Modeling Data

Howard Nguyen

2022-09-28

INSTALL AND LOAD PACKAGES

```
# Install pacman ("package manager") if needed
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
# pacman must already be installed; then load contributed
# packages (including pacman) with pacman
pacman::p_load(
  caret,          # Predictive analytics
  GGally,         # Scatterplot matrix
  magrittr,       # Pipes
  pacman,         # Load/unload packages
  parallel,       # Parallel processing
  randomForest,   # Random forests (obviously)
  rattle,         # Plot decision trees
  rio,            # Import/export data
  tictoc,         # Time operations
  tidyverse       # So many reasons
  )
```

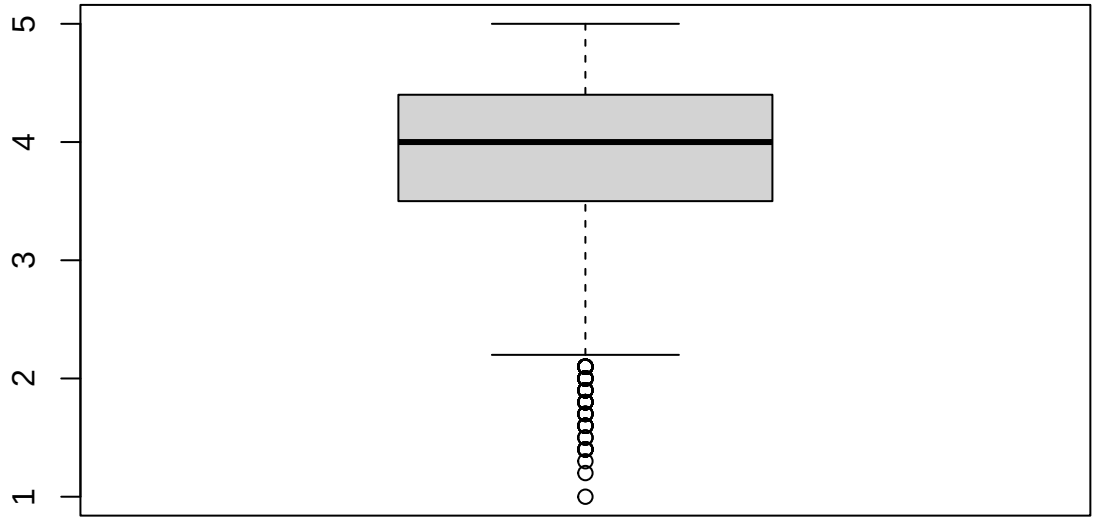
LOAD AND PREPARE DATA

```
# Import Big 5 data
df <- import("../data/b5_df.rds") %>%
  print()
```

```
## # A tibble: 18,837 x 8
##      age gender engnat Extrav Neurot Agree Consc  Open
##    <int> <fct>  <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl>
##  1    53 Male   Yes       4.4    1.1   4.6   4.7   4.3
##  2    46 Female Yes       2.2    3.1   3.5   4.2   2.6
##  3    14 Female No        3.5    4.6   3.8   4.9   4.5
##  4    19 Female No        2.2    4.3   3.7   2.6   4.1
##  5    25 Female No        3.4    3     4.4   3.4   3.4
##  6    31 Female Yes       1.6    2.4   3.6   3.1   3.3
```

```
##  7    20 Female Yes       4.6    2.1   4.5   2.8   4.1
##  8    23 Male   No        3.9    1.5   4.1   4.4   4.2
##  9    39 Female Yes       4.5    3.5   4.9   4.1   4.1
## 10    18 Female Yes       1.5    3.7   3.5   4     4.1
## # ... with 18,827 more rows
```

```
# Boxplot and median for "Open"
df %>%
  pull(Open) %T>%  # use pull() for vector; T-pipe
  boxplot() %>%
  median()
```



