# Data Mining Workflow

## Set Up the R Notebook for Analysis

```r
# Load necessary packages
library('swat')
```

```
## SWAT 1.0.0
```

```r
library('ggplot2')
library('reshape2')
options(cas.print.messages = FALSE)

# Data name
indata <- 'hmeq'

# Hostname, port, username, password
conn <- CAS(hostname, 8777, protocol = 'http')
```

```
## NOTE: Connecting to CAS and generating CAS action functions for loaded
##       action sets...
## NOTE: To generate the functions with signatures (for tab completion), set
##       options(cas.gen.function.sig=TRUE).
```

```r
# Read in the dataset
castbl <- cas.read.csv(conn, paste0('http://support.sas.com/documentation',
                                    '/onlinedoc/viya/exampledatasets/hmeq.csv'))
```

## View Data

```r
# Print the first few rows
head(castbl)
```

```
##   BAD LOAN MORTDUE  VALUE  REASON    JOB  YOJ DEROG DELINQ     CLAGE NINQ
## 1   1 1100   25860  39025 HomeImp  Other 10.5     0      0  94.36667    1
## 2   1 1300   70053  68400 HomeImp  Other  7.0     0      2 121.83333    0
## 3   1 1500   13500  16700 HomeImp  Other  4.0     0      0 149.46667    1
## 4   1 1500     NaN    NaN                 NaN   NaN    NaN       NaN  NaN
## 5   0 1700   97800 112000 HomeImp Office  3.0     0      0  93.33333    0
## 6   1 1700   30548  40320 HomeImp  Other  9.0     0      0 101.46600    1
##   CLNO  DEBTINC
## 1    9      NaN
## 2   14      NaN
## 3   10      NaN
## 4  NaN      NaN
## 5   14      NaN
## 6    8 37.11361
```

