#============================================================

# Rattle is Copyright (c) 2006-2017 Togaware Pty Ltd.

# It is open source software and is freely available.

# 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: 2018-05-01 15:31:12 x86\_64-apple-darwin15.6.0

# Rattle version 5.1.0 user 'tpm'

# This log captures Rattle interactions as an R script.

# For repeatability export this log of all activity to a

# file using the Export button or the Tools menu. This

# script can serve as a starting point for developing your

# own scripts. Exporting to a file called 'model.R' will

# allow you to 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 weather dataset and utilities.

library(magrittr) # For the %>% 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: 2018-05-01 15:31:36 x86\_64-apple-darwin15.6.0

# Load the dataset from file.

fname <- "file:///Users/tpm/Desktop/BDA/Project/bank.csv"

crs$dataset <- read.csv(fname,

na.strings=c(".", "NA", "", "?"),

strip.white=TRUE, encoding="UTF-8")

#============================================================

# Rattle timestamp: 2018-05-01 15:31:37 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("age", "job", "marital", "education", "default",

"housing", "loan", "contact", "month", "day\_of\_week",

"duration", "campaign", "pdays", "previous", "poutcome",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$numeric <- c("age", "duration", "campaign", "pdays", "previous",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$categoric <- c("job", "marital", "education", "default", "housing",

"loan", "contact", "month", "day\_of\_week", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- NULL

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 15:33:15 x86\_64-apple-darwin15.6.0

# CLEANUP the Dataset

#============================================================

# Rattle timestamp: 2018-05-01 15:33:23 x86\_64-apple-darwin15.6.0

# CLEANUP the Dataset

#============================================================

# Rattle timestamp: 2018-05-01 15:34:14 x86\_64-apple-darwin15.6.0

# Display histogram plots for the selected variables.

# Use ggplot2 to generate histogram plot for age

# Generate the plot.

p01 <- crs %>%

with(dataset[sample,]) %>%

dplyr::select(age) %>%

ggplot2::ggplot(ggplot2::aes(x=age)) +

ggplot2::geom\_density(lty=3) +

ggplot2::xlab("age\n\nRattle 2018-May-01 15:34:14 tpm") +

ggplot2::ggtitle("Distribution of age (sample)") +

ggplot2::labs(y="Density")

# Display the plots.

gridExtra::grid.arrange(p01)

#============================================================

# Rattle timestamp: 2018-05-01 15:34:51 x86\_64-apple-darwin15.6.0

# Display box plots for the selected variables.

# Use ggplot2 to generate box plot for age

# Generate a box plot.

p01 <- crs %>%

with(dataset[sample,]) %>%

ggplot2::ggplot(ggplot2::aes(y=age)) +

ggplot2::geom\_boxplot(ggplot2::aes(x="All"), notch=TRUE, fill="grey") +

ggplot2::stat\_summary(ggplot2::aes(x="All"), fun.y=mean, geom="point", shape=8) +

ggplot2::xlab("Rattle 2018-May-01 15:34:51 tpm") +

ggplot2::ggtitle("Distribution of age (sample)") +

ggplot2::theme(legend.position="none")

# Display the plots.

gridExtra::grid.arrange(p01)

#============================================================

# Rattle timestamp: 2018-05-01 15:35:13 x86\_64-apple-darwin15.6.0

# The 'gplots' package provides the 'barplot2' function.

library(gplots, quietly=TRUE)

#============================================================

# Rattle timestamp: 2018-05-01 15:35:13 x86\_64-apple-darwin15.6.0

# Bar Plot

# Generate the summary data for plotting.

ds <- rbind(summary(na.omit(crs$dataset[crs$sample,]$job)))

# Sort the entries.

ord <- order(ds[1,], decreasing=TRUE)

# Plot the data.

bp <- barplot2(ds[,ord], beside=TRUE, ylab="Frequency", xlab="job", ylim=c(0, 8738), col=colorspace::rainbow\_hcl(1))

# Add the actual frequencies.

text(bp, ds[,ord]+291, ds[,ord])

# Add a title to the plot.

title(main="Distribution of job (sample)",

sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))

#============================================================

# Rattle timestamp: 2018-05-01 15:46:13 x86\_64-apple-darwin15.6.0

# Generate a correlation plot for the variables.

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

library(corrplot, quietly=TRUE)