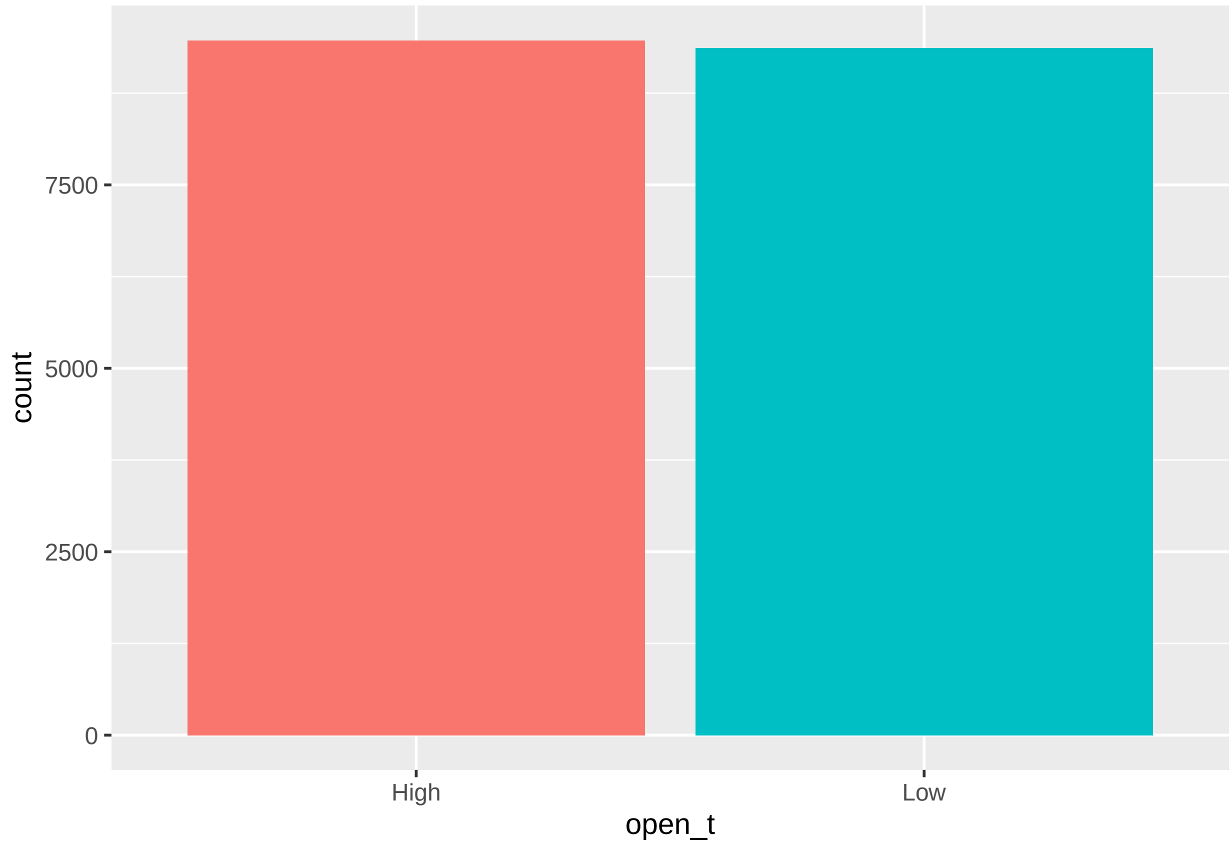

```
## [1] 4
```

```
# Dichotomize "Open" for outcome
df %<>%                          # Overwrite data
  mutate(
    open_t = ifelse(             # open_t ("text")
      Open >= 4,                 # Test
      "High",                    # Value if true
      "Low"                      # Value if false
    ),
    open_t = as_factor(open_t)   # Convert to factor
  ) %>%
  print()
```

```
## # A tibble: 18,837 x 9
##      age gender engnat Extrav Neurot Agree Consc  Open open_t
##    <int> <fct>  <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl> <fct>
##  1    53 Male   Yes       4.4    1.1   4.6   4.7   4.3 High
##  2    46 Female Yes       2.2    3.1   3.5   4.2   2.6 Low
##  3    14 Female No        3.5    4.6   3.8   4.9   4.5 High
##  4    19 Female No        2.2    4.3   3.7   2.6   4.1 High
##  5    25 Female No        3.4    3     4.4   3.4   3.4 Low
##  6    31 Female Yes       1.6    2.4   3.6   3.1   3.3 Low
##  7    20 Female Yes       4.6    2.1   4.5   2.8   4.1 High
##  8    23 Male   No        3.9    1.5   4.1   4.4   4.2 High
##  9    39 Female Yes       4.5    3.5   4.9   4.1   4.1 High
## 10    18 Female Yes       1.5    3.7   3.5   4     4.1 High
## # ... with 18,827 more rows
```