## Get Summary Statistics

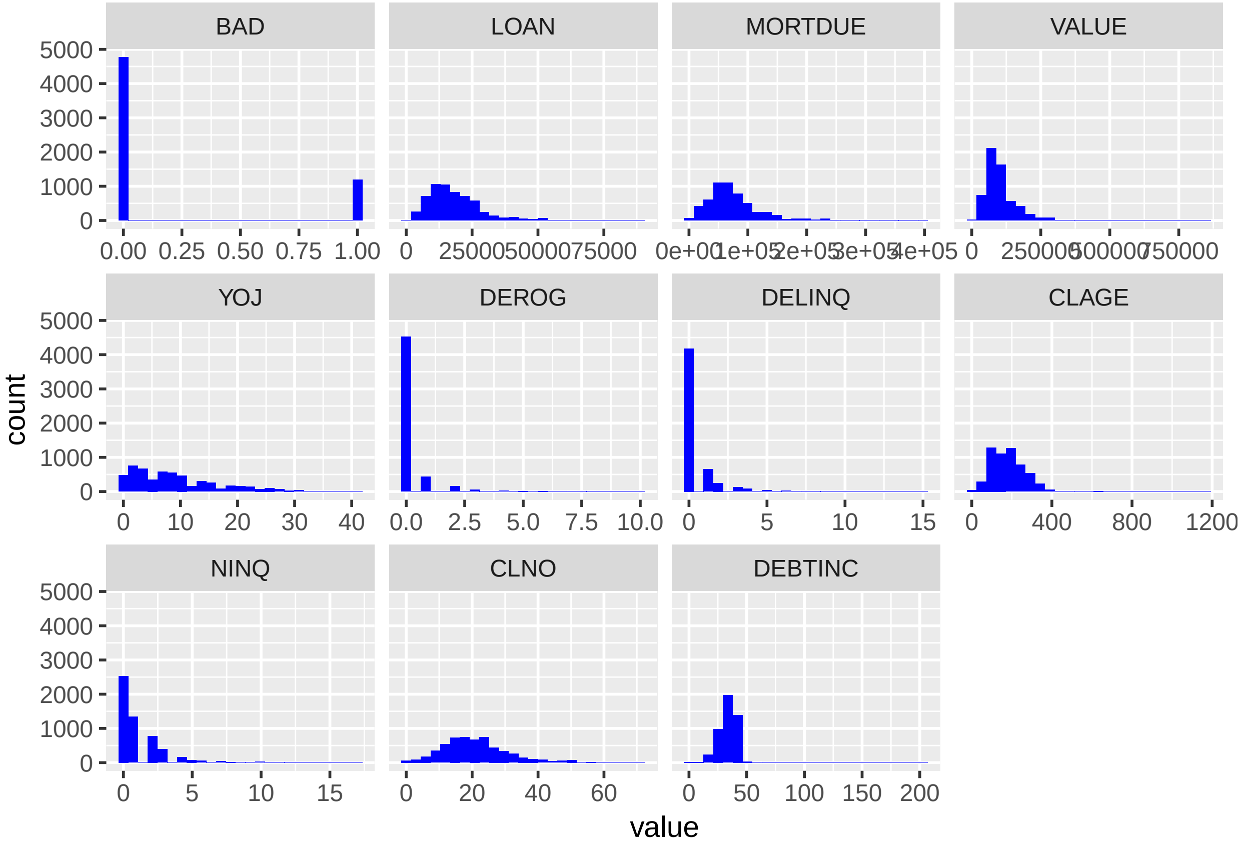
```r
# Use summary function to get variable summary
summary(castbl)
```

```
##       BAD             LOAN          MORTDUE      
##  Min.   :0.0000   Min.    : 1100  Min.   :2063   
##  1st Qu.:0.0000   1st Qu.:11100   1st Qu.:46268  
##  Median :0.0000   Median :16300   Median :65019  
##  Mean   :0.1995   Mean    :18608  Mean   :73760.8171995589  
##  3rd Qu.:0.0000   3rd Qu.:23300   3rd Qu.:91491  
##  Max.   :1.0000   Max.    :89900  Max.   :399550 
##                                   NA's   :518    
##      VALUE                     REASON          JOB      
##  Min.   :8000              DebtCon:3928   Mgr    : 767  
##  1st Qu.:66069             HomeImp:1780   Office : 948  
##  Median :89235.5           NA's   : 252   Other  :2388  
##  Mean   :101776.04874145                  ProfExe:1276  
##  3rd Qu.:119831.5                         Sales  : 109  
##  Max.   :855909                           Self   : 193  
##  NA's   :112                              NA's   : 279  
##       YOJ                       DEROG      
##  Min.   :0                  Min.   :0      
##  1st Qu.:3                  1st Qu.:0      
##  Median :7                  Median :0      
##  Mean   :8.92226813590449   Mean   :0.254569687738  
##  3rd Qu.:13                 3rd Qu.:0      
##  Max.   :41                 Max.   :10     
##  NA's   :515                NA's   :708    
##      DELINQ                     CLAGE      
##  Min.   :0                  Min.   :0      
##  1st Qu.:0                  1st Qu.:115.103196832924  
##  Median :0                  Median :173.466666666667  
##  Mean   :0.44944237918215   Mean   :179.766275186577  
##  3rd Qu.:0                  3rd Qu.:231.574833599946  
##  Max.   :15                 Max.   :1168.23356094464  
##  NA's   :580                NA's   :308    
##       NINQ                       CLNO      
##  Min.   :0                  Min.   :0      
##  1st Qu.:0                  1st Qu.:15     
##  Median :1                  Median :20     
##  Mean   :1.18605504587155   Mean   :21.2960962007668  
##  3rd Qu.:2                  3rd Qu.:26     
##  Max.   :17                 Max.   :71     
##  NA's   :510                NA's   :222    
##     DEBTINC      
##  Min.   :0.52449921542988  
##  1st Qu.:29.1400313718617  
##  Median :34.818261818587  
##  Mean   :33.7799153487192  
##  3rd Qu.:39.0031406283719  
##  Max.   :203.312148691165  
##  NA's   :1267
```

### Visualize Numeric Variables

```r
# Bring data locally
df <- to.casDataFrame(castbl, obs = nrow(castbl))

# Use reshape2's melt to help with data formatting
d <- melt(df[sapply(df, is.numeric)], id.vars=NULL)
ggplot(d, aes(x = value)) +
    facet_wrap(~variable,scales = 'free_x') +
    geom_histogram(fill = 'blue', bins = 25)
```

