Data Science Template
End-to-End **audit** Analysis

Graham Williams

10th September 2018

This template provides a minimal example of a data science template. We will build a decision tree model to predict future events from historic observations of that event. In this case the event is whether a customer was found non-compliant in an audit.

The concept of templates for Data Science are developed in the book The Essentials of Data Science (2017). The actual source files and scripts, with regular updates, available from the Essentials web site (essentials.togaware.com).

As with all of our templates and reports we collect up front here the packages used to support the creation of this document.

```
# Load required packages from local library into R.

library(magrittr)     # Pipe operator %>% %<>% %T>% equals().
library(lubridate)    # Dates and time.
library(rattle)       # normVarNames().
library(ROCR)         # Use prediction() for evaluation.
library(rpart)        # Model: decision tree.
library(scales)       # Include commas in numbers.
library(stringi)      # String concat operator %s+%.
library(tidyverse)    # ggplot2, tibble, tidyr, readr, purr, dplyr, stringr
```

1 Data Source

```
# Original dataset source/location.

dsorig <- file.path("https://rattle.togaware.com/audit.csv")

# Name of the dataset.

dsname <- "audit"

# Identify the Essentials location of the dataset.

dsloc  <- "https://essentials.togaware.com"
dspath <- file.path(dsloc, dsname %s+% ".csv") %T>% print()
```

```
## [1] "https://essentials.togaware.com/audit.csv"
```

2 Data Ingestion

```
# Ingest the dataset.

dspath %>% read_csv() %>% assign(dsname, ., envir=.GlobalEnv)
```

3 Generic Template Variable and Initial View

```
# Store the dataset with a generic template variable name.

dsname %>% get() %T>% print() -> ds
```

```
## # A tibble: 2,000 x 13
##        ID   Age Employment Education Marital Occupation Income Gender
##     <int> <int> <chr>      <chr>     <chr>   <chr>       <dbl> <chr>
##  1 1.00e6    38 Private    College   Unmarr~ Service    8.18e4 Female
##  2 1.01e6    35 Private    Associate Absent  Transport  7.21e4 Male
##  3 1.02e6    32 Private    HSgrad    Divorc~ Clerical   1.55e5 Male
##  4 1.04e6    45 Private    Bachelor  Married Repair     2.77e4 Male
##  5 1.04e6    60 Private    College   Married Executive  7.57e3 Male
##  6 1.05e6    74 Private    HSgrad    Married Service    3.31e4 Male
##  7 1.05e6    43 Private    Bachelor  Married Executive  4.34e4 Male
##  8 1.05e6    35 Private    Yr12      Married Machinist  5.99e4 Male
##  9 1.06e6    25 Private    Associate Divorc~ Clerical   1.27e5 Female
## 10 1.06e6    22 Private    HSgrad    Absent  Sales      5.25e4 Female
## # ... with 1,990 more rows, and 5 more variables: Deductions <dbl>,
## #   Hours <int>, IGNORE_Accounts <chr>, RISK_Adjustment <int>,
## #   TARGET_Adjusted <int>
```

4 Normalise Variable Names

```
# Normalise the variable names.

names(ds) %<>% normVarNames() %T>% print()
```

```
##  [1] "id"              "age"             "employment"
##  [4] "education"       "marital"         "occupation"
##  [7] "income"          "gender"          "deductions"
## [10] "hours"           "ignore_accounts" "risk_adjustment"
## [13] "target_adjusted"
```

```
# Fix specific variable names.

names(ds)[11:13] <- c("accounts", "adjustment", "adjusted")

# Check the names.

names(ds)
```

```
##  [1] "id"         "age"        "employment" "education"  "marital"
##  [6] "occupation" "income"     "gender"     "deductions" "hours"
## [11] "accounts"   "adjustment" "adjusted"
```

5 Key Variables

```
# Note any identifiers.

id <- c("id")

# Note the target variable.

target <- "adjusted"

# Note any risk variable - measures the severity of the outcome.

risk <- "adjustment"
```

