XGBoost Classification Model

KNN, Mice (PMM), Mean, Median, Piecewise, and RandomForest Imputed Tables

Lewis Rincon Castano

04-01-2023

- TrainControl Documentation: https://rdrr.io/cran/caret/man/trainControl.html
- Smote sampling (pg 71): It requires the themis and ROSE packages.
- TrainControl: https://www.r-bloggers.com/2018/05/tuning-xgboost-in-r-part-i/amp/
- VIF source: https://xgboost.readthedocs.io/en/stable/R-package/xgboostPresentation.html

Model Summary: We obtained the following model results:

- Imputation tables accuracy: autoTune_VIM_KNN (93,28%), Mice-PMM method: (81,79%), median (80,95%), mean (80,95%), RandomForest (79,83%) and Piecewise (N/A).
- ROC values: autoTune_VIM_KNN (96,6%), Mice-PMM method: (85,72%), median (83,58%), mean (84,96%), RandomForest (79,83%) and Piecewise (N/A).
  - Piecewise had some missing values, and KNN classification model do not compute datasets with NaN values.
- Out **best model** came from the autoTune_VIM_KNN table due to its higher ROC and accuracy rate.

Notes:

- The list of variables to conduct our analysis came from the stepwise multiple linear regression. This process was completed on SPSS using the sponsor's file. file that is not imputed. Lewis tried to conduct this same analysis at RStudio, but it was impossible to obtain an outcome due to the missing variables. Each iteration of the variables and its accuracy was recorded on the autoTuneVIM KNN file section because it is the model with the higher accuracy rate.

```r
# option to get rid of scientific notation
options(scipen = 999)

# install libraries
library(caret) # v6.0-93
library(ISLR) # v1.4
library(pROC) # v1.18.0
library(plotROC) # v2.3.0
library(ROCit) # v2.1.1
library(precrec) # v0.14.1
library(dplyr) # v1.0.10
```

```r
library(ggplot2) # v3.4.0
library(rattle) # v5.5.1
library(DMwR) # v0.4.1
library(ROSE)# v0.0-4
library(AppliedPredictiveModeling) # v1.1-7
library(NeuralNetTools) # v1.5.3
library(gbm) # v2.1.8.1
library(xgboost) # v1.6.0.1
library(recipes) # v1.0.3
library(themis) # v1.0.0
library(readxl) # v1.4.1, to read Excel files
library(readr) # v2.1.3, to read csv files
```

```r
# Documenting data Prep Workflow and Code
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          1.0
## year           2021
## month          05
## day            18
## svn rev        80317
## language       R
## version.string R version 4.1.0 (2021-05-18)
## nickname       Camp Pontanezen
```

```r
# Specific package number
# sessionInfo()
```

Import Excel file: AutoTuneVIM KNN imputed table

```r
data <- read_excel("C:/Users/lewis/Desktop/completeData_autotuneVIM_KNN_method_v2.xlsx")

# Filter table with the following columns
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
```

```
                OVERNIGHT_SHIFTS,
                JOB_DESCRIPTION,
                SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE

                )
```

```
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the echo=T, results='hide', message=F, warning=F to avoid printing unnecessary pages of iterations.

```
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
```

```r
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)

# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```r
# let's look at the model results - this is for training only
XGB_model
```

```
## eXtreme Gradient Boosting
##
## 537 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 483, 483, 484, 484, 483, 483, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  logLoss
##   0.3  1          0.6               0.5         50      0.3475453
##   0.3  1          0.6               0.5        100      0.3008306
##   0.3  1          0.6               1.0         50      0.3438666
##   0.3  1          0.6               1.0        100      0.2912256
##   0.3  1          0.8               0.5         50      0.3402939
##   0.3  1          0.8               0.5        100      0.2943756
##   0.3  1          0.8               1.0         50      0.3411825
##   0.3  1          0.8               1.0        100      0.2901443
##   0.3  2          0.6               0.5         50      0.2371054
##   0.3  2          0.6               0.5        100      0.2212727
##   0.3  2          0.6               1.0         50      0.2274788
##   0.3  2          0.6               1.0        100      0.2036844
##   0.3  2          0.8               0.5         50      0.2309776
##   0.3  2          0.8               0.5        100      0.2137947
##   0.3  2          0.8               1.0         50      0.2299553
```

```
##   0.3  2          0.8               1.0        100      0.2075136
##   0.4  1          0.6               0.5         50      0.3203188
##   0.4  1          0.6               0.5        100      0.2898199
##   0.4  1          0.6               1.0         50      0.3168857
##   0.4  1          0.6               1.0        100      0.2734390
##   0.4  1          0.8               0.5         50      0.3227823
##   0.4  1          0.8               0.5        100      0.2900603
##   0.4  1          0.8               1.0         50      0.3156050
##   0.4  1          0.8               1.0        100      0.2740617
##   0.4  2          0.6               0.5         50      0.2322548
##   0.4  2          0.6               0.5        100      0.2305761
##   0.4  2          0.6               1.0         50      0.2239784
##   0.4  2          0.6               1.0        100      0.2160805
##   0.4  2          0.8               0.5         50      0.2459762
##   0.4  2          0.8               0.5        100      0.2300911
##   0.4  2          0.8               1.0         50      0.2220165
##   0.4  2          0.8               1.0        100      0.2121364
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 100, max_depth = 2, eta
##  = 0.3, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsample
##  = 1.
```

```
summary(XGB_model)
```

```
##                Length Class              Mode
## handle             1  xgb.Booster.handle externalptr
## raw            92046  -none-             raw
## niter              1  -none-             numeric
## call               6  -none-             call
## params             9  -none-             list
## callbacks          1  -none-             list
## feature_names     12  -none-             character
## nfeatures          1  -none-             numeric
## xNames            12  -none-             character
## problemType        1  -none-             character
## tuneValue          7  data.frame         list
## obsLevels          2  -none-             character
## param              1  -none-             list
```

```
head(XGB_model$pred)
```

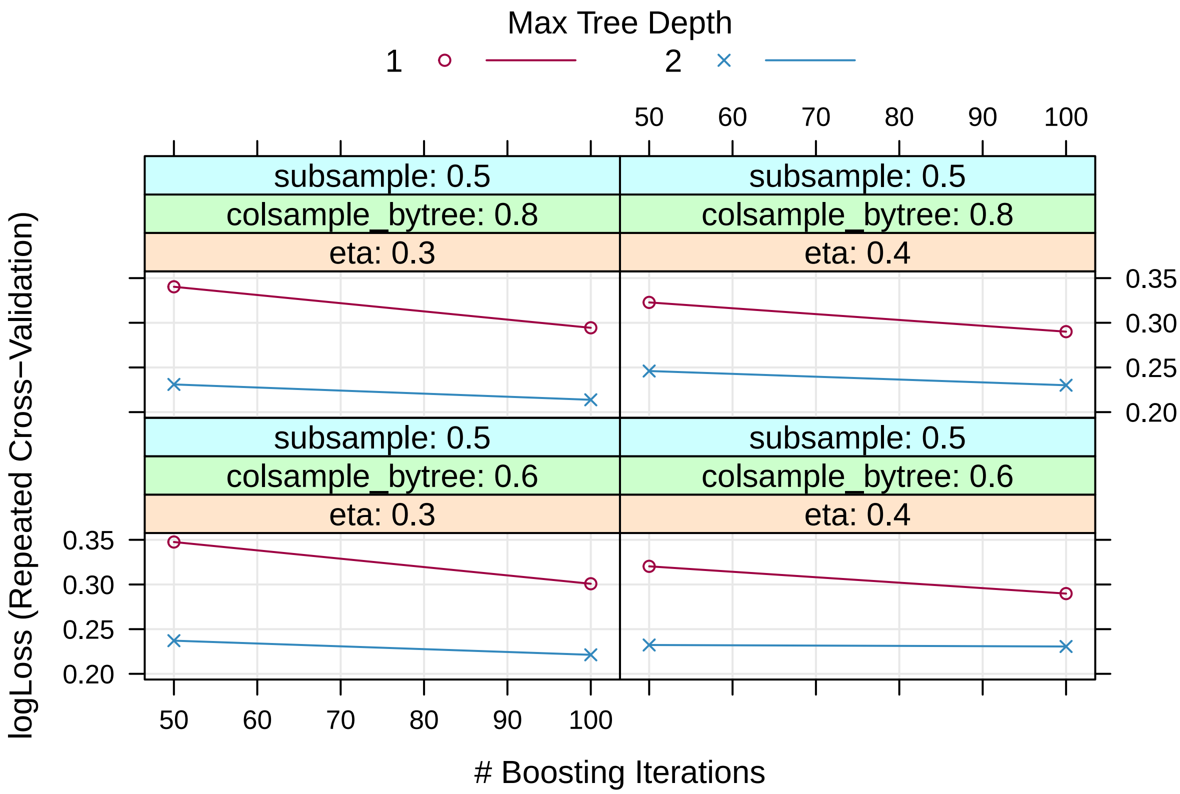
```
##         pred      obs rowIndex  EMPLOYEE TERMINATED eta max_depth gamma
## 1   EMPLOYEE EMPLOYEE       31 0.9614876 0.03851241 0.3         1     0
## 2   EMPLOYEE EMPLOYEE       36 0.9377227 0.06227726 0.3         1     0
## 3   EMPLOYEE EMPLOYEE       54 0.9703442 0.02965581 0.3         1     0
## 4 TERMINATED EMPLOYEE       56 0.3620723 0.63792765 0.3         1     0
## 5   EMPLOYEE EMPLOYEE       71 0.7507306 0.24926943 0.3         1     0
## 6   EMPLOYEE EMPLOYEE       82 0.8910710 0.10892904 0.3         1     0
##   colsample_bytree min_child_weight subsample nrounds    Resample
```

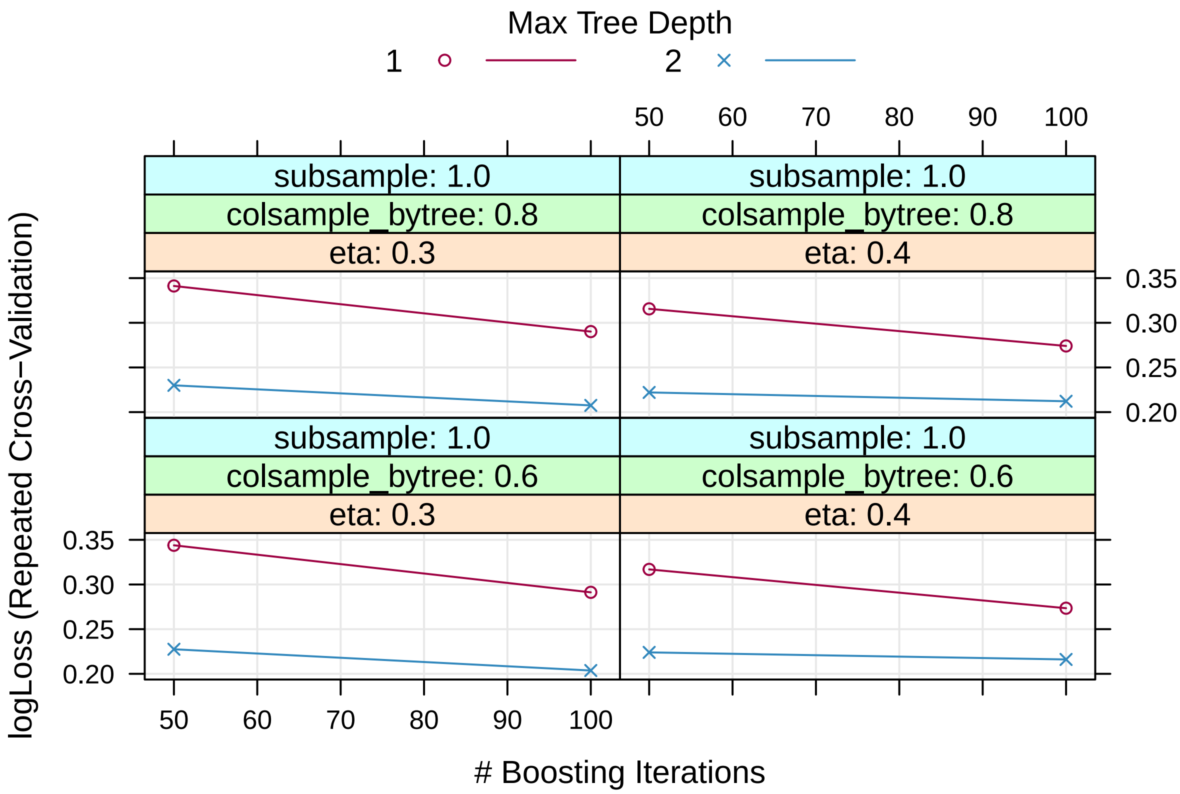
```
## 1              0.6                 1       0.5     100 Fold01.Rep1
## 2              0.6                 1       0.5     100 Fold01.Rep1
## 3              0.6                 1       0.5     100 Fold01.Rep1
## 4              0.6                 1       0.5     100 Fold01.Rep1
## 5              0.6                 1       0.5     100 Fold01.Rep1
## 6              0.6                 1       0.5     100 Fold01.Rep1
```

