Load packages

```
library(h2o)
## Warning: package 'h2o' was built under R version 3.5.1
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai

## Attaching package: 'h2o'
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc

h2o.init(startH2O = TRUE, max_mem_size="16G", nthreads = 1)
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 hours 4 minutes
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.20.0.2
##     H2O cluster version age:    1 month and 21 days
##     H2O cluster name:           H2O_started_from_R_daveh_teh407
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   15.67 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  1
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.0 (2018-04-23)

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

library(ggplot2)

library(mlbench)
## Warning: package 'mlbench' was built under R version 3.5.1

library(caret)
## Warning: package 'caret' was built under R version 3.5.1
## Loading required package: lattice
```

Importing and preparing data

```
c.card <- h2o.importFile(path ="E:/creditcard.csv")

dim(c.card)

## [1] 284807     31
head(c.card)
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##           V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##            V19         V20          V21          V22         V23
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767
##            V24        V25        V26          V27         V28 Amount Class
## 1  0.06692807  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2 -0.33984648  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.68928096 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4 -1.17557533  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5  0.14126698 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.37142658 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0

summary(c.card,exact_quantiles=TRUE)
##  Time             V1                    V2
##  Min.   :     0   Min.   :-5.641e+01   Min.   :-7.272e+01
##  1st Qu.: 54202   1st Qu.:-9.204e-01   1st Qu.:-5.985e-01
##  Median : 84692   Median : 1.811e-02   Median : 6.549e-02
##  Mean   : 94814   Mean   : 1.073e-15   Mean   : 4.151e-16
##  3rd Qu.:139321   3rd Qu.: 1.316e+00   3rd Qu.: 8.037e-01
##  Max.   :172792   Max.   : 2.455e+00   Max.   : 2.206e+01
##  V3                   V4                   V5
##  Min.   :-4.833e+01   Min.   :-5.683e+00   Min.   :-1.137e+02
##  1st Qu.:-8.904e-01   1st Qu.:-8.486e-01   1st Qu.:-6.916e-01
##  Median : 1.798e-01   Median :-1.985e-02   Median :-5.434e-02
##  Mean   :-1.022e-15   Mean   : 2.031e-15   Mean   : 1.022e-15
##  3rd Qu.: 1.027e+00   3rd Qu.: 7.433e-01   3rd Qu.: 6.119e-01
##  Max.   : 9.383e+00   Max.   : 1.688e+01   Max.   : 3.480e+01
##  V6                   V7                   V8
##  Min.   :-2.616e+01   Min.   :-4.356e+01   Min.   :-7.322e+01
##  1st Qu.:-7.683e-01   1st Qu.:-5.541e-01   1st Qu.:-2.086e-01
```

```
##  Median :-2.742e-01   Median : 4.010e-02   Median : 2.236e-02
##  Mean   : 1.418e-15   Mean   :-6.004e-16   Mean   : 1.405e-16
##  3rd Qu.: 3.986e-01   3rd Qu.: 5.704e-01   3rd Qu.: 3.273e-01
##  Max.   : 7.330e+01   Max.   : 1.206e+02   Max.   : 2.001e+01
##  V9                   V10                  V11
##  Min.   :-1.343e+01   Min.   :-2.459e+01   Min.   :-4.797e+00
##  1st Qu.:-6.431e-01   1st Qu.:-5.354e-01   1st Qu.:-7.625e-01
##  Median :-5.143e-02   Median :-9.292e-02   Median :-3.276e-02
##  Mean   :-2.516e-15   Mean   : 2.208e-15   Mean   : 1.788e-15
##  3rd Qu.: 5.971e-01   3rd Qu.: 4.539e-01   3rd Qu.: 7.396e-01
##  Max.   : 1.559e+01   Max.   : 2.375e+01   Max.   : 1.202e+01
##  V12                  V13                  V14
##  Min.   :-1.868e+01   Min.   :-5.792e+00   Min.   :-1.921e+01
##  1st Qu.:-4.056e-01   1st Qu.:-6.485e-01   1st Qu.:-4.256e-01
##  Median : 1.400e-01   Median :-1.357e-02   Median : 5.060e-02
##  Mean   :-1.169e-15   Mean   : 7.983e-16   Mean   : 1.099e-15
##  3rd Qu.: 6.182e-01   3rd Qu.: 6.625e-01   3rd Qu.: 4.931e-01
##  Max.   : 7.848e+00   Max.   : 7.127e+00   Max.   : 1.053e+01
##  V15                  V16                  V17
##  Min.   :-4.499e+00   Min.   :-1.413e+01   Min.   :-2.516e+01
##  1st Qu.:-5.829e-01   1st Qu.:-4.680e-01   1st Qu.:-4.837e-01
##  Median : 4.807e-02   Median : 6.641e-02   Median :-6.568e-02
##  Mean   : 4.982e-15   Mean   : 1.421e-15   Mean   :-3.513e-16
##  3rd Qu.: 6.488e-01   3rd Qu.: 5.233e-01   3rd Qu.: 3.997e-01
##  Max.   : 8.878e+00   Max.   : 1.732e+01   Max.   : 9.254e+00
##  V18                  V19                  V20
##  Min.   :-9.499e+00   Min.   :-7.214e+00   Min.   :-5.450e+01
##  1st Qu.:-4.988e-01   1st Qu.:-4.563e-01   1st Qu.:-2.117e-01
##  Median :-3.636e-03   Median : 3.735e-03   Median :-6.248e-02
##  Mean   : 1.015e-15   Mean   : 1.023e-15   Mean   : 6.259e-16
##  3rd Qu.: 5.008e-01   3rd Qu.: 4.589e-01   3rd Qu.: 1.330e-01
##  Max.   : 5.041e+00   Max.   : 5.592e+00   Max.   : 3.942e+01
##  V21                  V22                  V23
##  Min.   :-3.483e+01   Min.   :-1.093e+01   Min.   :-4.481e+01
##  1st Qu.:-2.284e-01   1st Qu.:-5.424e-01   1st Qu.:-1.618e-01
##  Median :-2.945e-02   Median : 6.782e-03   Median :-1.119e-02
##  Mean   : 1.661e-16   Mean   :-3.577e-16   Mean   : 2.746e-16
##  3rd Qu.: 1.864e-01   3rd Qu.: 5.286e-01   3rd Qu.: 1.476e-01
##  Max.   : 2.720e+01   Max.   : 1.050e+01   Max.   : 2.253e+01
##  V24                  V25                  V26
##  Min.   :-2.837e+00   Min.   :-1.030e+01   Min.   :-2.605e+00
##  1st Qu.:-3.546e-01   1st Qu.:-3.171e-01   1st Qu.:-3.270e-01
##  Median : 4.098e-02   Median : 1.659e-02   Median :-5.214e-02
##  Mean   : 4.472e-15   Mean   : 4.982e-16   Mean   : 1.700e-15
##  3rd Qu.: 4.395e-01   3rd Qu.: 3.507e-01   3rd Qu.: 2.410e-01
##  Max.   : 4.585e+00   Max.   : 7.520e+00   Max.   : 3.517e+00
##  V27                  V28                  Amount
##  Min.   :-2.257e+01   Min.   :-1.543e+01   Min.   :    0.00
##  1st Qu.:-7.084e-02   1st Qu.:-5.296e-02   1st Qu.:    5.60
##  Median : 1.342e-03   Median : 1.124e-02   Median :   22.00
##  Mean   :-3.688e-16   Mean   :-1.211e-16   Mean   :   88.35
##  3rd Qu.: 9.105e-02   3rd Qu.: 7.828e-02   3rd Qu.:   77.17
##  Max.   : 3.161e+01   Max.   : 3.385e+01   Max.   :25691.16
##  Class
##  Min.   :0.000000
##  1st Qu.:0.000000
##  Median :0.000000
##  Mean   :0.001727
##  3rd Qu.:0.000000
##  Max.   :1.000000
c.card[is.na(c.card)] <- 0
```