6 Variables for Analysis

```
# Note available variables ignoring identifiers and risk, with target first.

ds %>%
  names() %>%
  setdiff(c(id, risk)) %>%
  c(target, .) %>%
  unique() %T>%
  print() ->
vars
```

```
##  [1] "adjusted"   "age"        "employment" "education"  "marital"
##  [6] "occupation" "income"     "gender"     "deductions" "hours"
## [11] "accounts"
```

7 Variables to Ignore

```
# We will sometimes want to ignore specific variables.

ignore <- c("accounts", "marital", "occupation", "education")

# Remove variables to ignore from the variable list.

vars %<>% setdiff(ignore) %T>% print()
```

```
## [1] "adjusted"   "age"        "employment" "income"     "gender"
## [6] "deductions" "hours"
```

8 Deal with Character Variables

```
# Identify the character variables by index.

chari <- ds[vars] %>% sapply(is.character) %>% which() %T>% print()
```

```
## employment     gender
##          3          5
```

```
# Identify the chracter variables by name.

charc <- ds[vars] %>% names() %>% '['(chari) %T>% print()
```

```
## [1] "employment" "gender"
```

9 Character Variable Levels

```
# Observe the unique levels.

ds[charc] %>% sapply(unique)
```

```
## $employment
## [1] "Private"    "Consultant" "SelfEmp"    "PSLocal"    "PSState"
## [6] "PSFederal"  "Unemployed" NA           "Volunteer"
##
## $gender
## [1] "Female" "Male"
```

10 Characters to Factors

```
# Convert all character to factor if determined appropriate.

ds[charc] %<>% map(factor)
```

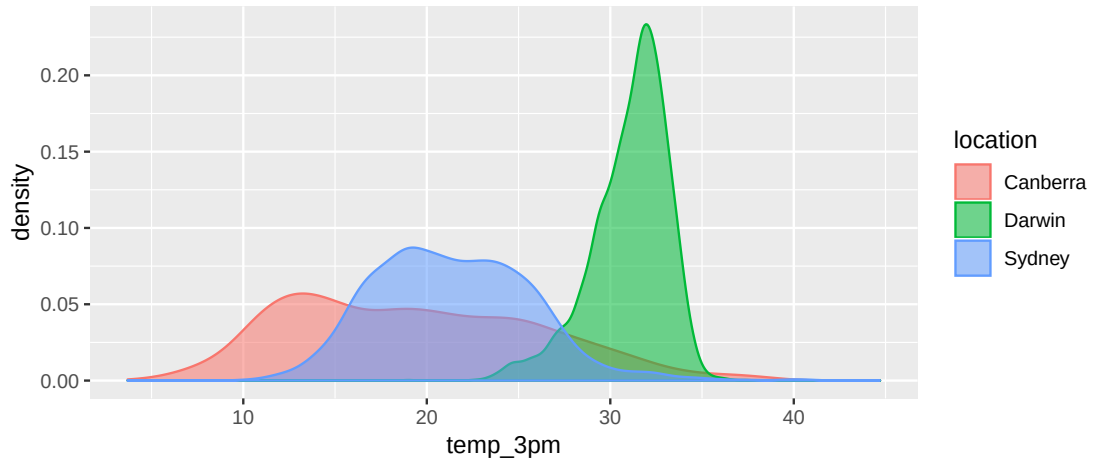
11 Data Observation

```
# A glimpse into the dataset.

glimpse(ds)
```

```
## Observations: 2,000
## Variables: 13
## $ id          <int> 1004641, 1010229, 1024587, 1038288, 1044221, 104709...
## $ age         <int> 38, 35, 32, 45, 60, 74, 43, 35, 25, 22, 48, 60, 21,...
## $ employment <fct> Private, Private, Private, Private, Private, Privat...
## $ education  <chr> "College", "Associate", "HSgrad", "Bachelor", "Coll...
## $ marital    <chr> "Unmarried", "Absent", "Divorced", "Married", "Marr...
## $ occupation <chr> "Service", "Transport", "Clerical", "Repair", "Exec...
## $ income     <dbl> 81838.00, 72099.00, 154676.74, 27743.82, 7568.23, 3...
## $ gender     <fct> Female, Male, Male, Male, Male, Male, Male, Male, F...
## $ deductions <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ hours      <int> 72, 30, 40, 55, 40, 30, 50, 40, 40, 37, 35, 40, 35,...
## $ accounts   <chr> "UnitedStates", "Jamaica", "UnitedStates", "UnitedS...
## $ adjustment <int> 0, 0, 0, 7298, 15024, 0, 22418, 0, 0, 0, 0, 0, 0, 0...
## $ adjusted   <int> 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ...
```