EXPLORE DATA

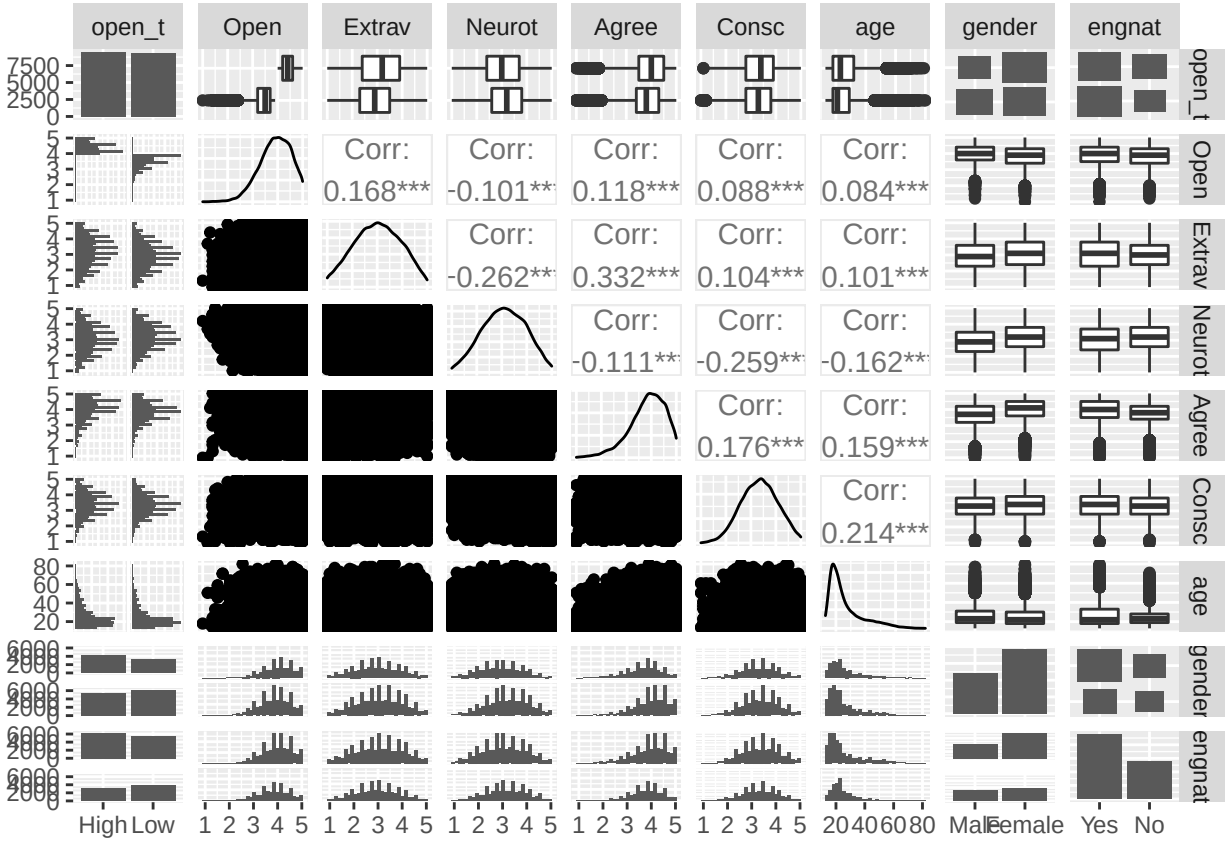
```
# Bar chart of "open_t"
df %>%
  ggplot() +
  geom_bar(
    aes(
      x    = open_t,  # Variable to chart
      fill = open_t   # Color bars by variable
    )
  ) +
  theme(legend.position = "none")
```




```
# Scatterplot matrix of all variables (takes a moment)
tic("Scatterplot matrix")
df %>%
  select(          # Reorder variables
    open_t,        # Put dichotomous "open" first
    Open,          # Then quantitative "open"
    Extrav:Consc,  # Then other Big 5
    age:engnat     # Then demographics
  ) %>%
  ggpairs()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
toc()  # 15.404 sec to calculate, about 10 sec to appear
```

```
## Scatterplot matrix: 3.4 sec elapsed
```

```
# Summary statistics
df %>% summary()
```

```
##       age          gender      engnat          Extrav          Neurot
##  Min.   :13.0   Male  : 7326   Yes:11865   Min.   :1.000   Min.   :1.000
##  1st Qu.:18.0   Female:11511   No : 6972   1st Qu.:2.300   1st Qu.:2.500
##  Median :22.0                               Median :3.000   Median :3.100
##  Mean   :26.2                               Mean   :3.015   Mean   :3.094
##  3rd Qu.:31.0                               3rd Qu.:3.700   3rd Qu.:3.700
##  Max.   :80.0                               Max.   :5.000   Max.   :5.000
##      Agree           Consc           Open        open_t
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   High:9470
##  1st Qu.:3.400   1st Qu.:2.800   1st Qu.:3.500   Low :9367
##  Median :3.900   Median :3.400   Median :4.000
##  Mean   :3.847   Mean   :3.346   Mean   :3.909
##  3rd Qu.:4.400   3rd Qu.:3.900   3rd Qu.:4.400
##  Max.   :5.000   Max.   :5.000   Max.   :5.000
```