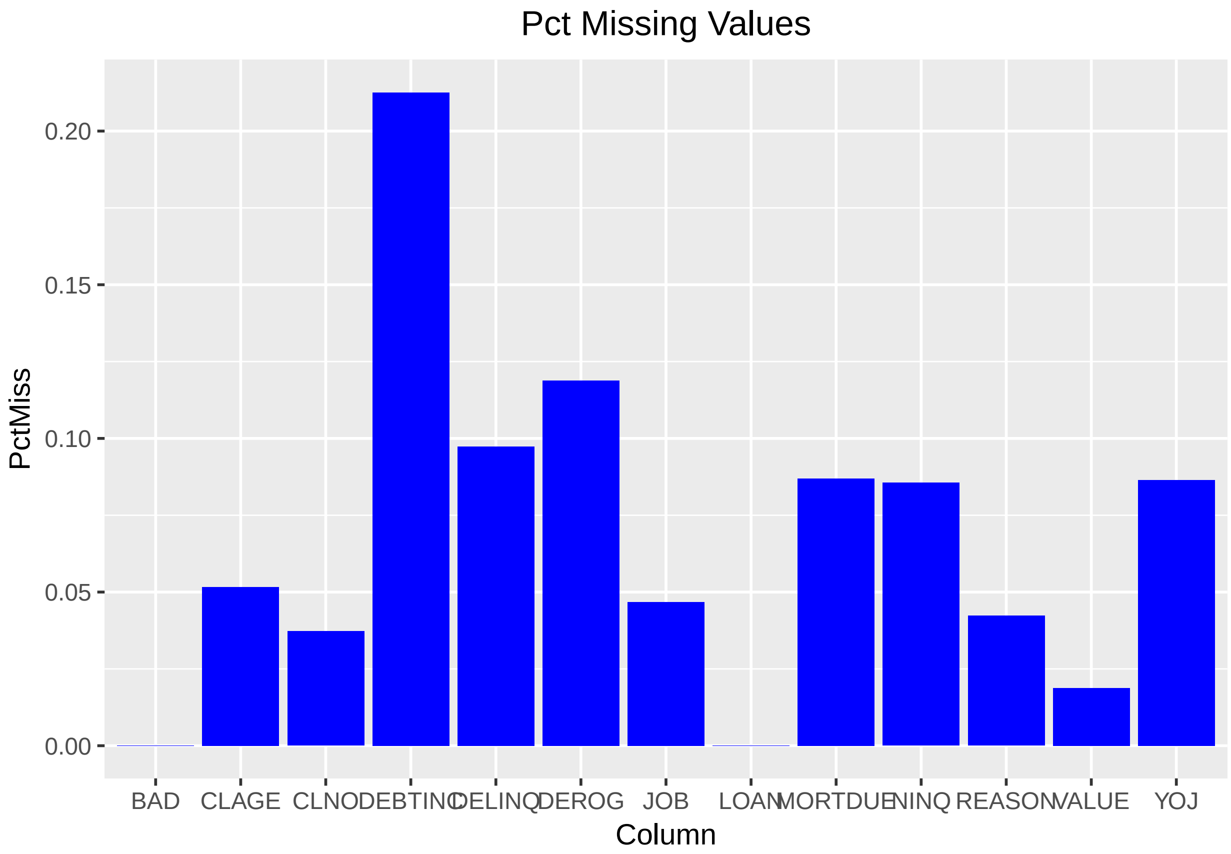


### Check for Missingness

```r
# Check for missing values
tbl <- cas.simple.distinct(castbl)$Distinct[,c('Column', 'NMiss')]
tbl
```

```
##      Column NMiss
## 1       BAD     0
## 2      LOAN     0
## 3   MORTDUE   518
## 4     VALUE   112
## 5    REASON   252
## 6       JOB   279
## 7       YOJ   515
## 8     DEROG   708
## 9    DELINQ   580
## 10    CLAGE   308
## 11     NINQ   510
## 12     CLNO   222
## 13 DEBTINC  1267
```

```
# Visualize the missing data
tbl$PctMiss <- tbl$NMiss/nrow(castbl)
ggplot(tbl, aes(Column, PctMiss)) +
  geom_col(fill = 'blue') +
  ggtitle('Pct Missing Values') +
  theme(plot.title = element_text(hjust = 0.5))
```