```r
# and produce a confusion matrix
lvs <- c("Employee","Terminated")
truth <- factor(XGB_model$pred$obs)
pred <- factor(XGB_model$pred$pred)
xtab <- table(pred, truth)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##             truth
## pred          EMPLOYEE TERMINATED
##   EMPLOYEE      38126       1756
##   TERMINATED     3346       8324
##
##                Accuracy : 0.901
##                  95% CI : (0.8984, 0.9036)
##     No Information Rate : 0.8045
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.7031
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9193
##             Specificity : 0.8258
##          Pos Pred Value : 0.9560
##          Neg Pred Value : 0.7133
##              Prevalence : 0.8045
##          Detection Rate : 0.7396
##    Detection Prevalence : 0.7736
##       Balanced Accuracy : 0.8726
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# LogLoss (Repeated Cross-Validation) plots using different parameters: eta,
# subsample and colsample bytree.
trellis.par.set(caretTheme())
plot(XGB_model)
```


Max Tree Depth
1
2
subsample: 0.5
colsample_bytree: 0.8
eta: 0.3
eta: 0.4
colsample_bytree: 0.6
logLoss (Repeated Cross−Validation)
0.20
0.25
0.30
0.35
50
60
70
80
90
100
Boosting Iterations

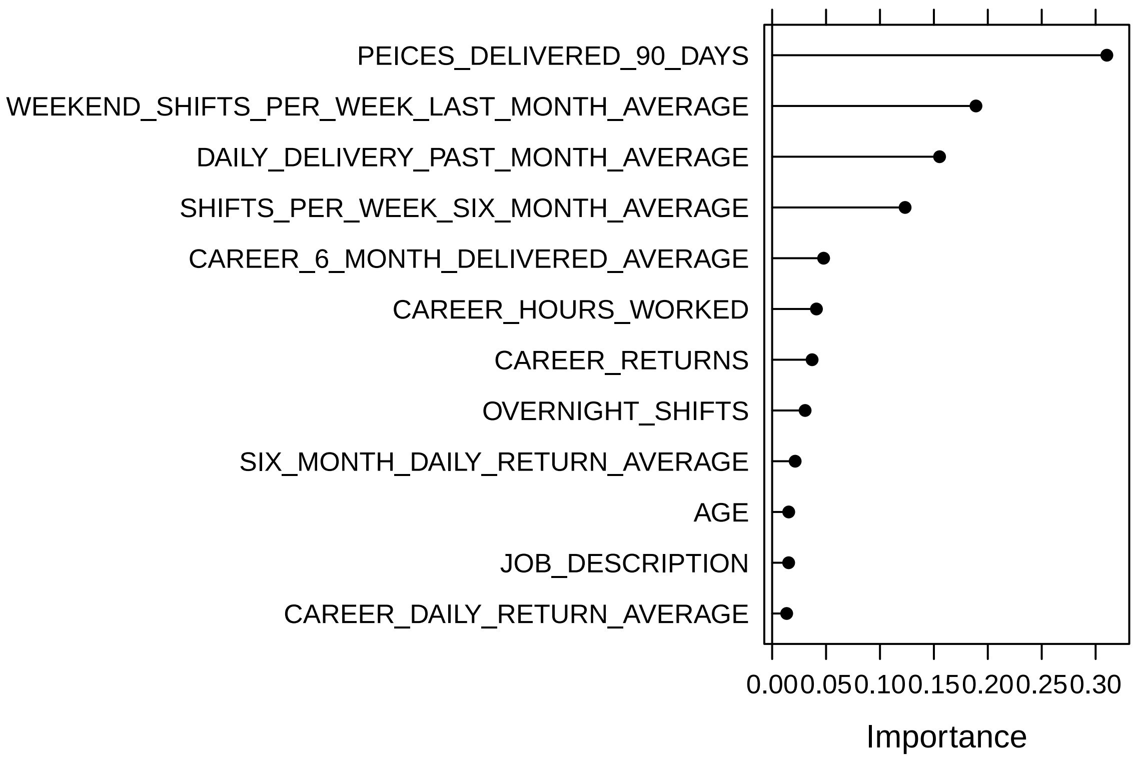



```r
# now apply the XGB model to the Test data
test_XGB <- predict(XGB_model, newdata = default_tst)
test_XGB_prob <- predict(XGB_model, newdata = default_tst, type="prob")
confusionMatrix(test_XGB, default_tst$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        278         15
##   TERMINATED        9         55
##
##                Accuracy : 0.9328
##                  95% CI : (0.9016, 0.9565)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.00000000000538
##
##                   Kappa : 0.7796
##
##  Mcnemar's Test P-Value : 0.3074
##
##             Sensitivity : 0.9686
##             Specificity : 0.7857
##          Pos Pred Value : 0.9488
##          Neg Pred Value : 0.8594
##              Prevalence : 0.8039
```

```
##          Detection Rate : 0.7787
##    Detection Prevalence : 0.8207
##       Balanced Accuracy : 0.8772
##
##        'Positive' Class : EMPLOYEE
##
```

```
# variable importance visualization
importance <- varImp(XGB_model, scale = FALSE)
plot(importance)
```



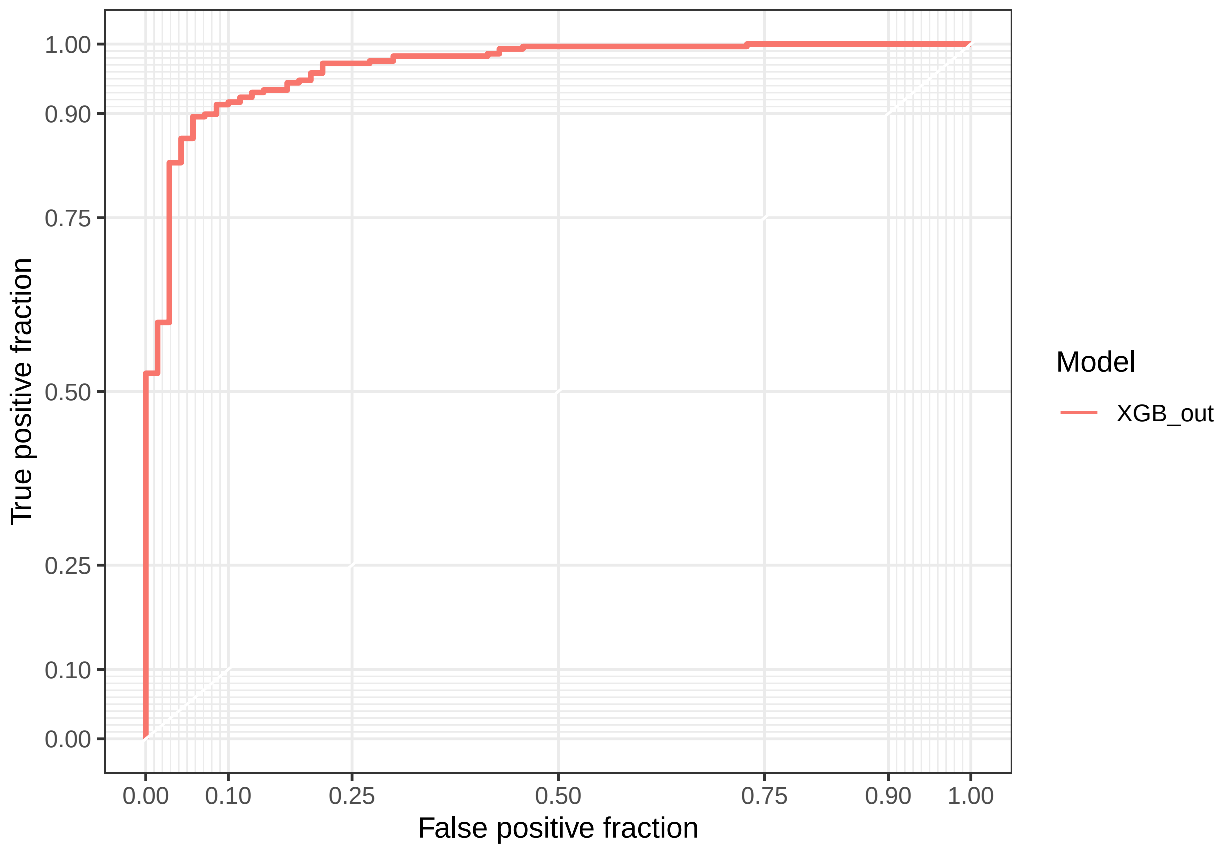

```
# Save the probabilities
results <- default_tst
results$XGB_out <- test_XGB_prob$EMPLOYEE

# this section of the code sets up the data frame for the plotROC format
results$D <- ifelse(results$STATUS=="TERMINATED",0,1)
longresult <- melt_roc(results, "D", c("XGB_out"))
head(longresult)
```

```
##          D         M    name
## XGB_out1 1 0.9990846 XGB_out
## XGB_out2 1 0.9849975 XGB_out
## XGB_out3 1 0.9975352 XGB_out
## XGB_out4 1 0.9972344 XGB_out
```

```
## XGB_out5 1 0.9923247 XGB_out
## XGB_out6 1 0.9997924 XGB_out
```

```
ggplot(longresult, aes(d=D, m=M, color=name))+
  geom_roc(n.cuts = 0)+style_roc()+
  labs(color = "Model")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: d, m
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```




```
# get AUC's for each model
XGB_Auc <- pROC::roc(results$STATUS, results$XGB_out)
```

```
## Setting levels: control = EMPLOYEE, case = TERMINATED
```

```
## Setting direction: controls > cases
```

```
XGB_Auc$auc
```

```
## Area under the curve: 0.966
```

```r
# create a confusion matrix
confusionMatrix(test_XGB, results$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        278         15
##   TERMINATED        9         55
##
##                Accuracy : 0.9328
##                  95% CI : (0.9016, 0.9565)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.00000000000538
##
##                   Kappa : 0.7796
##
##  Mcnemar's Test P-Value : 0.3074
##
##             Sensitivity : 0.9686
##             Specificity : 0.7857
##          Pos Pred Value : 0.9488
##          Neg Pred Value : 0.8594
##              Prevalence : 0.8039
##          Detection Rate : 0.7787
##    Detection Prevalence : 0.8207
##       Balanced Accuracy : 0.8772
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Mice (PMM) imputed table

```r
data <- read_excel("C:/Users/lewis/Downloads/completeData_mice_method.xlsx")
# Select and filter table with the key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```r
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```r
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the echo=T, results='hide', message=F, warning=F to avoid printing unnecessary pages of iterations.

```r
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)
```

```r
# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```r
# let's look at the model results - this is for training only
XGB_model
```

```
## eXtreme Gradient Boosting
##
## 537 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 483, 483, 484, 484, 483, 483, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  logLoss
##   0.3  1          0.6               0.5         50      0.4092974
##   0.3  1          0.6               0.5        100      0.4094055
##   0.3  1          0.6               1.0         50      0.4084532
##   0.3  1          0.6               1.0        100      0.4084039
##   0.3  1          0.8               0.5         50      0.4145520
##   0.3  1          0.8               0.5        100      0.4099158
##   0.3  1          0.8               1.0         50      0.4108314
##   0.3  1          0.8               1.0        100      0.4100910
##   0.3  2          0.6               0.5         50      0.4080891
##   0.3  2          0.6               0.5        100      0.4352851
##   0.3  2          0.6               1.0         50      0.4039593
##   0.3  2          0.6               1.0        100      0.4142974
##   0.3  2          0.8               0.5         50      0.4164943
##   0.3  2          0.8               0.5        100      0.4477270
##   0.3  2          0.8               1.0         50      0.4084562
##   0.3  2          0.8               1.0        100      0.4193871
##   0.4  1          0.6               0.5         50      0.4216838
##   0.4  1          0.6               0.5        100      0.4230440
##   0.4  1          0.6               1.0         50      0.4106666
##   0.4  1          0.6               1.0        100      0.4112159
##   0.4  1          0.8               0.5         50      0.4151491
##   0.4  1          0.8               0.5        100      0.4228447
```

```
##   0.4  1          0.8              1.0        50      0.4044064
##   0.4  1          0.8              1.0       100      0.4121496
##   0.4  2          0.6              0.5        50      0.4364915
##   0.4  2          0.6              0.5       100      0.4543455
##   0.4  2          0.6              1.0        50      0.4099843
##   0.4  2          0.6              1.0       100      0.4366158
##   0.4  2          0.8              0.5        50      0.4286449
##   0.4  2          0.8              0.5       100      0.4685753
##   0.4  2          0.8              1.0        50      0.4178382
##   0.4  2          0.8              1.0       100      0.4432241
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, max_depth = 2, eta
##  = 0.3, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsample
##  = 1.
```

```
summary(XGB_model)
```

```
##                Length Class              Mode
## handle             1  xgb.Booster.handle externalptr
## raw            48346  -none-             raw
## niter              1  -none-             numeric
## call               6  -none-             call
## params             9  -none-             list
## callbacks          1  -none-             list
## feature_names     12  -none-             character
## nfeatures          1  -none-             numeric
## xNames            12  -none-             character
## problemType        1  -none-             character
## tuneValue          7  data.frame         list
## obsLevels          2  -none-             character
## param              1  -none-             list
```