12 Data Visualisation

```
# Visualise relationships in the data.

ds %>%
  select(gender, income) %>%
  ggplot(aes(x=income, colour=gender, fill=gender)) +
  geom_density(alpha=0.55) +
  scale_y_continuous() +
  scale_x_continuous(labels=comma)
```



13 Model Formula

```
# Formula for modelling.

ds[vars] %>%
  formula() %T>%
  print() ->
form
```

```
## adjusted ~ age + employment + income + gender + deductions +
##     hours
## <environment: 0x56173ce18828>
```

14 Target as a Categoric

```
# Ensure the target is categoric.

ds[[target]] %<>% factor()
```

15 Variables and Observations

```
# Identify the input variables by name.

inputs <- setdiff(vars, target) %T>% print()
```

```
## [1] "age"         "employment" "income"     "gender"     "deductions"
## [6] "hours"
```

```
# Record the number of observations.

nobs <- nrow(ds) %T>% comcat()
```

```
## 2,000
```

16 Training Dataset

```
# Initialise random numbers for repeatable results.

seed <- 123
set.seed(seed)

# Partition the full dataset into three: train (70%), validate (15%), test (15%).

nobs %>%
  sample(0.70*nobs) %T>%
  {length(.) %>% comma() %>% cat("Size of training dataset:", ., "\n")} ->
train
```

```
## Size of training dataset: 1,400
```

17 Validation Dataset

```
# Create a validation dataset of 15% of the observations.

nobs %>%
  seq_len() %>%
  setdiff(train) %>%
  sample(0.15*nobs) %T>%
  {length(.) %>% comma() %>% cat("Size of validation dataset:", ., "\n")} ->
validate
```

```
## Size of validation dataset: 300
```

18 Test Dataset

```
# Create a testing dataset of 15% (the remainder) of the observations.

nobs %>%
  seq_len() %>%
  setdiff(union(train, validate)) %T>%
  {length(.) %>% comma() %>% cat("Size of validation dataset:", ., "\n")} ->
test
```

```
## Size of validation dataset: 300
```

19 Evaluation Subsets

```
# Cache the various actual values for target and risk.

tr_target <- ds[train,][[target]]    %T>% {head(., 15) %>% print()}
```

```
##  [1] 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1
## Levels: 0 1
```

```
tr_risk   <- ds[train,][[risk]]      %T>% {head(., 15) %>% print()}
```

```
##  [1]    0 7489    0 4229    0    0    0    0    0    0    0 2452    0    0
## [15] 5911
```

```
va_target <- ds[validate,][[target]] %T>% {head(., 15) %>% print()}
```

```
##  [1] 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```
va_risk   <- ds[validate,][[risk]]   %T>% {head(., 15) %>% print()}
```

```
##  [1] 4267    0    0    0    0    0    0    0    0    0    0    0    0    0
## [15]    0
```

```
te_target <- ds[test,][[target]]     %T>% {head(., 15) %>% print()}
```

```
##  [1] 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0
## Levels: 0 1
```

```
te_risk   <- ds[test,][[risk]]       %T>% {head(., 15) %>% print()}
```

```
##  [1]     0 22418     0     0     0     0     0     0     0     0     0
## [12]     0  5239     0     0
```

