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

# Rattle is Copyright (c) 2006-2018 Togaware Pty Ltd.
# It is free (as in libre) open source software.
# It is licensed under the GNU General Public License,
# Version 2. Rattle comes with ABSOLUTELY NO WARRANTY.
# Rattle was written by Graham Williams with contributions
# from others as acknowledged in 'library(help=rattle)'.
# Visit https://rattle.togaware.com/ for details.

#============================================================================
# Rattle timestamp: 2019-08-10 11:32:24 x86_64-w64-mingw32

# Rattle version 5.2.0 user 'bassa'

# This log captures interactions with Rattle as an R script.

# For repeatability, export this activity log to a
# file, like 'model.R' using the Export button or
# through the Tools menu. Th script can then serve as a
# starting point for developing your own scripts.
# After xporting to a file called 'model.R', for exmample,
# you can type into a new R Console the command
# "source('model.R')" and so repeat all actions. Generally,
# you will want to edit the file to suit your own needs.
# You can also edit this log in place to record additional
# information before exporting the script.

# Note that saving/loading projects retains this log.

# We begin most scripts by loading the required packages.
# Here are some initial packages to load and others will be
# identified as we proceed through the script. When writing
# our own scripts we often collect together the library
# commands at the beginning of the script here.

library(rattle)   # Access the weather dataset and utilities.
library(magrittr) # Utilise %>% and %<>% pipeline operators.

# This log generally records the process of building a model.
# However, with very little effort the log can also be used
# to score a new dataset. The logical variable 'building'
# is used to toggle between generating transformations,
# when building a model and using the transformations,
# when scoring a dataset.

building <- TRUE
scoring  <- ! building

# A pre-defined value is used to reset the random seed
```

```
# so that results are repeatable.

crv$seed <- 42

#=======================================================================
# Rattle timestamp: 2019-08-10 11:32:52 x86_64-w64-mingw32

# Load an R data frame.

crs$dataset <- SnP100full

# Display a simple summary (structure) of the dataset.

str(crs$dataset)

#=======================================================================
# Rattle timestamp: 2019-08-10 11:34:35 x86_64-w64-mingw32

# Action the user selections from the Data tab.

# Build the train/validate/test datasets.

# nobs=303361 train=212353 validate=45504 test=45504

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("open", "High", "Low", "Close", "Volume",
                   "Adjusted", "Volatility", "Symbol", "ON_PPC",
                   "HLppc")

crs$numeric   <- c("open", "High", "Low", "Close", "Volume",
                   "Adjusted", "Volatility", "ON_PPC", "HLppc")
```

```
crs$categoric <- "Symbol"

crs$target    <- "Dividends"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- NULL
crs$weights   <- NULL

#=======================================================================
# Rattle timestamp: 2019-08-10 11:36:15 x86_64-w64-mingw32

# Action the user selections from the Data tab.

# Build the train/validate/test datasets.

# nobs=303361 train=212353 validate=45504 test=45504

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

crs$train <- sample(crs$nobs, 0.7*crs$nobs)

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  sample(0.15*crs$nobs) ->
crs$validate

crs$nobs %>%
  seq_len() %>%
  setdiff(crs$train) %>%
  setdiff(crs$validate) ->
crs$test

# The following variable selections have been noted.

crs$input     <- c("open", "High", "Low", "Close", "Volume",
                   "Adjusted", "Volatility", "Symbol", "Dividends",
                   "ON_PPC")

crs$numeric   <- c("open", "High", "Low", "Close", "Volume",
                   "Adjusted", "Volatility", "Dividends", "ON_PPC")

crs$categoric <- "Symbol"

crs$target    <- "HLppc"
crs$risk      <- NULL
crs$ident     <- NULL
```

```
crs$ignore    <- NULL
crs$weights   <- NULL

#=======================================================================
# Rattle timestamp: 2019-08-10 11:38:39 x86_64-w64-mingw32

# Decision Tree

# The 'rpart' package provides the 'rpart' function.

library(rpart, quietly=TRUE)

# Reset the random number seed to obtain the same results each time.

set.seed(crv$seed)

# Build the Decision Tree model.

crs$rpart <- rpart(HLppc ~ .,
    data=crs$dataset[crs$train, c(crs$input, crs$target)],
    method="anova",
    parms=list(split="information"),
    control=rpart.control(usesurrogate=0,
        maxsurrogate=0),
    model=TRUE)

# Generate a textual view of the Decision Tree model.

print(crs$rpart)
printcp(crs$rpart)
cat("\n")

# Time taken: 7.08 secs

#=======================================================================
# Rattle timestamp: 2019-08-10 11:46:00 x86_64-w64-mingw32