### Impute Missing Values

```
# Impute missing values
cas.dataPreprocess.impute(castbl,
    methodContinuous  = 'MEDIAN',
    methodNominal     = 'MODE',
    inputs            = colnames(castbl)[-1],
    copyAllVars       = TRUE,
    casOut            = list(name = indata, replace = TRUE)
)
```

```
## $ImputeInfo
##    Variable ImputeTech   ResultVar    N NMiss ImputedValueContinuous
## 1      LOAN     Median    IMP_LOAN 5960     0            16300.00000
## 2   MORTDUE     Median IMP_MORTDUE 5442   518            65019.00000
## 3     VALUE     Median   IMP_VALUE 5848   112            89235.50000
## 4    REASON       Mode  IMP_REASON 5708   252                    NaN
## 5       JOB       Mode     IMP_JOB 5681   279                    NaN
## 6       YOJ     Median     IMP_YOJ 5445   515                7.00000
## 7     DEROG     Median   IMP_DEROG 5252   708                0.00000
## 8    DELINQ     Median  IMP_DELINQ 5380   580                0.00000
## 9     CLAGE     Median   IMP_CLAGE 5652   308              173.46667
## 10     NINQ     Median    IMP_NINQ 5450   510                1.00000
## 11     CLNO     Median    IMP_CLNO 5738   222               20.00000
## 12  DEBTINC     Median IMP_DEBTINC 4693  1267               34.81826
##    ImputedValueNominal
## 1
## 2
## 3
## 4             DebtCon
## 5               Other
## 6
## 7
## 8
## 9
## 10
## 11
## 12
##
## $OutputCasTables
##                casLib Name Rows Columns
## 1 CASUSERHDFS(jelueb) hmeq 5960      25
```

### Split the Data into Training and Validation

```r
# Load the sampling actionset
loadActionSet(conn, 'sampling')

# Partition the data
cas.sampling.srs(conn,
    table   = indata,
    samppct = 30,
    partind = TRUE,
    output  = list(casOut = list(name = indata, replace = T), copyVars = 'ALL')
)
```

```r
# Load the fedsql actionset
loadActionSet(conn, 'fedsql')

# Make sure the partition worked correctly using SQL
cas.fedsql.execDirect(conn, query = paste0("
    SELECT
        CASE WHEN _PartInd_ = 0 THEN 'Training' ELSE 'Validation' END AS name,
        _PartInd_,
        COUNT(*) AS obs
    FROM ", indata, "
    GROUP BY
        CASE WHEN _PartInd_ = 0 THEN 'Training' ELSE 'Validation' END,
        _PartInd_;
"))$`Result Set`
```

```
##         NAME _PartInd_  OBS
## 1   Training         0 4172
## 2 Validation         1 1788
```

### Variable Shortcuts

Note: I do not want to hard code any of my variable names.

```
# Get variable info and types
colinfo <- head(cas.table.columnInfo(conn, table = indata)$ColumnInfo, -1)

# My target variable is the first column
target <- colinfo$Column[1]

# For models that can inherently handle missing values (ex: Decision Tree)
inputs <- colinfo$Column[-1]
nominals <- c(target, subset(colinfo, Type == 'varchar')$Column)

# For models that cannot handle missing values (ex: Neural Network)
imp.inputs <- grep('IMP_', inputs, value = T)
imp.nominals <- c(target, grep('IMP_', nominals, value = T))
```

