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)

# Hostname, port, username, password
conn <- CAS(hostname, port, username, password)
```

```
## 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
# Change the active caslib to public
cas.sessionProp.setSessOpt(conn, caslib = 'public')
```

View Data

```r
# Create a CAS table for the prepped data set
castbl <- defCasTable(conn, 'hmeq_prepped')

# 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 1500     NaN   NaN                NaN   NaN    NaN       NaN  NaN
## 3   1 1800   48649 57037 HomeImp Other  5.0     3      2  77.10000    1
## 4   1 2000     NaN 62250 HomeImp Sales 16.0     0      0 115.80000    0
## 5   1 2000   45000 55000 HomeImp Other  3.0     0      0  86.06667    2
## 6   1 2200   24280 34687 HomeImp Other  NaN     0      1 300.86667    0
##   CLNO DEBTINC IMP_CLAGE IMP_CLNO IMP_DEBTINC IMP_DELINQ IMP_DEROG
## 1    9     NaN  94.36667        9    34.81826          0         0
## 2  NaN     NaN 173.46667       20    34.81826          0         0
## 3   17     NaN  77.10000       17    34.81826          2         3
## 4   13     NaN 115.80000       13    34.81826          0         0
## 5   25     NaN  86.06667       25    34.81826          0         0
## 6    8     NaN 300.86667        8    34.81826          1         0
##   IMP_LOAN IMP_MORTDUE IMP_MORTPAID IMP_NINQ IMP_VALUE IMP_YOJ IMP_JOB
## 1     1100       25860        13165        1   39025.0    10.5   Other
## 2     1500       65019        26623        1   89235.5     7.0   Other
## 3     1800       48649         8388        1   57037.0     5.0   Other
## 4     2000       65019        26623        0    62250.0     16.0    Sales
## 5     2000       45000        10000        2    55000.0      3.0    Other
## 6     2200       24280        10407        0    34687.0      7.0    Other
##   IMP_REASON MORTPAID _PartInd_
## 1    HomeImp    13165         1
## 2    DebtCon      NaN         0
## 3    HomeImp    8388          1
## 4    HomeImp      NaN         0
## 5    HomeImp    10000         0
## 6    HomeImp    10407         0
```

Variable Shortcuts

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

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

# My target variable is the first column
target <- colinfo$Column[1]
vars <- colinfo$Column[-1]
noms <- c(target, subset(colinfo, Type == 'varchar')$Column)

# For models that can inherently handle missing values (ex: Decision Tree)
inputs <- grep('IMP_', vars, value = TRUE, invert = TRUE)
nominals <- grep('IMP_', noms, value = TRUE, invert = TRUE)

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

Model Building

Decision Tree

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

# Train the decision tree model
cas.decisionTree.dtreeTrain(conn,
    table    = list(name = 'hmeq_prepped', 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 442.166677 183.520139     2
## 2   DELINQ  47.167293   0.000000     1
## 3    DEROG  11.843676   2.650169     2
## 4    VALUE   8.545419   0.000000     1
## 5  MORTDUE   1.819619   0.000000     1
##
## $ModelInfo
##                           Descr      Value
## 1          Number of Tree Nodes   15.00000
## 2        Max Number of Branches    2.00000
## 3              Number of Levels    6.00000
## 4              Number of Leaves    8.00000
## 5                Number of Bins   20.00000
## 6        Minimum Size of Leaves    5.00000
## 7        Maximum Size of Leaves 3214.00000
## 8           Number of Variables   13.00000
## 9  Confidence Level for Pruning    0.25000
## 10  Number of Observations Used 4172.00000
## 11  Misclassification Error (%)   13.75839
##
## $OutputCasTables
##   casLib     Name Rows Columns
## 1 Public dt_model   15      27
```

Random Forest

```
# Train the random forest model
cas.decisionTree.forestTrain(conn,
    table    = list(name = 'hmeq_prepped', 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)    4.00000
## 3                Random Number Seed    0.00000
## 4          Bootstrap Percentage (%)   63.21206
## 5                    Number of Bins   20.00000
## 6               Number of Variables   13.00000
## 7      Confidence Level for Pruning    0.25000
## 8          Max Number of Tree Nodes   33.00000
## 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   17.00000
## 15             Min Number of Leaves    6.00000
## 16           Maximum Size of Leaves 2580.00000
## 17           Minimum Size of Leaves    5.00000
## 18               Out-of-Bag MCR (%)        NaN
```

```
##
## $OutputCasTables
##   casLib     Name Rows Columns
## 1 Public rf_model  852      41
```

Gradient Boosting

```
# Train the gradient boosting model
cas.decisionTree.gbtreeTrain(conn,
    table    = list(name = 'hmeq_prepped', 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)   13.0
## 6                    Number of Bins   20.0
## 7               Number of Variables   13.0
## 8          Max Number of Tree Nodes   61.0
## 9          Min Number of Tree Nodes   31.0
## 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   16.0
## 16           Maximum Size of Leaves 1736.0
## 17           Minimum Size of Leaves    5.0
## 18               Random Number Seed    0.0
##
## $OutputCasTables
##   casLib      Name Rows Columns
## 1 Public gbt_model 2470      31
```