# Correlations work for numeric variables only.

crs$cor <- cor(crs$dataset[crs$sample, crs$numeric], use="pairwise", method="pearson")

# Order the correlations by their strength.

crs$ord <- order(crs$cor[1,])

crs$cor <- crs$cor[crs$ord, crs$ord]

# Display the actual correlations.

print(crs$cor)

# Graphically display the correlations.

corrplot(crs$cor, mar=c(0,0,1,0))

title(main="Correlation bank.csv using Pearson",

sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))

#============================================================

# Rattle timestamp: 2018-05-01 15:47:17 x86\_64-apple-darwin15.6.0

# Hierarchical Variable Correlation

# Generate the correlations (numerics only).

cc <- cor(crs$dataset[crs$sample, crs$numeric], use="pairwise", method="pearson")

# Generate hierarchical cluster of variables.

hc <- hclust(dist(cc), method="average")

# Generate the dendrogram.

dn <- as.dendrogram(hc)

# Now draw the dendrogram.

op <- par(mar = c(3, 4, 3, 4.57))

plot(dn, horiz = TRUE, nodePar = list(col = 3:2, cex = c(2.0, 0.75), pch = 21:22, bg= c("light blue", "pink"), lab.cex = 0.75, lab.col = "tomato"), edgePar = list(col = "gray", lwd = 2), xlab="Height")

title(main="Variable Correlation Clusters

bank.csv using Pearson",

sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))

par(op)

#============================================================

# Rattle timestamp: 2018-05-01 16:12:33 x86\_64-apple-darwin15.6.0

# Principal Components Analysis (on numerics only).

pc <- prcomp(na.omit(crs$dataset[crs$sample, crs$numeric]), scale=TRUE, center=TRUE, tol=0)

# Show the output of the analysis.

pc

# Summarise the importance of the components found.

summary(pc)

# Display a plot showing the relative importance of the components.

plot(pc, main="")

title(main="Principal Components Importance bank.csv",

sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))

axis(1, at=seq(0.7, ncol(pc$rotation)\*1.2, 1.2), labels=colnames(pc$rotation), lty=0)

# Display a plot showing the two most principal components.

biplot(pc, main="")

title(main="Principal Components bank.csv",

sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))

#============================================================

# Rattle timestamp: 2018-05-01 17:41:33 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("job", "default", "loan", "duration", "campaign",

"pdays", "previous", "poutcome", "emp.var.rate",

"cons.price.idx", "cons.conf.idx", "euribor3m",

"nr.employed")

crs$numeric <- c("duration", "campaign", "pdays", "previous",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$categoric <- c("job", "default", "loan", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- c("age", "marital", "education", "housing", "contact", "month", "day\_of\_week")

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 17:41:48 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("job", "default", "loan", "campaign", "pdays",

"previous", "poutcome", "emp.var.rate", "cons.price.idx",

"cons.conf.idx", "euribor3m", "nr.employed")

crs$numeric <- c("campaign", "pdays", "previous", "emp.var.rate",

"cons.price.idx", "cons.conf.idx", "euribor3m",

"nr.employed")

crs$categoric <- c("job", "default", "loan", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- c("age", "marital", "education", "housing", "contact", "month", "day\_of\_week", "duration")

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 17:42:21 x86\_64-apple-darwin15.6.0

# 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(y ~ .,

data=crs$dataset[crs$train, c(crs$input, crs$target)],

method="class",

parms=list(split="information"),

control=rpart.control(usesurrogate=0,

maxsurrogate=0))

# Generate a textual view of the Decision Tree model.

print(crs$rpart)

printcp(crs$rpart)

cat("\n")

# Time taken: 0.56 secs

#============================================================

# Rattle timestamp: 2018-05-01 17:42:33 x86\_64-apple-darwin15.6.0

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree bank.csv $ y")

#============================================================

# Rattle timestamp: 2018-05-01 17:43:09 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# Generate an Error Matrix for the Decision Tree model.