## Model Building

### Decision Tree

```
# Load the decsion tree actionset
loadActionSet(conn, 'decisionTree')

# Train the decision tree model
cas.decisionTree.dtreeTrain(conn,
    table    = list(name = indata, where = '_PartInd_ = 0'),
    target   = target,
    inputs   = inputs,
    nominals = nominals,
    varImp   = TRUE,
    casOut   = list(name = 'dt_model', replace = TRUE)
)
```

```
## $DTreeVarImpInfo
##   Variable Importance        Std Count
## 1  DEBTINC 438.877290 181.520795     2
## 2   DELINQ  50.426019  24.948820     2
## 3    DEROG  22.090378   2.396851     2
## 4    VALUE   8.621421   0.000000     1
## 5     NINQ   1.716695   0.000000     1
##
## $ModelInfo
##                             Descr      Value
## 1           Number of Tree Nodes   17.00000
## 2         Max Number of Branches    2.00000
## 3               Number of Levels    6.00000
## 4               Number of Leaves    9.00000
## 5                 Number of Bins   20.00000
## 6         Minimum Size of Leaves    5.00000
## 7         Maximum Size of Leaves 3203.00000
## 8            Number of Variables   24.00000
## 9  Confidence Level for Pruning    0.25000
## 10  Number of Observations Used 4172.00000
## 11  Misclassification Error (%)   13.68648
##
## $OutputCasTables
##                 casLib     Name Rows Columns
## 1 CASUSERHDFS(jelueb) dt_model   17      27
```

### Random Forest

```
# Train the random forest model
cas.decisionTree.forestTrain(conn,
    table    = list(name = indata, where = '_PartInd_ = 0'),
    target   = target,
    inputs   = inputs,
    nominals = nominals,
    casOut   = list(name = 'rf_model', replace = TRUE)
)
```

```
## $ModelInfo
##                                 Descr      Value
## 1                     Number of Trees   50.00000
## 2  Number of Selected Variables (M)    5.00000
## 3                  Random Number Seed    0.00000
## 4            Bootstrap Percentage (%)   63.21206
## 5                      Number of Bins   20.00000
## 6                 Number of Variables   24.00000
## 7        Confidence Level for Pruning    0.25000
## 8            Max Number of Tree Nodes   29.00000
## 9            Min Number of Tree Nodes   11.00000
## 10             Max Number of Branches    2.00000
## 11             Min Number of Branches    2.00000
## 12               Max Number of Levels    6.00000
## 13               Min Number of Levels    6.00000
## 14               Max Number of Leaves   15.00000
## 15               Min Number of Leaves    6.00000
## 16             Maximum Size of Leaves 2590.00000
## 17             Minimum Size of Leaves    5.00000
## 18                 Out-of-Bag MCR (%)        NaN
##
## $OutputCasTables
##                 casLib     Name Rows Columns
## 1 CASUSERHDFS(jelueb) rf_model  742      41
```

### Gradient Boosting

```
# Train the gradient boosting model
cas.decisionTree.gbtreeTrain(conn,
    table    = list(name = indata, where = '_PartInd_ = 0'),
    target   = target,
    inputs   = inputs,
    nominals = nominals,
    casOut   = list(name = 'gbt_model', replace = TRUE)
)
```

```
## $ModelInfo
##                               Descr  Value
## 1                   Number of Trees   50.0
## 2                      Distribution    2.0
## 3                     Learning Rate    0.1
## 4                  Subsampling Rate    0.5
## 5  Number of Selected Variables (M)   24.0
## 6                    Number of Bins   20.0
## 7               Number of Variables   24.0
## 8           Max Number of Tree Nodes   61.0
## 9           Min Number of Tree Nodes   29.0
## 10           Max Number of Branches    2.0
## 11           Min Number of Branches    2.0
## 12             Max Number of Levels    6.0
## 13             Min Number of Levels    6.0
## 14             Max Number of Leaves   31.0
## 15             Min Number of Leaves   15.0
## 16           Maximum Size of Leaves 1492.0
## 17           Minimum Size of Leaves    5.0
## 18               Random Number Seed    0.0
##
## $OutputCasTables
##                 casLib      Name Rows Columns
## 1 CASUSERHDFS(jelueb) gbt_model 2492      31
```