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree SnP100full $ HLppc")

#=======================================================================
# Rattle timestamp: 2019-08-10 11:48:34 x86_64-w64-mingw32

# Plot the resulting Decision Tree.

# We use the rpart.plot package.
```

```
fancyRpartPlot(crs$rpart, main="Decision Tree SnP100full $ HLppc")

#=======================================================================
# Rattle timestamp: 2019-08-10 12:02:59 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input,
crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$validate, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
    crs$dataset[crs$validate, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Decision Tree SnP100full [validate] ",
show.lift=FALSE, show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 12:30:57 x86_64-w64-mingw32

# Decision Tree

# The 'rpart' package provides the 'rpart' function.

library(rpart, quietly=TRUE)

# Reset the random number seed to obtain the same results each time.

set.seed(crv$seed)

# Build the Decision Tree model.

crs$rpart <- rpart(HLppc ~ .,
    data=crs$dataset[crs$train, c(crs$input, crs$target)],
    method="anova",
    parms=list(split="information"),
      control=rpart.control(minsplit=10,
           minbucket=4,
        usesurrogate=0,
        maxsurrogate=0),
    model=TRUE)
```

```
# Generate a textual view of the Decision Tree model.

print(crs$rpart)
printcp(crs$rpart)
cat("\n")

# Time taken: 4.83 secs

#=======================================================================
# Rattle timestamp: 2019-08-10 12:31:31 x86_64-w64-mingw32

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree SnP100full $ HLppc")

#=======================================================================
# Rattle timestamp: 2019-08-10 12:35:33 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input,
crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$validate, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
    crs$dataset[crs$validate, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Decision Tree SnP100full [validate] ",
show.lift=FALSE, show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 12:37:17 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# RPART: Generate a Predicted v Observed plot for rpart model on SnP100full
[validate].
```

```
crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input,
crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$validate, c(crs$input, crs$target)],
select=crs$target)

# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))

# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)

# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")

# Add a title and grid to the plot.
```

```
title(main="Predicted vs. Observed
 Decision Tree Model
 SnP100full [validate]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

#=======================================================================
# Rattle timestamp: 2019-08-10 12:39:36 x86_64-w64-mingw32

# Score the validation dataset.

# Obtain predictions for the Decision Tree model on SnP100full [validate].

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input)])

# Extract the relevant variables from the dataset.

sdata <- subset(crs$dataset[crs$validate,], select=c("HLppc"))

# Output the combined data.

write.csv(cbind(sdata, crs$pr),
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full_validate_score_identsDT.csv", row.names=FALSE)

#=======================================================================
# Rattle timestamp: 2019-08-10 12:40:25 x86_64-w64-mingw32

# Score the testing dataset.

#=======================================================================
# Rattle timestamp: 2019-08-10 12:40:36 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$test, c(crs$input,
crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$test, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
```

```
    crs$dataset[crs$test, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Decision Tree SnP100full [test] ", show.lift=FALSE,
show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 12:42:38 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.

# RPART: Generate a Predicted v Observed plot for rpart model on SnP100full [test].

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$test, c(crs$input,
crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$test, c(crs$input, crs$target)], select=crs$target)

# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))

# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)
```

```
# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")

# Add a title and grid to the plot.

title(main="Predicted vs. Observed
 Decision Tree Model
 SnP100full [test]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

#=======================================================================
# Rattle timestamp: 2019-08-10 12:47:53 x86_64-w64-mingw32

# Save the project data (variable crs) to file.

save(crs,
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full.rattlev3.rattle", compress=TRUE)

#=======================================================================
# Rattle timestamp: 2019-08-10 12:54:31 x86_64-w64-mingw32

# Build a Random Forest model using conditional inference trees.

set.seed(crv$seed)

crs$rf <- party::cforest(HLppc ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  controls=party::cforest_unbiased(
    ntree=500,
    mtry=3))

#=======================================================================
# Rattle timestamp: 2019-08-10 12:54:53 x86_64-w64-mingw32

# Neural Network

# Build a neural network model using the nnet package.

library(nnet, quietly=TRUE)

# Build the NNet model.
```

```
set.seed(199)
crs$nnet <- nnet(HLppc ~ .,
    data=crs$dataset[crs$train,c(crs$input, crs$target)],
    size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)

# Print the results of the modelling.

cat(sprintf("A %s network with %d weights.\n",
    paste(crs$nnet$n, collapse="-"),
    length(crs$nnet$wts)))
cat(sprintf("Inputs: %s.\n",
    paste(crs$nnet$coefnames, collapse=", ")))
cat(sprintf("Output: %s.\n",
    names(attr(crs$nnet$terms, "dataClasses"))[1]))
cat(sprintf("Sum of Squares Residuals: %.4f.\n",
    sum(residuals(crs$nnet) ^ 2)))
cat("\n")
print(summary(crs$nnet))
cat('\n')