Neural Network

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

# Build a neural network model
cas.neuralNet.annTrain(conn,
    table    = list(name = 'hmeq_prepped', 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           21
## 6        Number of Input Nodes           19
## 7       Number of Output Nodes            2
## 8       Number of Hidden Nodes            0
## 9  Number of Weight Parameters           19
## 10   Number of Bias Parameters            2
## 11                Architecture         GLIM
## 12       Number of Neural Nets            1
## 13             Objective Value 1.5687790628
##
## $OptIterHistory
##    Progress Objective     Loss
## 1         1  4.568050 4.568050
## 2         2  2.879156 2.879156
## 3         3  1.748308 1.748308
## 4         4  1.660549 1.660549
## 5         5  1.613690 1.613690
## 6         6  1.595178 1.595178
## 7         7  1.578170 1.578170
## 8         8  1.573189 1.573189
## 9         9  1.569972 1.569972
## 10       10  1.568779 1.568779
##
## $OutputCasTables
##   casLib     Name Rows Columns
## 1 Public nn_model   21      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, 'hmeq_prepped'),
    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 244 163  89 1292
## 2     Random Forest 125  13 208 1442
## 3 Gradient Boosting 234  63  99 1392
## 4    Neural Network 116  51 217 1404
```

Compare Misclassification

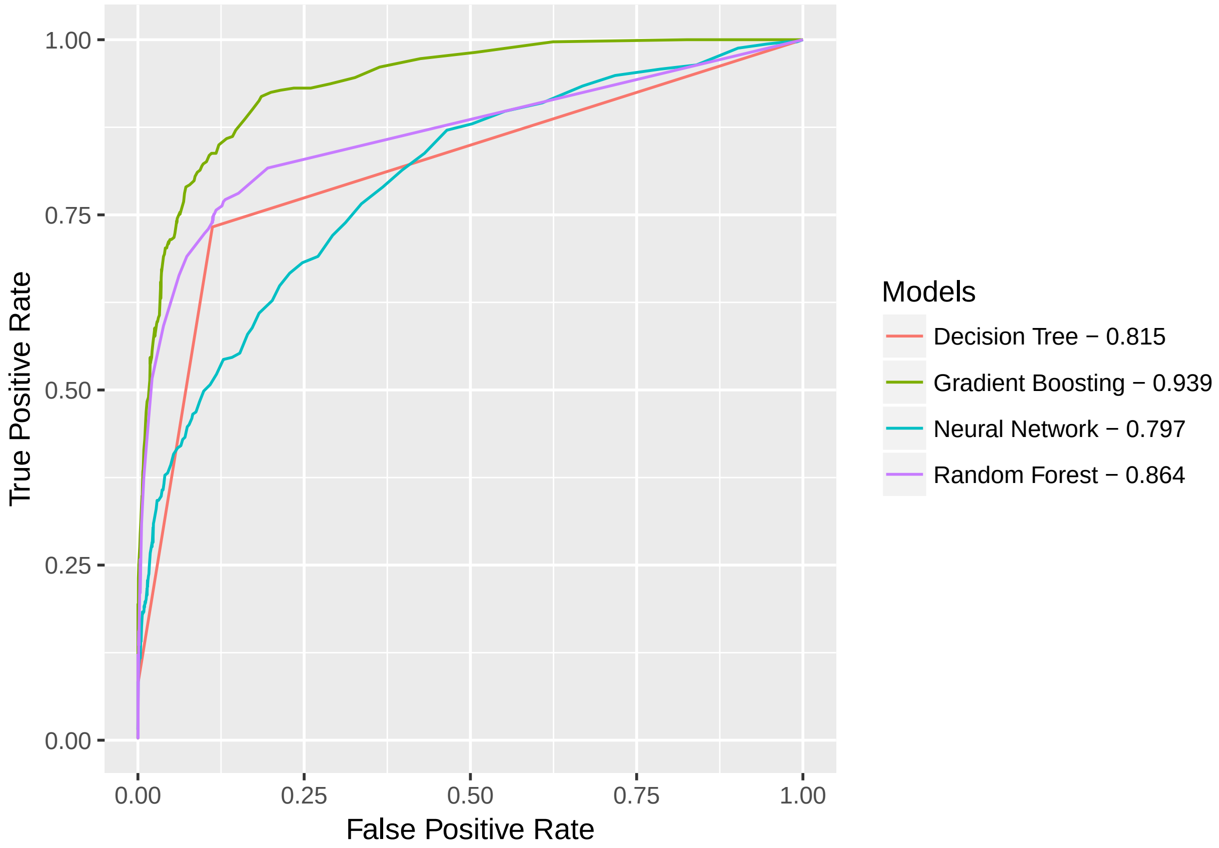
```
# Build a dataframe to compare the misclassification rates
compare$Misclassification <- 1 - compare$ACC
miss <- compare[order(compare$Misclassification), c('Model','Misclassification')]
rownames(miss) <- NULL
miss
```

