Student Dropout Competition: Modelling process

Hamed

3/25/2020

Modelling

```
# Load the engineered and transformed features
transformed_train=read.csv("transformed_train.csv",header = T)
transformed_test=read.csv("transformed_test.csv",header = T)

# Convert the response variable from an integer to a factor
transformed_train$Dropout=as.factor(transformed_train$Dropout)

# Load required packages
library(class)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# Split the transformed_train dataset into training and validation datasets
set.seed(123)
## 80% of the sample size
smp_size <- floor(0.80 * nrow(transformed_train))
train_ind <- sample(seq_len(nrow(transformed_train)), size = smp_size)
train.set <- transformed_train[train_ind, ]
validation.set <- transformed_train[-train_ind, ]

dim(train.set)

## [1] 9808   53
```

```r
str(train.set)

## 'data.frame':    9808 obs. of  53 variables:
##  $ cohort_term.1                            : num  0.495 0.495 0.495
0.495 0.495 ...
##  $ cohort_term.3                            : num  -0.495 -0.495 -0.495 -
0.495 -0.495 ...
##  $ Marital.Status.Divorced                  : num  -0.131 -0.131 -0.131 -
0.131 -0.131 ...
##  $ Marital.Status.Married                   : num  -0.285 -0.285 3.503 -
0.285 -0.285 ...
##  $ Marital.Status.Separated                 : num  -0.124 -0.124 -0.124 -
0.124 -0.124 ...
##  $ Marital.Status.Single                    : num  0.347 0.347 -2.883
0.347 0.347 ...
##  $ Adjusted.Gross.Income                    : num  -0.276 -0.255 -0.126 -
0.257 -0.329 ...
##  $ Parent.Adjusted.Gross.Income             : num  2.0827 3.8143 -0.5842
-0.5842 -0.0579 ...
##  $ Father.s.Highest.Grade.Level.College     : num  1.79 -0.559 -0.559 -
0.559 -0.559 ...
##  $ Father.s.Highest.Grade.Level.High.School  : num  -1.07 0.934 -1.07
0.934 0.934 ...
##  $ Father.s.Highest.Grade.Level.Middle.School: num  -0.33 -0.33 -0.33 -
0.33 -0.33 ...
##  $ Father.s.Highest.Grade.Level.Unknown     : num  -0.387 -0.387 2.583 -
0.387 -0.387 ...
##  $ Mother.s.Highest.Grade.Level.College     : num  1.798 1.798 -0.556 -
0.556 -0.556 ...
##  $ Mother.s.Highest.Grade.Level.High.School  : num  -1.093 -1.093 -1.093
0.914 0.914 ...
##  $ Mother.s.Highest.Grade.Level.Middle.School: num  -0.322 -0.322 -0.322 -
0.322 -0.322 ...
##  $ Mother.s.Highest.Grade.Level.Unknown     : num  -0.378 -0.378 2.643 -
0.378 -0.378 ...
##  $ Housing.Off.Campus                       : num  -1.1 -1.1 0.909 0.909
0.909 ...
##  $ Housing.On.Campus.Housing                : num  -0.363 -0.363 -0.363 -
0.363 -0.363 ...
##  $ Housing.With.Parent                      : num  1.406 1.406 -0.711 -
0.711 -0.711 ...
##  $ Total_loan                               : num  1.381 -0.441 2.786 -
0.731 -0.731 ...
##  $ Total_grant                              : num  -0.773 -0.773 -0.773
1.156 0.203 ...
##  $ Total_scholarship                        : num  -0.24 -0.24 -0.24 -
0.24 -0.24 ...
##  $ Total_WorkStudy                          : num  -0.213 -0.213 -0.213 -
0.213 -0.213 ...
##  $ Cohort.2011.12                           : num  -0.459 -0.459 -0.459 -
```

```
0.459 -0.459 ...
##  $ Cohort.2012.13                                : num  2.226 2.226 -0.449 -
0.449 2.226 ...
##  $ Cohort.2013.14                                : num  -0.433 -0.433 -0.433 -
0.433 -0.433 ...
##  $ Cohort.2014.15                                : num  -0.452 -0.452 -0.452 -
0.452 -0.452 ...
##  $ Cohort.2015.16                                : num  -0.466 -0.466 -0.466
2.148 -0.466 ...
##  $ Cohort.2016.17                                : num  -0.424 -0.424 2.356 -
0.424 -0.424 ...
##  $ CohortTerm                                    : int  1 1 1 1 1 3 1 1 1 1
...
##  $ Hispanic                                      : int  0 1 1 1 1 0 1 0 0 1
...
##  $ Black                                         : int  0 0 0 0 0 0 0 0 0 0
...
##  $ NativeHawaiian                                : int  0 0 0 0 0 0 0 0 0 0
...
##  $ White                                         : int  1 0 0 0 0 1 0 0 1 0
...
##  $ HSDipYr                                       : num  -1.2811 -2.1034 0.3636
-0.0476 0.3636 ...
##  $ HSGPAUnwtd                                    : num  1.225 1.32 -0.962
2.009 2.282 ...
##  $ EnrollmentStatus                              : int  1 1 2 1 1 2 1 1 1 2
...
##  $ NumColCredAttemptTransfer                     : num  -0.91 -0.91 -0.864 -
0.91 -0.91 ...
##  $ NumColCredAcceptTransfer                      : num  -0.989 -0.989 -0.931 -
0.989 -0.989 ...
##  $ CumLoanAtEntry                                : num  -1.19 -1.19 0.58 -1.19
-1.19 ...
##  $ HighDeg                                       : int  0 0 0 0 0 0 0 0 0 0
...
##  $ MathPlacement                                 : int  1 0 0 1 0 0 1 0 1 0
...
##  $ EngPlacement                                  : int  1 1 0 0 1 0 1 1 0 0
...
##  $ GatewayMathStatus                             : int  0 1 0 0 0 0 0 1 0 0
...
##  $ GatewayEnglishStatus                          : int  0 1 0 1 0 0 0 1 1 0
...
##  $ CompleteDevMath                               : num  0.273 -2 -2 0.25 -2
...
##  $ CompleteDevEnglish                            : num  0.0909 0 -2 -2 0 ...
##  $ Major1                                        : num  -0.477 0.943 0.885
0.807 0.927 ...
##  $ Complete1                                     : num  0 0 2.67 0 0 ...
##  $ CompleteCIP1                                  : num  -0.515 -0.515 2.819 -
```

```
0.515 -0.515 ...
##  $ TermGPA                                      : num  0.043 -0.276 0.663
0.618 0.611 ...
##  $ CumGPA                                       : num  0.043 -0.276 0.663
0.618 0.611 ...
##  $ Dropout                                      : Factor w/ 2 levels "0","1":
1 2 1 1 2 1 2 1 1 2 ...
```