# Time taken: 8.06 mins

#=======================================================================
# Rattle timestamp: 2019-08-10 13:08:08 x86_64-w64-mingw32

# Save the project data (variable crs) to file.

save(crs,
file="C:/Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full.rattlev3.rattle", compress=TRUE)

#=======================================================================
# Rattle timestamp: 2019-08-10 13:09:00 x86_64-w64-mingw32

# Neural Network

# Build a neural network model using the nnet package.

library(nnet, quietly=TRUE)

# Build the NNet model.

set.seed(199)
crs$nnet <- nnet(HLppc ~ .,
    data=crs$dataset[crs$train,c(crs$input, crs$target)],
    size=4, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)

# Print the results of the modelling.
```

```
cat(sprintf("A %s network with %d weights.\n",
    paste(crs$nnet$n, collapse="-"),
    length(crs$nnet$wts)))
cat(sprintf("Inputs: %s.\n",
    paste(crs$nnet$coefnames, collapse=", ")))
cat(sprintf("Output: %s.\n",
    names(attr(crs$nnet$terms, "dataClasses"))[1]))
cat(sprintf("Sum of Squares Residuals: %.4f.\n",
    sum(residuals(crs$nnet) ^ 2)))
cat("\n")
print(summary(crs$nnet))
cat('\n')

# Time taken: 3.36 mins

#=======================================================================
# Rattle timestamp: 2019-08-10 13:52:01 x86_64-w64-mingw32

# Save the project data (variable crs) to file.

save(crs,
file="C:/Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full.rattlev3.rattle", compress=TRUE)

#=======================================================================
# Rattle timestamp: 2019-08-10 13:52:46 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# NNET: Generate a Predicted v Observed plot for nnet model on SnP100full
[validate].

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$validate, c(crs$input,
crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$validate, c(crs$input, crs$target)],
select=crs$target)

# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))
```

```
# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)

# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")

# Add a title and grid to the plot.

title(main="Predicted vs. Observed
 Neural Net Model
 SnP100full [validate]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

#=======================================================================
# Rattle timestamp: 2019-08-10 14:00:40 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.
```

```
# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$validate, c(crs$input,
crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$validate, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
    crs$dataset[crs$validate, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Neural Net SnP100full [validate] ", show.lift=FALSE,
show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 14:03:33 x86_64-w64-mingw32

# Score the validation dataset.

# Obtain predictions for the Neural Net model on SnP100full [validate].

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$validate, c(crs$input)])

# Extract the relevant variables from the dataset.

sdata <- subset(crs$dataset[crs$validate,], select=c("HLppc"))

# Output the combined data.

write.csv(cbind(sdata, crs$pr),
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full_validate_score_identsNN.csv", row.names=FALSE)

#=======================================================================
# Rattle timestamp: 2019-08-10 14:08:58 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$test, c(crs$input,
crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$test, c(crs$input,
crs$target)]$HLppc)
```

```
print(riskchart(crs$pr,
    crs$dataset[crs$test, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Neural Net SnP100full [test] ", show.lift=FALSE,
show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 14:12:09 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.

# NNET: Generate a Predicted v Observed plot for nnet model on SnP100full [test].

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$test, c(crs$input,
crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$test, c(crs$input, crs$target)], select=crs$target)

# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))

# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)
```

```
# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")

# Add a title and grid to the plot.

title(main="Predicted vs. Observed
 Neural Net Model
 SnP100full [test]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

#=======================================================================
# Rattle timestamp: 2019-08-10 14:14:34 x86_64-w64-mingw32

# Score the testing dataset.

# Obtain predictions for the Neural Net model on SnP100full [test].

crs$pr <- predict(crs$nnet, newdata=crs$dataset[crs$test, c(crs$input)])

# Extract the relevant variables from the dataset.

sdata <- subset(crs$dataset[crs$test,], select=c("HLppc"))

# Output the combined data.

write.csv(cbind(sdata, crs$pr),
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full_test_score_identsNN.csv", row.names=FALSE)

#=======================================================================
# Rattle timestamp: 2019-08-10 14:16:02 x86_64-w64-mingw32

# Regression model

# Build a Regression model.

crs$glm <- lm(HLppc ~ ., data=crs$dataset[crs$train,c(crs$input, crs$target)])

# Generate a textual view of the Linear model.

print(summary(crs$glm))
```

```
cat('==== ANOVA ====

')
print(anova(crs$glm))
print("
")

