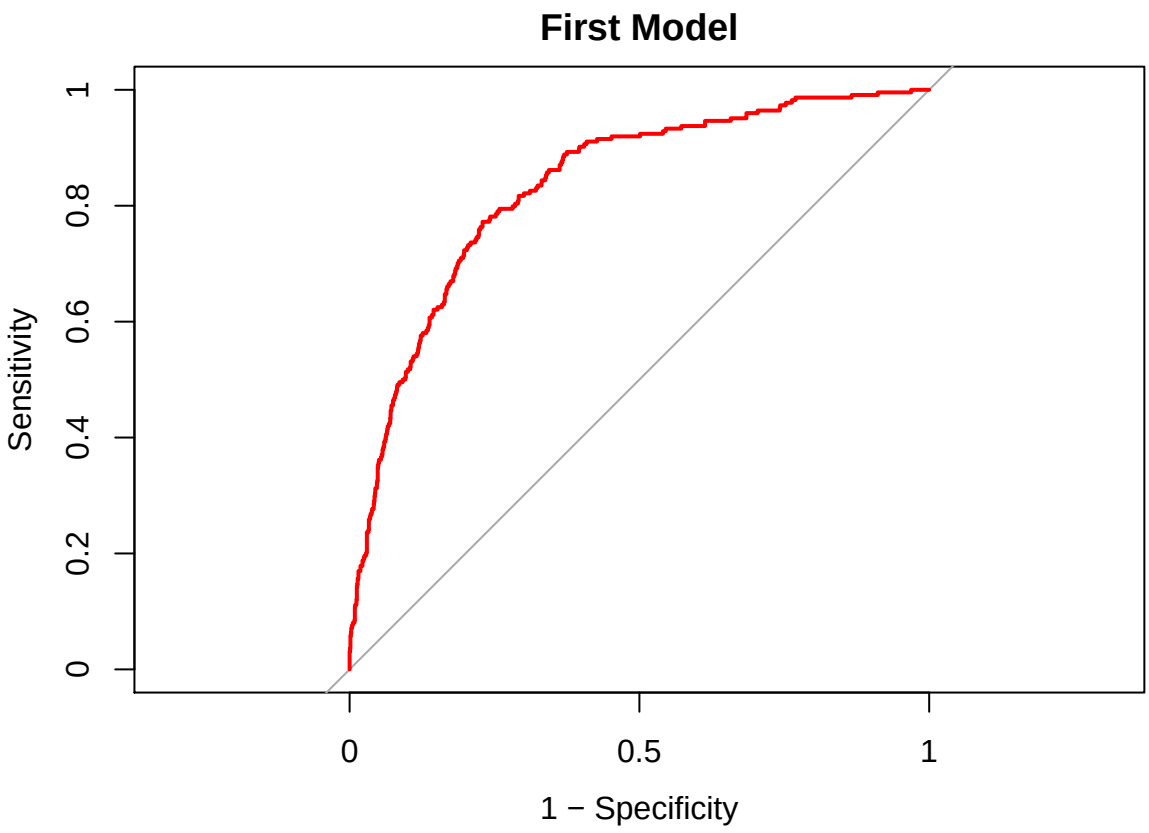
```
firstroc <- roc(response = combined_test$churn, predictor = logpred)
steproc <- roc(response = combined_test$churn, predictor = steppred)

print(paste("The first model AUC:", auc(firstroc)))
```

```
## [1] "The first model AUC: 0.833082739332739"
```