Gradient Boosting Models

GBM model training

```
# Stochastic Gradient Boosting GBM model

library(caret)
library(gbm)

## Loaded gbm 2.1.5

set.seed(123)

fit.gbm <- train(Dropout~., data=train.set, method="gbm", metric="Accuracy",
trControl=trainControl(method="repeatedcv", number=10, repeats=3),
verbose=FALSE)

fit.gbm

## Stochastic Gradient Boosting
##
## 9808 samples
##   52 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 8827, 8827, 8827, 8827, 8827, 8828, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8796901  0.7506720
##   1                  100      0.8949842  0.7795752
##   1                  150      0.9001499  0.7898122
##   2                   50      0.8963097  0.7828849
##   2                  100      0.9113310  0.8127880
##   2                  150      0.9185358  0.8280719
##   3                   50      0.9054172  0.8008253
##   3                  100      0.9190456  0.8293683
##   3                  150      0.9243137  0.8404221
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth
=
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

GBM: Model Evaluation and Prediction

```r
# Evaluation of model accuracy
predict_gbm<-predict.train(object=fit.gbm,validation.set,type="raw")

confusionMatrix(predict_gbm,validation.set$Dropout)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1436   87
##          1   83  847
##
##                Accuracy : 0.9307
##                  95% CI : (0.9199, 0.9404)
##     No Information Rate : 0.6192
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8529
##
##  Mcnemar's Test P-Value : 0.818
##
##             Sensitivity : 0.9454
##             Specificity : 0.9069
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.9108
##              Prevalence : 0.6192
##          Detection Rate : 0.5854
##    Detection Prevalence : 0.6209
##       Balanced Accuracy : 0.9261
##
##        'Positive' Class : 0
##

# Predict new data
pred.gbm<-predict.train(object=fit.gbm,transformed_test,type="raw")
pred.gbm=as.data.frame(pred.gbm)

head(pred.gbm)

##   pred.gbm
## 1        1
## 2        1
```

```
## 3        0
## 4        0
## 5        0
## 6        1

# save the prediction
#write.csv(pred.gbm,"D:/Hamed/KAGGLE
COMPETITION/FEATURES/my_new_submission/submission_gbm.csv")
```

Logistic Regression

```
library(mlbench)     # for PimaIndiansDiabetes2 dataset
library(dplyr)       # for data manipulation (dplyr)
library(broom)       # for making model summary tidy
library(visreg)      # for potting logodds and probability
library(margins)     # to calculate Average Marginal Effects
library(rcompanion)  # to calculate pseudo R2
library(ROCR)        # to compute and plot Reciever Opering Curve

## Loading required package: gplots

## Registered S3 method overwritten by 'gdata':
##   method         from
##   reorder.factor DescTools

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

##
## Attaching package: 'ROCR'

## The following object is masked from 'package:margins':
##
##     prediction

#Fitting a binary logistic regression
model_logi <- glm(Dropout~., data = train.set, family = "binomial")
#Model summary
summary(model_logi)

##
## Call:
## glm(formula = Dropout ~ ., family = "binomial", data = train.set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1789  -0.1849  -0.0001   0.2184   4.8151
##
## Coefficients: (8 not defined because of singularities)
```