# Obtain the response from the Decision Tree model.

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

type="class")

# Generate the confusion matrix showing counts.

rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr, count=TRUE)

# Generate the confusion matrix showing proportions.

(per <- rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr))

# Calculate the overall error percentage.

cat(100-sum(diag(per), na.rm=TRUE))

# Calculate the averaged class error percentage.

cat(mean(per[,"Error"], na.rm=TRUE))

#============================================================

# Rattle timestamp: 2018-05-01 17:43:49 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("age", "job", "marital", "education", "default",

"housing", "loan", "contact", "month", "day\_of\_week",

"duration", "campaign", "pdays", "previous", "poutcome",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$numeric <- c("age", "duration", "campaign", "pdays", "previous",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$categoric <- c("job", "marital", "education", "default", "housing",

"loan", "contact", "month", "day\_of\_week", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- NULL

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 17:44:12 x86\_64-apple-darwin15.6.0

# 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(y ~ .,

data=crs$dataset[crs$train, c(crs$input, crs$target)],

method="class",

parms=list(split="information"),

control=rpart.control(usesurrogate=0,

maxsurrogate=0))

# Generate a textual view of the Decision Tree model.

print(crs$rpart)

printcp(crs$rpart)

cat("\n")

# Time taken: 0.57 secs

#============================================================

# Rattle timestamp: 2018-05-01 17:44:23 x86\_64-apple-darwin15.6.0

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree bank.csv $ y")

#============================================================

# Rattle timestamp: 2018-05-01 17:47:12 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# Generate an Error Matrix for the Decision Tree model.

# Obtain the response from the Decision Tree model.

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

type="class")

# Generate the confusion matrix showing counts.

rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr, count=TRUE)

# Generate the confusion matrix showing proportions.

(per <- rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr))

# Calculate the overall error percentage.

cat(100-sum(diag(per), na.rm=TRUE))

# Calculate the averaged class error percentage.

cat(mean(per[,"Error"], na.rm=TRUE))

#============================================================

# Rattle timestamp: 2018-05-01 17:53:16 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("default", "loan", "duration", "campaign", "pdays",

"previous", "poutcome", "emp.var.rate", "cons.price.idx",

"cons.conf.idx", "euribor3m", "nr.employed")

crs$numeric <- c("duration", "campaign", "pdays", "previous",

"emp.var.rate", "cons.price.idx", "cons.conf.idx",

"euribor3m", "nr.employed")

crs$categoric <- c("default", "loan", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- c("age", "job", "marital", "education", "housing", "contact", "month", "day\_of\_week")

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 17:53:23 x86\_64-apple-darwin15.6.0

# 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(y ~ .,

data=crs$dataset[crs$train, c(crs$input, crs$target)],

method="class",

parms=list(split="information"),

control=rpart.control(usesurrogate=0,

maxsurrogate=0))

# Generate a textual view of the Decision Tree model.

print(crs$rpart)

printcp(crs$rpart)

cat("\n")

# Time taken: 0.41 secs

#============================================================

# Rattle timestamp: 2018-05-01 17:53:28 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# Generate an Error Matrix for the Decision Tree model.

# Obtain the response from the Decision Tree model.

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

type="class")

# Generate the confusion matrix showing counts.

rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr, count=TRUE)

# Generate the confusion matrix showing proportions.

(per <- rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr))

# Calculate the overall error percentage.

cat(100-sum(diag(per), na.rm=TRUE))

# Calculate the averaged class error percentage.

cat(mean(per[,"Error"], na.rm=TRUE))

#============================================================

# Rattle timestamp: 2018-05-01 17:53:54 x86\_64-apple-darwin15.6.0

# Regression model

# Build a Regression model.

crs$glm <- glm(y ~ .,

data=crs$dataset[crs$train, c(crs$input, crs$target)],

family=binomial(link="logit"))

# Generate a textual view of the Linear model.

print(summary(crs$glm))

cat(sprintf("Log likelihood: %.3f (%d df)\n",

logLik(crs$glm)[1],

attr(logLik(crs$glm), "df")))

cat(sprintf("Null/Residual deviance difference: %.3f (%d df)\n",

crs$glm$null.deviance-crs$glm$deviance,

crs$glm$df.null-crs$glm$df.residual))

cat(sprintf("Chi-square p-value: %.8f\n",

dchisq(crs$glm$null.deviance-crs$glm$deviance,

crs$glm$df.null-crs$glm$df.residual)))

cat(sprintf("Pseudo R-Square (optimistic): %.8f\n",

cor(crs$glm$y, crs$glm$fitted.values)))

cat('\n==== ANOVA ====\n\n')

print(anova(crs$glm, test="Chisq"))

cat("\n")