Splitting the dataframe for machine learning setup - Split the dataframe

```
# the ratios should sum up to to be less than 1.0.
#60% for training, 20% for validation, 20% for testing
# the ratios should sum up to to be less than 1.0.
cc.splits <- h2o.splitFrame(data = c.card,
                         ratios = c(0.6,0.2),
                         destination_frames = c("train", "valid", "test"),
                         seed = 1234)

train <- cc.splits[[1]]
valid <- cc.splits[[2]]
test  <- cc.splits[[3]]
valid

##   Time        V1          V2          V3         V4         V5          V6
## 1    0 -1.359807 -0.07278117  2.53634674  1.3781552 -0.3383208  0.46238778
## 2    1 -1.358354 -1.34016307  1.77320934  0.3797796 -0.5031981  1.80049938
## 3   12  1.103215 -0.04029621  1.26733209  1.2890915 -0.7359972  0.28806916
## 4   18  1.166616  0.50212009 -0.06730031  2.2615692  0.4288042  0.08947352
## 5   22 -1.946525 -0.04490051 -0.40557007 -1.0130573  2.9419677  2.95505340
## 6   25  1.114009  0.08554609  0.49370249  1.3357600 -0.3001886 -0.01075378
##          V7        V8         V9         V10         V11         V12
## 1  0.23959855 0.0986979  0.3637870  0.09079417 -0.55159953 -0.61780086
## 2  0.79146096 0.2476758 -1.5146543  0.20764287  0.62450146  0.06608369
## 3 -0.58605679 0.1893797  0.7823329 -0.26797507 -0.45031128  0.93670771
## 4  0.24114658 0.1380817 -0.9891624  0.92217497  0.74478579 -0.53137725
## 5 -0.06306315 0.8555463  0.0499669  0.57374251 -0.08125651 -0.21574500
## 6 -0.11876002 0.1886167  0.2056868  0.08226226  1.13355567  0.62669900
##          V13         V14          V15        V16          V17         V18
## 1 -0.99138985 -0.31116935  1.468176972 -0.4704005  0.207971242  0.02579058
## 2  0.71729273 -0.16594592  2.345864949 -2.8900832  1.109969379 -0.12135931
## 3  0.70838041 -0.46864729  0.354574063 -0.2466347 -0.009212378 -0.59591241
## 4 -2.10534645  1.12687010  0.003075323  0.4244245 -0.454475292 -0.09887063
## 5  0.04416063  0.03389776  1.190717675  0.5788435 -0.975667025  0.04406282
## 6 -1.49278039  0.52078789 -0.674592597 -0.5291082  0.158256198 -0.39875148
##         V19        V20         V21          V22         V23         V24
## 1  0.4039930  0.2514121 -0.01830678  0.277837576 -0.11047391  0.06692807
## 2 -2.2618571  0.5249797  0.24799815  0.771679402  0.90941226 -0.68928096
## 3 -0.5756816 -0.1139102 -0.02461201  0.196001953  0.01380165  0.10375833
## 4 -0.8165973 -0.3071685  0.01870187 -0.061972267 -0.10385492 -0.37041518
## 5  0.4886029 -0.2167153 -0.57952593 -0.799228953  0.87030022  0.98342149
## 6 -0.1457089 -0.2738324 -0.05323366 -0.004760151 -0.03147017  0.19805372
##         V25        V26         V27          V28 Amount Class
## 1  0.1285394 -0.1891148  0.13355838 -0.021053053 149.62     0
## 2 -0.3276418 -0.1390966 -0.05535279 -0.059751841 378.66     0
## 3  0.3642975 -0.3822606  0.09280919  0.037050517  12.99     0
## 4  0.6032003  0.1085559 -0.04052071 -0.011417815   2.28     0
## 5  0.3212011  0.1496499  0.70751884  0.014599752   0.89     0
## 6  0.5650073 -0.3377181  0.02905740  0.004452631   4.45     0
##
## [56904 rows x 31 columns]
# Identify predictors and response
y <- "Class"
x <- setdiff(names(train), y)

# For binary classification, response should be a factor
train[,y] <- as.factor(train[,y])
test[,y] <- as.factor(test[,y])
valid[,y] <- as.factor(valid[,y])
```

```
# Number of CV folds (to generate level-one data for stacking)
nfolds <- 5
```

Baseline performance, default GBM, trained on the 60% training split

```
#Required parameters, and defaults
gbm <- h2o.gbm(x = x, y = y, training_frame = train)

summary(gbm)
## Model Details:
## H2OBinomialModel: gbm
## Model Key:  GBM_model_R_1533532011214_1014
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1              50                       50               10454         5
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1         5    5.00000          6         20    11.70000

## H2OBinomialMetrics: gbm
## ** Reported on training data. **
## MSE:  0.0003522879
## RMSE:  0.01876933
## LogLoss:  0.005832577
## Mean Per-Class Error:  0.08218382
## AUC:  0.9423526

## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal
threshold:
##             0   1    Error         Rate
## 0      170887  14 0.000082  =14/170901
## 1          46 234 0.164286     =46/280
## Totals 170933 248 0.000351  =60/171181
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.262952 0.886364  13
## 2                       max f2  0.262952 0.855263  13
## 3                 max f0point5  0.838048 0.921474   7
## 4                 max accuracy  0.262952 0.999649  13
## 5                max precision  1.000000 0.957265   0
## 6                   max recall  0.000000 1.000000 261
## 7              max specificity  1.000000 0.999941   0
## 8             max absolute_mcc  0.262952 0.887826  13
## 9   max min_per_class_accuracy  0.000213 0.909111 125
## 10 max mean_per_class_accuracy  0.000244 0.937572 121

## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
## Scoring History:
##              timestamp   duration number_of_trees training_rmse
## 1  2018-08-06 04:11:45  0.016 sec               0       0.04041
## 2  2018-08-06 04:11:45  0.356 sec               1       0.02357
## 3  2018-08-06 04:11:46  0.691 sec               2       0.02245
## 4  2018-08-06 04:11:46  1.061 sec               3       0.02252
## 5  2018-08-06 04:11:47  1.722 sec               4       0.02475
## 6  2018-08-06 04:11:47  2.171 sec               5       0.02548
## 7  2018-08-06 04:11:48  2.583 sec               6       0.02661
## 8  2018-08-06 04:11:48  2.997 sec               7       0.02633
## 9  2018-08-06 04:11:48  3.405 sec               8       0.02622
## 10 2018-08-06 04:11:49  3.847 sec               9       0.02612
## 11 2018-08-06 04:11:53  8.050 sec              24       0.02409
## 12 2018-08-06 04:11:57 12.281 sec              40       0.01886
## 13 2018-08-06 04:12:00 14.987 sec              50       0.01877
```

```
##    training_logloss training_auc training_lift
## 1           0.01213      0.50000       1.00000
## 2           0.00798      0.93036       1.00122
## 3           0.00931      0.85826      21.13551
## 4           0.01042      0.92895      22.57814
## 5           0.01414      0.91105      22.08740
## 6           0.01546      0.88605      21.46903
## 7           0.01590      0.88960      21.46735
## 8           0.01577      0.88960      21.46735
## 9           0.01568      0.88960      21.46735
## 10          0.01553      0.88960      21.46735
## 11          0.01373      0.80052      19.41860
## 12          0.00581      0.94211      21.18285
## 13          0.00583      0.94235      21.11882

##    training_classification_error
## 1                        0.99836
## 2                        0.00041
## 3                        0.00047
## 4                        0.00048
## 5                        0.00060
## 6                        0.00064
## 7                        0.00065
## 8                        0.00063
## 9                        0.00063
## 10                       0.00063
## 11                       0.00058
## 12                       0.00036
## 13                       0.00035

## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      V17          212.026199          1.000000   0.200459
## 2      V10          116.907967          0.551385   0.110530
## 3      V11           96.059898          0.453057   0.090819
## 4      V14           94.677246          0.446536   0.089512
## 5      V12           83.939354          0.395891   0.079360
##

##    variable relative_importance scaled_importance percentage
## 25      V23            3.207329          0.015127   0.003032
## 26      V22            2.747638          0.012959   0.002598
## 27       V7            2.637098          0.012438   0.002493
## 28      V28            2.069043          0.009758   0.001956
## 29       V2            1.937746          0.009139   0.001832
## 30   Amount            1.557692          0.007347   0.001473
```