```
##                                             Estimate Std. Error z value
## (Intercept)                                1.428e+01  3.553e+01   0.402
## cohort_term.1                              4.069e-01  4.292e-02   9.482
## cohort_term.3                                     NA         NA      NA
## Marital.Status.Divorced                    3.212e-02  4.622e-02   0.695
## Marital.Status.Married                    -2.311e-03  4.584e-02  -0.050
## Marital.Status.Separated                   1.342e-02  4.241e-02   0.316
## Marital.Status.Single                             NA         NA      NA
## Adjusted.Gross.Income                      1.807e-02  3.113e-02   0.581
## Parent.Adjusted.Gross.Income              -4.429e-01  5.208e-02  -8.503
## Father.s.Highest.Grade.Level.College       7.676e-02  6.695e-02   1.146
## Father.s.Highest.Grade.Level.High.School  -5.578e-02  7.165e-02  -0.778
## Father.s.Highest.Grade.Level.Middle.School -2.473e-02  5.571e-02  -0.444
## Father.s.Highest.Grade.Level.Unknown              NA         NA      NA
## Mother.s.Highest.Grade.Level.College       5.520e-02  6.679e-02   0.826
## Mother.s.Highest.Grade.Level.High.School  -3.212e-02  7.062e-02  -0.455
## Mother.s.Highest.Grade.Level.Middle.School  6.917e-02  5.412e-02   1.278
## Mother.s.Highest.Grade.Level.Unknown              NA         NA      NA
## Housing.Off.Campus                         8.099e-02  5.150e-02   1.572
## Housing.On.Campus.Housing                  1.699e-01  4.801e-02   3.539
## Housing.With.Parent                               NA         NA      NA
## Total_loan                                -8.723e-01  4.722e-02 -18.473
## Total_grant                               -1.126e+00  5.049e-02 -22.295
## Total_scholarship                         -4.822e-01  5.696e-02  -8.465
## Total_WorkStudy                           -5.924e-02  4.330e-02  -1.368
## Cohort.2011.12                             9.348e+00  8.817e+01   0.106
## Cohort.2012.13                             8.976e+00  8.697e+01   0.103
## Cohort.2013.14                             8.269e+00  8.484e+01   0.097
## Cohort.2014.15                             8.020e+00  8.733e+01   0.092
## Cohort.2015.16                             7.503e+00  8.902e+01   0.084
## Cohort.2016.17                                    NA         NA      NA
## CohortTerm                                        NA         NA      NA
## Hispanic                                  -1.155e-01  1.132e-01  -1.021
## Black                                      2.421e-01  1.295e-01   1.869
## NativeHawaiian                            -1.711e+01  2.158e+03  -0.008
## White                                      1.457e-01  1.202e-01   1.212
## HSDipYr                                    1.102e-02  4.926e-02   0.224
## HSGPAUnwtd                                -2.124e-01  6.075e-02  -3.497
## EnrollmentStatus                          -7.910e+00  6.263e-01 -12.629
## NumColCredAttemptTransfer                 -8.476e-02  8.733e-02  -0.971
## NumColCredAcceptTransfer                   1.603e-01  1.102e-01   1.455
## CumLoanAtEntry                             3.336e+00  2.791e-01  11.955
## HighDeg                                    5.053e-03  5.821e-02   0.087
## MathPlacement                             -3.497e+00  7.347e-01  -4.759
## EngPlacement                              -1.435e+00  6.261e-01  -2.292
## GatewayMathStatus                         -2.164e-01  1.340e-01  -1.615
## GatewayEnglishStatus                      -3.266e-01  1.235e-01  -2.645
## CompleteDevMath                            1.195e+00  3.357e-01   3.561
## CompleteDevEnglish                         3.731e-01  2.795e-01   1.335
## Major1                                    -1.547e-01  4.395e-02  -3.519
```

```
## Complete1                                    -5.401e+00  5.374e-01 -10.049
## CompleteCIP1                                 -1.248e-01  4.688e-01  -0.266
## TermGPA                                      -7.429e-01  5.277e-02 -14.078
## CumGPA                                               NA         NA      NA
##                                              Pr(>|z|)
## (Intercept)                                  0.687630
## cohort_term.1                                 < 2e-16 ***
## cohort_term.3                                      NA
## Marital.Status.Divorced                      0.487098
## Marital.Status.Married                       0.959789
## Marital.Status.Separated                     0.751628
## Marital.Status.Single                              NA
## Adjusted.Gross.Income                        0.561571
## Parent.Adjusted.Gross.Income                  < 2e-16 ***
## Father.s.Highest.Grade.Level.College         0.251621
## Father.s.Highest.Grade.Level.High.School     0.436305
## Father.s.Highest.Grade.Level.Middle.School 0.657101
## Father.s.Highest.Grade.Level.Unknown               NA
## Mother.s.Highest.Grade.Level.College         0.408565
## Mother.s.Highest.Grade.Level.High.School     0.649233
## Mother.s.Highest.Grade.Level.Middle.School 0.201209
## Mother.s.Highest.Grade.Level.Unknown               NA
## Housing.Off.Campus                           0.115837
## Housing.On.Campus.Housing                    0.000402 ***
## Housing.With.Parent                                NA
## Total_loan                                    < 2e-16 ***
## Total_grant                                   < 2e-16 ***
## Total_scholarship                             < 2e-16 ***
## Total_WorkStudy                              0.171217
## Cohort.2011.12                               0.915561
## Cohort.2012.13                               0.917800
## Cohort.2013.14                               0.922364
## Cohort.2014.15                               0.926825
## Cohort.2015.16                               0.932837
## Cohort.2016.17                                     NA
## CohortTerm                                         NA
## Hispanic                                     0.307415
## Black                                        0.061641 .
## NativeHawaiian                               0.993676
## White                                        0.225555
## HSDipYr                                      0.822956
## HSGPAUnwtd                                   0.000471 ***
## EnrollmentStatus                              < 2e-16 ***
## NumColCredAttemptTransfer                    0.331738
## NumColCredAcceptTransfer                     0.145646
## CumLoanAtEntry                                < 2e-16 ***
## HighDeg                                      0.930828
## MathPlacement                                1.94e-06 ***
## EngPlacement                                 0.021881 *
## GatewayMathStatus                            0.106278
```

```
## GatewayEnglishStatus                            0.008181 **
## CompleteDevMath                                 0.000370 ***
## CompleteDevEnglish                              0.181826
## Major1                                          0.000433 ***
## Complete1                                        < 2e-16 ***
## CompleteCIP1                                    0.790123
## TermGPA                                          < 2e-16 ***
## CumGPA                                                NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13095.4  on 9807  degrees of freedom
## Residual deviance:  4201.5  on 9763  degrees of freedom
## AIC: 4291.5
##
## Number of Fisher Scoring iterations: 18
```

Model fit statistics

```
# Pseudo R_squared values and Likelyhood ratio test
nagelkerke(model_logi)
```

```
## $Models
##
## Model: "glm, Dropout ~ ., binomial, train.set"
## Null:  "glm, Dropout ~ 1, binomial, train.set"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.679161
## Cox and Snell (ML)                   0.596186
## Nagelkerke (Cragg and Uhler)         0.809058
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq p.value
##      -44     -4446.9 8893.9       0
##
## $Number.of.observations
##
## Model: 9808
## Null:  9808
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on
refitting with ML"
##
## $Warnings
## [1] "None"
```

ODDS Ratios

```
# The ODDS ratio can be retrieved in a beautiful tidy formatted table
# using the tidy( ) function of broom package.
tidy(model_logi, exponentiate = TRUE, conf.level = 0.95) #odds ratio
```