20 Build Model: Decision Tree

```
# Splitting function: "anova" "poisson" "class" "exp"

mthd <- "class"

# Splitting function parameters.

prms <- list(split="information")

# Control the training.

ctrl <- rpart.control(maxdepth=5)

# Build the model

m_rp <- rpart(form, ds[train, vars], method=mthd, parms=prms, control=ctrl)
```

21 Model Generic Variables

```
# Capture the model in generic variables.

model <- m_rp
mtype <- "rpart"
mdesc <- "Decision Tree"
```

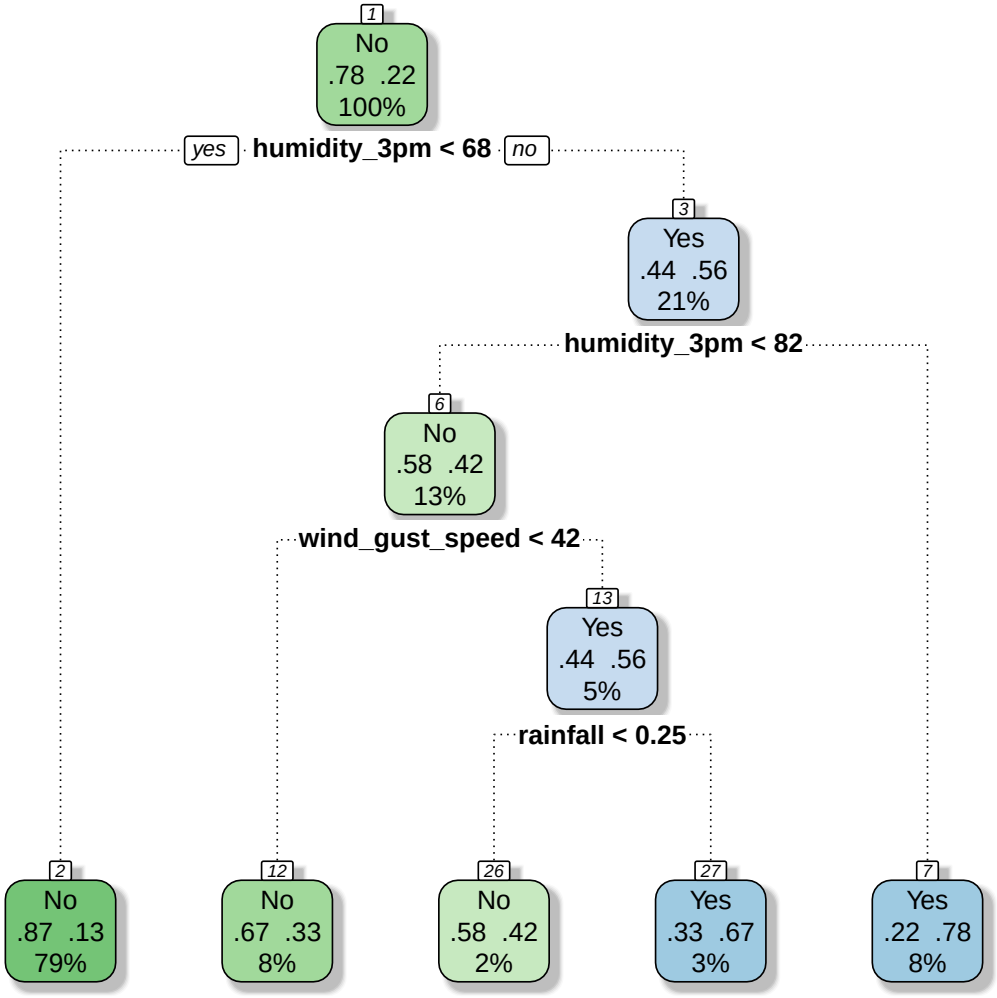
22 Review Model

```
# Basic model structure.

model
```

```
## n= 1400
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1400 335 0 (0.76071429 0.23928571)
##    2) age< 27.5 346  13 0 (0.96242775 0.03757225) *
##    3) age>=27.5 1054 322 0 (0.69449715 0.30550285)
##      6) income>=61848.59 440  73 0 (0.83409091 0.16590909) *
##      7) income< 61848.59 614 249 0 (0.59446254 0.40553746)
##       14) deductions< 1537.667 580 215 0 (0.62931034 0.37068966)
##         28) gender=Female 64   7 0 (0.89062500 0.10937500) *
##         29) gender=Male 516 208 0 (0.59689922 0.40310078)
##           58) hours< 44.5 319 104 0 (0.67398119 0.32601881) *
##           59) hours>=44.5 197  93 1 (0.47208122 0.52791878) *
##       15) deductions>=1537.667 34   0 1 (0.00000000 1.00000000) *
```