# Time taken: 3.05 secs

#=======================================================================
# Rattle timestamp: 2019-08-10 14:19:56 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$validate, c(crs$input, crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$validate, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
    crs$dataset[crs$validate, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Linear SnP100full [validate] ", show.lift=FALSE,
show.precision=FALSE, legend.horiz=FALSE))


#=======================================================================
# Rattle timestamp: 2019-08-10 15:36:43 x86_64-w64-mingw32

# Evaluate model performance on the validation dataset.

# GLM: Generate a Predicted v Observed plot for glm model on SnP100full [validate].

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$validate, c(crs$input, crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$validate, c(crs$input, crs$target)],
select=crs$target)
```

```
# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))

# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)

# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")

# Add a title and grid to the plot.

title(main="Predicted vs. Observed
 Linear Model
 SnP100full [validate]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

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

```
# Rattle timestamp: 2019-08-10 15:41:03 x86_64-w64-mingw32

# Score the validation dataset.

# Obtain predictions for the Linear model on SnP100full [validate].

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$validate, c(crs$input)])

# Extract the relevant variables from the dataset.

sdata <- subset(crs$dataset[crs$validate,], select=c("HLppc"))

# Output the combined data.

write.csv(cbind(sdata, crs$pr),
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full_validate_score_identsLN.csv", row.names=FALSE)

#=====================================================================
# Rattle timestamp: 2019-08-10 15:41:50 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.

# Risk Chart: requires the ggplot2 package.

library(ggplot2)

# Generate a risk chart.

# Rattle provides evaluateRisk() and riskchart().

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$test, c(crs$input, crs$target)])

crs$eval <- evaluateRisk(crs$pr, crs$dataset[crs$test, c(crs$input,
crs$target)]$HLppc)
print(riskchart(crs$pr,
    crs$dataset[crs$test, c(crs$input, crs$target)]$HLppc,
    title="Performance Chart Linear SnP100full [test] ", show.lift=FALSE,
show.precision=FALSE, legend.horiz=FALSE))


#=====================================================================
# Rattle timestamp: 2019-08-10 15:43:47 x86_64-w64-mingw32

# Evaluate model performance on the testing dataset.
```

```
# GLM: Generate a Predicted v Observed plot for glm model on SnP100full [test].

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$test, c(crs$input, crs$target)])

# Obtain the observed output for the dataset.

obs <- subset(crs$dataset[crs$test, c(crs$input, crs$target)], select=crs$target)

# Handle in case categoric target treated as numeric.

obs.rownames <- rownames(obs)
obs <- as.numeric(obs[[1]])
obs <- data.frame(HLppc=obs)
rownames(obs) <- obs.rownames

# Combine the observed values with the predicted.

fitpoints <- na.omit(cbind(obs, Predicted=crs$pr))

# Obtain the pseudo R2 - a correlation.

fitcorr <- format(cor(fitpoints[,1], fitpoints[,2])^2, digits=4)

# Plot settings for the true points and best fit.

op <- par(c(lty="solid", col="blue"))

# Display the observed (X) versus predicted (Y) points.

plot(fitpoints[[1]], fitpoints[[2]], asp=1, xlab="HLppc", ylab="Predicted")

# Generate a simple linear fit between predicted and observed.

prline <- lm(fitpoints[,2] ~ fitpoints[,1])

# Add the linear fit to the plot.

abline(prline)

# Add a diagonal representing perfect correlation.

par(c(lty="dashed", col="black"))
abline(0, 1)

# Include a pseudo R-square on the plot

legend("bottomright",  sprintf(" Pseudo R-square=%s ", fitcorr),  bty="n")
```

```
# Add a title and grid to the plot.

title(main="Predicted vs. Observed
 Linear Model
 SnP100full [test]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
Sys.info()["user"]))
grid()

#=======================================================================
# Rattle timestamp: 2019-08-10 15:45:28 x86_64-w64-mingw32

# Score the testing dataset.

# Obtain predictions for the Linear model on SnP100full [test].

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[crs$test, c(crs$input)])

# Extract the relevant variables from the dataset.

sdata <- subset(crs$dataset[crs$test,], select=c("HLppc"))

# Output the combined data.

write.csv(cbind(sdata, crs$pr),
file="C:\Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full_test_score_identsLN.csv", row.names=FALSE)

#=======================================================================
# Rattle timestamp: 2019-08-10 15:46:11 x86_64-w64-mingw32

# Save the project data (variable crs) to file.

save(crs,
file="C:/Users\bassa\OneDrive\Files\GitHub\CSDA-1050F18S1\BassamRizk_303525\sprint
2\rattle\SnP100full.rattlev3.rattle", compress=TRUE)
```