```
## # A tibble: 45 x 5
##    term                                      estimate std.error statistic
p.value
##    <chr>                                        <dbl>     <dbl>     <dbl>
<dbl>
##  1 (Intercept)                                1.60e+6   35.5       0.402
6.88e- 1
##  2 cohort_term.1                              1.50e+0    0.0429    9.48
2.49e-21
##  3 Marital.Status.Divorced                    1.03e+0    0.0462    0.695
4.87e- 1
##  4 Marital.Status.Married                     9.98e-1    0.0458   -0.0504
9.60e- 1
##  5 Marital.Status.Separated                   1.01e+0    0.0424    0.316
7.52e- 1
##  6 Adjusted.Gross.Income                      1.02e+0    0.0311    0.581
5.62e- 1
##  7 Parent.Adjusted.Gross.Income               6.42e-1    0.0521   -8.50
1.85e-17
##  8 Father.s.Highest.Grade.Level.College       1.08e+0    0.0670    1.15
2.52e- 1
##  9 Father.s.Highest.Grade.Level.High.Sc~      9.46e-1    0.0717   -0.778
4.36e- 1
## 10 Father.s.Highest.Grade.Level.Middle.~      9.76e-1    0.0557   -0.444
6.57e- 1
## # ... with 35 more rows
```

Model Evaluation on Test Data Set

```
# Confusion matrix
# predict the test dataset
pred <- predict(model_logi, validation.set, type="response")
predicted <-ifelse(pred>0.5,1,0) # round of the value; >0.5 will convert to 1
else 0
table(predicted)
```

```
## predicted
##    0    1
## 1507  946
```

```
# Creating a contigency table
tab <- table(Predicted = predicted, Reference = validation.set$Dropout)
tab
```

```
##          Reference
## Predicted    0    1
##         0 1423   84
##         1   96  850
```

Accuracy

```
# Creating a dataframe of observed and predicted data
library(yardstick)

## For binary classification, the first factor level is assumed to be the
event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.

##
## Attaching package: 'yardstick'

## The following object is masked from 'package:rcompanion':
##
##     accuracy

## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity

act_pred <- data.frame(observed = validation.set$Dropout,
predicted=factor(predicted))

# Calculating Accuracy
accuracy_est <- accuracy(act_pred, observed, predicted)
print(accuracy_est)

## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.927
```

Classification Report

```
# Precision, F1-score and recall values
library(yardstick)
# Creating a actual/observed vs predicted dataframe
act_pred <- data.frame(observed = validation.set$Dropout, predicted =
                         factor(predicted))
# Calculating precision, recall and F1_score
prec <- precision(act_pred, observed, predicted)
rec <- recall(act_pred, observed, predicted)
F1_score <- f_meas(act_pred, observed, predicted) #called f_measure
print(prec)

## # A tibble: 1 x 3
##   .metric   .estimator .estimate
```

```
##   <chr>     <chr>          <dbl>
## 1 precision binary         0.944
```

```
print(rec)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 recall  binary         0.937
```

```
print(F1_score)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.941
```

Prediction using new data

```
pred_log <- predict(model_logi, transformed_test, type="response")

predicted <- round(pred_log) # round of the value; >0.5 will convert to 1
else 0
submission_lr=as.data.frame(predicted)

head(submission_lr)
```

```
##   predicted
## 1         1
## 2         1
## 3         0
## 4         0
## 5         1
## 6         1
```

```
# save the file
#write.csv(submission_lr,"D:/Hamed/KAGGLE
COMPETITION/FEATURES/my_new_submission/submission_lr.csv")
```

Support Vector Machine (SVM)

```
library(dplyr)
library(mlr)
```

```
## Loading required package: ParamHelpers

## 'mlr' is in maintenance mode since July 2019. Future development
## efforts will go into its successor 'mlr3' (<https://mlr3.mlr-org.com>).

##
## Attaching package: 'mlr'
```

```
## The following object is masked from 'package:ROCR':
##
##     performance

## The following object is masked from 'package:caret':
##
##     train

library(caret)
library(ROCR)
library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

library(e1071)

##
## Attaching package: 'e1071'

## The following object is masked from 'package:mlr':
##
##     impute

library(foreach)
library(doParallel)

## Loading required package: iterators

## Loading required package: parallel

model_svm <-
svm(Dropout~.,data=train.set,trControl=trainControl("cv",number=10),
                 tuneGrid = expand.grid(C=c(.01,.02,.05,.1,.2,.5,1,2,5,10)
                  ,degree=c(1:5),scale=c(0.01:1)),tuneLength = 4)

summary(model_svm)

##
## Call:
## svm(formula = Dropout ~ ., data = train.set, trControl =
trainControl("cv",
##     number = 10), tuneGrid = expand.grid(C = c(0.01, 0.02, 0.05,
##     0.1, 0.2, 0.5, 1, 2, 5, 10), degree = c(1:5), scale = c(0.01:1)),
##     tuneLength = 4)
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  3064
##
##  ( 1584 1480 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
model_svm$cost#displays cost , error, degree and scale of the model
```

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

```
model_svm$epsilon #displays the accuracy of the model crossvalidated
```

```
## [1] 0.1
```

Model Evaluation

```
x=validation.set[,-53]
y=validation.set[,53]
pred <- predict(model_svm,x)
confusionMatrix(pred,y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1436  114
##          1   83  820
##
##                Accuracy : 0.9197
##                  95% CI : (0.9082, 0.9301)
##     No Information Rate : 0.6192
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8286
##
##  Mcnemar's Test P-Value : 0.03256
##
##             Sensitivity : 0.9454
##             Specificity : 0.8779
##          Pos Pred Value : 0.9265
##          Neg Pred Value : 0.9081
##              Prevalence : 0.6192
##          Detection Rate : 0.5854
```

```
##    Detection Prevalence : 0.6319
##       Balanced Accuracy : 0.9117
##
##        'Positive' Class : 0
##
```

Predict the test data

```
pred_test=predict(model_svm,transformed_test)
table(pred_test)