23 Visualise the Model

```
# Visually expose the discovered knowledge.

fancyRpartPlot(model)
```



24 Summary of Model

```
# Complete model build summary.

summary(model)
```

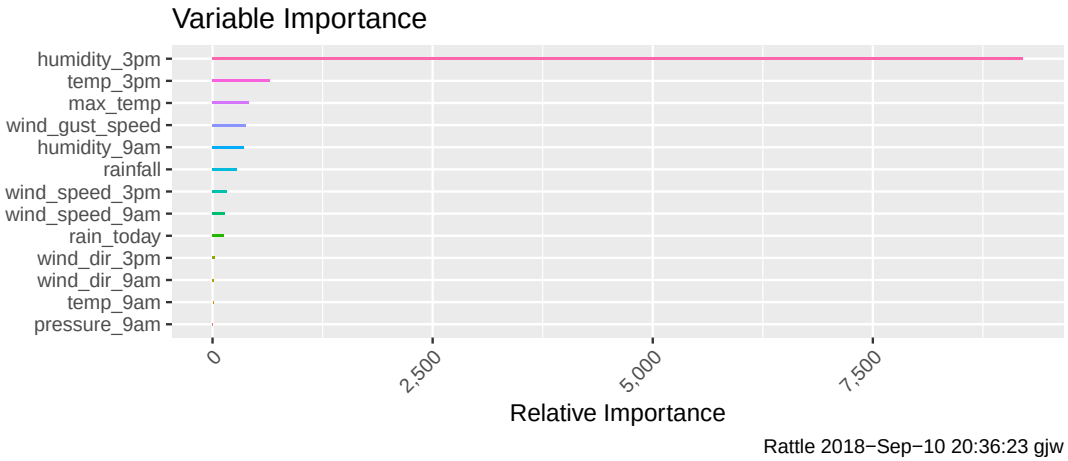
```
## Call:
## rpart(formula = form, data = ds[train, vars], method = mthd,
##     parms = prms, control = ctrl)
##   n= 1400
##
##           CP nsplit rel error    xerror       xstd
## 1 0.03383085      0 1.0000000 1.0000000 0.04765280
## 2 0.01641791      3 0.8985075 0.9014925 0.04594051
## 3 0.01000000      5 0.8656716 0.8985075 0.04588527
##
## Variable importance
##        age     income deductions     gender      hours
##         38         21         18         16          7
##
## Node number 1: 1400 observations,    complexity param=0.03383085
##   predicted class=0  expected loss=0.2392857  P(node) =1
##     class counts:  1065   335
##    probabilities: 0.761 0.239
##   left son=2 (346 obs) right son=3 (1054 obs)
##   Primary splits:
##       age        < 27.5     to the left,  improve=66.25566, (0 missing)
##       income     < 65263.6  to the right, improve=58.15583, (0 missing)
##       deductions < 1561.667 to the left,  improve=44.22407, (0 missing)
##       hours      < 49.5     to the left,  improve=34.54047, (0 missing)
##       gender     splits as  LR,           improve=29.47286, (0 missing)
##
## Node number 2: 346 observations
##   predicted class=0  expected loss=0.03757225  P(node) =0.2471429
##     class counts:   333    13
##    probabilities: 0.962 0.038
##
## Node number 3: 1054 observations,    complexity param=0.03383085
##   predicted class=0  expected loss=0.3055028  P(node) =0.7528571
##     class counts:   732   322
##    probabilities: 0.694 0.306
##   left son=6 (440 obs) right son=7 (614 obs)
##   Primary splits:
##       income     < 61848.59 to the right, improve=36.411450, (0 missing)
##       deductions < 1561.667 to the left,  improve=35.772190, (0 missing)
##       gender     splits as  LR,           improve=27.576510, (0 missing)
##       hours      < 49.5     to the left,  improve=22.845260, (0 missing)
##       age        < 36.5     to the left,  improve= 8.560643, (0 missing)
##   Surrogate splits:
```

```
##       gender     splits as  LR,           agree=0.764, adj=0.434, (0 split)
##       hours      < 37.5     to the left,  agree=0.616, adj=0.080, (0 split)
##       age        < 29.5     to the left,  agree=0.588, adj=0.014, (0 split)
##       deductions < 2464.167 to the right, agree=0.584, adj=0.005, (0 split)
##
## Node number 6: 440 observations
##   predicted class=0  expected loss=0.1659091  P(node) =0.3142857
##     class counts:   367    73
##    probabilities: 0.834 0.166
##
## Node number 7: 614 