### Neural Network

```
# Load the neuralNet actionset
loadActionSet(conn, 'neuralNet')

# Build a neural network model
cas.neuralNet.annTrain(conn,
    table    = list(name = indata, where = '_PartInd_ = 0'),
    target   = target,
    inputs   = imp.inputs,
    nominals = imp.nominals,
    casOut   = list(name = 'nn_model', replace = TRUE)
)
```

```
## $ConvergenceStatus
##                                           Reason
## 1 The optimization exited on maximum iterations.
##
## $ModelInfo
##                           Descr        Value
## 1                         Model   Neural Net
## 2  Number of Observations Used         4172
## 3  Number of Observations Read         4172
## 4     Target/Response Variable          BAD
## 5               Number of Nodes           20
## 6         Number of Input Nodes           18
## 7        Number of Output Nodes            2
## 8        Number of Hidden Nodes            0
## 9  Number of Weight Parameters           18
## 10   Number of Bias Parameters            2
## 11                 Architecture         GLIM
## 12        Number of Neural Nets            1
## 13              Objective Value 1.5329196041
##
## $OptIterHistory
##    Progress Objective     Loss
## 1         1  4.184992 4.184992
## 2         2  2.456393 2.456393
## 3         3  1.658346 1.658346
## 4         4  1.584687 1.584687
## 5         5  1.548499 1.548499
## 6         6  1.540727 1.540727
## 7         7  1.535718 1.535718
## 8         8  1.534470 1.534470
## 9         9  1.533433 1.533433
## 10       10  1.532920 1.532920
##
## $OutputCasTables
##                casLib     Name Rows Columns
## 1 CASUSERHDFS(jelueb) nn_model   20      15
```

## Score the Models

```
# Score the models
models <- c('dt','rf','gbt','nn')
scores <- c(cas.decisionTree.dtreeScore, cas.decisionTree.forestScore,
            cas.decisionTree.gbtreeScore, cas.neuralNet.annScore)
names(scores) <- models

# Function to help automate prediction process on new data
score.params <- function(model){return(list(
    object       = defCasTable(conn, indata),
    modelTable   = list(name = paste0(model, '_model')),
    copyVars     = list(target, '_PartInd_'),
    assessonerow = TRUE,
    casOut       = list(name = paste0(model, '_scored'), replace = T)
))}
```

```
lapply(models, function(x) {do.call(scores[[x]], score.params(x))})
```

## Compare Confusion Matrix

```
# Load the percentile actionset for scoring
loadActionSet(conn, 'percentile')

# Useful function for model assessment
assess.model <- function(model){
    cas.percentile.assess(conn,
        table    = list(name = paste0(model,'_scored'),
                        where = '_PartInd_ = 1'),
        inputs   = paste0('_', model, '_P_           1'),
        response = target,
        event    = '1')
}

model.names <- c('Decision Tree', 'Random Forest',
                 'Gradient Boosting', 'Neural Network')
roc.df <- data.frame()
for (i in 1:length(models)){
    tmp <- (assess.model(models[i]))$ROCInfo
    tmp$Model <- model.names[i]
    roc.df <- rbind(roc.df, tmp)
}

# Manipulate the dataframe
compare <- subset(roc.df, CutOff == 0.5)
rownames(compare) <- NULL
compare[,c('Model','TP','FP','FN','TN')]
```

```
##               Model  TP  FP  FN   TN
## 1     Decision Tree 244 152 103 1289
## 2     Random Forest  35   1 312 1440
## 3 Gradient Boosting 224  58 123 1383
## 4    Neural Network 106  50 241 1391
```