## pred_test
##   0   1
## 618 382

# Create a submission file
submission_svm=as.data.frame(pred_test)

head(submission_svm)

##   pred_test
## 1         0
## 2         1
## 3         0
## 4         0
## 5         1
## 6         1

#write.csv(pred_test,"D:/Hamed/KAGGLE
COMPETITION/FEATURES/my_new_submission/submission_svm.csv")
```

Decision Tree Model

```
# load libraries
library(rpart)
library(rattle)

## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(mlr)
library(FSelector)
library(rpart.plot)
```

First we have to make a classification task with our training set. This is where we can define which type of machine learning problem we're trying to solve and define the target variable

```
(dt_task <- makeClassifTask(data=train.set, target="Dropout"))

## Supervised task: train.set
## Type: classif
```

```
## Target: Dropout
## Observations: 9808
## Features:
##    numerics     factors     ordered functionals
##          52           0           0           0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
## Has coordinates: FALSE
## Classes: 2
##    0    1
## 6008 3800
## Positive class: 0
```

After creating a classification task we need to make a learner that will later take our task to learn the data. I have chosen the rpart decision tree algorithm. This is the Recursive Partitioning Decision Tree.

```
(dt_prob <- makeLearner('classif.rpart', predict.type="prob"))

## Learner classif.rpart from package rpart
## Type: classif
## Name: Decision Tree; Short name: rpart
## Class: classif.rpart
## Properties:
twoclass,multiclass,missings,numerics,factors,ordered,prob,weights,featimp
## Predict-Type: prob
## Hyperparameters: xval=0
```

Hyper Parameter Tuning

```
getParamSet("classif.rpart")

##                     Type len  Def   Constr Req Tunable Trafo
## minsplit         integer   -   20 1 to Inf   -    TRUE     -
## minbucket        integer   -    - 1 to Inf   -    TRUE     -
## cp               numeric   - 0.01   0 to 1   -    TRUE     -
## maxcompete       integer   -    4 0 to Inf   -    TRUE     -
## maxsurrogate     integer   -    5 0 to Inf   -    TRUE     -
## usesurrogate    discrete   -    2    0,1,2   -    TRUE     -
## surrogatestyle  discrete   -    0      0,1   -    TRUE     -
## maxdepth         integer   -   30  1 to 30   -    TRUE     -
## xval             integer   -   10 0 to Inf   -   FALSE     -
## parms            untyped   -    -        -   -    TRUE     -

dt_param <- makeParamSet( makeDiscreteParam("minsplit", values=seq(5,10,1)),
            makeDiscreteParam("minbucket",values=seq(round(5/3,0),
round(10/3,0), 1)),
            makeNumericParam("cp",lower = 0.01, upper = 0.05),
            makeDiscreteParam("maxcompete",
```

```
            values=6), makeDiscreteParam("usesurrogate", values=0),
            makeDiscreteParam("maxdepth", values=10) )
```

Optimization Algorithm

```
ctrl = makeTuneControlGrid()

# Evaluating Tuning with Resampling
rdesc = makeResampleDesc("CV", iters = 3L, stratify=TRUE)
```

We can now use tuneParams to show us what combination of hyperparameter values as specified by us will give us the optimal result.

```
set.seed(1000)
(dt_tuneparam <-
tuneParams(learner=dt_prob,resampling=rdesc,measures=list(tpr,auc,
            fnr, mmce, tnr, setAggregation(tpr, test.sd)),
            par.set=dt_param,control=ctrl,task=dt_task,show.info = TRUE) )

## [Tune] Started tuning learner classif.rpart for parameter set:

##                  Type len Def       Constr Req Tunable Trafo
## minsplit     discrete   -   - 5,6,7,8,9,10   -    TRUE     -
## minbucket    discrete   -   -          2,3   -    TRUE     -
## cp            numeric   -   - 0.01 to 0.05   -    TRUE     -
## maxcompete   discrete   -   -            6   -    TRUE     -
## usesurrogate discrete   -   -            0   -    TRUE     -
## maxdepth     discrete   -   -           10   -    TRUE     -

## With control class: TuneControlGrid

## Imputation value: -0Imputation value: -0Imputation value: 1Imputation
value: 1Imputation value: -0Imputation value: Inf

## [Tune-x] 1: minsplit=5; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 1:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 2: minsplit=6; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 2:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 3: minsplit=7; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 3:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 4: minsplit=8; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 4:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 5: minsplit=9; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 5:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 6: minsplit=10; minbucket=2; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 6:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 7: minsplit=5; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 7:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 8: minsplit=6; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 8:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 9: minsplit=7; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 9:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min
```

```
## [Tune-x] 10: minsplit=8; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 10:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 11: minsplit=9; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 11:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 12: minsplit=10; minbucket=3; cp=0.01; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 12:
tpr.test.mean=0.9097838,auc.test.mean=0.9467121,fnr.test.mean=0.0902162,mmce.
test.mean=0.0973686,tnr.test.mean=0.8913217,tpr.test.sd=0.0192740; time: 0.0
min

## [Tune-x] 13: minsplit=5; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 13:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 14: minsplit=6; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 14:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 15: minsplit=7; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 15:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 16: minsplit=8; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 16:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
```

```
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 17: minsplit=9; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 17:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 18: minsplit=10; minbucket=2; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 18:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 19: minsplit=5; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 19:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 20: minsplit=6; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 20:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 21: minsplit=7; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 21:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 22: minsplit=8; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 22:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 23: minsplit=9; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 23:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 24: minsplit=10; minbucket=3; cp=0.0144; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 24:
tpr.test.mean=0.8899809,auc.test.mean=0.9333891,fnr.test.mean=0.1100191,mmce.
test.mean=0.1131719,tnr.test.mean=0.8818507,tpr.test.sd=0.0192406; time: 0.0
min

## [Tune-x] 25: minsplit=5; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 25:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 26: minsplit=6; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 26:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 27: minsplit=7; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 27:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 28: minsplit=8; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 28:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 29: minsplit=9; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 29:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min
```