 # AUC, validation set

```
h2o.auc(h2o.performance(gbm, newdata = valid))
## [1] 0.835717
```

GBM/1. Trained on 80% of the data/2. Combine the training andalidationfoldsn splits to get more training data/3. Cross-validated using 5 fold

```
gbm <- h2o.gbm(x = x, y = y , training_frame = h2o.rbind(train, valid),
nfolds = nfolds, seed = 1234)

# Cross validation metrics summary and variance between the folds
gbm@model$cross_validation_metrics_summary
## Cross-Validation Metrics Summary:
##                                mean           sd  cv_1_valid  cv_2_valid
## accuracy                  0.9992152 6.3865744E-5      0.9993  0.99934185
## auc                       0.7839946  0.026804693   0.7975272  0.81427443
```

```
## err                        7.8480644E-4 6.3865744E-5 6.999891E-4 6.581401E-4
## err_count                          35.8     2.912044        32.0        30.0
## f0point5                      0.7920115  0.041773416   0.8199357  0.88235295
## f1                            0.7405418  0.036291275  0.76119405   0.8076923
## f2                           0.69576555   0.03410931   0.7103064   0.7446808
## lift_top_group                 10.07801     5.444641   19.111908   15.576619
## logloss                     0.019771403 0.0017265858  0.01783132 0.016363015
## max_per_class_error          0.33100817  0.033892713        0.32  0.29213482
## mcc                          0.74484164   0.03696787    0.766347   0.8155474
## mean_per_class_accuracy       0.8343817  0.016957954  0.83991235  0.85388863
## mean_per_class_error          0.1656183  0.016957954  0.16008765  0.14611138
## mse                         8.314242E-4 4.8960574E-5 7.957461E-4 7.118524E-4
## precision                    0.83086777  0.047829207  0.86440676   0.9402985
## r2                           0.50335467  0.059035446  0.51416916  0.63469833
## recall                        0.6689918  0.033892713        0.68   0.7078652
## rmse                        0.028808504 8.6436846E-4 0.028208973 0.026680563
## specificity                   0.9997716 6.0404764E-5   0.9998247   0.9999121
##                              cv_3_valid   cv_4_valid   cv_5_valid
## accuracy                     0.99912727   0.99918646   0.99912035
## auc                          0.76528853    0.8234984   0.7193847
## err                         8.727336E-4  8.135265E-4 8.7964284E-4
## err_count                          40.0         37.0        40.0
## f0point5                       0.751634    0.7969152   0.7092199
## f1                            0.6969697   0.77018636   0.6666667
## f2                            0.6497175    0.7451923   0.6289308
## lift_top_group                13.958975    0.9227803  0.81976825
## logloss                      0.02294165  0.019858725 0.021862304
## max_per_class_error           0.3783784   0.27058825   0.3939394
## mcc                            0.701728   0.77098995   0.6695959
## mean_per_class_accuracy       0.8106797   0.86455166   0.8028761
## mean_per_class_error         0.18932031   0.13544832  0.19712386
## mse                        8.9546497E-4 8.6865353E-4 8.8540383E-4
## precision                    0.79310346   0.81578946   0.7407407
## r2                           0.44448355    0.5343388  0.38908347
## recall                        0.6216216    0.7294118   0.6060606
## rmse                         0.02992432  0.029472928 0.029755736
## specificity                  0.99973774    0.9996916  0.99969167
```

 # AUC ross-validated AUC by combined holdout predictions

```
h2o.auc(h2o.performance(gbm, xval = TRUE))
## [1] 0.7906815
```

 # Cross-validated performance (0.7465849) is worse than the validation set performance (0.8851712)

GBM training/ 1. Use early stopping to automatically tune the number of trees using the validation AUC

```
gbm <- h2o.gbm(x = x, y = y ,
               training_frame = train,
               validation_frame = valid,
               learn_rate = 0.01,
               learn_rate_annealing =.99,
               ntrees=1000,
               stopping_rounds = 5,
               stopping_tolerance = 1e-8,
               stopping_metric = "AUC",
               sample_rate = 0.8,
               col_sample_rate = 0.8,
               score_tree_interval = 10,
               seed = 1234)
summary(gbm)
```

Model Details:

```
## ==============
```

```
## H2OBinomialModel: gbm
## Model Key:  GBM_model_R_1533532011214_1134
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             170                      170               50358         5
##   max_depth mean_depth min_leaves max_leaves mean_leaves

## H2OBinomialMetrics: gbm
## ** Reported on training data. **
## MSE:  0.0004431494
## RMSE:  0.02105111
## LogLoss:  0.003045868
## Mean Per-Class Error:  0.07325525
## AUC:  0.9768579
```

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:

```
##             0   1    Error        Rate
## 0      170887  14 0.000082  =14/170901
## 1          41 239 0.146429     =41/280
## Totals 170928 253 0.000321  =55/171181

## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.175371 0.896811 206
## 2                       max f2  0.122607 0.870827 211
## 3                 max f0point5  0.362046 0.930851 166
## 4                 max accuracy  0.178988 0.999679 204
## 5                max precision  0.939610 1.000000   0
## 6                   max recall  0.000819 1.000000 399
## 7              max specificity  0.939610 1.000000   0
## 8             max absolute_mcc  0.175371 0.897806 206
## 9   max min_per_class_accuracy  0.000987 0.939029 381
## 10 max mean_per_class_accuracy  0.001241 0.955781 365

## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
## H2OBinomialMetrics: gbm
## ** Reported on validation data. **
##
## MSE:  0.0007525779
## RMSE:  0.02743315
## LogLoss:  0.004628192
## Mean Per-Class Error:  0.1056278
## AUC:  0.9649468
## Gini:  0.9298935
```

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:

```
##            0   1    Error       Rate
## 0      56781  14 0.000247  =14/56795
## 1         23  86 0.211009    =23/109
## Totals 56804 100 0.000650  =37/56904
##

## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.089558 0.822967  97
## 2                       max f2  0.089558 0.802239  97
## 3                 max f0point5  0.150233 0.852391  90
## 4                 max accuracy  0.089558 0.999350  97
## 5                max precision  0.927094 1.000000   0
## 6                   max recall  0.000819 1.000000 397
## 7              max specificity  0.927094 1.000000   0
```

```
## 8            max absolute_mcc  0.089558 0.823408  97
## 9   max min_per_class_accuracy  0.000939 0.913584 354
## 10 max mean_per_class_accuracy  0.001011 0.929257 329
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

Scoring History:

```
##               timestamp   duration number_of_trees training_rmse
## 1  2018-08-06 04:13:55  0.015 sec               0       0.04041
## 2  2018-08-06 04:13:59  3.507 sec              10       0.02945
## 3  2018-08-06 04:14:01  5.728 sec              20       0.02799
## 4  2018-08-06 04:14:03  7.940 sec              30       0.02679
## 5  2018-08-06 04:14:05 10.177 sec              40       0.02583
## 6  2018-08-06 04:14:07 12.414 sec              50       0.02503
## 7  2018-08-06 04:14:10 14.645 sec              60       0.02437
## 8  2018-08-06 04:14:12 16.885 sec              70       0.02380
## 9  2018-08-06 04:14:14 19.144 sec              80       0.02331
## 10 2018-08-06 04:14:16 21.397 sec              90       0.02290
## 11 2018-08-06 04:14:19 23.624 sec             100       0.02254
## 12 2018-08-06 04:14:21 25.857 sec             110       0.02222
## 13 2018-08-06 04:14:23 28.310 sec             120       0.02196
## 14 2018-08-06 04:14:26 30.743 sec             130       0.02173
## 15 2018-08-06 04:14:28 33.381 sec             140       0.02152
## 16 2018-08-06 04:14:31 36.159 sec             150       0.02134
## 17 2018-08-06 04:14:35 39.615 sec             160       0.02119
## 18 2018-08-06 04:14:37 42.335 sec             170       0.02105

##    training_logloss training_auc training_lift
## 1           0.01213      0.50000       1.00000
## 2           0.00497      0.96958      14.63632
## 3           0.00455      0.97280      14.64404
## 4           0.00425      0.97510      51.72899
## 5           0.00403      0.97514      53.98051
## 6           0.00385      0.97506      87.72889
## 7           0.00371      0.97704      87.28872
## 8           0.00360      0.97687      91.41842
## 9           0.00349      0.97682      87.24026
## 10          0.00341      0.97689      88.61799
## 11          0.00333      0.97686      89.17123
## 12          0.00327      0.97688      89.01966
## 13          0.00322      0.97687      86.85445
## 14          0.00318      0.97692      91.72195
## 15          0.00314      0.97687      91.29559
## 16          0.00310      0.97687      88.41852
## 17          0.00307      0.97687      88.17043
## 18          0.00305      0.97686      88.71807

##    training_classification_error validation_rmse validation_logloss
## 1                        0.99836         0.04373            0.01392
## 2                        0.00046         0.03400            0.00646
## 3                        0.00045         0.03270            0.00604
## 4                        0.00041         0.03174            0.00575
## 5                        0.00039         0.03097            0.00554
## 6                        0.00037         0.03033            0.00537
## 7                        0.00035         0.02982            0.00524
## 8                        0.00034         0.02939            0.00512
## 9                        0.00034         0.02904            0.00503
## 10                       0.00033         0.02875            0.00496
## 11                       0.00033         0.02850            0.00489
## 12                       0.00033         0.02827            0.00484
## 13                       0.00033         0.02808            0.00479
## 14                       0.00033         0.02791            0.00475
## 15                       0.00033         0.02777            0.00471
## 16                       0.00033         0.02764            0.00468
## 17                       0.00032         0.02753            0.00465
```

```
## 18                       0.00032         0.02743            0.00463
##    validation_auc validation_lift validation_classification_error
## 1         0.50000         1.00000                         0.99808
## 2         0.95215        13.27851                         0.00070
## 3         0.96044        13.20778                         0.00069
## 4         0.96409        45.81197                         0.00065
## 5         0.96358        47.69780                         0.00065
## 6         0.96359        82.24155                         0.00065
## 7         0.96440        83.34563                         0.00065
## 8         0.96518        81.18916                         0.00065
## 9         0.96531        82.29003                         0.00065
## 10        0.96539        79.07348                         0.00065
## 11        0.96519        79.33189                         0.00065
## 12        0.96518        79.33189                         0.00065
## 13        0.96531        84.14405                         0.00065
## 14        0.96512        85.94251                         0.00065
## 15        0.96503        85.79226                         0.00065
## 16        0.96498        80.31616                         0.00065
## 17        0.96497        79.92374                         0.00065
## 18        0.96495        79.92374                         0.00065
##
## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
##
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      V17         3572.485107          1.000000   0.521447
## 2      V12         1099.678345          0.307819   0.160511
## 3      V10          649.227905          0.181730   0.094763
## 4      V14          549.718079          0.153876   0.080238
## 5       V4          125.715721          0.035190   0.018350
##
## ---
##    variable relative_importance scaled_importance percentage
## 25      V22           13.792594          0.003861   0.002013
## 26       V8           11.460245          0.003208   0.001673
## 27      V25           11.043874          0.003091   0.001612
## 28      V19            8.518763          0.002385   0.001243
## 29      V23            7.204707          0.002017   0.001052
## 30      V18            4.966079          0.001390   0.000725
```

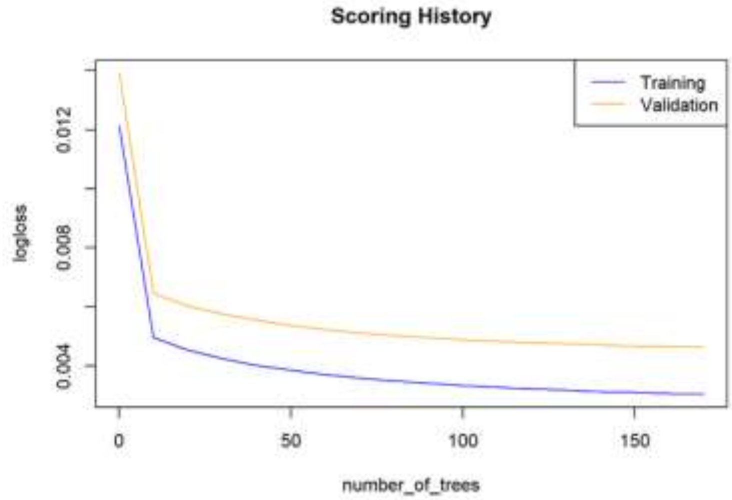
#AUC on the validation set