```
##               Model Misclassification
## 1 Gradient Boosting        0.09060403
## 2     Random Forest        0.12360179
## 3     Decision Tree        0.14093960
## 4    Neural Network        0.14988814
```

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')
```



Compare XGBoost Model

```
library('xgboost')
suppressPackageStartupMessages(library('caret'))

# Bring data to R client
df <- to.casDataFrame(castbl, obs = nrow(castbl))
df <- df[,c(target, inputs, '_PartInd_')]
```

```r
# Create dummy variables through one-hot encoding
df.dum <- df[,nominals[-1]]
dummies <- dummyVars('~ .', data = df.dum)
df.ohe <- as.data.frame(predict(dummies, newdata = df))
df.all.combined <- cbind(df[,-c(which(colnames(df) %in% nominals[-1]))], df.ohe)

# Split into training and validation
train <- df.all.combined[df.all.combined['_PartInd_'] == 0,]
valid <- df.all.combined[df.all.combined['_PartInd_'] == 1,]

# Train the XGBoost model
bst <- xgboost(
    data = data.matrix(train[,-1]),
    label = data.matrix(train[,1]),
    objective = "binary:logistic",
    nround = 50,
    eta = 0.1,
    subsample = 0.5,
    colsample_bytree = 0.5
)
```

```
## [1]  train-error:0.107143
## [2]  train-error:0.100192
## [3]  train-error:0.098754
## [4]  train-error:0.100671
## [5]  train-error:0.095638
## [6]  train-error:0.093720
## [7]  train-error:0.089406
## [8]  train-error:0.090364
## [9]  train-error:0.087248
## [10] train-error:0.086769
## [11] train-error:0.086050
## [12] train-error:0.082694
## [13] train-error:0.083174
## [14] train-error:0.080297
## [15] train-error:0.079338
## [16] train-error:0.079818
## [17] train-error:0.079578
## [18] train-error:0.077900
## [19] train-error:0.074065
## [20] train-error:0.074305
## [21] train-error:0.073826
## [22] train-error:0.074065
## [23] train-error:0.073346
## [24] train-error:0.072627
## [25] train-error:0.072148
## [26] train-error:0.071668
## [27] train-error:0.070470
## [28] train-error:0.069271
## [29] train-error:0.068792
## [30] train-error:0.068073
## [31] train-error:0.068552
## [32] train-error:0.066395
## [33] train-error:0.065436
## [34] train-error:0.064477
## [35] train-error:0.063279
## [36] train-error:0.061361
## [37] train-error:0.061361
## [38] train-error:0.060163
## [39] train-error:0.059923
## [40] train-error:0.059444
## [41] train-error:0.058965
## [42] train-error:0.058245
## [43] train-error:0.056807
## [44] train-error:0.057287
## [45] train-error:0.057287
## [46] train-error:0.056568
## [47] train-error:0.054890
## [48] train-error:0.055129
## [49] train-error:0.054410
## [50] train-error:0.053931
```

Score and Assess XGBoost on Validation Data

```
# Create a dataframe with the misclassification rate for XGBoost
pred <- as.numeric(predict(bst, data.matrix(valid[,-1]), missing = 'NAN') > 0.5)
Misclassification <- mean(as.numeric(pred > 0.5) != valid[,1])
xgb <- data.frame(cbind(Model = 'R - XGBoost', Misclassification))
xgb
```

```
##         Model  Misclassification
## 1 R - XGBoost 0.0911633109619687
```

Final Assessment with CAS and R Models

```
# Combine the assessments and order by most accurate on validation data
err <- data.frame(rbind(miss, xgb))
err[,-1] <- round(as.numeric(as.character(err[,-1])),7)
err <- err[order(err[,-1]),]
rownames(err) <- NULL
err
```

```
##               Model Misclassification
## 1 Gradient Boosting         0.0906040
## 2       R - XGBoost         0.0911633
## 3     Random Forest         0.1236018
## 4     Decision Tree         0.1409396
## 5    Neural Network         0.1498881
```

Save the CAS Gradient Boosting Model

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

```
## $caslib
## [1] "Public"
##
## $name
## [1] "Jesse_SAS_gbt.sashdat"
```

```
# Promote the champion model to public memory to share with team
cas.table.promote(conn, name = 'gbt_model', target = 'Jesse_SAS_gbt', targetLib = 'public')
```

```
## list()
```

```
# Save the challenger (XGBoost) model for later use
xgb.save(bst, "Jesse_R_xgb.model")
```

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

End the Session

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