```
## [Tune-x] 30: minsplit=10; minbucket=2; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 30:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 31: minsplit=5; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 31:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 32: minsplit=6; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 32:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 33: minsplit=7; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 33:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 34: minsplit=8; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 34:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 35: minsplit=9; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 35:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 36: minsplit=10; minbucket=3; cp=0.0189; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 36:
tpr.test.mean=0.8826549,auc.test.mean=0.9281592,fnr.test.mean=0.1173451,mmce.
```

```
test.mean=0.1154152,tnr.test.mean=0.8876387,tpr.test.sd=0.0204596; time: 0.0
min

## [Tune-x] 37: minsplit=5; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 37:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 38: minsplit=6; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 38:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 39: minsplit=7; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 39:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 40: minsplit=8; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 40:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 41: minsplit=9; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 41:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 42: minsplit=10; minbucket=2; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 42:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 43: minsplit=5; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 43:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 44: minsplit=6; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 44:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 45: minsplit=7; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 45:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 46: minsplit=8; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 46:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 47: minsplit=9; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 47:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 48: minsplit=10; minbucket=3; cp=0.0233; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 48:
tpr.test.mean=0.8966409,auc.test.mean=0.9241868,fnr.test.mean=0.1033591,mmce.
test.mean=0.1230628,tnr.test.mean=0.8458076,tpr.test.sd=0.0246099; time: 0.0
min

## [Tune-x] 49: minsplit=5; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 49:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min
```

```
## [Tune-x] 50: minsplit=6; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 50:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 51: minsplit=7; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 51:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 52: minsplit=8; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 52:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 53: minsplit=9; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 53:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 54: minsplit=10; minbucket=2; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 54:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 55: minsplit=5; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 55:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 56: minsplit=6; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 56:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
```

```
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 57: minsplit=7; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 57:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 58: minsplit=8; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 58:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 59: minsplit=9; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 59:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 60: minsplit=10; minbucket=3; cp=0.0278; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 60:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 61: minsplit=5; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 61:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 62: minsplit=6; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 62:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 63: minsplit=7; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 63:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 64: minsplit=8; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 64:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 65: minsplit=9; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 65:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 66: minsplit=10; minbucket=2; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 66:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 67: minsplit=5; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 67:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 68: minsplit=6; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 68:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 69: minsplit=7; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 69:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min
```

```
## [Tune-x] 70: minsplit=8; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 70:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 71: minsplit=9; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 71:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 72: minsplit=10; minbucket=3; cp=0.0322; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 72:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 73: minsplit=5; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 73:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 74: minsplit=6; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 74:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 75: minsplit=7; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 75:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 76: minsplit=8; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 76:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
```

```
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 77: minsplit=9; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 77:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 78: minsplit=10; minbucket=2; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 78:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 79: minsplit=5; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 79:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 80: minsplit=6; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 80:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 81: minsplit=7; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 81:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 82: minsplit=8; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 82:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 83: minsplit=9; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 83:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 84: minsplit=10; minbucket=3; cp=0.0367; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 84:
tpr.test.mean=0.9129498,auc.test.mean=0.9205301,fnr.test.mean=0.0870502,mmce.
test.mean=0.1274474,tnr.test.mean=0.8086828,tpr.test.sd=0.0060318; time: 0.0
min

## [Tune-x] 85: minsplit=5; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 85:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 86: minsplit=6; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 86:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 87: minsplit=7; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 87:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 88: minsplit=8; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 88:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 89: minsplit=9; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 89:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min
```

```
## [Tune-x] 90: minsplit=10; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 90:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 91: minsplit=5; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 91:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 92: minsplit=6; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 92:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 93: minsplit=7; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 93:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 94: minsplit=8; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 94:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 95: minsplit=9; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 95:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 96: minsplit=10; minbucket=3; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 96:
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
```

```
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814; time: 0.0
min

## [Tune-x] 97: minsplit=5; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 97:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 98: minsplit=6; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 98:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 99: minsplit=7; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 99:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 100: minsplit=8; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 100:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 101: minsplit=9; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 101:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 102: minsplit=10; minbucket=2; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 102:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 103: minsplit=5; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10
```

```
## [Tune-y] 103:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 104: minsplit=6; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 104:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 105: minsplit=7; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 105:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 106: minsplit=8; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 106:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 107: minsplit=9; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 107:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 108: minsplit=10; minbucket=3; cp=0.0456; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 108:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 109: minsplit=5; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 109:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min
```

```
## [Tune-x] 110: minsplit=6; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 110:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 111: minsplit=7; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 111:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 112: minsplit=8; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 112:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 113: minsplit=9; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 113:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 114: minsplit=10; minbucket=2; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 114:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 115: minsplit=5; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 115:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 116: minsplit=6; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 116:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
```

```
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 117: minsplit=7; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 117:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 118: minsplit=8; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 118:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 119: minsplit=9; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 119:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune-x] 120: minsplit=10; minbucket=3; cp=0.05; maxcompete=6;
usesurrogate=0; maxdepth=10

## [Tune-y] 120:
tpr.test.mean=0.8903170,auc.test.mean=0.9091048,fnr.test.mean=0.1096830,mmce.
test.mean=0.1400902,tnr.test.mean=0.8118519,tpr.test.sd=0.0382770; time: 0.0
min

## [Tune] Result: minsplit=9; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10 :
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814