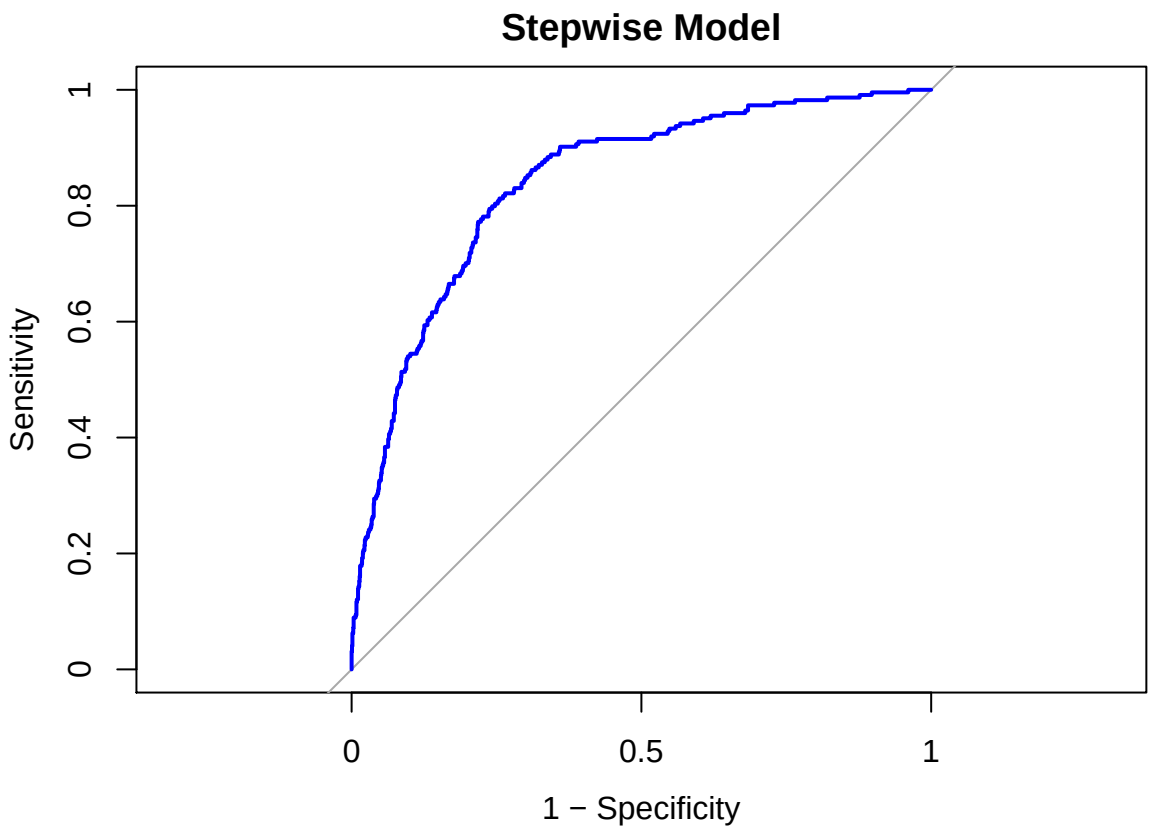
```
print(paste("The stepwise model AUC:", auc(steproc)))
```

```
## [1] "The stepwise model AUC: 0.839854964854965"
```

```
plot(firstroc, legacy.axes = TRUE, main = 'First Model', col= 'red')
```




```
plot(steproc, legacy.axes = TRUE, main = 'Stepwise Model', col = 'blue')
```



The AUC of the final model is lightly bigger than the AUC of the raw model. So I will keep the final model, since it is more interpertable with better performance.

```
coefmessage <- cbind(coefficient = coef(logstep),
      odds_ratio = exp(coef(logstep)),
      odds_ratio_lcl = exp(confint(logstep))[,1],
      odds_ratio_ucl = exp(confint(logstep))[,2])[order(-coef(logstep)),]
round(coefmessage,5)
```

```
##                                coefficient odds_ratio odds_ratio_lcl
## dummy_int                          2.17286    8.78340        6.54652
## MT                                 1.20044    3.32158        1.66202
## CA                                 1.16424    3.20350        1.24268
## SC                                 1.12358    3.07584        1.43181
## TX                                 1.00334    2.72738        1.43657
## NJ                                 0.93774    2.55419        1.32983
## WA                                 0.77580    2.17234        1.05793
## MI                                 0.73735    2.09038        1.06729
## MS                                 0.70350    2.02081        0.96626
## ME                                 0.69122    1.99615        0.95154
## NV                                 0.59451    1.81214        0.87640
## number_customer_service_calls      0.52667    1.69329        1.56615
## NY                                 0.52326    1.68751        0.83088
## MN                                 0.50730    1.66080        0.83294
## MD                                 0.49949    1.64787        0.83116
## total_intl_charge                  0.31808    1.37449        1.18187
## total_night_charge                 0.08810    1.09209        1.03940
```

```
## total_day_charge                  0.07700    1.08005        1.06657
## number_vmail_messages             0.03522    1.03585        0.99935
## total_eve_minutes                 0.00761    1.00764        1.00535
## total_day_calls                   0.00410    1.00411        0.99861
## total_intl_calls                 -0.09269    0.91148        0.86642
## RI                               -0.76787    0.46400        0.15258
## HI                               -0.84693    0.42873        0.09893
## IL                               -0.85689    0.42448        0.13139
## VA                               -1.07463    0.34143        0.10752
## dummy_voice                      -2.02821    0.13157        0.04044
## (Intercept)                      -8.85505    0.00014        0.00004
##                               odds_ratio_ucl
## dummy_int                          11.81473
## MT                                  6.27747
## CA                                  7.69146
## SC                                  6.29250
## TX                                  5.00350
## NJ                                  4.71373
## WA                                  4.22632
## MI                                  3.90692
## MS                                  3.99309
## ME                                  3.92375
## NV                                  3.55043
## number_customer_service_calls       1.83281
## NY                                  3.22038
## MN                                  3.12031
## MD                                  3.12792
## total_intl_charge                   1.60153
## total_night_charge                  1.14779
## total_day_charge                    1.09399
## number_vmail_messages               1.07429
## total_eve_minutes                   1.00996
## total_day_calls                     1.00966
## total_intl_calls                    0.95734
## RI                                  1.20103
## HI                                  1.28363
## IL                                  1.12983
## VA                                  0.86640
## dummy_voice                         0.40379
## (Intercept)                         0.00046
```