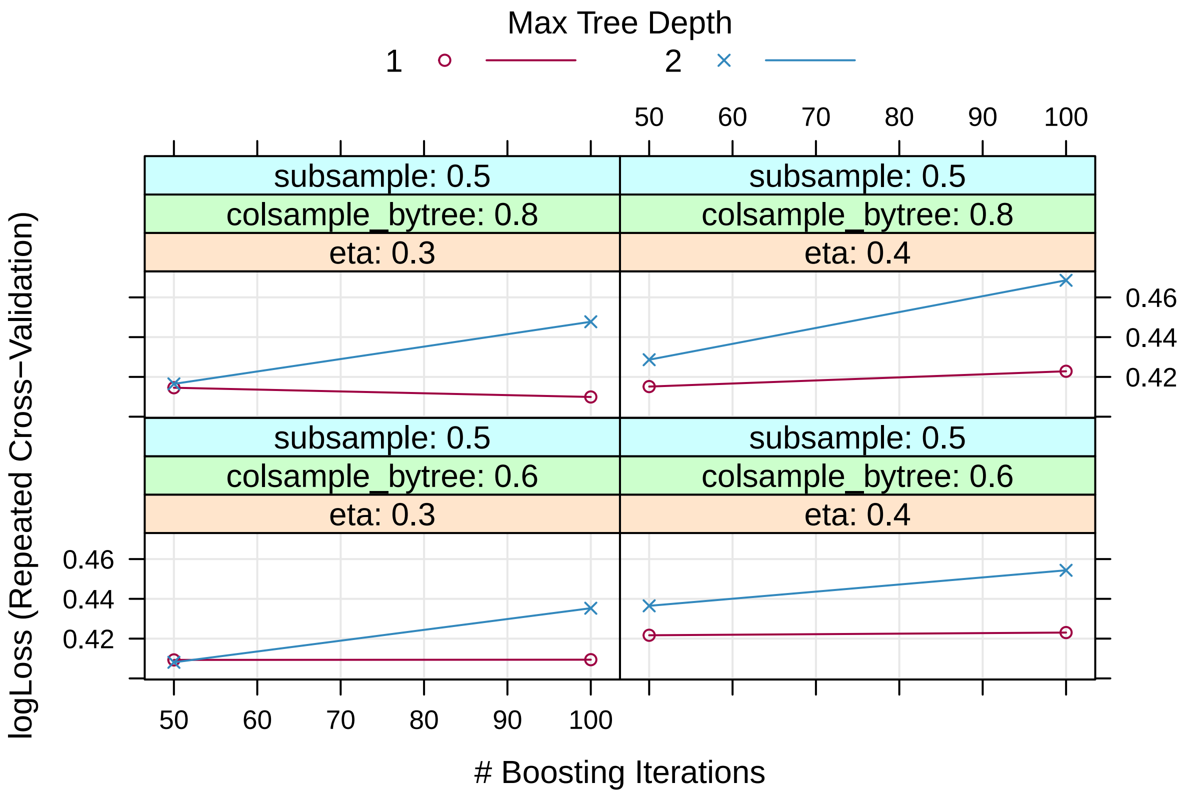
```
head(XGB_model$pred)
```

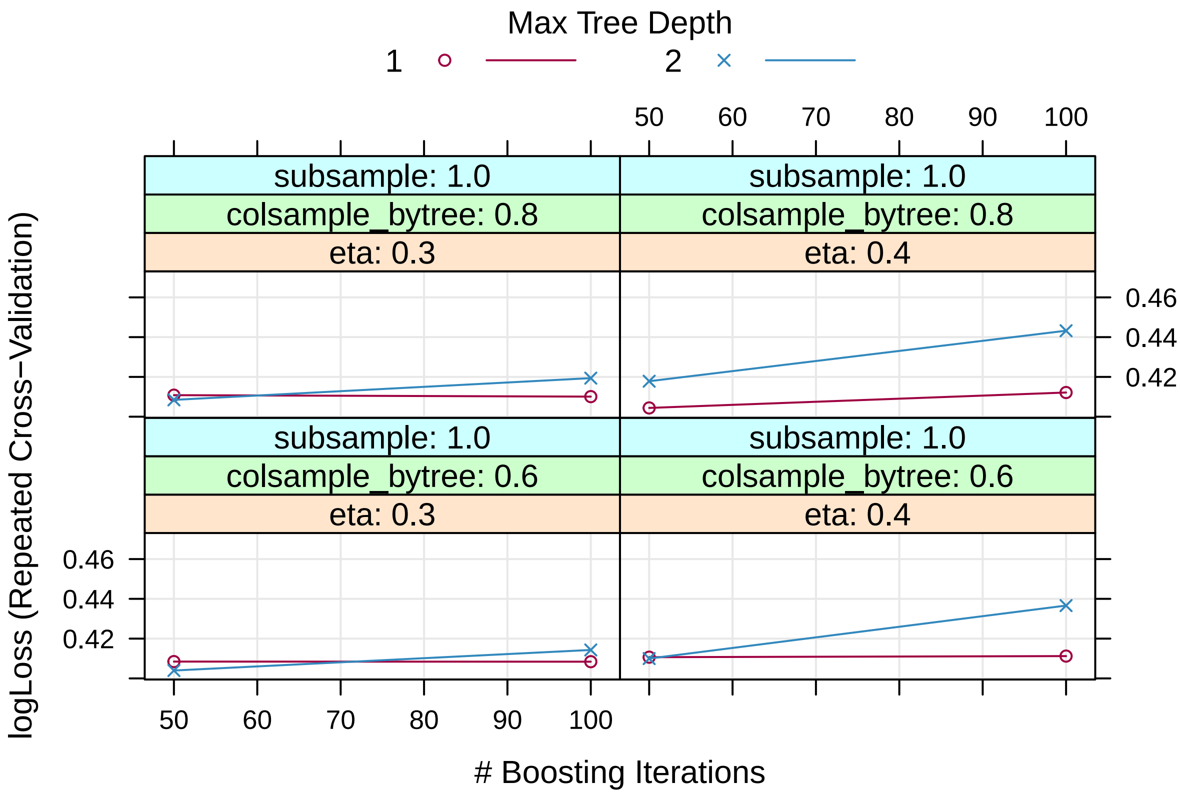
```
##       pred      obs rowIndex  EMPLOYEE TERMINATED eta max_depth gamma
## 1 EMPLOYEE EMPLOYEE       31 0.9541165 0.04588354 0.3         1     0
## 2 EMPLOYEE EMPLOYEE       36 0.9822209 0.01777905 0.3         1     0
## 3 EMPLOYEE EMPLOYEE       54 0.9444863 0.05551368 0.3         1     0
## 4 EMPLOYEE EMPLOYEE       56 0.8648786 0.13512141 0.3         1     0
## 5 EMPLOYEE EMPLOYEE       71 0.6242208 0.37577915 0.3         1     0
## 6 EMPLOYEE EMPLOYEE       82 0.6678925 0.33210754 0.3         1     0
##   colsample_bytree min_child_weight subsample nrounds    Resample
## 1              0.6                1       0.5     100 Fold01.Rep1
## 2              0.6                1       0.5     100 Fold01.Rep1
## 3              0.6                1       0.5     100 Fold01.Rep1
## 4              0.6                1       0.5     100 Fold01.Rep1
## 5              0.6                1       0.5     100 Fold01.Rep1
## 6              0.6                1       0.5     100 Fold01.Rep1
```

```r
# and produce a confusion matrix
lvs <- c("Employee","Terminated")
truth <- factor(XGB_model$pred$obs)
pred <- factor(XGB_model$pred$pred)
xtab <- table(pred, truth)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##             truth
## pred          EMPLOYEE TERMINATED
##   EMPLOYEE      35680       3113
##   TERMINATED     5792       6967
##
##                Accuracy : 0.8273
##                  95% CI : (0.824, 0.8305)
##     No Information Rate : 0.8045
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.5011
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##             Sensitivity : 0.8603
##             Specificity : 0.6912
##          Pos Pred Value : 0.9198
##          Neg Pred Value : 0.5460
##              Prevalence : 0.8045
##          Detection Rate : 0.6921
##    Detection Prevalence : 0.7525
##       Balanced Accuracy : 0.7758
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# LogLoss (Repeated Cross-Validation) plots using different parameters: eta,
# subsample and colsample bytree.
trellis.par.set(caretTheme())
plot(XGB_model)
```


Max Tree Depth
1
2
subsample: 0.5
colsample_bytree: 0.8
eta: 0.3
subsample: 0.5
colsample_bytree: 0.8
eta: 0.4
subsample: 0.5
colsample_bytree: 0.6
eta: 0.3
subsample: 0.5
colsample_bytree: 0.6
eta: 0.4
0.46
0.44
0.42
50
60
70
80
90
100
logLoss (Repeated Cross−Validation)
Boosting Iterations

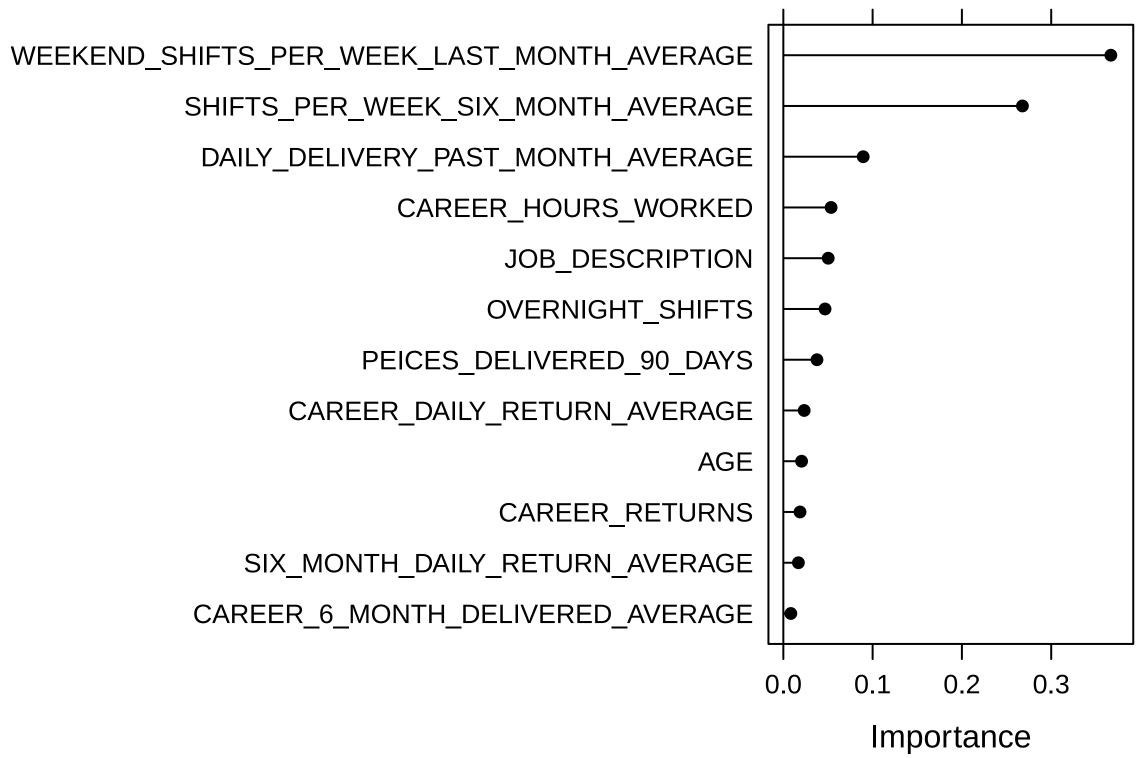



```r
# now apply the XGB model to the Test data
test_XGB <- predict(XGB_model, newdata = default_tst)
test_XGB_prob <- predict(XGB_model, newdata = default_tst, type="prob")
confusionMatrix(test_XGB, default_tst$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        241         19
##   TERMINATED       46         51
##
##                Accuracy : 0.8179
##                  95% CI : (0.7739, 0.8566)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.27723
##
##                   Kappa : 0.496
##
##  Mcnemar's Test P-Value : 0.00126
##
##             Sensitivity : 0.8397
##             Specificity : 0.7286
##          Pos Pred Value : 0.9269
##          Neg Pred Value : 0.5258
##              Prevalence : 0.8039
```

```
##          Detection Rate : 0.6751
##    Detection Prevalence : 0.7283
##       Balanced Accuracy : 0.7841
##
##        'Positive' Class : EMPLOYEE
##
```

```
# variable importance visualization
importance <- varImp(XGB_model, scale = FALSE)
plot(importance)
```