# Time taken: 2.66 secs

# Plot the model evaluation.

ttl <- genPlotTitleCmd("Linear Model",crs$dataname,vector=TRUE)

plot(crs$glm, main=ttl[1])

#============================================================

# Rattle timestamp: 2018-05-01 17:54:47 x86\_64-apple-darwin15.6.0

# Note the user selections.

# Build the train/validate/test datasets.

# nobs=41188 train=28831 validate=6178 test=6179

set.seed(crv$seed)

crs$nobs <- nrow(crs$dataset)

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

crs$validate <- sample(setdiff(seq\_len(crs$nobs), crs$train), 0.15\*crs$nobs)

crs$test <- setdiff(setdiff(seq\_len(crs$nobs), crs$train), crs$validate)

# The following variable selections have been noted.

crs$input <- c("default", "loan", "campaign", "pdays", "previous",

"poutcome", "emp.var.rate", "cons.price.idx",

"cons.conf.idx", "euribor3m", "nr.employed")

crs$numeric <- c("campaign", "pdays", "previous", "emp.var.rate",

"cons.price.idx", "cons.conf.idx", "euribor3m",

"nr.employed")

crs$categoric <- c("default", "loan", "poutcome")

crs$target <- "y"

crs$risk <- NULL

crs$ident <- NULL

crs$ignore <- c("age", "job", "marital", "education", "housing", "contact", "month", "day\_of\_week", "duration")

crs$weights <- NULL

#============================================================

# Rattle timestamp: 2018-05-01 17:54:52 x86\_64-apple-darwin15.6.0

# 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(y ~ .,

data=crs$dataset[crs$train, c(crs$input, crs$target)],

method="class",

parms=list(split="information"),

control=rpart.control(usesurrogate=0,

maxsurrogate=0))

# Generate a textual view of the Decision Tree model.

print(crs$rpart)

printcp(crs$rpart)

cat("\n")

# Time taken: 0.50 secs

#============================================================

# Rattle timestamp: 2018-05-01 17:54:55 x86\_64-apple-darwin15.6.0

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree bank.csv $ y")

#============================================================

# Rattle timestamp: 2018-05-01 17:55:35 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# Generate an Error Matrix for the Decision Tree model.

# Obtain the response from the Decision Tree model.

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

type="class")

# Generate the confusion matrix showing counts.

rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr, count=TRUE)

# Generate the confusion matrix showing proportions.

(per <- rattle::errorMatrix(crs$dataset[crs$validate, c(crs$input, crs$target)]$y, crs$pr))

# Calculate the overall error percentage.

cat(100-sum(diag(per), na.rm=TRUE))

# Calculate the averaged class error percentage.

cat(mean(per[,"Error"], na.rm=TRUE))

#============================================================

# Rattle timestamp: 2018-05-01 17:55:56 x86\_64-apple-darwin15.6.0

# Plot the resulting Decision Tree.

# We use the rpart.plot package.

fancyRpartPlot(crs$rpart, main="Decision Tree bank.csv $ y")

#============================================================

# Rattle timestamp: 2018-05-01 17:57:32 x86\_64-apple-darwin15.6.0

# Support vector machine.

# The 'kernlab' package provides the 'ksvm' function.

library(kernlab, quietly=TRUE)

# Build a Support Vector Machine model.

set.seed(crv$seed)

crs$ksvm <- ksvm(as.factor(y) ~ .,

data=crs$dataset[crs$train,c(crs$input, crs$target)],

kernel="rbfdot",

prob.model=TRUE)

# Generate a textual view of the SVM model.

crs$ksvm

# Time taken: 54.97 secs

#============================================================

# Rattle timestamp: 2018-05-01 18:10:54 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# Generate an Error Matrix for the SVM model.