Policy based on this model

According to the output, the final model included 24 predictors. If the customer has international plan, The number of voice messages, the number of customer service calls, the total call minutes (or the charge) drive customer churn. Also, the customers in some certain state have a higher chances to leave.

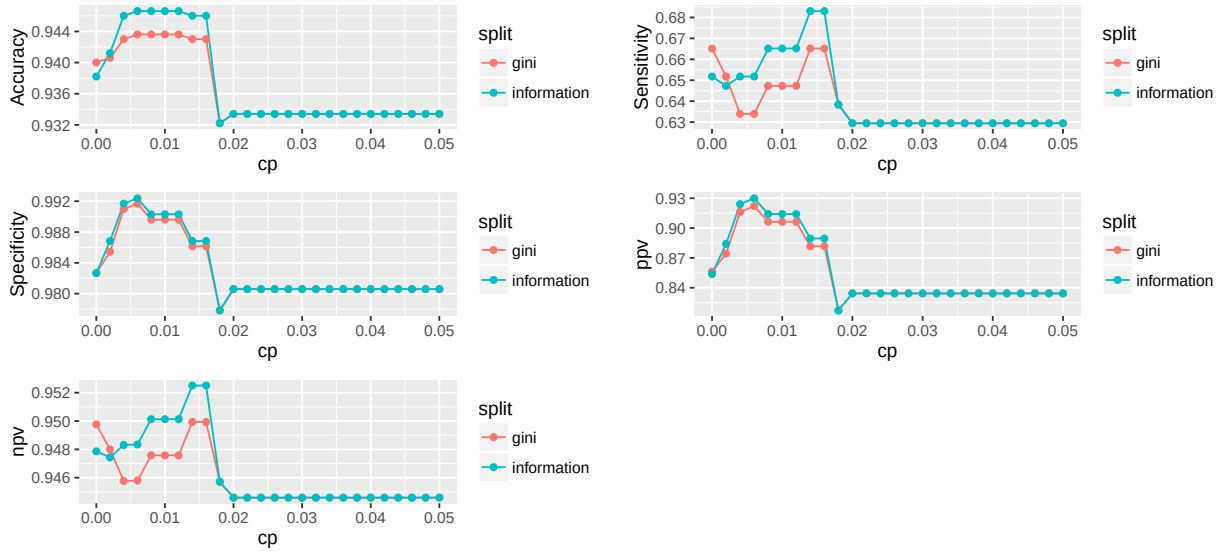
The company should consider offering discount for those people who have internation plan. Because there are high chance that those people will churn. The company should consider making special plan for the customer who has made more service calls than the other people. Because those customers may have more complaint than others. So the company can provide those customer some additional service to increase their satifaction. For the people who have more call time. They are usually charged more than others. The company should consider offering discount for those customer. The company also need to focus on several states, like California and Texas, etc. Trying to figure out the reason drving customer churn in these states.