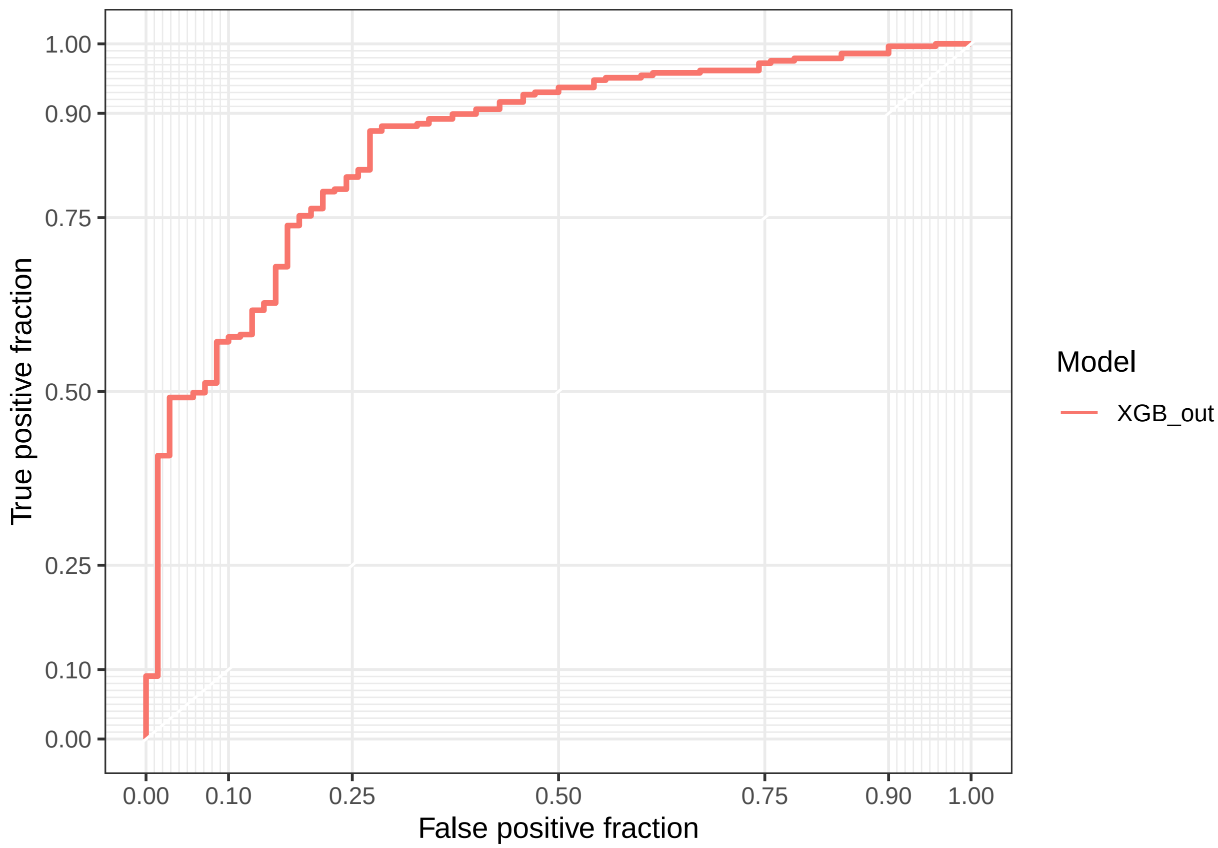

```
# Save the probabilities
results <- default_tst
results$XGB_out <- test_XGB_prob$EMPLOYEE

# this section of the code sets up the data frame for the plotROC format
results$D <- ifelse(results$STATUS=="TERMINATED",0,1)
longresult <- melt_roc(results, "D", c("XGB_out"))
head(longresult)
```

```
##          D         M    name
## XGB_out1 1 0.9876707 XGB_out
## XGB_out2 1 0.9985647 XGB_out
## XGB_out3 1 0.6024330 XGB_out
## XGB_out4 1 0.9963380 XGB_out
```

```
## XGB_out5 1 0.8479584 XGB_out
## XGB_out6 1 0.9864114 XGB_out
```

```
ggplot(longresult, aes(d=D, m=M, color=name))+
  geom_roc(n.cuts = 0)+style_roc()+
  labs(color = "Model")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: d, m
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a 'group' aesthetic or to convert a numerical
##   variable into a factor?
```




```
# get AUC's for each model
XGB_Auc <- pROC::roc(results$STATUS, results$XGB_out)
```

```
## Setting levels: control = EMPLOYEE, case = TERMINATED
```

```
## Setting direction: controls > cases
```

```
XGB_Auc$auc
```

```
## Area under the curve: 0.8572
```

```r
# create a confusion matrix
confusionMatrix(test_XGB, results$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        241         19
##   TERMINATED       46         51
##
##                Accuracy : 0.8179
##                  95% CI : (0.7739, 0.8566)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.27723
##
##                   Kappa : 0.496
##
##  Mcnemar's Test P-Value : 0.00126
##
##             Sensitivity : 0.8397
##             Specificity : 0.7286
##          Pos Pred Value : 0.9269
##          Neg Pred Value : 0.5258
##              Prevalence : 0.8039
##          Detection Rate : 0.6751
##    Detection Prevalence : 0.7283
##       Balanced Accuracy : 0.7841
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Median imputed table

```r
data <- read_excel("C:/Users/lewis/Downloads/CompleteeData_median_imp.xlsx")

# Select and filter table with the key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```r
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```r
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the echo=T, results='hide', message=F, warning=F to avoid printing unnecessary pages of iterations.

```r
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)
```

```r
# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```r
# let's look at the model results - this is for training only
XGB_model
```

```
## eXtreme Gradient Boosting
##
## 537 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 483, 483, 484, 484, 483, 483, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  logLoss
##   0.3  1          0.6               0.5         50      0.4548703
##   0.3  1          0.6               0.5        100      0.4417524
##   0.3  1          0.6               1.0         50      0.4575375
##   0.3  1          0.6               1.0        100      0.4435806
##   0.3  1          0.8               0.5         50      0.4557631
##   0.3  1          0.8               0.5        100      0.4537590
##   0.3  1          0.8               1.0         50      0.4564731
##   0.3  1          0.8               1.0        100      0.4447723
##   0.3  2          0.6               0.5         50      0.4320139
##   0.3  2          0.6               0.5        100      0.4431639
##   0.3  2          0.6               1.0         50      0.4243146
##   0.3  2          0.6               1.0        100      0.4300662
##   0.3  2          0.8               0.5         50      0.4287079
##   0.3  2          0.8               0.5        100      0.4439333
##   0.3  2          0.8               1.0         50      0.4265144
##   0.3  2          0.8               1.0        100      0.4256798
##   0.4  1          0.6               0.5         50      0.4542449
##   0.4  1          0.6               0.5        100      0.4516378
##   0.4  1          0.6               1.0         50      0.4470668
##   0.4  1          0.6               1.0        100      0.4441327
##   0.4  1          0.8               0.5         50      0.4474580
##   0.4  1          0.8               0.5        100      0.4533969
```

```
##   0.4  1          0.8               1.0        50      0.4473823
##   0.4  1          0.8               1.0       100      0.4444718
##   0.4  2          0.6               0.5        50      0.4439332
##   0.4  2          0.6               0.5       100      0.4583297
##   0.4  2          0.6               1.0        50      0.4203194
##   0.4  2          0.6               1.0       100      0.4402906
##   0.4  2          0.8               0.5        50      0.4465515
##   0.4  2          0.8               0.5       100      0.4558056
##   0.4  2          0.8               1.0        50      0.4197978
##   0.4  2          0.8               1.0       100      0.4470887
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, max_depth = 2, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 1.
```

```
summary(XGB_model)
```

```
##               Length Class              Mode
## handle            1  xgb.Booster.handle externalptr
## raw           48006  -none-             raw
## niter             1  -none-             numeric
## call              6  -none-             call
## params            9  -none-             list
## callbacks         1  -none-             list
## feature_names    12  -none-             character
## nfeatures         1  -none-             numeric
## xNames           12  -none-             character
## problemType       1  -none-             character
## tuneValue         7  data.frame         list
## obsLevels         2  -none-             character
## param             1  -none-             list
```

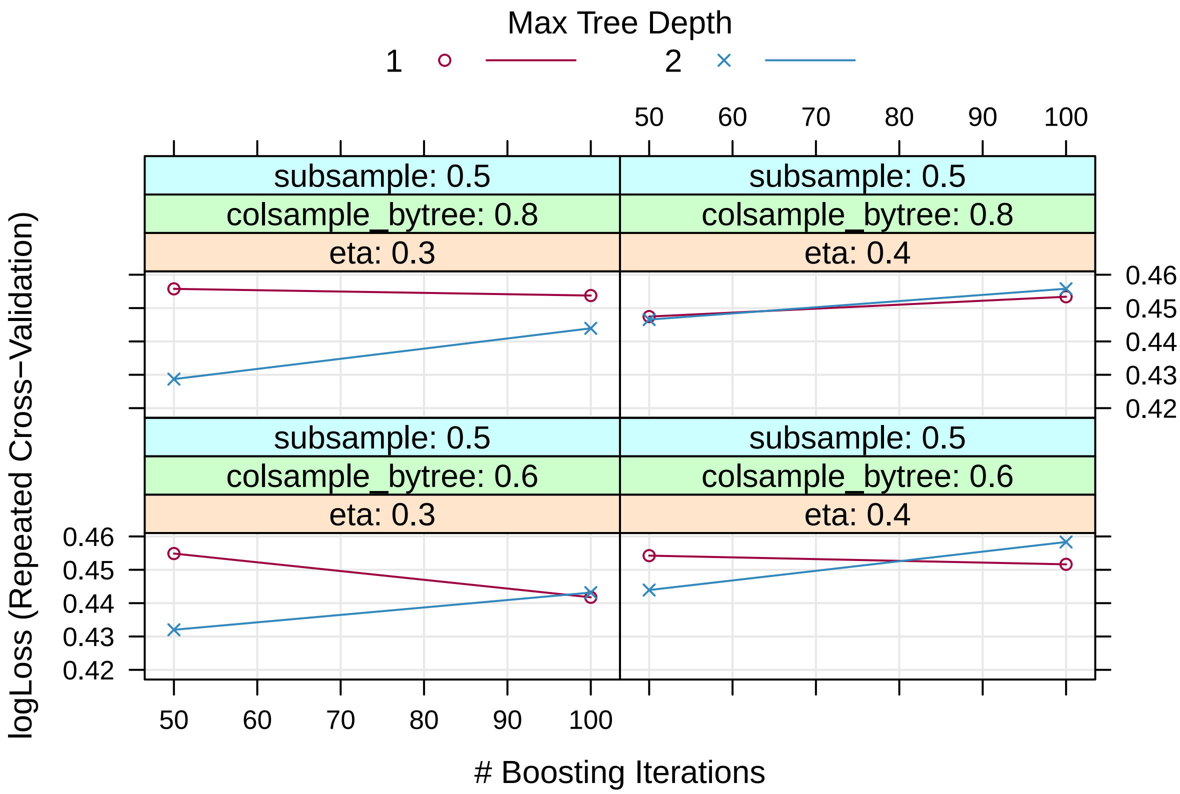
```
head(XGB_model$pred)
```

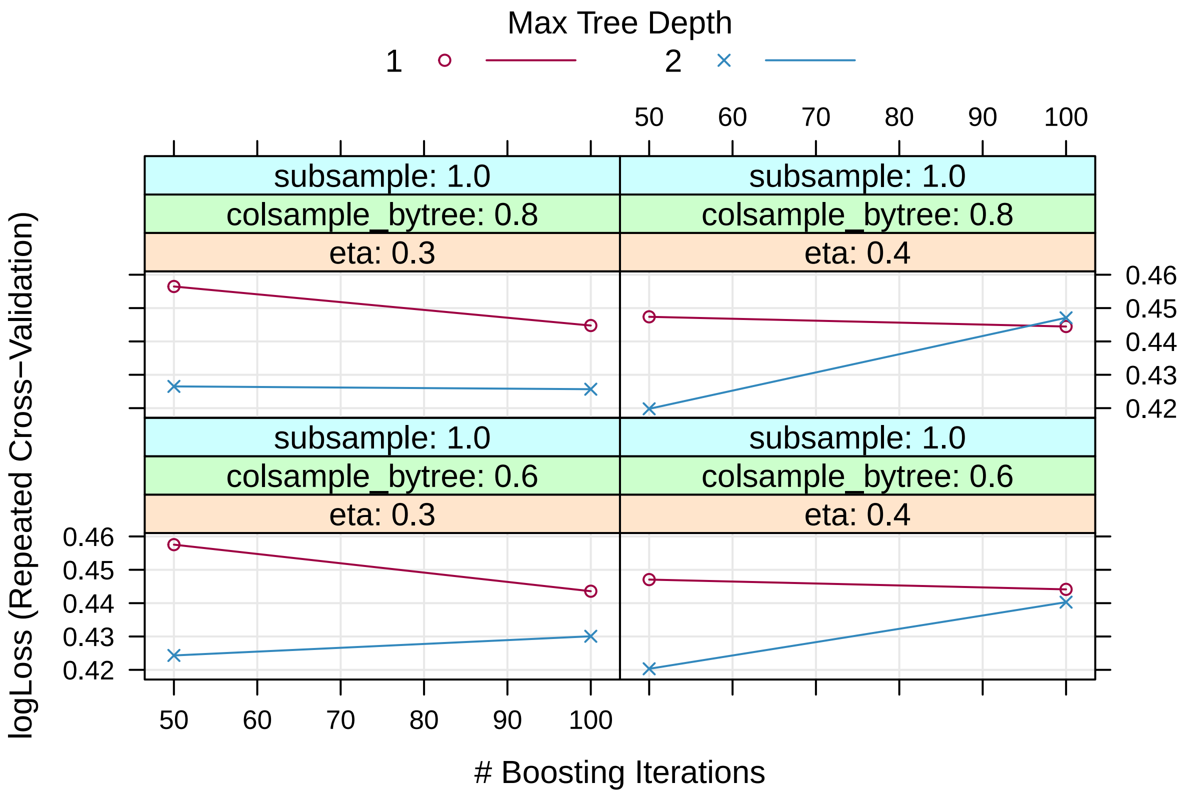
```
##         pred      obs rowIndex  EMPLOYEE TERMINATED eta max_depth gamma
## 1   EMPLOYEE EMPLOYEE       31 0.9569260 0.04307395 0.3         1     0
## 2   EMPLOYEE EMPLOYEE       36 0.9703610 0.02963901 0.3         1     0
## 3   EMPLOYEE EMPLOYEE       54 0.9631881 0.03681195 0.3         1     0
## 4   EMPLOYEE EMPLOYEE       56 0.8894240 0.11057597 0.3         1     0
## 5   EMPLOYEE EMPLOYEE       71 0.8020302 0.19796979 0.3         1     0
## 6 TERMINATED EMPLOYEE       82 0.4833187 0.51668134 0.3         1     0
##   colsample_bytree min_child_weight subsample nrounds    Resample
## 1              0.6                1       0.5     100 Fold01.Rep1
## 2              0.6                1       0.5     100 Fold01.Rep1
## 3              0.6                1       0.5     100 Fold01.Rep1
## 4              0.6                1       0.5     100 Fold01.Rep1
## 5              0.6                1       0.5     100 Fold01.Rep1
## 6              0.6                1       0.5     100 Fold01.Rep1
```

```r
# and produce a confusion matrix
lvs <- c("Employee","Terminated")
truth <- factor(XGB_model$pred$obs)
pred <- factor(XGB_model$pred$pred)
xtab <- table(pred, truth)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##             truth
## pred          EMPLOYEE TERMINATED
##   EMPLOYEE      33767       3398
##   TERMINATED     7705       6682
##
##                Accuracy : 0.7846
##                  95% CI : (0.7811, 0.7882)
##     No Information Rate : 0.8045
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4107
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.8142
##             Specificity : 0.6629
##          Pos Pred Value : 0.9086
##          Neg Pred Value : 0.4644
##              Prevalence : 0.8045
##          Detection Rate : 0.6550
##    Detection Prevalence : 0.7209
##       Balanced Accuracy : 0.7386
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# LogLoss (Repeated Cross-Validation) plots using different parameters: eta,
# subsample and colsample bytree.
trellis.par.set(caretTheme())
plot(XGB_model)
```