```
h2o.auc(h2o.performance(gbm, valid = TRUE))
## [1] 0.9649468
```

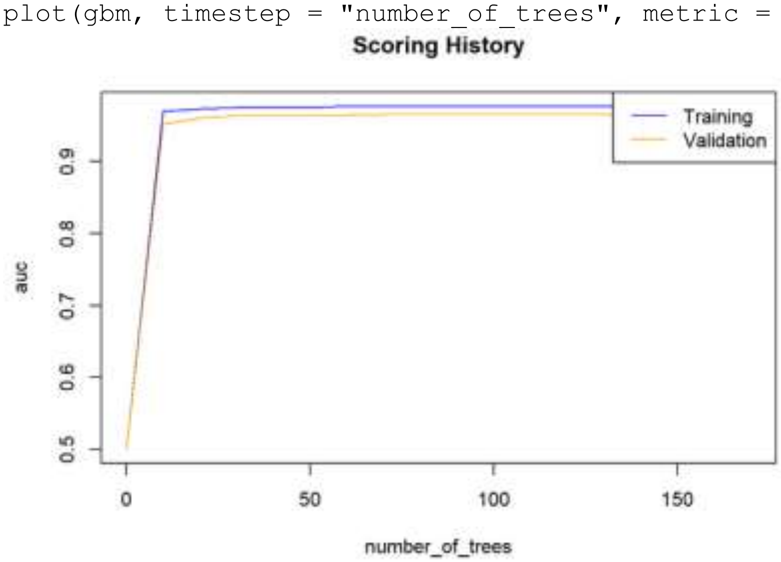
#The model performance has improved dramatically from the validation set performance (0.8851712)

GBM Training Model Partial Dependence Plot

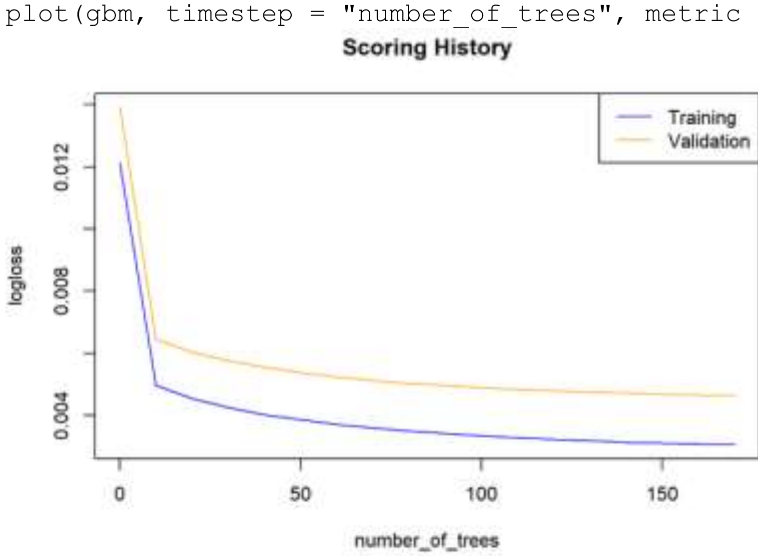
```
plot(gbm)
```



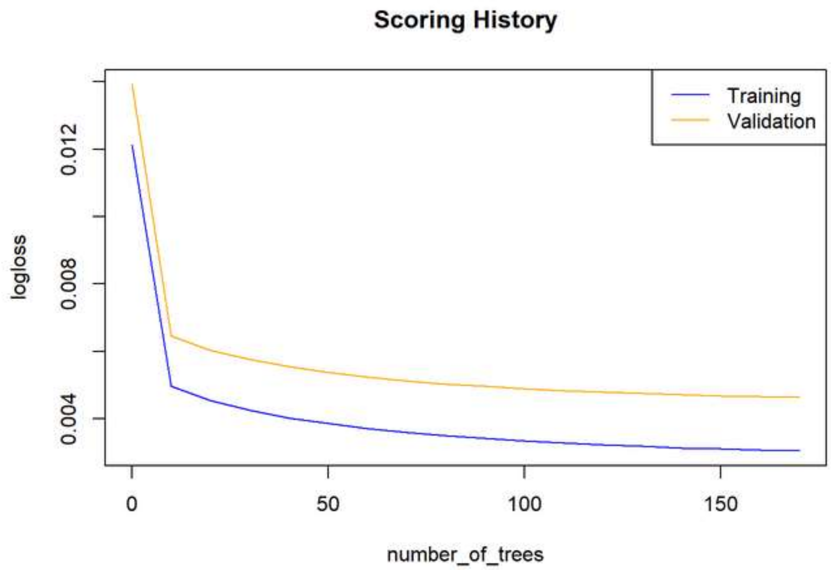

```
plot(gbm, timestep = "number_of_trees", metric = "rmse")
```




```
plot(gbm, timestep = "number_of_trees", metric = "auc")
```

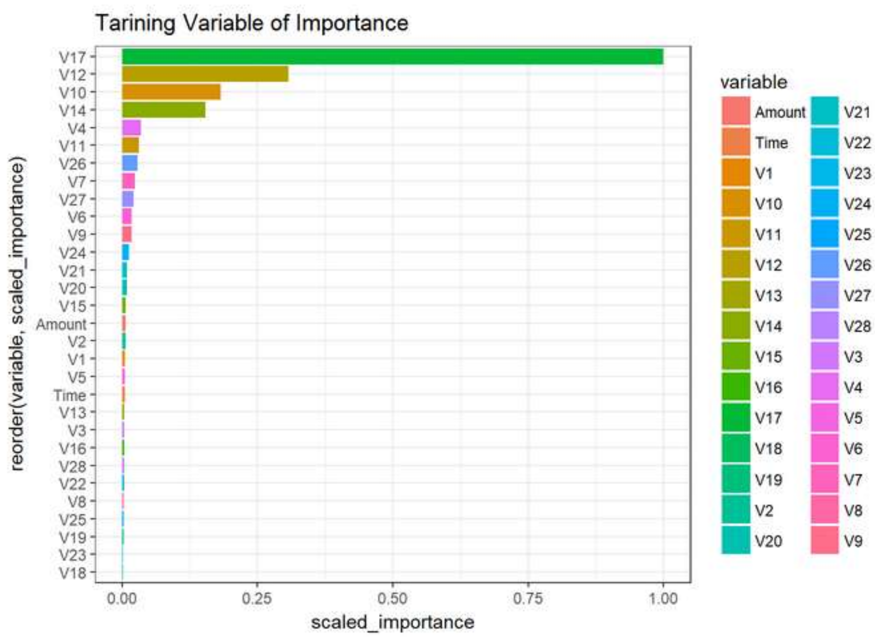


```
plot(gbm, timestep = "number_of_trees", metric = "logloss")
```



#Training variable of Importance

```
p <- gbm@model$variable_importances

ggplot(p, aes(x = reorder(variable, scaled_importance) , y =
scaled_importance, fill = variable)) +
  theme_bw()+
  geom_bar(stat = "identity") +
  ggtitle("Tarining Variable of Importance")+
  coord_flip()
```



Traning Model Hyper-Parameter Search

```
hyper_params = list( ntrees = 10000,
                     max_depth=seq(1,29,2))
cc.grid <- h2o.grid(hyper_params = hyper_params,
                  search_criteria = list(strategy = "Cartesian"),
                  algorithm="gbm",
                  grid_id="my.grid",
                  x = x,
                  y = y,
                  training_frame = train,
                  validation_frame = valid,
                  learn_rate = 0.02,
                  learn_rate_annealing = 0.99,
                  sample_rate = 0.8,
                  col_sample_rate = 0.8,
                  seed = 123456,
                  stopping_rounds = 5,
                  stopping_tolerance = 1e-8,
                  stopping_metric = "AUC",
                  score_tree_interval = 10)
cc.grid
## H2O Grid Details
## ================
##
## Grid ID: my.grid
## Used hyper parameters:
##   -  max_depth
##   -  ntrees
## Number of models: 15
## Number of failed models: 0
##
## Hyper-Parameter Search Summary: ordered by increasing logloss
##    max_depth ntrees        model_ids               logloss
## 1          3  10000  my.grid_model_1  0.004596453559600422
## 2          5  10000  my.grid_model_2 0.0046327627110317555
## 3         15  10000  my.grid_model_7  0.004688471276981163
## 4         13  10000  my.grid_model_6  0.004743346990864267
## 5         19  10000  my.grid_model_9  0.004746555930508997
## 6         23  10000 my.grid_model_11  0.004754278808414476
## 7         29  10000 my.grid_model_14   0.00475977583415095
## 8         21  10000 my.grid_model_10  0.004768430662394952
## 9         27  10000 my.grid_model_13  0.004772099008564593
## 10         9  10000  my.grid_model_4  0.004784380038727926
## 11         7  10000  my.grid_model_3  0.004784507391774722
## 12        17  10000  my.grid_model_8  0.004787759168946505
## 13        25  10000 my.grid_model_12  0.004800160800757093
## 14        11  10000  my.grid_model_5  0.004830196727455461
## 15         1  10000  my.grid_model_0  0.006175649286002828
```

sort the grid models by decreasing AUC

```
sortedGrid <- h2o.getGrid("my.grid", sort_by="auc", decreasing = TRUE)
sortedGrid
## H2O Grid Details
## Grid ID: my.grid
## Used hyper parameters:
##   -  max_depth
##   -  ntrees
## Number of models: 15
## Number of failed models: 0