Build the best tree-based predictive model you can and measure its performance

classification tree

```
fit_tree <- list()
spl <- c("gini", "information")
cpl <- seq(0, 0.05, 0.002)
classperform <- data.frame(split = c(), cp = c(), Accuracy = c(),
                           Sensitivity = c(), Specificity = c(), ppv = c(), npv = c())
for(i in 1:length(spl)){
  fit_tree[[i]] <- list()
  for(e in 1:length(cpl)){
    fit_tree[[i]][[e]] <- rpart(make_formula(input_features),
                                data=combined_train, method = 'class',
                                parms = list(split = spl[i]),
                                control = list(cp = cpl[e]))
    preds_tree <- predict(fit_tree[[i]][[e]], combined_test)
    preds_result <- c()
    preds_result[preds_tree[,2] >= .5] <- 1
    preds_result[preds_tree[,2] < .5] <- 0
    preds_result <- factor(preds_result, levels = c(0,1), labels = c("no", "yes"))
    ccfm <- confusionMatrix(data = preds_result,
                            reference = combined_test$churn, positive = 'yes')
    cAcc <- ccfm$overall["Accuracy"]
    cSen <- ccfm$byClass["Sensitivity"]
    cSpe <- ccfm$byClass["Specificity"]
    cPPV <- ccfm$byClass["Pos Pred Value"]
    cNPV <- ccfm$byClass["Neg Pred Value"]
    cnewrow <- data.frame(split = spl[i], cp = cpl[e], Accuracy = cAcc,
                          Sensitivity = cSen, Specificity = cSpe,
                          ppv = cPPV, npv = cNPV)
    classperform <- rbind(classperform, cnewrow)
  }
}
```

```
rownames(classperform) <- NULL
class_ac <- ggplot(data = classperform)+
  geom_line(aes(x = cp, y = Accuracy, colour = split))+
  geom_point(aes(x = cp, y = Accuracy, colour = split))
class_sen <-ggplot(data = classperform)+
  geom_line(aes(x = cp, y = Sensitivity, colour = split))+
  geom_point(aes(x = cp, y = Sensitivity, colour = split))
class_spe <- ggplot(data = classperform)+
  geom_line(aes(x = cp, y = Specificity, colour = split))+
  geom_point(aes(x = cp, y = Specificity, colour = split))
class_ppv <- ggplot(data = classperform)+
  geom_line(aes(x = cp, y = ppv, colour = split))+
  geom_point(aes(x = cp, y = ppv, colour = split))
class_npv <- ggplot(data = classperform)+
  geom_line(aes(x = cp, y = npv, colour = split))+
  geom_point(aes(x = cp, y = npv, colour = split))
grid.arrange(class_ac, class_sen, class_spe, class_ppv, class_npv, ncol = 2)
```




```
classperform[which(classperform$Accuracy == max(classperform$Accuracy)),]
```

```
##          split    cp  Accuracy Sensitivity Specificity       ppv       npv
## 30 information 0.006 0.9466107   0.6517857    0.992377 0.9299363 0.9483444
## 31 information 0.008 0.9466107   0.6651786    0.990298 0.9141104 0.9501330
## 32 information 0.010 0.9466107   0.6651786    0.990298 0.9141104 0.9501330
## 33 information 0.012 0.9466107   0.6651786    0.990298 0.9141104 0.9501330
```

```
classperform[which(classperform$Sensitivity == max(classperform$Sensitivity)),]
```

```
##          split    cp  Accuracy Sensitivity Specificity       ppv       npv
## 34 information 0.014 0.9460108   0.6830357    0.986833 0.8895349 0.9525084
## 35 information 0.016 0.9460108   0.6830357    0.986833 0.8895349 0.9525084
```

```
classperform[which(classperform$ppv == max(classperform$ppv)),]
```

```
##          split    cp  Accuracy Sensitivity Specificity       ppv       npv
## 30 information 0.006 0.9466107   0.6517857    0.992377 0.9299363 0.9483444
```

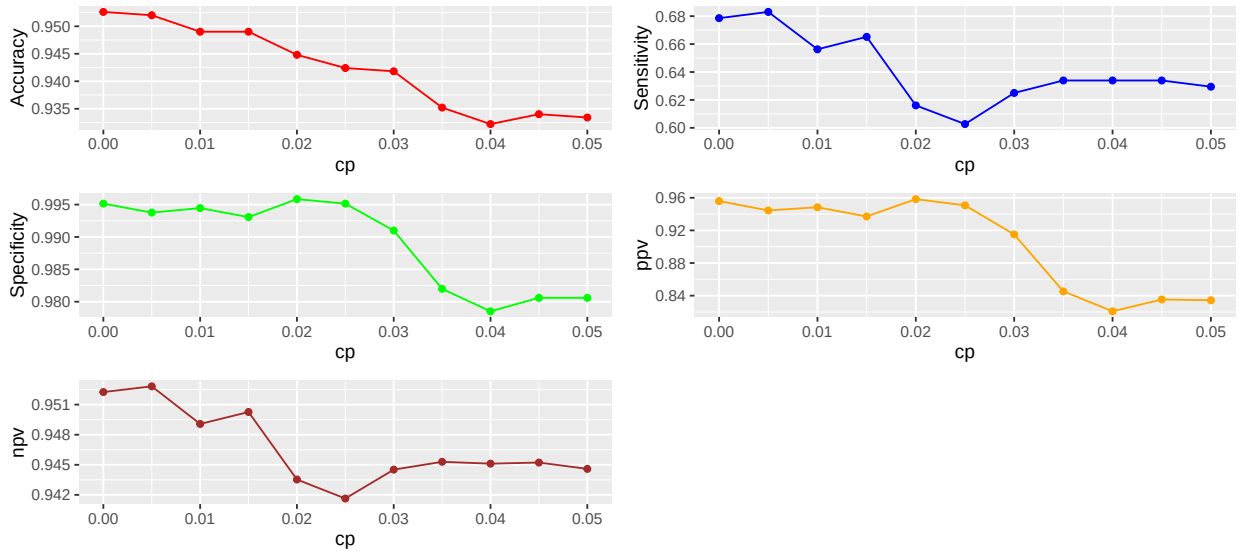
For the single tree model, the model with the highest Accuracy, Sensitivity and PPV are shown above.