observations,    complexity param=0.03383085
##   predicted class=0  expected loss=0.4055375  P(node) =0.4385714
##     class counts:   365   249
##    probabilities: 0.594 0.406
##   left son=14 (580 obs) right son=15 (34 obs)
##   Primary splits:
##       deductions < 1537.667 to the left,  improve=32.161990, (0 missing)
##       gender     splits as  LR,           improve=13.883610, (0 missing)
##       hours      < 41.5     to the left,  improve=12.101410, (0 missing)
##       age        < 36.5     to the left,  improve= 5.783609, (0 missing)
##       income     < 42134.44 to the right, improve= 1.915395, (0 missing)
##
## Node number 14: 580 observations,    complexity param=0.01641791
##   predicted class=0  expected loss=0.3706897  P(node) =0.4142857
##     class counts:   365   215
##    probabilities: 0.629 0.371
##   left son=28 (64 obs) right son=29 (516 obs)
##   Primary splits:
##       gender     splits as  LR,           improve=12.400930, (0 missing)
##       hours      < 49.5     to the left,  improve=12.107460, (0 missing)
##       age        < 36.5     to the left,  improve= 5.974525, (0 missing)
##       income     < 42134.44 to the right, improve= 2.086069, (0 missing)
##       employment splits as  LLRLRR-L,     improve= 1.921256, (28 missing)
##
## Node number 15: 34 observations
##   predicted class=1  expected loss=0  P(node) =0.02428571
##     class counts:     0    34
##    probabilities: 0.000 1.000
##
## Node number 28: 64 observations
##   predicted class=0  expected loss=0.109375  P(node) =0.04571429
##     class counts:    57     7
##    probabilities: 0.891 0.109
##
## Node number 29: 516 observations,    complexity param=0.01641791
##   predicted class=0  expected loss=0.4031008  P(node) =0.3685714
##     class counts:   308   208
##    probabilities: 0.597 0.403
```

```
##   left son=58 (319 obs) right son=59 (197 obs)
##   Primary splits:
##       hours      < 44.5     to the left,  improve=10.2775800, (0 missing)
##       age        < 36.5     to the left,  improve= 5.8638660, (0 missing)
##       employment splits as  LLRLRR-L,     improve= 2.2070110, (21 missing)
##       deductions < 549.3333 to the right, improve= 1.1473320, (0 missing)
##       income     < 59630.25 to the left,  improve= 0.4964137, (0 missing)
##   Surrogate splits:
##       age < 29.5     to the right, agree=0.622, adj=0.01, (0 split)
##
## Node number 58: 319 observations
##   predicted class=0  expected loss=0.3260188  P(node) =0.2278571
##     class counts:   215   104
##    probabilities: 0.674 0.326
##
## Node number 59: 197 observations
##   predicted class=1  expected loss=0.4720812  P(node) =0.1407143
##     class counts:    93   104
##    probabilities: 0.472 0.528
```