## Hyper-Parameter Search Summary: ordered by decreasing auc
##    max_depth ntrees        model_ids                auc
## 1         19  10000  my.grid_model_9 0.9717289204454134
## 2         21  10000 my.grid_model_10 0.9708261888281612
```

```
## 3         17  10000  my.grid_model_8 0.9698675665175979
## 4         15  10000  my.grid_model_7 0.9688109739599444
## 5         13  10000  my.grid_model_6 0.9682088567364843
## 6          9  10000  my.grid_model_4 0.9673891534902204
## 7         27  10000 my.grid_model_13 0.9653138480500044
## 8         11  10000  my.grid_model_5 0.9652440654502633
## 9         23  10000 my.grid_model_11 0.9646235333740938
## 10         7  10000  my.grid_model_3 0.9624937102778302
## 11        29  10000 my.grid_model_14 0.9558180031030642
## 12         5  10000  my.grid_model_2 0.9527588760801563
## 13        25  10000 my.grid_model_12 0.9503348191750308
## 14         3  10000  my.grid_model_1 0.9077037082505809
## 15         1  10000  my.grid_model_0 0.9033031238213081
```

find the range of max_depth for the top 5 models

```
topDepths = sortedGrid@summary_table$max_depth[1:5]
minDepth = min(as.numeric(topDepths))
maxDepth = max(as.numeric(topDepths))

minDepth
## [1] 13

maxDepth
## [1] 21
```

Inspect the best 10 models from the grid search/query their validation AUC

```
 for (i in 1:10) {
  gbm <- h2o.getModel(sortedGrid@model_ids[[i]])
  print(h2o.auc(h2o.performance(gbm, valid = TRUE)))
}
## [1] 0.9717289
## [1] 0.9708262
## [1] 0.9698676
## [1] 0.968811
## [1] 0.9682089
## [1] 0.9673892
## [1] 0.9653138
## [1] 0.9652441
## [1] 0.9646235
## [1] 0.9624937
```

Model inspection and final test set scoring

 # Judge best model of the grid search by AUC validation on held out test

```
gbm <- h2o.getModel(sortedGrid@model_ids[[1]])
print(h2o.auc(h2o.performance(gbm, newdata = test)))
## [1] 0.9750609
# It performs well on the test set as on the validation set
# Thebest GBM model generalizes well to the unseen test set
```

Inspect the winning model’s parameters

```
gbm@parameters
## $model_id
## [1] "my.grid_model_9"
##
## $training_frame
## [1] "RTMP_sid_a168_101"
##
## $validation_frame
## [1] "RTMP_sid_a168_102"
##
## $score_tree_interval
```

```
## [1] 10
##
## $ntrees
## [1] 10000
##
## $max_depth
## [1] 19
##
## $stopping_rounds
## [1] 5
##
## $stopping_metric
## [1] "AUC"
##
## $stopping_tolerance
## [1] 1e-08
##
## $max_runtime_secs
## [1] 1.797693e+308
##
## $seed
## [1] 123456
##
## $learn_rate
## [1] 0.02
##
## $learn_rate_annealing
## [1] 0.99
##
## $distribution
## [1] "bernoulli"
##
## $sample_rate
## [1] 0.8
##
## $col_sample_rate
## [1] 0.8
##
## $x
##  [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"
##  [8] "V7"     "V8"     "V9"     "V10"    "V11"    "V12"    "V13"
## [15] "V14"    "V15"    "V16"    "V17"    "V18"    "V19"    "V20"
## [22] "V21"    "V22"    "V23"    "V24"    "V25"    "V26"    "V27"
## [29] "V28"    "Amount"
##
## $y
## [1] "Class"
```

Best Model Features

```
best.model <- h2o.getModel(sortedGrid@model_ids[[1]])
summary(best.model)
```

```
## H2OBinomialModel: gbm
## Model Key:  my.grid_model_9
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             190                      190              469065        17
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1        19   18.98947         61        506   191.64737
##
## H2OBinomialMetrics: gbm
## ** Reported on training data. **
##
## MSE:  8.507187e-05
## RMSE:  0.009223441
```

```
## LogLoss:  0.0006086346
## Mean Per-Class Error:  0.007160411
## AUC:  0.9999966
## Gini:  0.9999931
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal
threshold:
##             0   1    Error        Rate
## 0      170895   6 0.000035   =6/170901
## 1           4 276 0.014286      =4/280
## Totals 170899 282 0.000058  =10/171181
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.347418 0.982206 143
## 2                       max f2  0.176064 0.987306 158
## 3                 max f0point5  0.373098 0.981375 140
## 4                 max accuracy  0.347418 0.999942 143
## 5                max precision  0.999853 1.000000   0
## 6                   max recall  0.176064 1.000000 158
## 7              max specificity  0.999853 1.000000   0
## 8             max absolute_mcc  0.347418 0.982183 143
## 9   max min_per_class_accuracy  0.176064 0.999895 158
## 10 max mean_per_class_accuracy  0.176064 0.999947 158
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
## H2OBinomialMetrics: gbm
## ** Reported on validation data. **
##
## MSE:  0.0007520406
## RMSE:  0.02742336
## LogLoss:  0.004746556
## Mean Per-Class Error:  0.1560337
## AUC:  0.9717289
## Gini:  0.9434578
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal
threshold:
##            0  1    Error       Rate
## 0      56787  8 0.000141   =8/56795
## 1         34 75 0.311927    =34/109
## Totals 56821 83 0.000738  =42/56904
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.895365 0.781250  63
## 2                       max f2  0.030142 0.764388  99
## 3                 max f0point5  0.895365 0.850340  63
## 4                 max accuracy  0.895365 0.999262  63
## 5                max precision  0.999929 1.000000   0
## 6                   max recall  0.000299 1.000000 389
## 7              max specificity  0.999929 1.000000   0
## 8             max absolute_mcc  0.895365 0.788171  63
## 9   max min_per_class_accuracy  0.000364 0.919148 336
## 10 max mean_per_class_accuracy  0.000437 0.928562 311
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
##
##
## Scoring History:
##              timestamp          duration number_of_trees training_rmse
## 1  2018-08-06 04:23:47  9 min  6.024 sec               0       0.04041
## 2  2018-08-06 04:23:54  9 min 13.206 sec              10       0.01942
```

