Costumer_churn

Loading libraries.

```
library(caret)
library(C50)
library(dplyr)
data(churn)
```

```
prop.table(table(churnTrain$churn))
```

```
##
##       yes        no
## 0.1449145 0.8550855
```

Notice that just 14% of custumers will churn.

Create train test indices for cross-validation

```
set.seed(42)
myFolds <- createFolds(churnTrain$churn, k = 5)
```

Compare class distribution

```
i <- myFolds$Fold1
table(churnTrain$churn[i])
```

```
##
## yes  no
##  96 570
```

Create myControl

. to have exact same cross validation folds for each model. THis allow us to compare these model and make a fair comparison.

```
myControl <- trainControl(summaryFunction = twoClassSummary,
                          classProbs = T,
                          verboseIter = F,
                          savePredictions = T,
                          index = myFolds)
```

Fit the models

1. glmnet model

```
set.seed(42)
model_glmnet <- train(churn ~ .,
                      data = churnTrain,
                      metric = "ROC",
                      method = "glmnet",
                      tuneGrid = expand.grid(
                        alpha = 0:1,
                        lambda = seq(.0001, 1, length = 20)
```

```r
                        ),
                      trControl = myControl
                      )
# Find the best ROC value
model_glmnet$results %>%
  filter(ROC == max(ROC))
```

```
##   alpha    lambda       ROC       Sens      Spec      ROCSD      SensSD
## 1     0 0.2106053 0.7686188 0.01397223 0.9996491 0.01167738 0.003908229
##         SpecSD
## 1 0.0004804584
```

```r
#glmnet_pred <- predict(model_glmnet, newdata = churnTest)
#confusionMatrix(glmnet_pred, churnTest$churn)
#library(Metrics)
#auc(actual = ifelse(churnTest$churn == "yes", 1, 0), predicted = glmnet_pred[, "yes"])
```

Notice that alpha = 0 and lambda = 0.2106053 give the best result "ROC" = .7686188.

2. glm model

```r
set.seed(42)
model_glm <- train(churn ~ .,
                   data = churnTrain,
                   method = "glm",
                   metric = "ROC",
                   trControl = myControl)

model_glm$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.7077977 0.2981229 0.9401754 0.03284925 0.02928006 0.01214128
```

```r
#glm_pred <-  predict(object = model_glmnet, newdata = churnTest)
#confusionMatrix(glm_pred, churnTest$churn)

#library(caTools)
#colAUC(glm_pred, churnTest$churn, plotROC = T)
```

3. Random Forest RF is slower to fit than glmnet but often(not always) more accurate than glmnet, aesier to tune, captures threshold effect and variable interactions systematically.

```r
set.seed(42)
model_rf <- train(churn ~ .,
                  data = churnTrain,
                  method = "ranger",
                  metric = "ROC",
                  tuneGrid = expand.grid(mtry = seq(4, ncol(churnTrain) * 0.8, 2),
                                         splitrule = "gini"),
                  trControl = myControl)

#library(dplyr)
model_rf$results %>%
  filter(ROC == max(ROC))
```

```
##   mtry splitrule      ROC      Sens      Spec       ROCSD    SensSD
## 1   16      gini 0.903621 0.5418926 0.9891228 0.007002841 0.0619377
```

```
##        SpecSD
## 1 0.003013257
```

```
#rf_pred <- predict(object = model_rf, newdata = churnTest)
#confusionMatrix(rf_pred, churnTest$churn)
```

4. Gradient Boosting

```
set.seed(42)
model_xgb <- train(churn ~ .,
                   data = churnTrain,
                   method = "xgbTree",
                   metric = "ROC",
                   tuneGrid = expand.grid(eta = .01,
                                          gamma = 0,
                                          max_depth = c(5, 10),
                                          colsample_bytree = 1,
                                          min_child_weight = 1,
                                          subsample = .75,
                                          nrounds = seq(100, 200, 50)),
                   trControl = myControl
                   )

 model_xgb$results %>%
  filter(ROC == max(ROC))
```

```
##    eta max_depth gamma colsample_bytree min_child_weight subsample nrounds
## 1 0.01         5     0                1                1      0.75     200
##         ROC      Sens      Spec      ROCSD     SensSD      SpecSD
## 1 0.9064618 0.6308993 0.9834211 0.00666029 0.06459166 0.004095655
```

Comparing models

Make sure they were fit on the same data

Selection criteria .Highest average AUC .Lowest standard deviation AUC A. Make a list of models

```
model_list <- list(glmnet = model_glmnet,
                   rf = model_rf,
                   xgb = model_xgb,
                   glm = model_glm
                  )

#Collect resamples from the cv folds
set.seed(42)
resamps <- resamples(model_list)
resamps
```