25 Variable Importance

```
# Review which importance of the variables.

ggVarImp(model)
```



26 Model Predictions on Validation

```
# Predict on validation dataset to judge performance.

model %>%
  predict(newdata=ds[validate, vars], type="class") %>%
  set_names(NULL) %T>%
  {head(., 20) %>% print()} ->
va_class
```

```
##  [1] 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0
## Levels: 0 1
```

```
model %>%
  predict(newdata=ds[validate, vars], type="prob") %>%
  .[,2] %>%
  set_names(NULL) %>%
  round(2) %T>%
  {head(., 20) %>% print()} ->
va_prob
```

```
##  [1] 0.33 0.04 0.17 0.33 0.53 0.11 0.17 0.11 0.33 0.17 0.04 0.53 0.17 0.17
## [15] 0.33 0.33 0.53 0.33 0.11 0.04
```

27 Overall Accuracy and Error

```
# Overall accuracy and error.

sum(va_class == va_target) %>%
  divide_by(length(va_target)) %T>%
  {
    percent(.) %>%
      sprintf("Overall accuracy = %s\n", .) %>%
      cat()
  } ->
va_acc
```

```
## Overall accuracy = 78.7%
```

```
sum(va_class != va_target) %>%
  divide_by(length(va_target)) %T>%
  {
    percent(.) %>%
      sprintf("Overall error = %s\n", .) %>%
      cat()
  } ->
va_err
```

```
## Overall error = 21.3%
```

28 Confusion Matrix

```
# Basic comparison of prediction/actual as a confusion matrix.

table(va_target, va_class, useNA="ifany", dnn=c("Actual", "Predicted"))
```

```
##        Predicted
## Actual   0   1
##      0 211  27
##      1  37  25
```

```
# Comparison as percentages of all observations.

errorMatrix(va_target, va_class) %T>%
  print() ->
va_matrix
```

```
##        Predicted
## Actual    0   1 Error
##      0 70.3 9.0  11.3
##      1 12.3 8.3  59.7
```

```
# Error rate and average of the class error rate.

va_matrix %>%
  diag() %>%
  sum(na.rm=TRUE) %>%
  subtract(100, .) %>%
  sprintf("Overall error percentage = %s%%\n", .) %>%
  cat()
```

```
## Overall error percentage = 21.4%
```

```
va_matrix[,"Error"] %>%
  mean(na.rm=TRUE) %>%
  sprintf("Averaged class error percentage = %s%%\n", .) %>%
  cat()
```

```
## Averaged class error percentage = 35.5%
```

29 Recall, Precision, F-Score

```
# Other performance metrics: recall, precision, and the F-score.

va_rec <- (va_matrix[2,2]/(va_matrix[2,2]+va_matrix[2,1])) %T>%
  {percent(.) %>% sprintf("Recall = %s\n", .) %>% cat()}
```

```
## Recall = 40.3%
```

```
va_pre <- (va_matrix[2,2]/(va_matrix[2,2]+va_matrix[1,2])) %T>%
  {percent(.) %>% sprintf("Precision = %s\n", .) %>% cat()}
```

```
## Precision = 48.0%
```

```
va_fsc <- ((2 * va_pre * va_rec)/(va_rec + va_pre))  %T>%
  {sprintf("F-Score = %.3f\n", .) %>% cat()}
```

```
## F-Score = 0.438
```