Max Tree Depth
1
2
subsample: 0.5
colsample_bytree: 0.8
eta: 0.3
subsample: 0.5
colsample_bytree: 0.8
eta: 0.4
subsample: 0.5
colsample_bytree: 0.6
eta: 0.3
subsample: 0.5
colsample_bytree: 0.6
eta: 0.4
50
60
70
80
90
100
0.42
0.43
0.44
0.45
0.46
logLoss (Repeated Cross−Validation)
Boosting Iterations

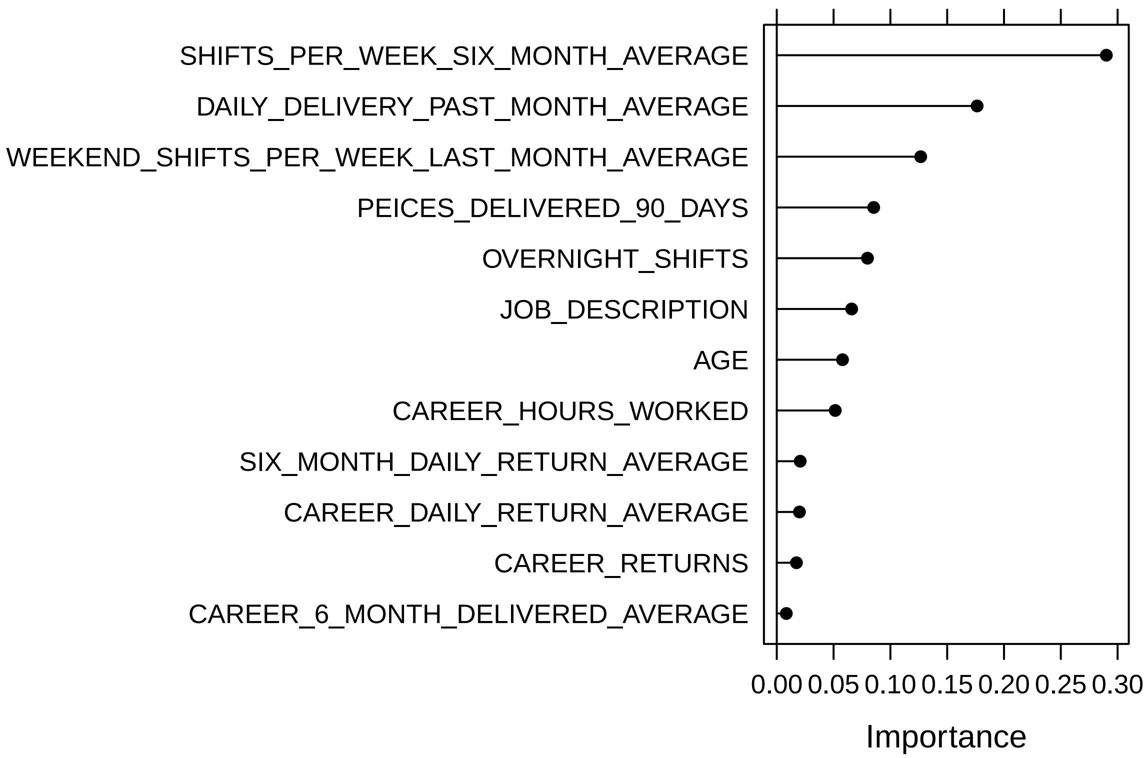



```r
# now apply the XGB model to the Test data
test_XGB <- predict(XGB_model, newdata = default_tst)
test_XGB_prob <- predict(XGB_model, newdata = default_tst, type="prob")
confusionMatrix(test_XGB, default_tst$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        246         27
##   TERMINATED       41         43
##
##                Accuracy : 0.8095
##                  95% CI : (0.7649, 0.8489)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.4259
##
##                   Kappa : 0.4383
##
##  Mcnemar's Test P-Value : 0.1149
##
##             Sensitivity : 0.8571
##             Specificity : 0.6143
##          Pos Pred Value : 0.9011
##          Neg Pred Value : 0.5119
##              Prevalence : 0.8039
```

```
##          Detection Rate : 0.6891
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.7357
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# variable importance visualization
importance <- varImp(XGB_model, scale = FALSE)
plot(importance)
```



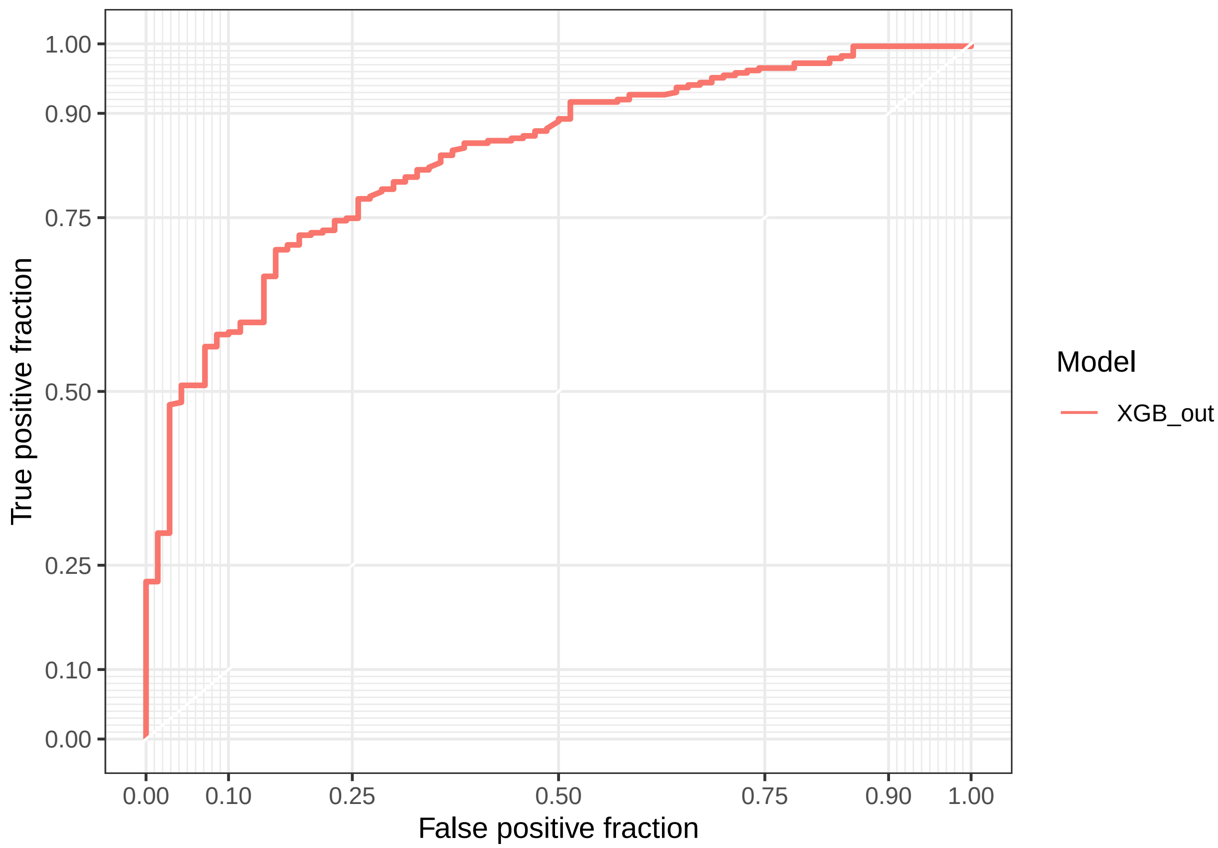

```r
# Save the probabilities
results <- default_tst
results$XGB_out <- test_XGB_prob$EMPLOYEE

# this section of the code sets up the data frame for the plotROC format
results$D <- ifelse(results$STATUS=="TERMINATED",0,1)
longresult <- melt_roc(results, "D", c("XGB_out"))
head(longresult)
```

```
##          D         M    name
## XGB_out1 1 0.9929443 XGB_out
## XGB_out2 1 0.9950969 XGB_out
## XGB_out3 1 0.9948764 XGB_out
## XGB_out4 1 0.9982334 XGB_out
```

```
## XGB_out5 1 0.9852828 XGB_out
## XGB_out6 1 0.9979205 XGB_out
```

```
ggplot(longresult, aes(d=D, m=M, color=name))+
  geom_roc(n.cuts = 0)+style_roc()+
  labs(color = "Model")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: d, m
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```




```
# get AUC's for each model
XGB_Auc <- pROC::roc(results$STATUS, results$XGB_out)
```

```
## Setting levels: control = EMPLOYEE, case = TERMINATED
```

```
## Setting direction: controls > cases
```

```
XGB_Auc$auc
```

```
## Area under the curve: 0.8358
```

```r
# create a confusion matrix
confusionMatrix(test_XGB, results$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        246         27
##   TERMINATED       41         43
##
##                Accuracy : 0.8095
##                  95% CI : (0.7649, 0.8489)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.4259
##
##                   Kappa : 0.4383
##
##  Mcnemar's Test P-Value : 0.1149
##
##             Sensitivity : 0.8571
##             Specificity : 0.6143
##          Pos Pred Value : 0.9011
##          Neg Pred Value : 0.5119
##              Prevalence : 0.8039
##          Detection Rate : 0.6891
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.7357
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Mean imputed table

```r
data <- read_excel("C:/Users/lewis/Downloads/completeData_mean_imp.xlsx")

# Select and filter table with the key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE

                )
```

```r
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```r
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```r
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the echo=T, results=‘hide’, message=F, warning=F to avoid printing unnecessary pages of iterations.

```r
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)
```

```r
# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```r
# let's look at the model results - this is for training only
XGB_model
```

```
## eXtreme Gradient Boosting
##
## 537 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 483, 483, 484, 484, 483, 483, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  logLoss
##   0.3  1          0.6               0.5         50      0.4408233
##   0.3  1          0.6               0.5        100      0.4406826
##   0.3  1          0.6               1.0         50      0.4365679
##   0.3  1          0.6               1.0        100      0.4310240
##   0.3  1          0.8               0.5         50      0.4420102
##   0.3  1          0.8               0.5        100      0.4465207
##   0.3  1          0.8               1.0         50      0.4385861
##   0.3  1          0.8               1.0        100      0.4294654
##   0.3  2          0.6               0.5         50      0.4156709
##   0.3  2          0.6               0.5        100      0.4335515
##   0.3  2          0.6               1.0         50      0.4079831
##   0.3  2          0.6               1.0        100      0.4130111
##   0.3  2          0.8               0.5         50      0.4180035
##   0.3  2          0.8               0.5        100      0.4388174
##   0.3  2          0.8               1.0         50      0.4098623
##   0.3  2          0.8               1.0        100      0.4154270
##   0.4  1          0.6               0.5         50      0.4430469
##   0.4  1          0.6               0.5        100      0.4471876
##   0.4  1          0.6               1.0         50      0.4335203
##   0.4  1          0.6               1.0        100      0.4343865
##   0.4  1          0.8               0.5         50      0.4436377
##   0.4  1          0.8               0.5        100      0.4511538
```

```
##   0.4  1          0.8               1.0        50      0.4357633
##   0.4  1          0.8               1.0       100      0.4352488
##   0.4  2          0.6               0.5        50      0.4316095
##   0.4  2          0.6               0.5       100      0.4447072
##   0.4  2          0.6               1.0        50      0.4164864
##   0.4  2          0.6               1.0       100      0.4399502
##   0.4  2          0.8               0.5        50      0.4425060
##   0.4  2          0.8               0.5       100      0.4730733
##   0.4  2          0.8               1.0        50      0.4109843
##   0.4  2          0.8               1.0       100      0.4327382
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, max_depth = 2, eta
##  = 0.3, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsample
##  = 1.
```