```
## 3  2018-08-06 04:24:01  9 min 20.824 sec                20       0.01775
## 4  2018-08-06 04:24:09  9 min 28.417 sec                30       0.01634
## 5  2018-08-06 04:24:17  9 min 36.091 sec                40       0.01525
## 6  2018-08-06 04:24:24  9 min 43.677 sec                50       0.01428
## 7  2018-08-06 04:24:32  9 min 51.582 sec                60       0.01352
## 8  2018-08-06 04:24:40  9 min 59.301 sec                70       0.01284
## 9  2018-08-06 04:24:48 10 min  7.032 sec                80       0.01229
## 10 2018-08-06 04:24:55 10 min 14.685 sec                90       0.01179
## 11 2018-08-06 04:25:03 10 min 22.431 sec               100       0.01135
## 12 2018-08-06 04:25:11 10 min 30.042 sec               110       0.01098
## 13 2018-08-06 04:25:18 10 min 37.643 sec               120       0.01065
## 14 2018-08-06 04:25:26 10 min 45.329 sec               130       0.01035
## 15 2018-08-06 04:25:34 10 min 52.996 sec               140       0.01011
## 16 2018-08-06 04:25:41 11 min  0.609 sec               150       0.00989
## 17 2018-08-06 04:25:49 11 min  8.370 sec               160       0.00969
## 18 2018-08-06 04:25:57 11 min 16.075 sec               170       0.00952
## 19 2018-08-06 04:26:04 11 min 23.666 sec               180       0.00937
## 20 2018-08-06 04:26:12 11 min 31.302 sec               190       0.00922
##    training_logloss training_auc training_lift
## 1           0.01213      0.50000       1.00000
## 2           0.00259      0.99995      99.98890
## 3           0.00208      0.99996      99.98890
## 4           0.00176      0.99997      99.98890
## 5           0.00153      0.99997      99.98890
## 6           0.00135      0.99998      99.98890
## 7           0.00121      0.99998      99.98890
## 8           0.00110      0.99998      99.93053
## 9           0.00101      0.99999      99.98890
## 10          0.00094      0.99999      99.98890
## 11          0.00088      0.99999      99.98890
## 12          0.00083      0.99999      99.98890
## 13          0.00078      0.99999      99.98890
## 14          0.00074      0.99999      99.98890
## 15          0.00071      0.99999      99.98890
## 16          0.00069      1.00000      99.98890
## 17          0.00066      1.00000      99.98890
## 18          0.00064      1.00000      99.98890
## 19          0.00063      1.00000      99.98890
## 20          0.00061      1.00000      99.98890
##    training_classification_error validation_rmse validation_logloss
## 1                        0.99836         0.04373            0.01392
## 2                        0.00030         0.02882            0.00569
## 3                        0.00024         0.02858            0.00545
## 4                        0.00020         0.02840            0.00527
## 5                        0.00018         0.02825            0.00516
## 6                        0.00016         0.02820            0.00507
## 7                        0.00016         0.02807            0.00500
## 8                        0.00014         0.02800            0.00495
## 9                        0.00013         0.02790            0.00490
## 10                       0.00012         0.02782            0.00486
## 11                       0.00012         0.02774            0.00484
## 12                       0.00012         0.02770            0.00481
## 13                       0.00011         0.02767            0.00480
## 14                       0.00009         0.02760            0.00479
## 15                       0.00008         0.02758            0.00478
## 16                       0.00007         0.02754            0.00477
## 17                       0.00006         0.02749            0.00476
## 18                       0.00006         0.02747            0.00475
## 19                       0.00006         0.02745            0.00475
## 20                       0.00006         0.02742            0.00475
##    validation_auc validation_lift validation_classification_error
## 1         0.50000         1.00000                         0.99808
## 2         0.93433        82.42974                         0.00084
## 3         0.94489        81.51386                         0.00077
## 4         0.95846        81.51386                         0.00076
```

```
## 5        0.96075       81.51386                          0.00076
## 6        0.96808       81.51386                          0.00076
## 7        0.96521       81.51386                          0.00076
## 8        0.96502       81.51386                          0.00076
## 9        0.96965       81.51386                          0.00076
## 10       0.97009       81.51386                          0.00076
## 11       0.97272       81.51386                          0.00076
## 12       0.97195       81.51386                          0.00083
## 13       0.97232       81.51386                          0.00083
## 14       0.97202       81.51386                          0.00081
## 15       0.97239       81.51386                          0.00081
## 16       0.97249       81.51386                          0.00081
## 17       0.97217       81.51386                          0.00074
## 18       0.97166       81.51386                          0.00074
## 19       0.97160       81.51386                          0.00074
## 20       0.97173       81.51386                          0.00074
##
## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
##
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      V17          208.194717          1.000000   0.238659
## 2      V11           98.617279          0.473678   0.113048
## 3      V14           62.697151          0.301147   0.071871
## 4       V4           51.119110          0.245535   0.058599
## 5      V12           49.196831          0.236302   0.056396
##
## ---
##    variable relative_importance scaled_importance percentage
## 25       V8            7.956810          0.038218   0.009121
## 26       V7            6.810081          0.032710   0.007807
## 27       V2            6.506504          0.031252   0.007459
## 28       V5            6.224453          0.029897   0.007135
## 29      V28            5.861414          0.028154   0.006719
## 30   Amount            5.687578          0.027319   0.006520
scoring_history <- as.data.frame(best.model@model$scoring_history)

#Training MSE
#Plot scoring history
plot(x= scoring_history$number_of_trees, y= scoring_history$training_logloss
, main = "Scoring History Curve", xlab="# of Trees", ylab="Training RMSE",
     col.axis="purple",col= 'Tomato2')

#Actual number of trees
ntrees <- best.model@model$model_summary$number_of_trees

#Plot the variable of importance
df <- best.model@model$variable_importances
df
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      V17          208.194717          1.000000   0.238659
## 2      V11           98.617279          0.473678   0.113048
## 3      V14           62.697151          0.301147   0.071871
## 4       V4           51.119110          0.245535   0.058599
## 5      V12           49.196831          0.236302   0.056396
##
## ---
##    variable relative_importance scaled_importance percentage
## 25       V8            7.956810          0.038218   0.009121
## 26       V7            6.810081          0.032710   0.007807
## 27       V2            6.506504          0.031252   0.007459
## 28       V5            6.224453          0.029897   0.007135
## 29      V28            5.861414          0.028154   0.006719
```

```
## 30   Amount            5.687578          0.027319   0.006520
 best.model@model$variable_importances
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1      V17          208.194717          1.000000   0.238659
## 2      V11           98.617279          0.473678   0.113048
## 3      V14           62.697151          0.301147   0.071871
## 4       V4           51.119110          0.245535   0.058599
## 5      V12           49.196831          0.236302   0.056396
##
## ---
##    variable relative_importance scaled_importance percentage
## 25       V8            7.956810          0.038218   0.009121
## 26       V7            6.810081          0.032710   0.007807
## 27       V2            6.506504          0.031252   0.007459
## 28       V5            6.224453          0.029897   0.007135
## 29      V28            5.861414          0.028154   0.006719
## 30   Amount            5.687578          0.027319   0.006520
ggplot(df, aes(x = reorder(variable, scaled_importance) , y =
scaled_importance, fill = variable)) +
  theme_bw()+
  geom_bar(stat = "identity") +
  ggtitle("variable of Importance")+
  coord_flip()
```

Model prediction

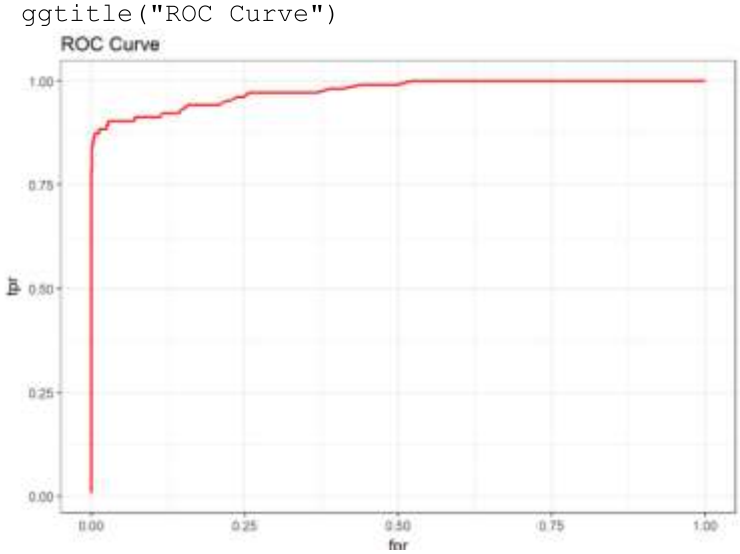
 # Calculate performance measures at threshold that maximizes precision