30 ROC Curve

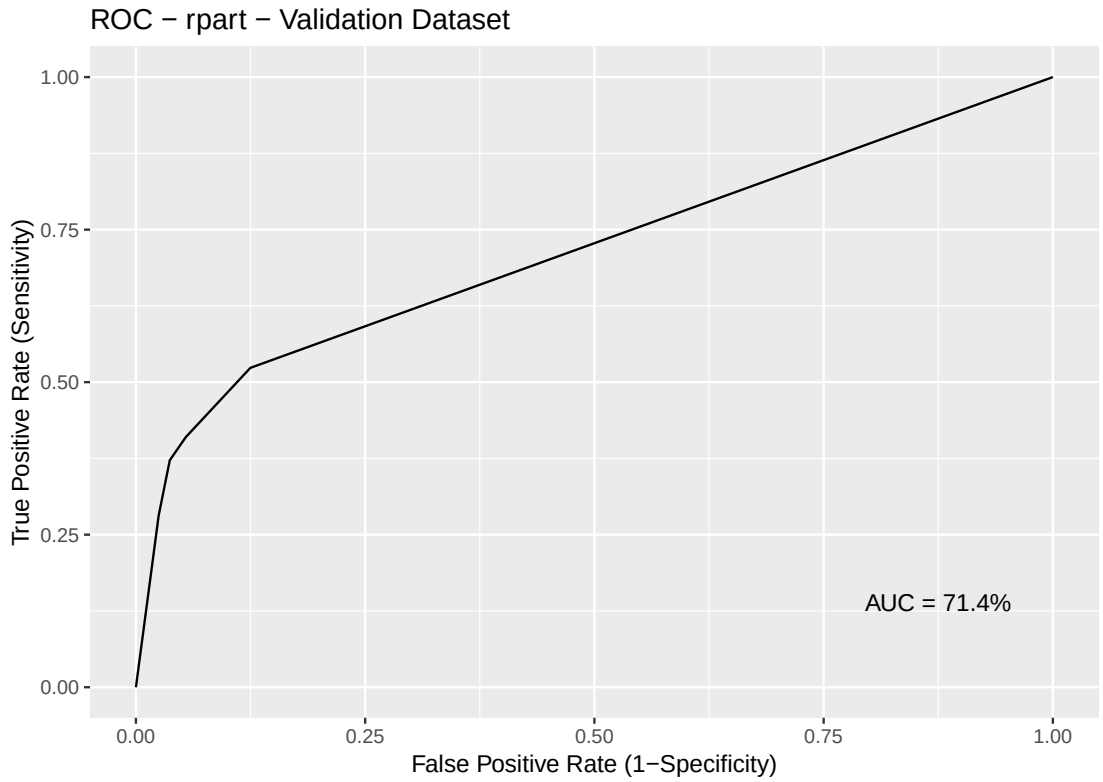
```
# Calculate the area under the curve (AUC).

va_prob %>%
  prediction(va_target) %>%
  performance("auc") %>%
  attr("y.values") %>%
  .[[1]] %T>%
  {
    percent(.) %>%
    sprintf("Percentage area under the ROC curve = %s\n", .) %>%
    cat()
  } ->
va_auc
```

```
## Percentage area under the ROC curve = 73.8%
```

```
# Calculate measures required to plot the ROC Curve.

va_prob %>%
  prediction(va_target) %>%
  performance("tpr", "fpr") ->
va_rates
```

31 ROC Curve Plot

```
# Plot the ROC Curve.

data_frame(tpr=attr(va_rates, "y.values")[[1]],
           fpr=attr(va_rates, "x.values")[[1]]) %>%
  ggplot(aes(fpr, tpr)) +
  geom_line() +
  annotate("text", x=0.875, y=0.125, vjust=0,
           label=paste("AUC =", percent(va_auc))) +
  labs(title="ROC - " %s+% mtype %s+% " - Validation Dataset",
       x="False Positive Rate (1-Specificity)",
       y="True Positive Rate (Sensitivity)")
```



32 Risk Chart

```
# Risk chart.

riskchart(va_prob, va_target, va_risk) +
  labs(title="Risk Chart - " %s+%
         mtype %s+%
         " - Validation Dataset") +
  theme(plot.title=element_text(size=14))
```