bagging

```
set.seed(63)
fit_bag <- list()
cpl <- seq(0, 0.05, 0.005)
bagperform <- data.frame(cp = c(), Accuracy = c(),
                         Sensitivity = c(), Specificity = c(), ppv = c(), npv = c())
for(e in 1:length(cpl)){
  fit_bag[[e]] <- bagging(make_formula(input_features),
                          data = combined_train,
                          coob = TRUE,
                          control = rpart.control(cp = cpl[e]))
  preds_bag <- predict(fit_bag[[e]], combined_test)
  bacfm <- confusionMatrix(data = preds_bag,
                           reference = combined_test$churn, positive = 'yes')
  baAcc <- bacfm$overall["Accuracy"]
```

```
  baSen <- bacfm$byClass["Sensitivity"]
  baSpe <- bacfm$byClass["Specificity"]
  baPPV <- bacfm$byClass["Pos Pred Value"]
  baNPV <- bacfm$byClass["Neg Pred Value"]
  banewrow <- data.frame(cp = cpl[e], Accuracy = baAcc,
                         Sensitivity = baSen, Specificity = baSpe,
                         ppv = baPPV, npv = baNPV)
  bagperform <- rbind(bagperform, banewrow)
}
```

```
rownames(bagperform) <- NULL
ba_ac <- ggplot(data = bagperform)+
  geom_line(aes(x = cp, y = Accuracy), color = 'red')+
  geom_point(aes(x = cp, y = Accuracy), color = 'red')+
  theme(legend.key.size=unit(0.3,'cm'))
ba_sen <-ggplot(data = bagperform)+
  geom_line(aes(x = cp, y = Sensitivity), color = 'blue')+
  geom_point(aes(x = cp, y = Sensitivity), color = 'blue')+
  theme(legend.key.size=unit(0.3,'cm'))
ba_spe <- ggplot(data = bagperform)+
  geom_line(aes(x = cp, y = Specificity), colour = 'green')+
  geom_point(aes(x = cp, y = Specificity), colour = 'green')+
  theme(legend.key.size=unit(0.3,'cm'))
ba_ppv <- ggplot(data = bagperform)+
  geom_line(aes(x = cp, y = ppv), colour = 'orange')+
  geom_point(aes(x = cp, y = ppv), colour = 'orange')
ba_npv <- ggplot(data = bagperform)+
  geom_line(aes(x = cp, y = npv), colour = 'brown')+
  geom_point(aes(x = cp, y = npv), colour = 'brown')
grid.arrange(ba_ac, ba_sen, ba_spe, ba_ppv, ba_npv, ncol = 2)
```




```
bagperform[which(bagperform$Accuracy == max(bagperform$Accuracy)),]
```

```
##   cp  Accuracy Sensitivity Specificity       ppv       npv
## 1  0 0.9526095   0.6785714    0.995149 0.9559748 0.9522546
```

```
bagperform[which(bagperform$Sensitivity == max(bagperform$Sensitivity)),]
```

```
##      cp  Accuracy Sensitivity Specificity       ppv       npv
## 2 0.005 0.9520096   0.6830357    0.993763 0.9444444 0.9528239
```

```
bagperform[which(bagperform$ppv == max(bagperform$ppv)),]
```

```
##     cp Accuracy Sensitivity Specificity       ppv       npv
## 5 0.02 0.944811   0.6160714    0.995842 0.9583333 0.9435325
```