## Tune result:
## Op. pars: minsplit=9; minbucket=2; cp=0.0411; maxcompete=6;
usesurrogate=0; maxdepth=10
##
tpr.test.mean=0.9141147,auc.test.mean=0.9185711,fnr.test.mean=0.0858853,mmce.
test.mean=0.1314229,tnr.test.mean=0.7965807,tpr.test.sd=0.0040814
```

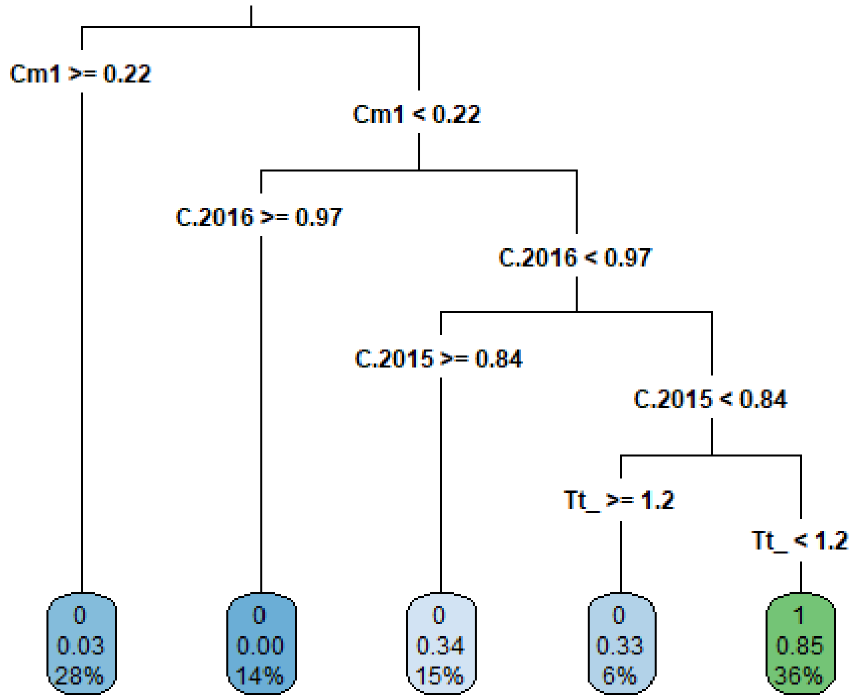
Optimal HyperParameters

```r
list( 'Optimal HyperParameters' = dt_tuneparam$x,
      'Optimal Metrics' = dt_tuneparam$y )
```

```
## $`Optimal HyperParameters`
## $`Optimal HyperParameters`$minsplit
## [1] 9
##
## $`Optimal HyperParameters`$minbucket
## [1] 2
##
## $`Optimal HyperParameters`$cp
## [1] 0.04111111
##
## $`Optimal HyperParameters`$maxcompete
## [1] 6
##
## $`Optimal HyperParameters`$usesurrogate
## [1] 0
##
## $`Optimal HyperParameters`$maxdepth
## [1] 10
##
##
## $`Optimal Metrics`
##  tpr.test.mean  auc.test.mean  fnr.test.mean mmce.test.mean  tnr.test.mean
##    0.914114675    0.918571143    0.085885325    0.131422915    0.796580720
##    tpr.test.sd
##    0.004081437

dtree <- setHyperPars(dt_prob, par.vals = dt_tuneparam$x)
```

Model Training

```
set.seed(1000)
dtree_train <- train(learner=dtree, task=dt_task)
getLearnerModel(dtree_train)

## n= 9808
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 9808 3800 0 (0.61256117 0.38743883)
##    2) Complete1>=0.21875 2789   78 0 (0.97203299 0.02796701) *
##    3) Complete1< 0.21875 7019 3297 1 (0.46972503 0.53027497)
##      6) Cohort.2016.17>=0.9660425 1420    0 0 (1.00000000 0.00000000) *
##      7) Cohort.2016.17< 0.9660425 5599 1877 1 (0.33523844 0.66476156)
##       14) Cohort.2015.16>=0.8412062 1487  508 0 (0.65837256 0.34162744) *
##       15) Cohort.2015.16< 0.8412062 4112  898 1 (0.21838521 0.78161479)
##         30) Total_grant>=1.22859 567  185 0 (0.67372134 0.32627866) *
##         31) Total_grant< 1.22859 3545  516 1 (0.14555712 0.85444288) *

rpart.plot(dtree_train$learner.model, roundint=FALSE, varlen=3, type = 3,
           clip.right.labs = FALSE, yesno = 2)
```




```
rpart.rules(dtree_train$learner.model, roundint = FALSE)

##  Dropout
##     0.00 when Complete1 <  0.22 & Cohort.2016.17 >= 0.97
##     0.03 when Complete1 >= 0.22
##     0.33 when Complete1 <  0.22 & Cohort.2016.17 <  0.97 & Cohort.2015.16
<  0.84 & Total_grant >= 1.2
##     0.34 when Complete1 <  0.22 & Cohort.2016.17 <  0.97 & Cohort.2015.16
>= 0.84
##     0.85 when Complete1 <  0.22 & Cohort.2016.17 <  0.97 & Cohort.2015.16
<  0.84 & Total_grant <  1.2
```

Model Prediction (Testing): We now pass the trained learner to be used to make predictions with our test data.

```
set.seed(1000)
(dtree_predict <- predict(dtree_train, newdata = validation.set))

## Prediction: 2453 observations
## predict.type: prob
## threshold: 0=0.50,1=0.50
## time: 0.00
##    truth    prob.0     prob.1 response
## 4      0 0.9720330 0.02796701        0
## 8      1 0.1455571 0.85444288        1
## 19     0 0.9720330 0.02796701        0
## 22     0 0.9720330 0.02796701        0
```

```
## 23     1 0.1455571 0.85444288        1
## 28     0 0.9720330 0.02796701        0
## ... (#rows: 2453, #cols: 4)

# The threshold for classifying each row is 50/50. This is by default
# but can be changed later (which I will do).
dtree_predict %>% calculateROCMeasures()