```
cc.prediction <- h2o.predict(best.model,test)
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
cc.prediction
##   predict        p0           p1
## 1       0 0.9996958 0.0003042013
## 2       0 0.9997001 0.0002998611
## 3       0 0.9996619 0.0003381131
## 4       0 0.9997004 0.0002996062
## 5       0 0.9996857 0.0003143171
## 6       0 0.9996967 0.0003032855
##
## [56722 rows x 3 columns]
cc.performance <- h2o.performance(best.model, test)
cc.performance
## H2OBinomialMetrics: gbm
##
## MSE:  0.0006181165
## RMSE:  0.02486195
## LogLoss:  0.003857607
## Mean Per-Class Error:  0.1359577
## AUC:  0.9750609
## Gini:  0.9501217
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal
threshold:
##            0  1    Error       Rate
## 0      56615  4 0.000071   =4/56619
## 1         28 75 0.271845    =28/103
## Totals 56643 79 0.000564  =32/56722
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
```

```
## 1                        max f1  0.947771 0.824176  60
## 2                        max f2  0.030576 0.800000  94
## 3                  max f0point5  0.947771 0.894988  60
## 4                  max accuracy  0.947771 0.999436  60
## 5                 max precision  0.999885 1.000000   0
## 6                    max recall  0.000304 1.000000 380
## 7               max specificity  0.999885 1.000000   0
## 8              max absolute_mcc  0.947771 0.831180  60
## 9    max min_per_class_accuracy  0.000371 0.912621 335
## 10 max mean_per_class_accuracy  0.000458 0.937565 307
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or
`h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

Plot ROC Curve

```
tpr<-as.data.frame(h2o.tpr(cc.performance))
fpr<-as.data.frame(h2o.fpr(cc.performance))

ROC<-merge(tpr,fpr,by='threshold')
head(ROC)
##      threshold tpr       fpr
## 1 1.877446e-05   1 1.0000000
## 2 4.905438e-05   1 0.9999470
## 3 1.837972e-04   1 0.9999294
## 4 2.213887e-04   1 0.9998940
## 5 2.434876e-04   1 0.9998410
## 6 2.648282e-04   1 0.9998057

ggplot(ROC, aes(x = fpr, y = tpr)) +
  theme_bw() +
  geom_line(size= 0.8, colour="red")+
  ggtitle("ROC Curve")
```

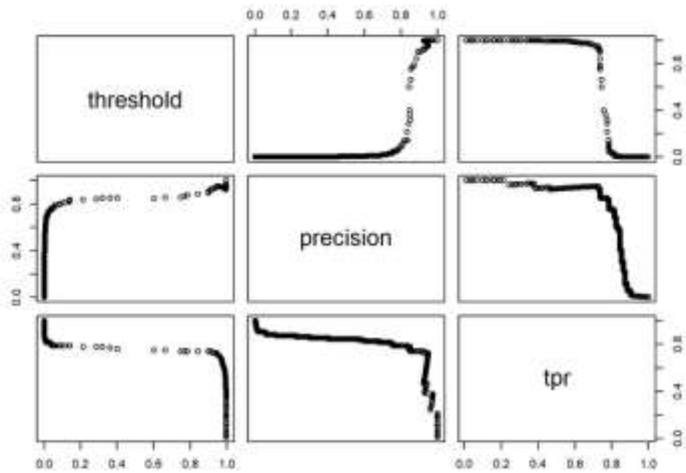


Precision & Threshold Curve, Precision Recall Curve Plots

```
h2o.F1(cc.performance)
##   threshold         f1
## 1 0.9998849 0.01923077
## 2 0.9997835 0.05660377
## 3 0.9997236 0.07476636
## 4 0.9996631 0.09259259
## 5 0.9996060 0.12727273

##        threshold          f1
## 395 2.648282e-04 0.003625867
## 396 2.434876e-04 0.003625739
## 397 2.213887e-04 0.003625548
## 398 1.837972e-04 0.003625420
```

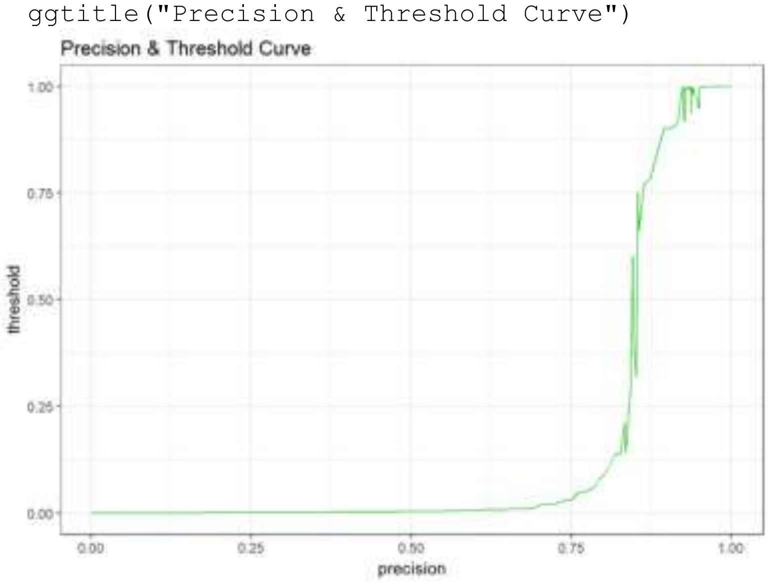
```
## 399 4.905438e-05 0.003625356
## 400 1.877446e-05 0.003625165
rf.precision <- as.data.frame(h2o.precision(cc.performance))

rf.recall <- as.data.frame(h2o.recall(cc.performance))

rf.pre.recall <-merge(rf.precision, rf.recall,by='threshold')
head(rf.pre.recall)
##      threshold   precision tpr
## 1 1.877446e-05 0.001815874   1
## 2 4.905438e-05 0.001815970   1
## 3 1.837972e-04 0.001816002   1
## 4 2.213887e-04 0.001816066   1
## 5 2.434876e-04 0.001816162   1
## 6 2.648282e-04 0.001816226   1
```

plot(rf.pre.recall)

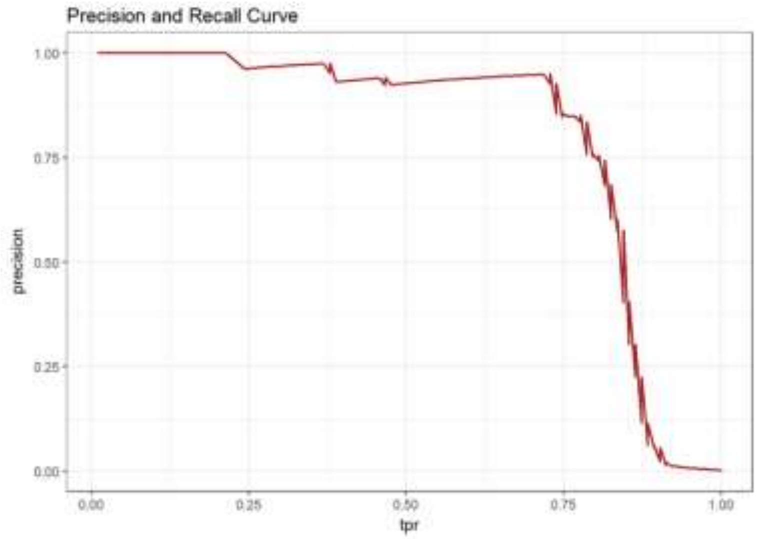


#Precision & Threshold Curve

```
ggplot(rf.pre.recall, aes(x = precision, y = threshold)) +
  theme_bw() +
  geom_line(size= 0.5, colour="lime green") +
  ggtitle("Precision & Threshold Curve")
```



```
#Precision Recall Curve
ggplot(rf.pre.recall, aes(x = tpr, y = precision)) +
  theme_bw() +
  geom_line(size= 0.8, colour="brown") +
  ggtitle("Precision and Recall Curve")
```




```
# All done. Shut down H2O.
h2o.shutdown(prompt=FALSE)
```