```
summary(XGB_model)
```

```
##                Length Class              Mode
## handle             1  xgb.Booster.handle externalptr
## raw            47870  -none-             raw
## niter              1  -none-             numeric
## call               6  -none-             call
## params             9  -none-             list
## callbacks          1  -none-             list
## feature_names     12  -none-             character
## nfeatures          1  -none-             numeric
## xNames            12  -none-             character
## problemType        1  -none-             character
## tuneValue          7  data.frame         list
## obsLevels          2  -none-             character
## param              1  -none-             list
```

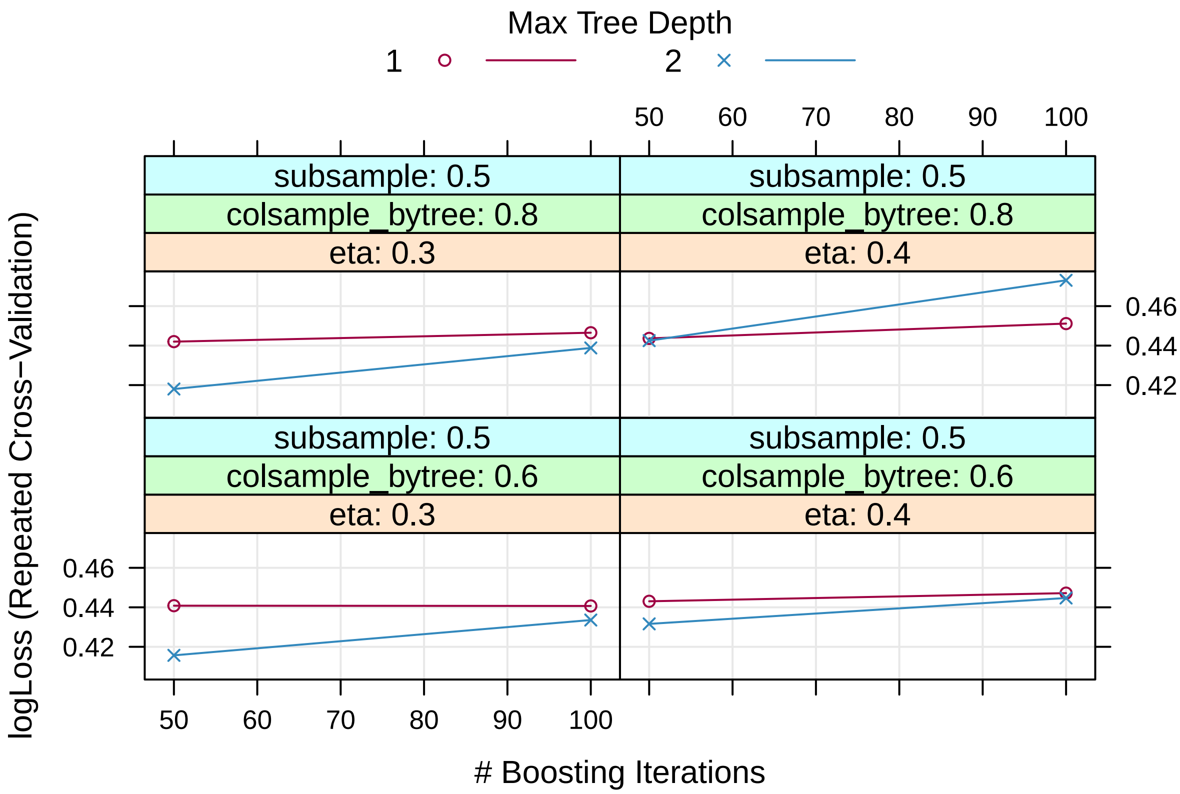
```
head(XGB_model$pred)
```

```
##       pred      obs rowIndex  EMPLOYEE TERMINATED eta max_depth gamma
## 1 EMPLOYEE EMPLOYEE       31 0.9564618 0.04353821 0.3         1     0
## 2 EMPLOYEE EMPLOYEE       36 0.9759111 0.02408886 0.3         1     0
## 3 EMPLOYEE EMPLOYEE       54 0.9718034 0.02819657 0.3         1     0
## 4 EMPLOYEE EMPLOYEE       56 0.9698185 0.03018153 0.3         1     0
## 5 EMPLOYEE EMPLOYEE       71 0.8654606 0.13453943 0.3         1     0
## 6 EMPLOYEE EMPLOYEE       82 0.6170212 0.38297880 0.3         1     0
##   colsample_bytree min_child_weight subsample nrounds    Resample
## 1              0.6                1       0.5     100 Fold01.Rep1
## 2              0.6                1       0.5     100 Fold01.Rep1
## 3              0.6                1       0.5     100 Fold01.Rep1
## 4              0.6                1       0.5     100 Fold01.Rep1
## 5              0.6                1       0.5     100 Fold01.Rep1
## 6              0.6                1       0.5     100 Fold01.Rep1
```

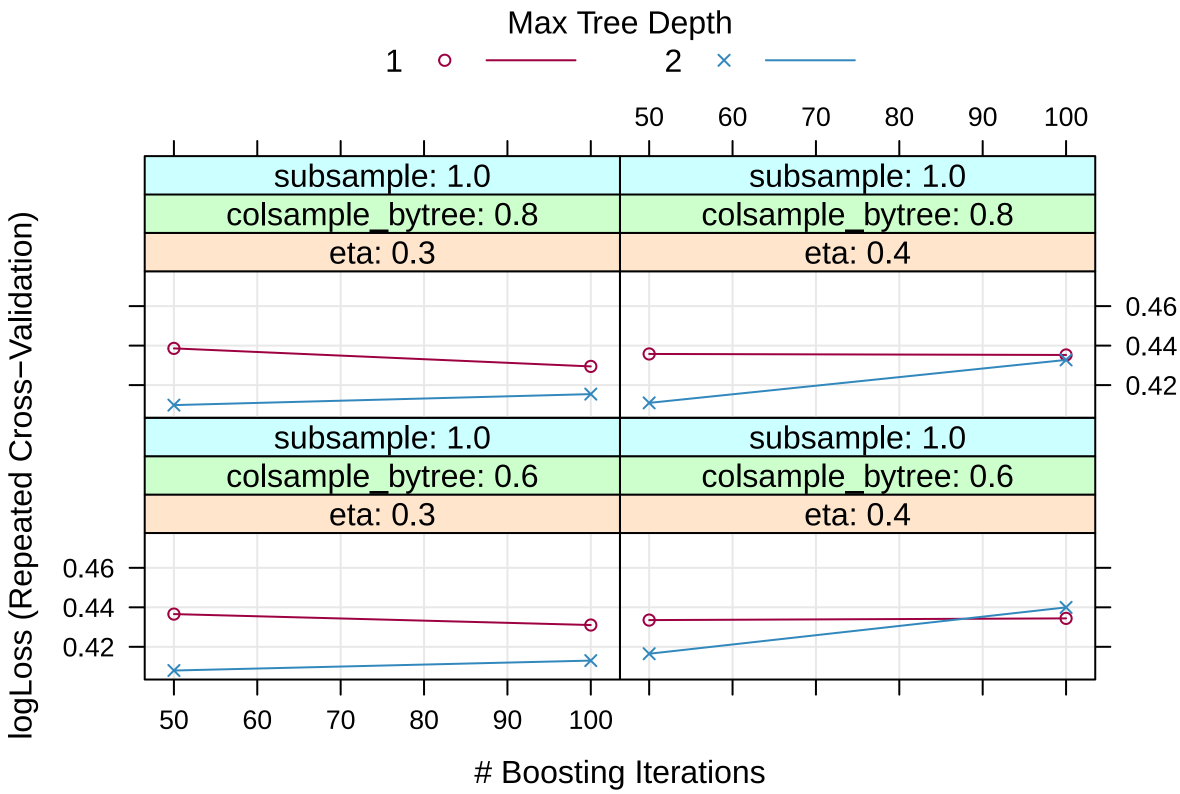
```r
# and produce a confusion matrix
lvs <- c("Employee","Terminated")
truth <- factor(XGB_model$pred$obs)
pred <- factor(XGB_model$pred$pred)
xtab <- table(pred, truth)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##             truth
## pred          EMPLOYEE TERMINATED
##   EMPLOYEE      33822       3151
##   TERMINATED     7650       6929
##
##                Accuracy : 0.7905
##                  95% CI : (0.7869, 0.794)
##     No Information Rate : 0.8045
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4303
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.8155
##             Specificity : 0.6874
##          Pos Pred Value : 0.9148
##          Neg Pred Value : 0.4753
##              Prevalence : 0.8045
##          Detection Rate : 0.6561
##    Detection Prevalence : 0.7172
##       Balanced Accuracy : 0.7515
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# LogLoss (Repeated Cross-Validation) plots using different parameters: eta,
# subsample and colsample bytree.
trellis.par.set(caretTheme())
plot(XGB_model)
```



Max Tree Depth
1
2
subsample: 0.5
colsample_bytree: 0.8
eta: 0.3
eta: 0.4
colsample_bytree: 0.6
logLoss (Repeated Cross−Validation)
0.42
0.44
0.46
50
60
70
80
90
100
Boosting Iterations

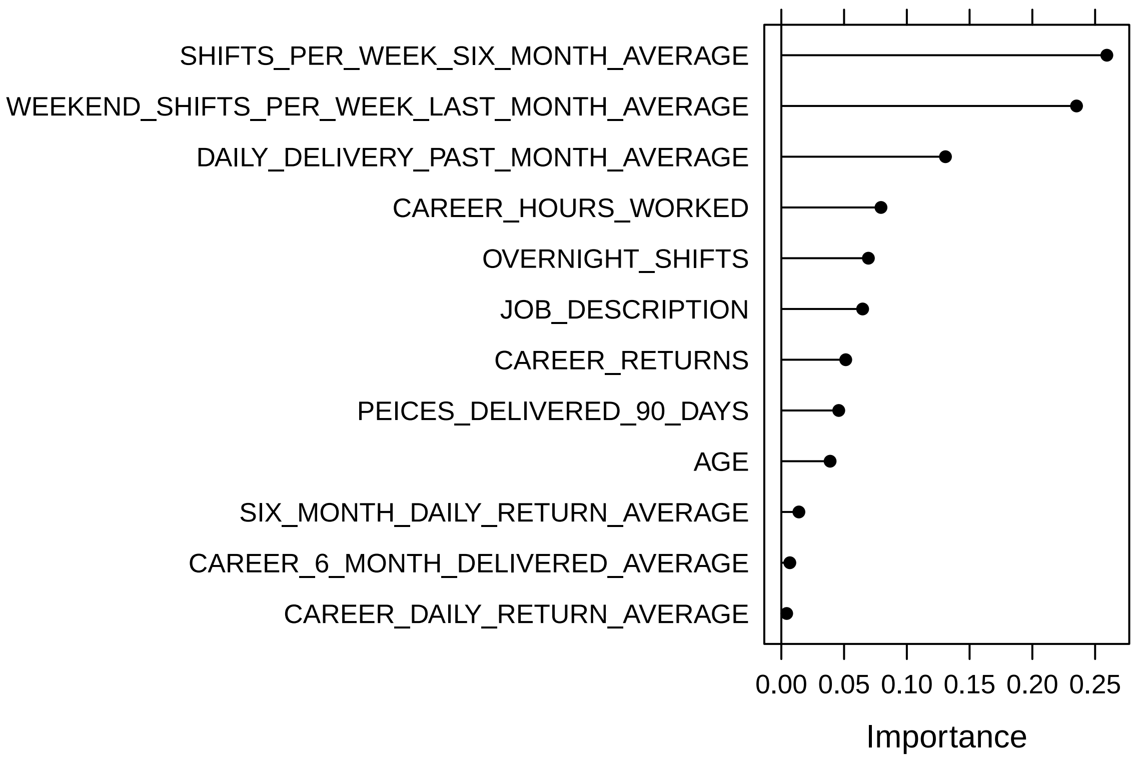



```r
# now apply the XGB model to the Test data
test_XGB <- predict(XGB_model, newdata = default_tst)
test_XGB_prob <- predict(XGB_model, newdata = default_tst, type="prob")
confusionMatrix(test_XGB, default_tst$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        244         25
##   TERMINATED       43         45
##
##                Accuracy : 0.8095
##                  95% CI : (0.7649, 0.8489)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.42588
##
##                   Kappa : 0.4493
##
##  Mcnemar's Test P-Value : 0.03925
##
##             Sensitivity : 0.8502
##             Specificity : 0.6429
##          Pos Pred Value : 0.9071
##          Neg Pred Value : 0.5114
##              Prevalence : 0.8039
```

```
##          Detection Rate : 0.6835
##    Detection Prevalence : 0.7535
##       Balanced Accuracy : 0.7465
##
##        'Positive' Class : EMPLOYEE
##
```