##     predicted
## true 0         1
##    0 1406      113       tpr: 0.93 fnr: 0.07
##    1 189       745       fpr: 0.2  tnr: 0.8
##      ppv: 0.88 for: 0.13 lrp: 4.57 acc: 0.88
##      fdr: 0.12 npv: 0.87 lrm: 0.09 dor: 49.05
##
##
## Abbreviations:
## tpr - True positive rate (Sensitivity, Recall)
## fpr - False positive rate (Fall-out)
## fnr - False negative rate (Miss rate)
## tnr - True negative rate (Specificity)
## ppv - Positive predictive value (Precision)
## for - False omission rate
## lrp - Positive likelihood ratio (LR+)
## fdr - False discovery rate
## npv - Negative predictive value
## acc - Accuracy
## lrm - Negative likelihood ratio (LR-)
## dor - Diagnostic odds ratio

dtree_predict.test <- predict(dtree_train, newdata = transformed_test)

sub_tree=as.data.frame(dtree_predict.test)
head(sub_tree)

##      prob.0     prob.1 response
## 1 0.6583726 0.34162744        0
## 2 0.1455571 0.85444288        1
## 3 0.6737213 0.32627866        0
## 4 0.6583726 0.34162744        0
## 5 0.9720330 0.02796701        0
## 6 0.1455571 0.85444288        1

#write.csv(sub_tree,"D:/Hamed/KAGGLE
COMPETITION/FEATURES/my_new_submission/sub_tree.csv")
```

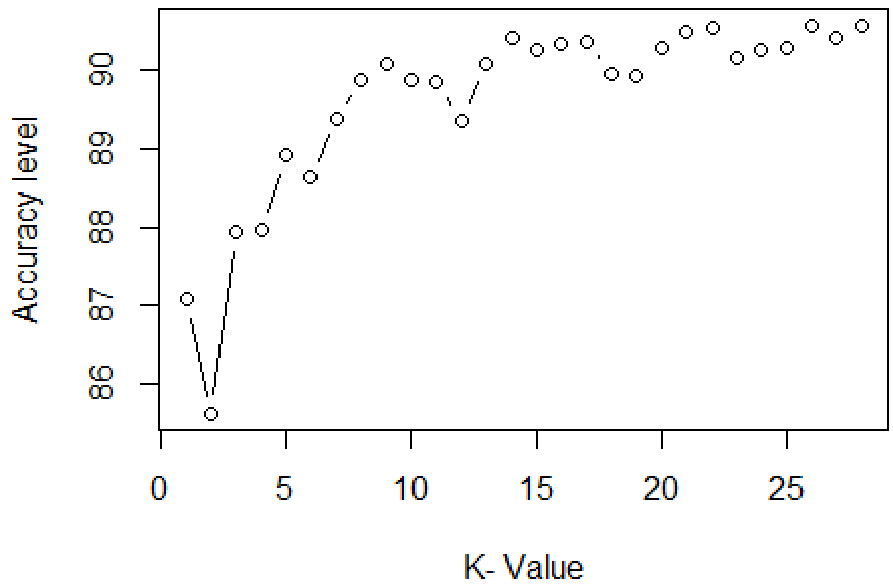
K-Nearest Neighbors (KNN)

```
library(class)
library(caret)
library(rpart)
library(dplyr)
```

Create the train and validation labels

```
train.dropout_labels <- train.set$Dropout
val.dropout_labels <-validation.set$Dropout



# Lets get a good k value

i=1
k.optm=1
for (i in 1:28){
  knn.mod <- knn(train=train.set, test=validation.set,
cl=train.dropout_labels, k=i)
  k.optm[i] <- 100 * sum(val.dropout_labels ==
knn.mod)/NROW(val.dropout_labels)
  k=i
  cat(k,'=',k.optm[i],'
')
}

## 1 = 87.07705
## 2 = 85.60946
## 3 = 87.93314
## 4 = 87.97391
## 5 = 88.91154
## 6 = 88.62617
## 7 = 89.40073
## 8 = 89.88993
## 9 = 90.09376
## 10 = 89.88993
## 11 = 89.84916
## 12 = 89.35997
## 13 = 90.09376
## 14 = 90.41989
## 15 = 90.25683
## 16 = 90.33836
## 17 = 90.37913
## 18 = 89.97146
## 19 = 89.9307
## 20 = 90.29759
## 21 = 90.50143
## 22 = 90.54219
## 23 = 90.1753
## 24 = 90.25683
## 25 = 90.29759
## 26 = 90.58296
## 27 = 90.41989
## 28 = 90.58296
```

```r
# Plot the K to check Accuracy plot for best k values
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



Use the best K value to fit a model to the data

```r
library(kknn)

##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy

knn.fit <- train.kknn(as.factor(Dropout)~., train.set, ks = 22,
                      kernel = "rectangular", scale = TRUE)
pred.train.kknn <- predict(knn.fit, validation.set)

# Lets look at the performance metrics
confusionMatrix(table(pred.train.kknn ,val.dropout_labels))

## Confusion Matrix and Statistics
##
##                val.dropout_labels
## pred.train.kknn    0    1
##               0 1384  205
##               1  135  729
##
```

```
##                Accuracy : 0.8614
##                  95% CI : (0.8471, 0.8748)
##     No Information Rate : 0.6192
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7018
##
##  Mcnemar's Test P-Value : 0.0001825
##
##             Sensitivity : 0.9111
##             Specificity : 0.7805
##          Pos Pred Value : 0.8710
##          Neg Pred Value : 0.8438
##              Prevalence : 0.6192
##          Detection Rate : 0.5642
##    Detection Prevalence : 0.6478
##       Balanced Accuracy : 0.8458
##
##        'Positive' Class : 0
##
```

Prediction using a new data

```
library(caret)

pred_knn<-predict(object=knn.fit,transformed_test,type="raw")
pred_knn=as.data.frame(pred_knn)


submission_knn=as.data.frame(pred_knn)
head(submission_knn)
```

```
##   pred_knn
## 1        0
## 2        1
## 3        0
## 4        0
## 5        0
## 6        1
```

```
# save the file
#write.csv(submission_knn,"D:/Hamed/KAGGLE
COMPETITION/FEATURES/my_new_submission/submission_knn.csv")
```