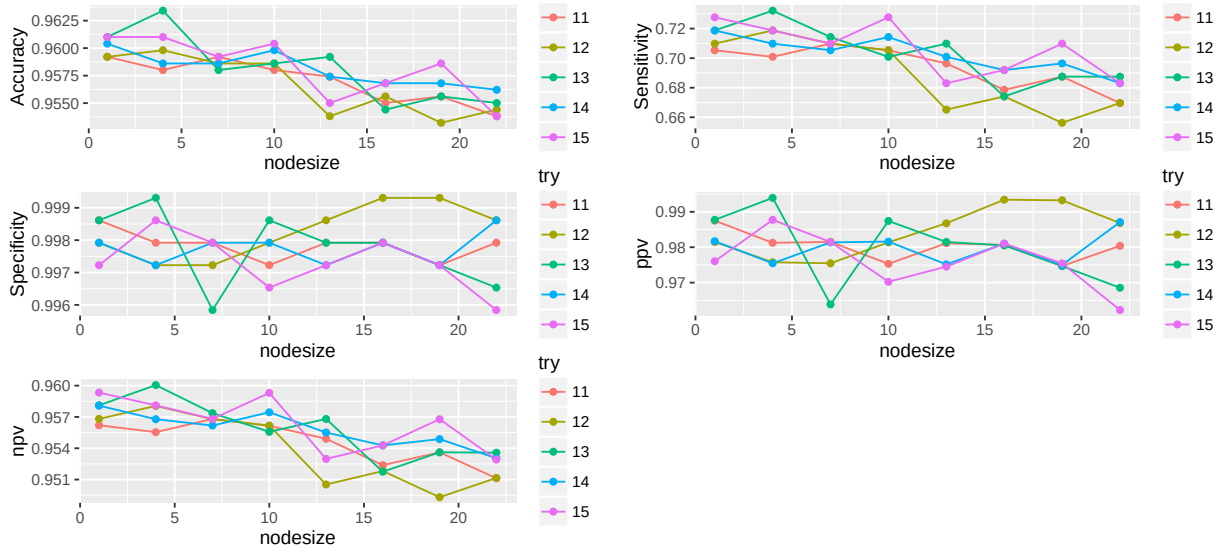
For the bagging tree model, the model with the highest Accuracy, Sensitivity and PPV are shown above.

randomForest

```
set.seed(1988)
fit_rf <- list()
nodesizel <- seq(1,22,3)
tryl <- c(11:15)
rfperform <- data.frame(try = c(), nodesize = c(), Accuracy = c(),
                        Sensitivity = c(), Specificity = c(), ppv = c(), npv = c())
for(e in 1:length(tryl)){
  fit_rf[[e]] <- list()
  for(i in 1:length(nodesizel)){
    fit_rf[[e]][[i]] <- randomForest(make_formula(input_features),
                                     data = combined_train,
                                     ntree = 120,
                                     mtry = tryl[e],
                                     replace = TRUE,
                                     nodesize = nodesizel[i],
                                     do.trace = FALSE)
    pred_rf <- predict(fit_rf[[e]][[i]], combined_test)
    rcfm <- confusionMatrix(data = pred_rf,
                            reference = combined_test$churn, positive = 'yes')
    rAcc <- rcfm$overall["Accuracy"]
    rSen <- rcfm$byClass["Sensitivity"]
    rSpe <- rcfm$byClass["Specificity"]
    rPPV <- rcfm$byClass["Pos Pred Value"]
    rNPV <- rcfm$byClass["Neg Pred Value"]
    rnewrow <- data.frame(try = tryl[e], nodesize = nodesizel[i],
                          Accuracy = rAcc, Sensitivity = rSen,
                          Specificity = rSpe, ppv = rPPV, npv = rNPV)
    rfperform <- rbind(rfperform, rnewrow)
  }
}
```

```
rownames(rfperform) <- NULL
rfperform$try <- factor(rfperform$try)
rf_ac <- ggplot(data = rfperform)+
  geom_line(aes(x = nodesize, y = Accuracy, colour = try))+
  geom_point(aes(x = nodesize, y = Accuracy, colour = try))
rf_sen <-ggplot(data = rfperform)+
  geom_line(aes(x = nodesize, y = Sensitivity, colour = try))+
  geom_point(aes(x = nodesize, y = Sensitivity, colour = try))
rf_spe <- ggplot(data = rfperform)+
```

```
  geom_line(aes(x = nodesize, y = Specificity, colour = try))+
  geom_point(aes(x = nodesize, y = Specificity, colour = try))
rf_ppv <- ggplot(data = rfperform)+
  geom_line(aes(x = nodesize, y = ppv, colour = try))+
  geom_point(aes(x = nodesize, y = ppv, colour = try))
rf_npv <- ggplot(data = rfperform)+
  geom_line(aes(x = nodesize, y = npv, colour = try))+
  geom_point(aes(x = nodesize, y = npv, colour = try))
grid.arrange(rf_ac, rf_sen, rf_spe,rf_ppv, rf_npv, ncol = 2)
```




```
rfperform[which(rfperform$Accuracy == max(rfperform$Accuracy)),]
```

```
##    try nodesize  Accuracy Sensitivity Specificity       ppv       npv
## 18  13        4 0.9634073   0.7321429    0.999307 0.9939394 0.9600533
```

```
rfperform[which(rfperform$Sensitivity == max(rfperform$Sensitivity)),]
```

```
##    try nodesize  Accuracy Sensitivity Specificity       ppv       npv
## 18  13        4 0.9634073   0.7321429    0.999307 0.9939394 0.9600533
```

```
rfperform[which(rfperform$ppv == max(rfperform$ppv)),]
```

```
##    try nodesize  Accuracy Sensitivity Specificity       ppv       npv
## 18  13        4 0.9634073   0.7321429    0.999307 0.9939394 0.9600533
```