```
# variable importance visualization
importance <- varImp(XGB_model, scale = FALSE)
plot(importance)
```



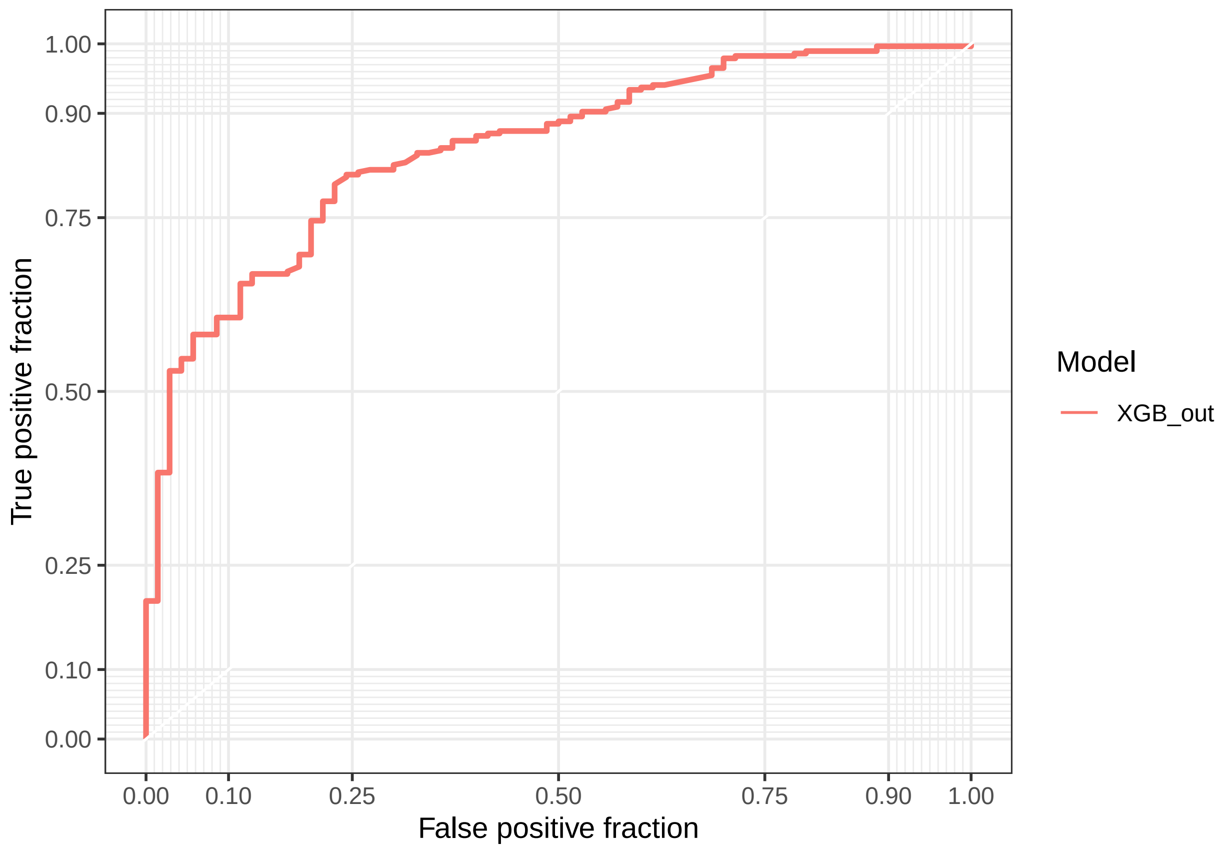

```
# Save the probabilities
results <- default_tst
results$XGB_out <- test_XGB_prob$EMPLOYEE

# this section of the code sets up the data frame for the plotROC format
results$D <- ifelse(results$STATUS=="TERMINATED",0,1)
longresult <- melt_roc(results, "D", c("XGB_out"))
head(longresult)
```

```
##          D         M    name
## XGB_out1 1 0.9548031 XGB_out
## XGB_out2 1 0.9955124 XGB_out
## XGB_out3 1 0.9732369 XGB_out
## XGB_out4 1 0.9882443 XGB_out
```

```
## XGB_out5 1 0.9269701 XGB_out
## XGB_out6 1 0.9948525 XGB_out
```

```
ggplot(longresult, aes(d=D, m=M, color=name))+
  geom_roc(n.cuts = 0)+style_roc()+
  labs(color = "Model")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: d, m
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```




```
# get AUC's for each model
XGB_Auc <- pROC::roc(results$STATUS, results$XGB_out)
```

```
## Setting levels: control = EMPLOYEE, case = TERMINATED
```

```
## Setting direction: controls > cases
```

```
XGB_Auc$auc
```

```
## Area under the curve: 0.8496
```

```r
# create a confusion matrix
confusionMatrix(test_XGB, results$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        244         25
##   TERMINATED       43         45
##
##                Accuracy : 0.8095
##                  95% CI : (0.7649, 0.8489)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.42588
##
##                   Kappa : 0.4493
##
##  Mcnemar's Test P-Value : 0.03925
##
##             Sensitivity : 0.8502
##             Specificity : 0.6429
##          Pos Pred Value : 0.9071
##          Neg Pred Value : 0.5114
##              Prevalence : 0.8039
##          Detection Rate : 0.6835
##    Detection Prevalence : 0.7535
##       Balanced Accuracy : 0.7465
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Random Forest method

```r
data <- read_excel("C:/Users/lewis/Downloads/randomforest output.xlsx")

# Select and filter table with the key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```r
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```r
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the echo=T, results='hide', message=F, warning=F to avoid printing unnecessary pages of iterations.

```r
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)
```

```r
# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```r
# let's look at the model results - this is for training only
XGB_model
```

```
## eXtreme Gradient Boosting
##
## 537 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 483, 483, 484, 484, 483, 483, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  logLoss
##   0.3  1          0.6               0.5         50      0.4819160
##   0.3  1          0.6               0.5        100      0.4815402
##   0.3  1          0.6               1.0         50      0.4719520
##   0.3  1          0.6               1.0        100      0.4707185
##   0.3  1          0.8               0.5         50      0.4817008
##   0.3  1          0.8               0.5        100      0.4826076
##   0.3  1          0.8               1.0         50      0.4694162
##   0.3  1          0.8               1.0        100      0.4703657
##   0.3  2          0.6               0.5         50      0.4756288
##   0.3  2          0.6               0.5        100      0.5070132
##   0.3  2          0.6               1.0         50      0.4612910
##   0.3  2          0.6               1.0        100      0.4676778
##   0.3  2          0.8               0.5         50      0.4737236
##   0.3  2          0.8               0.5        100      0.5019866
##   0.3  2          0.8               1.0         50      0.4603349
##   0.3  2          0.8               1.0        100      0.4803445
##   0.4  1          0.6               0.5         50      0.4854472
##   0.4  1          0.6               0.5        100      0.4885337
##   0.4  1          0.6               1.0         50      0.4750547
##   0.4  1          0.6               1.0        100      0.4724146
##   0.4  1          0.8               0.5         50      0.4911136
##   0.4  1          0.8               0.5        100      0.5003510
```

```
##   0.4  1          0.8              1.0        50      0.4687804
##   0.4  1          0.8              1.0       100      0.4686183
##   0.4  2          0.6              0.5        50      0.4796273
##   0.4  2          0.6              0.5       100      0.5153935
##   0.4  2          0.6              1.0        50      0.4709253
##   0.4  2          0.6              1.0       100      0.4970168
##   0.4  2          0.8              0.5        50      0.5007174
##   0.4  2          0.8              0.5       100      0.5264250
##   0.4  2          0.8              1.0        50      0.4600267
##   0.4  2          0.8              1.0       100      0.4941453
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, max_depth = 2, eta
##  = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 1.
```

```
summary(XGB_model)
```

```
##                Length Class              Mode
## handle            1   xgb.Booster.handle externalptr
## raw           48482   -none-             raw
## niter             1   -none-             numeric
## call              6   -none-             call
## params            9   -none-             list
## callbacks         1   -none-             list
## feature_names    12   -none-             character
## nfeatures         1   -none-             numeric
## xNames           12   -none-             character
## problemType       1   -none-             character
## tuneValue         7   data.frame         list
## obsLevels         2   -none-             character
## param             1   -none-             list
```

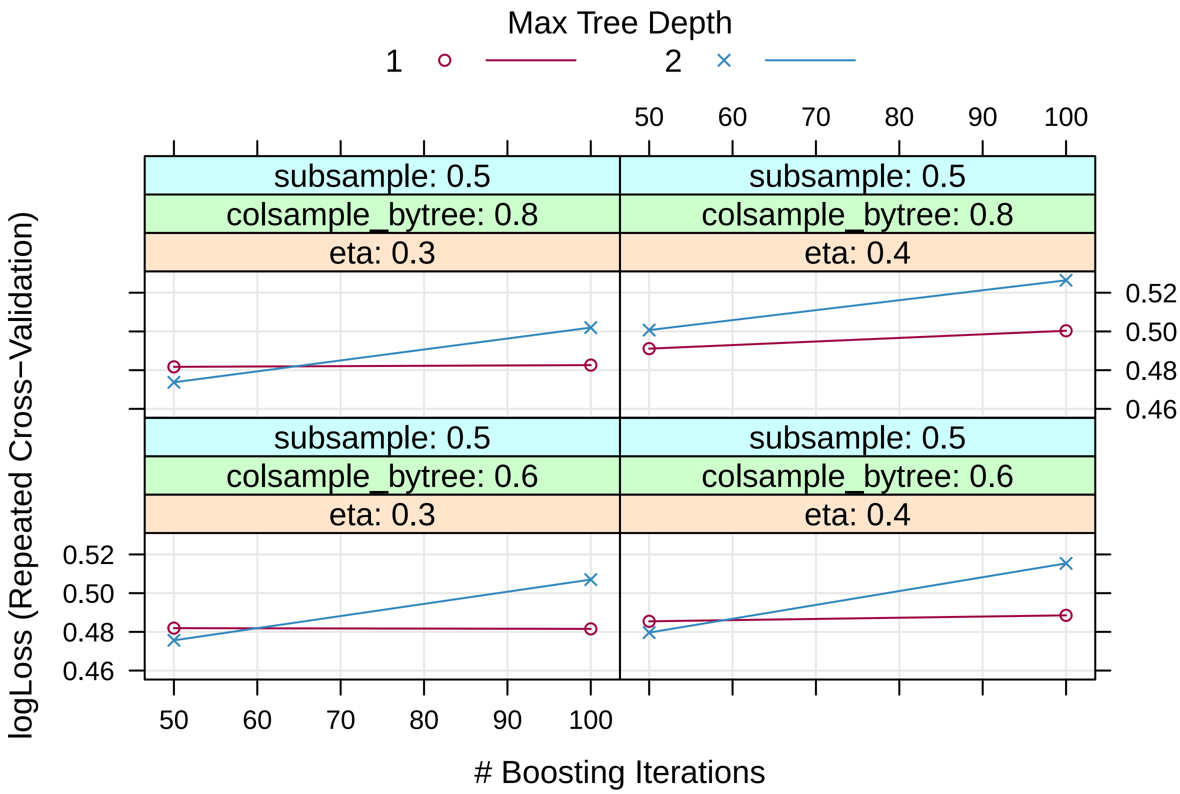
```
head(XGB_model$pred)
```

```
##         pred      obs rowIndex  EMPLOYEE TERMINATED eta max_depth gamma
## 1   EMPLOYEE EMPLOYEE       31 0.9864262 0.01357377 0.3         1     0
## 2   EMPLOYEE EMPLOYEE       36 0.9777490 0.02225101 0.3         1     0
## 3   EMPLOYEE EMPLOYEE       54 0.9779293 0.02207071 0.3         1     0
## 4   EMPLOYEE EMPLOYEE       56 0.6148996 0.38510036 0.3         1     0
## 5 TERMINATED EMPLOYEE       71 0.4739183 0.52608168 0.3         1     0
## 6   EMPLOYEE EMPLOYEE       82 0.7366651 0.26333493 0.3         1     0
##   colsample_bytree min_child_weight subsample nrounds    Resample
## 1              0.6                1       0.5     100 Fold01.Rep1
## 2              0.6                1       0.5     100 Fold01.Rep1
## 3              0.6                1       0.5     100 Fold01.Rep1
## 4              0.6                1       0.5     100 Fold01.Rep1
## 5              0.6                1       0.5     100 Fold01.Rep1
## 6              0.6                1       0.5     100 Fold01.Rep1
```

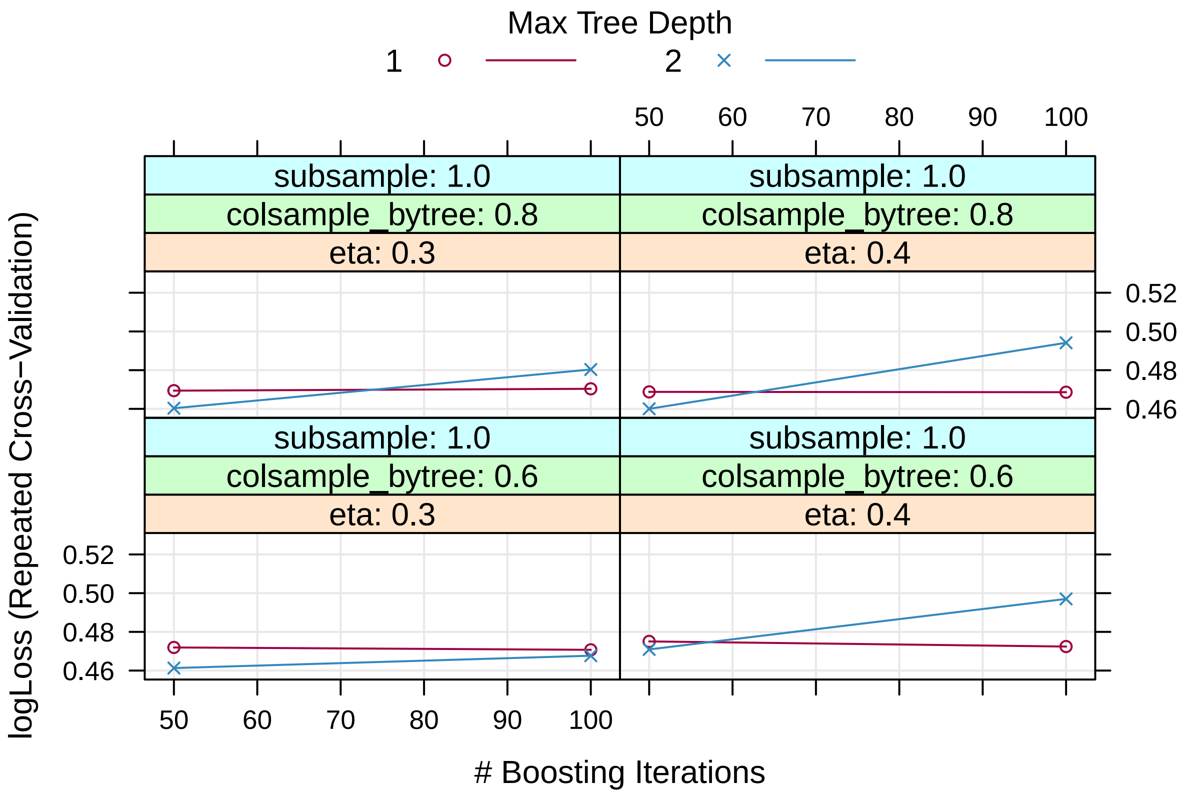
```r
# and produce a confusion matrix
lvs <- c("Employee","Terminated")
truth <- factor(XGB_model$pred$obs)
pred <- factor(XGB_model$pred$pred)
xtab <- table(pred, truth)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##             truth
## pred          EMPLOYEE TERMINATED
##   EMPLOYEE      33613       3752
##   TERMINATED     7859       6328
##
##                Accuracy : 0.7748
##                  95% CI : (0.7711, 0.7784)
##     No Information Rate : 0.8045
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3797
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.8105
##             Specificity : 0.6278
##          Pos Pred Value : 0.8996
##          Neg Pred Value : 0.4460
##              Prevalence : 0.8045
##          Detection Rate : 0.6520
##    Detection Prevalence : 0.7248
##       Balanced Accuracy : 0.7191
##
##        'Positive' Class : EMPLOYEE
##
```

```r
# LogLoss (Repeated Cross-Validation) plots using different parameters: eta,
# subsample and colsample bytree.
trellis.par.set(caretTheme())
plot(XGB_model)
```