# Obtain the response from the SVM model.

crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]))

# Generate the confusion matrix showing counts.

rattle::errorMatrix(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$y, crs$pr, count=TRUE)

# Generate the confusion matrix showing proportions.

(per <- rattle::errorMatrix(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$y, crs$pr))

# Calculate the overall error percentage.

cat(100-sum(diag(per), na.rm=TRUE))

# Calculate the averaged class error percentage.

cat(mean(per[,"Error"], na.rm=TRUE))

#============================================================

# Rattle timestamp: 2018-05-01 18:11:14 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# ROC Curve: requires the ROCR package.

library(ROCR)

# ROC Curve: requires the ggplot2 package.

library(ggplot2, quietly=TRUE)

# Generate an ROC Curve for the ksvm model on bank.csv [validate].

crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]),

type = "probabilities")[,2]

# Remove observations with missing target.

no.miss <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$y)

miss.list <- attr(no.miss, "na.action")

attributes(no.miss) <- NULL

if (length(miss.list))

{

pred <- prediction(crs$pr[-miss.list], no.miss)

} else

{

pred <- prediction(crs$pr, no.miss)

}

pe <- performance(pred, "tpr", "fpr")

au <- performance(pred, "auc")@y.values[[1]]

pd <- data.frame(fpr=unlist(pe@x.values), tpr=unlist(pe@y.values))

p <- ggplot(pd, aes(x=fpr, y=tpr))

p <- p + geom\_line(colour="red")

p <- p + xlab("False Positive Rate") + ylab("True Positive Rate")

p <- p + ggtitle("ROC Curve SVM bank.csv [validate] y")

p <- p + theme(plot.title=element\_text(size=10))

p <- p + geom\_line(data=data.frame(), aes(x=c(0,1), y=c(0,1)), colour="grey")

p <- p + annotate("text", x=0.50, y=0.00, hjust=0, vjust=0, size=5,

label=paste("AUC =", round(au, 2)))

print(p)

# Calculate the area under the curve for the plot.

# Remove observations with missing target.

no.miss <- na.omit(na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)])$y)

miss.list <- attr(no.miss, "na.action")

attributes(no.miss) <- NULL

if (length(miss.list))

{

pred <- prediction(crs$pr[-miss.list], no.miss)

} else

{

pred <- prediction(crs$pr, no.miss)

}

performance(pred, "auc")

#============================================================

# Rattle timestamp: 2018-05-01 18:11:28 x86\_64-apple-darwin15.6.0

# Evaluate model performance on the validation dataset.

# ROC Curve: requires the ROCR package.

library(ROCR)

# ROC Curve: requires the ggplot2 package.

library(ggplot2, quietly=TRUE)

# Generate an ROC Curve for the rpart model on bank.csv [validate].

crs$pr <- predict(crs$rpart, newdata=crs$dataset[crs$validate, c(crs$input, crs$target)])[,2]

# Remove observations with missing target.

no.miss <- na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]$y)

miss.list <- attr(no.miss, "na.action")

attributes(no.miss) <- NULL

if (length(miss.list))

{

pred <- prediction(crs$pr[-miss.list], no.miss)

} else

{

pred <- prediction(crs$pr, no.miss)

}

pe <- performance(pred, "tpr", "fpr")

au <- performance(pred, "auc")@y.values[[1]]

pd <- data.frame(fpr=unlist(pe@x.values), tpr=unlist(pe@y.values))

p <- ggplot(pd, aes(x=fpr, y=tpr))

p <- p + geom\_line(colour="red")

p <- p + xlab("False Positive Rate") + ylab("True Positive Rate")

p <- p + ggtitle("ROC Curve Decision Tree bank.csv [validate] y")

p <- p + theme(plot.title=element\_text(size=10))

p <- p + geom\_line(data=data.frame(), aes(x=c(0,1), y=c(0,1)), colour="grey")

p <- p + annotate("text", x=0.50, y=0.00, hjust=0, vjust=0, size=5,

label=paste("AUC =", round(au, 2)))

print(p)

# Calculate the area under the curve for the plot.

# Remove observations with missing target.

no.miss <- na.omit(crs$dataset[crs$validate, c(crs$input, crs$target)]$y)

miss.list <- attr(no.miss, "na.action")

attributes(no.miss) <- NULL

if (length(miss.list))

{

pred <- prediction(crs$pr[-miss.list], no.miss)

} else

{

pred <- prediction(crs$pr, no.miss)

}

performance(pred, "auc")