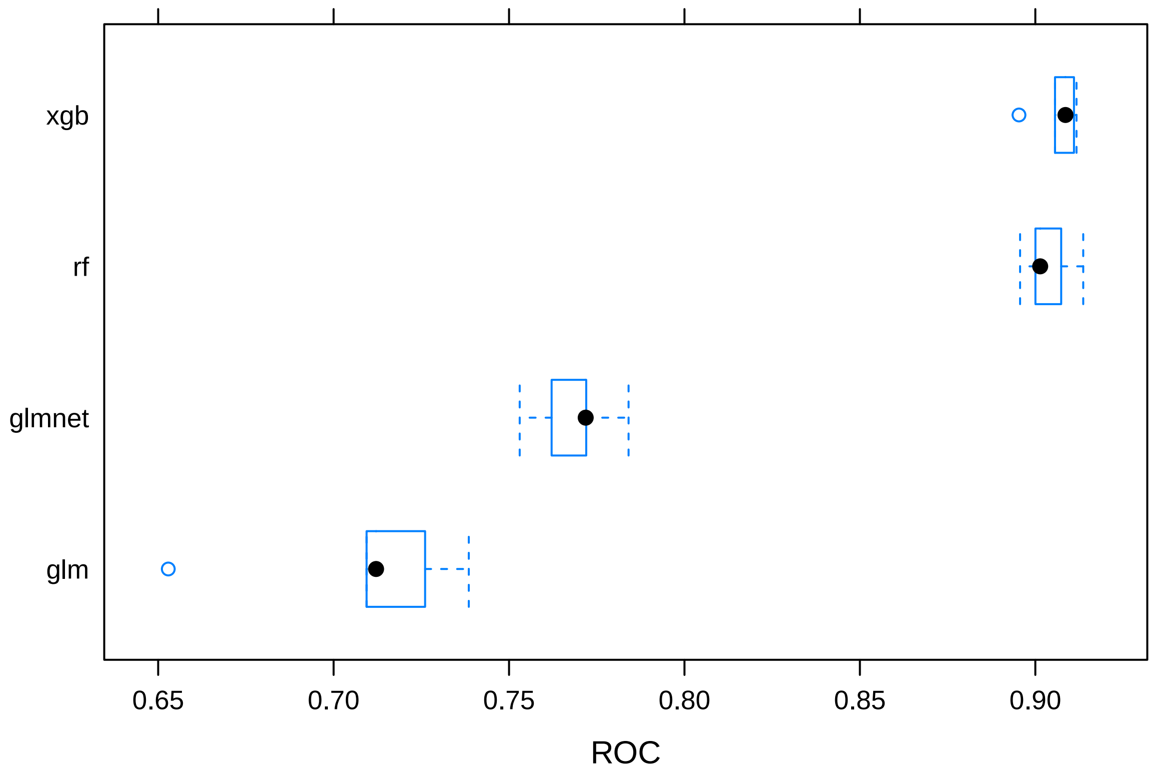
```
##
## Call:
## resamples.default(x = model_list)
##
## Models: glmnet, rf, xgb, glm
## Number of resamples: 5
## Performance metrics: ROC, Sens, Spec
```

```
## Time estimates for: everything, final model fit
```

```r
#Summarize the results
summary(resamps)
```

```
## 
## Call:
## summary.resamples(object = resamps)
## 
## Models: glmnet, rf, xgb, glm 
## Number of resamples: 5 
## 
## ROC 
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glmnet 0.7530372 0.7621334 0.7718537 0.7686188 0.7720012 0.7840685    0
## rf     0.8956544 0.9000341 0.9013845 0.9036210 0.9073745 0.9136577    0
## xgb    0.8953476 0.9056080 0.9085941 0.9064618 0.9110159 0.9117435    0
## glm    0.6528747 0.7094009 0.7121057 0.7077977 0.7260767 0.7385306    0
## 
## Sens 
##              Min.   1st Qu.     Median       Mean    3rd Qu.       Max.
## glmnet 0.01036269 0.0129199 0.01295337 0.01397223 0.01295337 0.02067183
## rf     0.46113990 0.5077720 0.53488372 0.54189260 0.59326425 0.61240310
## xgb    0.54145078 0.6010363 0.64766839 0.63089931 0.64857881 0.71576227
## glm    0.27461140 0.2772021 0.28165375 0.29812293 0.31606218 0.34108527
##        NA's
## glmnet    0
## rf        0
## xgb       0
## glm       0
## 
## Spec 
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glmnet 0.9991228 0.9991228 1.0000000 0.9996491 1.0000000 1.0000000    0
## rf     0.9846491 0.9881579 0.9890351 0.9891228 0.9916667 0.9921053    0
## xgb    0.9771930 0.9815789 0.9846491 0.9834211 0.9868421 0.9868421    0
## glm    0.9263158 0.9311404 0.9421053 0.9401754 0.9438596 0.9574561    0
```

```r
bwplot(resamps, metric = "ROC")
```


xgb
rf
glmnet
glm
0.65
0.70
0.75
0.80
0.85
0.90
ROC