Max Tree Depth
1
2
subsample: 0.5
colsample_bytree: 0.8
eta: 0.3
eta: 0.4
colsample_bytree: 0.6
logLoss (Repeated Cross−Validation)
0.46
0.48
0.50
0.52
50
60
70
80
90
100
Boosting Iterations

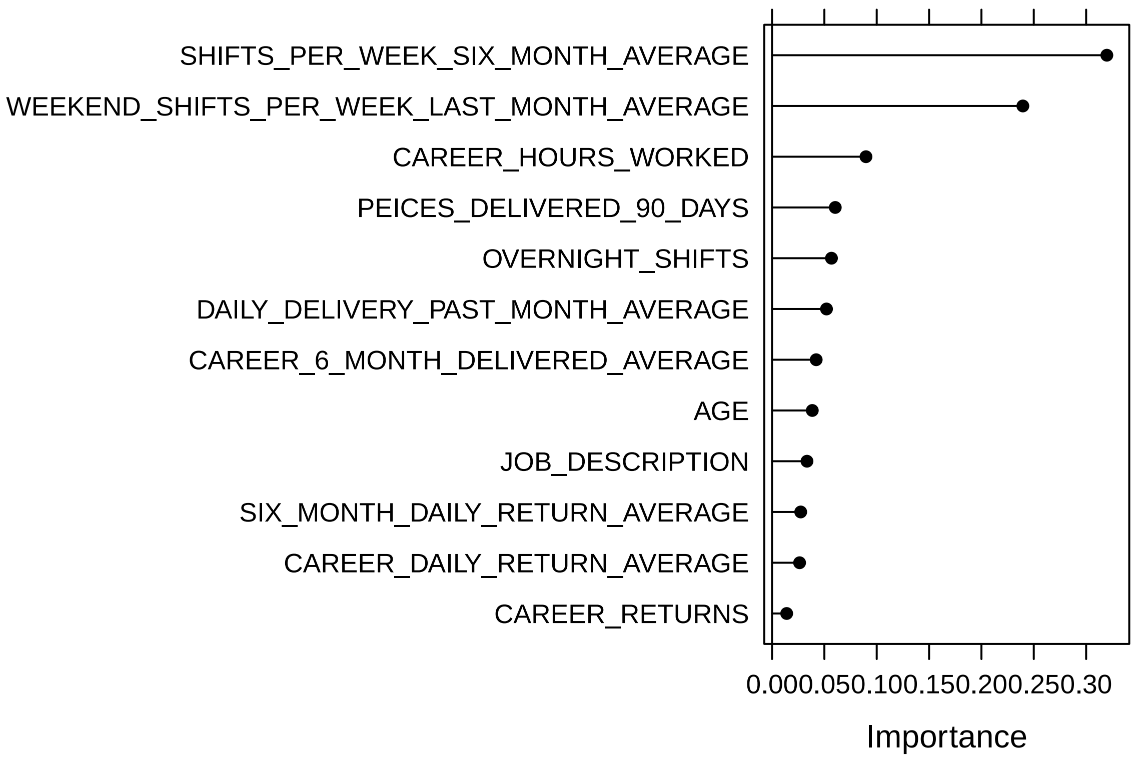



```r
# now apply the XGB model to the Test data
test_XGB <- predict(XGB_model, newdata = default_tst)
test_XGB_prob <- predict(XGB_model, newdata = default_tst, type="prob")
confusionMatrix(test_XGB, default_tst$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        243         28
##   TERMINATED       44         42
##
##                Accuracy : 0.7983
##                  95% CI : (0.7529, 0.8387)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.6350
##
##                   Kappa : 0.4112
##
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.8467
##             Specificity : 0.6000
##          Pos Pred Value : 0.8967
##          Neg Pred Value : 0.4884
##              Prevalence : 0.8039
```

```
##          Detection Rate : 0.6807
##    Detection Prevalence : 0.7591
##       Balanced Accuracy : 0.7233
##
##        'Positive' Class : EMPLOYEE
##
```

```
# variable importance visualization
importance <- varImp(XGB_model, scale = FALSE)
plot(importance)
```



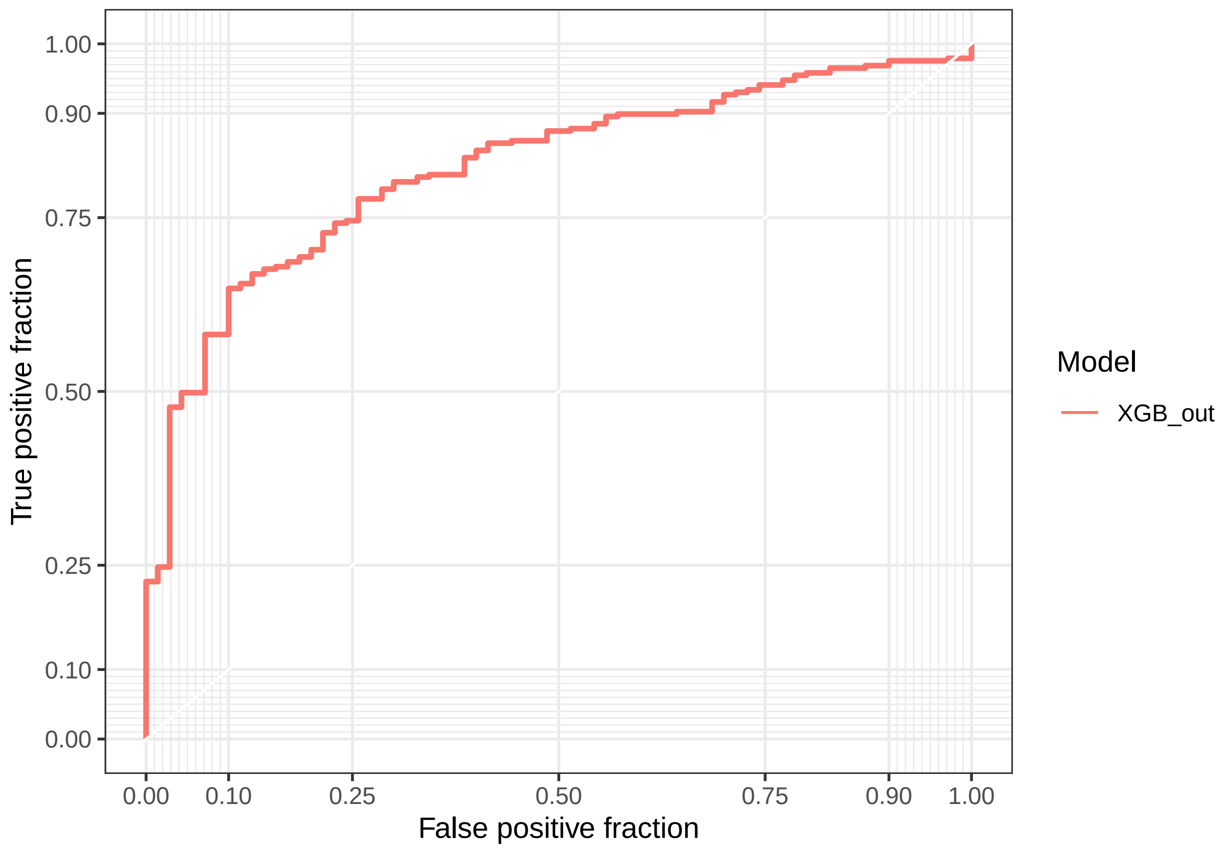

```
# Save the probabilities
results <- default_tst
results$XGB_out <- test_XGB_prob$EMPLOYEE

# this section of the code sets up the data frame for the plotROC format
results$D <- ifelse(results$STATUS=="TERMINATED",0,1)
longresult <- melt_roc(results, "D", c("XGB_out"))
head(longresult)
```

```
##          D         M    name
## XGB_out1 1 0.9828735 XGB_out
## XGB_out2 1 0.9966173 XGB_out
## XGB_out3 1 0.3685744 XGB_out
## XGB_out4 1 0.9882782 XGB_out
```

```
## XGB_out5 1 0.9851206 XGB_out
## XGB_out6 1 0.9901319 XGB_out
```

```r
ggplot(longresult, aes(d=D, m=M, color=name))+
  geom_roc(n.cuts = 0)+style_roc()+
  labs(color = "Model")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: d, m
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```




```r
# get AUC's for each model
XGB_Auc <- pROC::roc(results$STATUS, results$XGB_out)
```

```
## Setting levels: control = EMPLOYEE, case = TERMINATED
```

```
## Setting direction: controls > cases
```

```r
XGB_Auc$auc
```

```
## Area under the curve: 0.8211
```

```r
# create a confusion matrix
confusionMatrix(test_XGB, results$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        243         28
##   TERMINATED       44         42
##
##                Accuracy : 0.7983
##                  95% CI : (0.7529, 0.8387)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.6350
##
##                   Kappa : 0.4112
##
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.8467
##             Specificity : 0.6000
##          Pos Pred Value : 0.8967
##          Neg Pred Value : 0.4884
##              Prevalence : 0.8039
##          Detection Rate : 0.6807
##    Detection Prevalence : 0.7591
##       Balanced Accuracy : 0.7233
##
##        'Positive' Class : EMPLOYEE
##
```

Import CSV File: Piecewise Imputed Table

```r
data <- read_csv("C:/Users/lewis/Downloads/interpolated_ops_features_piecewise.csv")
```

```
## Rows: 894 Columns: 30
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): DELETE_CODE, JOB_DESCRIPTION
## dbl (28): Unnamed: 0, TECH, AGE, STATUS, WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Select and filter table with the key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
```

```
                CAREER_6_MONTH_DELIVERED_AVERAGE,
                WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
                AGE,
                DAILY_DELIVERY_PAST_MONTH_AVERAGE,
                CAREER_DAILY_RETURN_AVERAGE,
                OVERNIGHT_SHIFTS,
                JOB_DESCRIPTION,
                SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                 )
```

```
# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(0,1),
                      labels = c("EMPLOYEE",
                                 "TERMINATED"
                      ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))


# Imbalance data: zero for employee and one for terminated
table(data$STATUS)
```

```
##
##   EMPLOYEE TERMINATED
##        719        175
```

```
# separate the file into train test subsets with 60/40 ratio, and using STATUS column
# as a predictable or y-label
set.seed(365)
default_idx <- createDataPartition(data$STATUS, p=0.6, list = FALSE)
default_trn <- data[default_idx, ]
default_tst <- data[-default_idx,]
table(default_trn$STATUS) # train table
```

```
##
##   EMPLOYEE TERMINATED
##        432        105
```

```
table(default_tst$STATUS) # test table
```

```
##
##   EMPLOYEE TERMINATED
##        287         70
```

We are using the error=TRUE to print our model error due to the prescense of missing values on this imputed file.

```
# now the data is ready to go into our machine learning models
# here is the process using caret's trainControl function
# https://stackoverflow.com/questions/65848998/smote-within-a-recipe-versus-smote-in-traincontrol
```

```r
set.seed(365)
# training control setup - note the SMOTE sampling and the k = 10 fold C-Validation
trn_ctrl <- trainControl(summaryFunction = mnLogLoss, #similar to "mlogloss" from the v5
                         savePredictions = TRUE,
                         sampling = "smote",
                         method = "repeatedcv",
                         number = 10,
                         repeats = 3,
                         classProbs = TRUE,
                         allowParallel = FALSE)

# Run the XGboost model
# https://www.kaggle.com/nagsdata/simple-r-xgboost-caret-kernel
# the metric = ROC means "repeated cross validation"

# Preprocess: Center subtracts the mean of the predictor's data (again from the data in x)
# from the predictor values and scale divides by the standard deviation.
set.seed(365)
XGB_model <- train(STATUS~ ., data=default_trn,
                   preProcess = c("center","scale"), # See comment above
                   method="xgbTree", # Machine learning model name
                   eval_metric = "mlogloss", #  multiclass logloss
                   tuneLength = 2, # using 3 has similar pattern plot as 2
                   trControl = trn_ctrl)
```

```
## Error in na.fail.default(structure(list(STATUS = structure(c(1L, 1L, 1L, : missing values in object
```