## Compare Misclassification

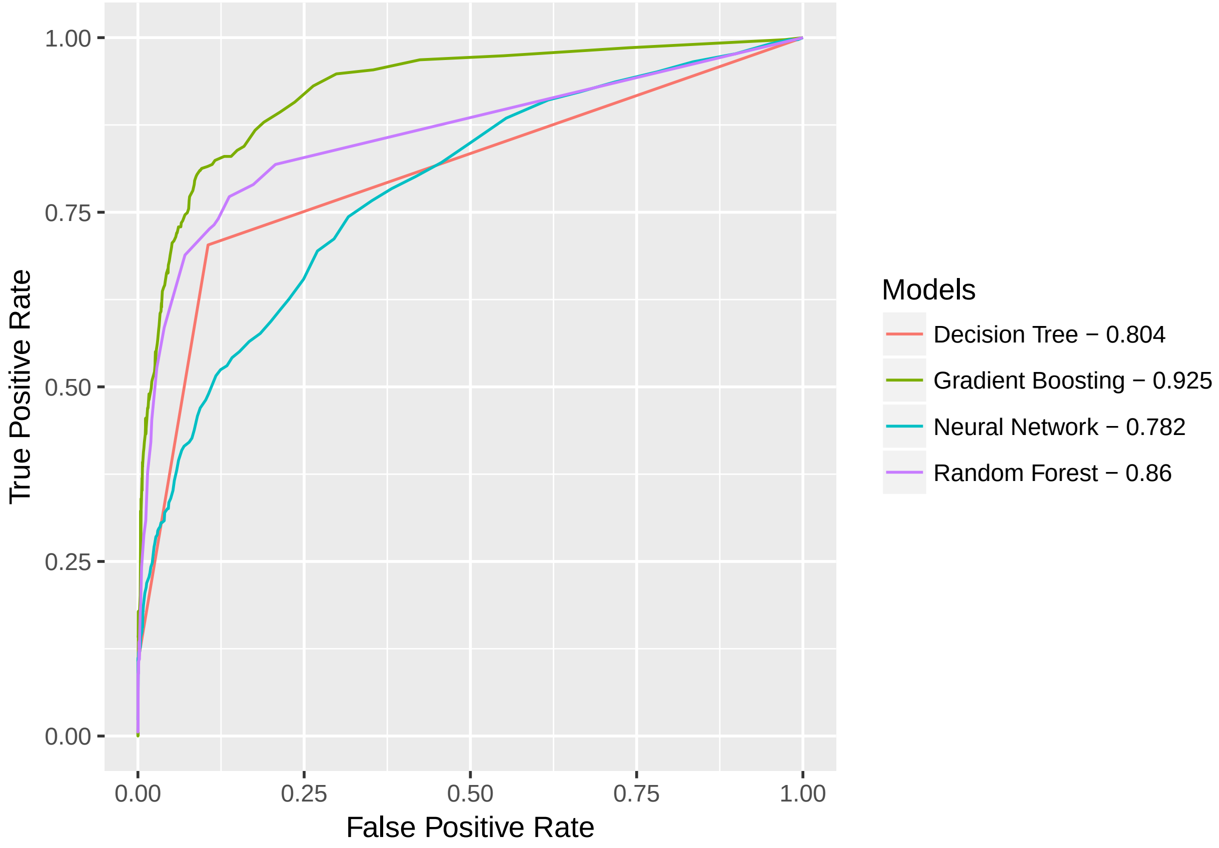
```
# Build a dataframe to compare the misclassification rates
compare$Misclassification <- 1 - compare$ACC
miss <- compare[order(compare$Misclassification), c('Model','Misclassification')]
rownames(miss) <- NULL
miss
```

```
##               Model Misclassification
## 1 Gradient Boosting         0.1012304
## 2     Decision Tree         0.1426174
## 3    Neural Network         0.1627517
## 4     Random Forest         0.1750559
```

### Compare ROC Curve

```
# Add a new column to be used as the ROC curve label
roc.df$Models <- paste(roc.df$Model, round(roc.df$C, 3), sep = ' - ')

# Create the ROC curve
ggplot(data = roc.df[c('FPR', 'Sensitivity', 'Models')],
       aes(x = as.numeric(FPR), y = as.numeric(Sensitivity), colour = Models)) +
       geom_line() +
       labs(x = 'False Positive Rate', y = 'True Positive Rate')
```



### Save the CAS Gradient Boosting Model

```
# Save the champion model for later use
cas.table.save(conn, table = list(name = 'gbt_model'), name = 'gbt_model', replace = T)
```

```
## $caslib
## [1] "CASUSERHDFS(jelueb)"
##
## $name
## [1] "gbt_model.sashdat"
```

### End the Session

```
# End the session
cas.session.endSession(conn)
```