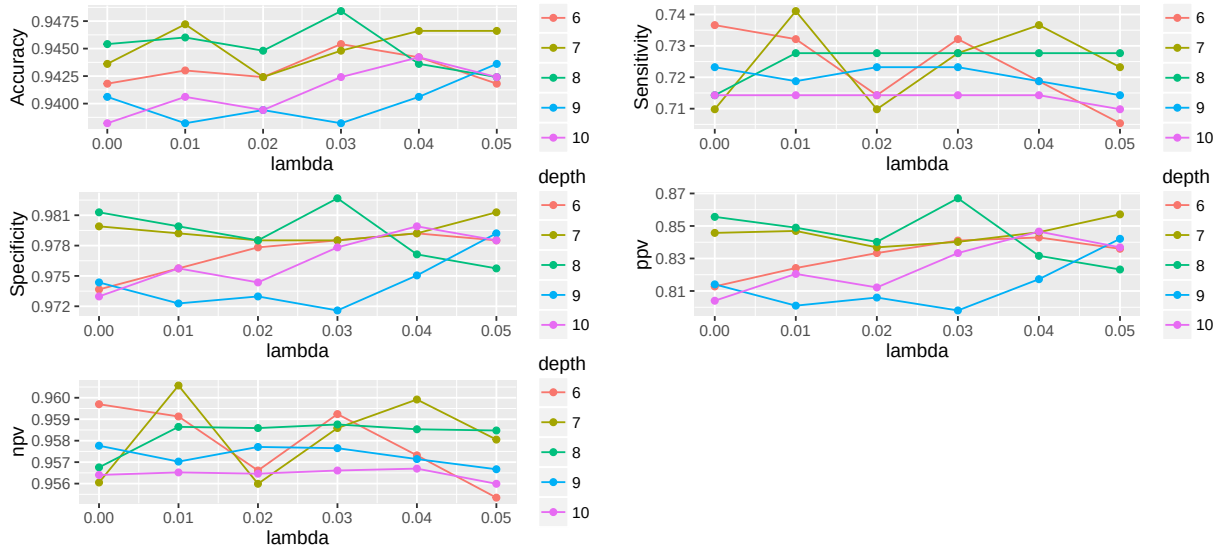
For the random forest model, the model with the highest Accuracy, Sensitivity and PPV are shown above.

boosting

```
set.seed(611)
X_matrix <- as.matrix(combined_train[,c(-1,-3, -4, -5, -20)])
y_matrix <- as.matrix(class2ind(combined_train$churn)[,2])
dtrain <- xgb.DMatrix(X_matrix, label =y_matrix)
depthl <- c(6:10)
lambdal <- seq(.0, .05, .01)
fit_bo <- list()
```

```
boperform <- data.frame(depth = c(), lambda = c(),  Accuracy = c(),
                        Sensitivity = c(), Specificity = c(), ppv = c(), npv =c())
for(i in 1: length(depthl)){
  fit_bo[[i]] <- list()
  for(e in 1: length(lambdal)){
    fit_bo[[i]][[e]] <- xgb.train(data = dtrain,
                params = list(silent = 1),
                nrounds = 100,
                max_depth = depthl[i],
                lambda = lambdal[e],
                alpha = .01)
    preds_bo <- predict(fit_bo[[i]][[e]],
                        as.matrix(combined_test[,c(-1,-3, -4, -5, -20)]))
    preds_bo_result <- c()
    preds_bo_result[preds_bo >= .5] <- 1
    preds_bo_result[preds_bo < .5] <- 0
    preds_bo_result <- factor(preds_bo_result, levels = c(0,1), labels = c("no", "yes"))
    bocfm <- confusionMatrix(data = preds_bo_result,
                             reference = combined_test$churn, positive = 'yes')
    boAcc <- bocfm$overall["Accuracy"]
    boSen <- bocfm$byClass["Sensitivity"]
    boSpe <- bocfm$byClass["Specificity"]
    boPPV <- bocfm$byClass["Pos Pred Value"]
    boNPV <- bocfm$byClass["Neg Pred Value"]
    bonewrow <- data.frame(depth = depthl[i], lambda = lambdal[e],
                           Accuracy = boAcc, Sensitivity = boSen,
                           Specificity = boSpe, ppv = boPPV, npv = boNPV)
    boperform <- rbind(boperform, bonewrow)
  }
}
```

```
rownames(boperform) <- NULL
boperform$depth <- factor(boperform$depth)
bo_ac <- ggplot(data = boperform)+
  geom_line(aes(x = lambda, y = Accuracy, colour = depth))+
  geom_point(aes(x = lambda, y = Accuracy, colour = depth))
bo_sen <-ggplot(data = boperform)+
  geom_line(aes(x = lambda, y = Sensitivity, colour = depth))+
  geom_point(aes(x = lambda, y = Sensitivity, colour = depth))
bo_spe <- ggplot(data = boperform)+
  geom_line(aes(x = lambda, y = Specificity, colour = depth))+
  geom_point(aes(x = lambda, y = Specificity, colour = depth))
bo_ppv <- ggplot(data = boperform)+
  geom_line(aes(x = lambda, y = ppv, colour = depth))+
  geom_point(aes(x = lambda, y = ppv, colour = depth))
bo_npv <- ggplot(data = boperform)+
  geom_line(aes(x = lambda, y = npv, colour = depth))+
  geom_point(aes(x = lambda, y = npv, colour = depth))
grid.arrange(bo_ac, bo_sen, bo_spe, bo_ppv, bo_npv, ncol = 2)
```




```
boperform[which(boperform$Accuracy == max(boperform$Accuracy)),]
```

```
##    depth lambda  Accuracy Sensitivity Specificity       ppv       npv
## 16     8   0.03 0.9484103   0.7276786    0.982675 0.8670213 0.9587559
```

```
boperform[which(boperform$Sensitivity == max(boperform$Sensitivity)),]
```

```
##   depth lambda  Accuracy Sensitivity Specificity       ppv      npv
## 8     7   0.01 0.9472106   0.7410714     0.97921 0.8469388 0.960571
```

```
boperform[which(boperform$ppv == max(boperform$ppv)),]
```

```
##    depth lambda  Accuracy Sensitivity Specificity       ppv       npv
## 16     8   0.03 0.9484103   0.7276786    0.982675 0.8670213 0.9587559
```

For the boosting model, the model with the highest Accuracy, Sensitivity and PPV are shown above.

I will choose the random forest model with mtry = 13 and nodesize = 4. Because it has the highest accuracy and PPV among all the different model. The Sensitivity of this model is also very high. So the overall performance of this model is the best.

Business Plan

I will give $20 reward per month to each customer that predicted to churn. For each customer, they can use the $20 to pay the phone charge. The customer will stay as a result of my intervention is 1 month. I assume that 80% of the customers who want to churn will stay after they receive the $20. The dollar gained by retaining the customer is $30 per month.

For the prediction model, TP means the number of customer who wants to leave and predicted to churn. FP means the number of customer who does not want to churn and predicted to churn. TN means the number of customer who does not want to churn and predicted not to churn. FN means the number of customer who wants to leave and predicted not to churn. If we use the plan I mentioned above, the profit will become $(TN + FP + 0.8 \times TP) \times \$30 - (TP + FP) \times \$20$ in the next month. If I choose to do nothing, the profit will become $(TN + FP) \times \$30$ in the next month.

Because I want to make sure that the profit after I apply this plan is more than the baseline profit, which means $0.8 \times TP \times \$30 - (TP + FP) \times \$20 > 0$. So we need to make sure that $\frac{TP}{FP} > 5$.

Set $\frac{TP}{FP} = r$. Because $PPV = \frac{TP}{TP+FP}$. So $PPV = \frac{r}{r+1}$, PPV is increased by increasing r. When $r > 5$, $PPV > \frac{5}{6}$. I just keep the model with the highest PPV. The PPV of this model is 0.9939394. The performance of this model is shown below. $0.9939394 > \frac{5}{6}$. So we can use this model to make prediction.

```
rfperform[which(rfperform$ppv == max(rfperform$ppv)),]
```

```
##    try nodesize  Accuracy Sensitivity Specificity       ppv       npv
## 18  13        4 0.9634073   0.7321429    0.999307 0.9939394 0.9600533
```

In general, we set C as the reward amount, P as the dollar gained by retaining the customer, the f as the probability of the customers who want to churn will stay after they receive reward. We should make sure that $\frac{TP}{FP} > \frac{C}{Pf-C}$ to make our plan more profitable than doing nothing.

The probability of the customers who want to churn will stay after they receive the $20 is given as 80%. If the probabiliy decrease to 67%, The profit we gained might be less than baseline profit. At that time, we may increase the reward amount to increase the probabiliy, but we need to make sure that the $\frac{TP}{FP} > \frac{C}{Pf-C}$. Because the decrease of C may also lead to the increase of $\frac{C}{Pf-C}$.