T-TEST ON FULL DATA

```
# Just an example
df %>% t.test(Extrav ~ open_t, data = .)
```

```
##
##  Welch Two Sample t-test
##
## data:  Extrav by open_t
## t = 18.371, df = 18724, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group High and group Low is not equal to 0
## 95 percent confidence interval:
##  0.2182320 0.2703623
## sample estimates:
## mean in group High  mean in group Low
##           3.136589           2.892292
```

```
# Not a good idea to do several t-tests
```

LINEAR REGRESSION ON FULL DATA

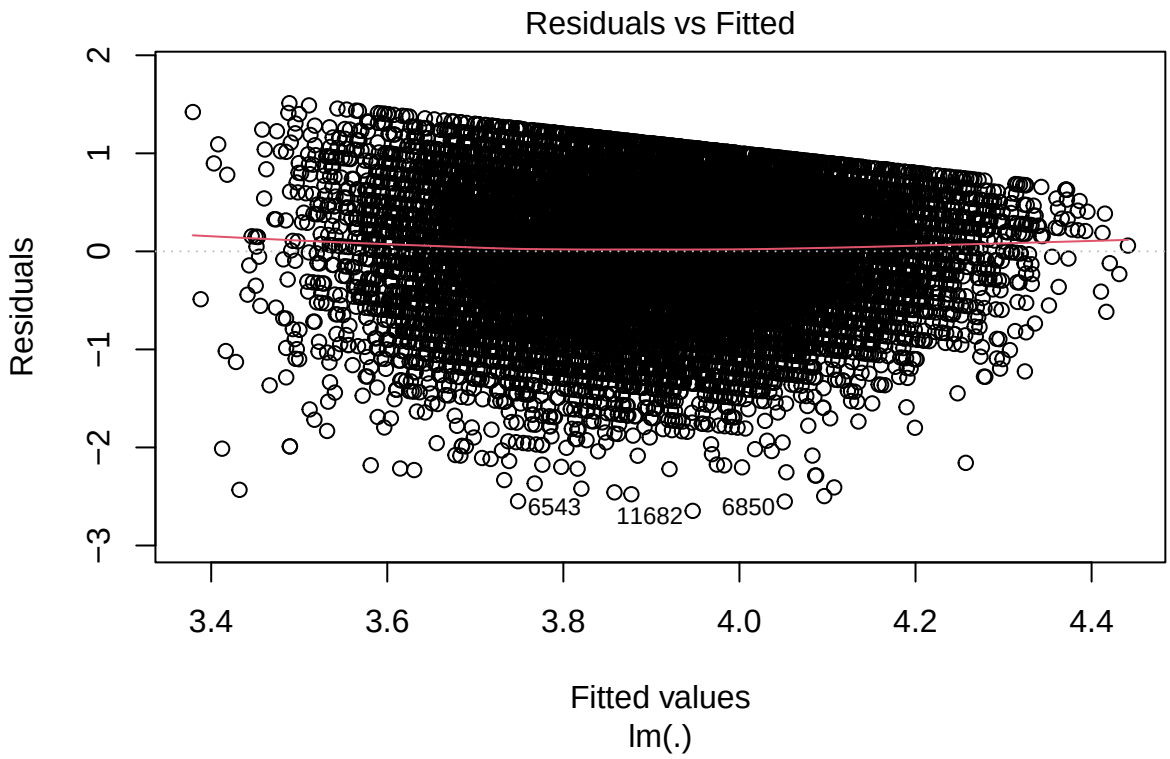
```
# Simultaneous entry linear regression
fit_lm <- df %>%   # Use full data, save as "fit_lm"
  select(          # Put outcome first
    Open,          # Then quantitative "open"
    Extrav:Consc,  # Then other Big 5
    age:engnat     # Then demographics
  ) %>%
  lm()             # Default linear regression
```

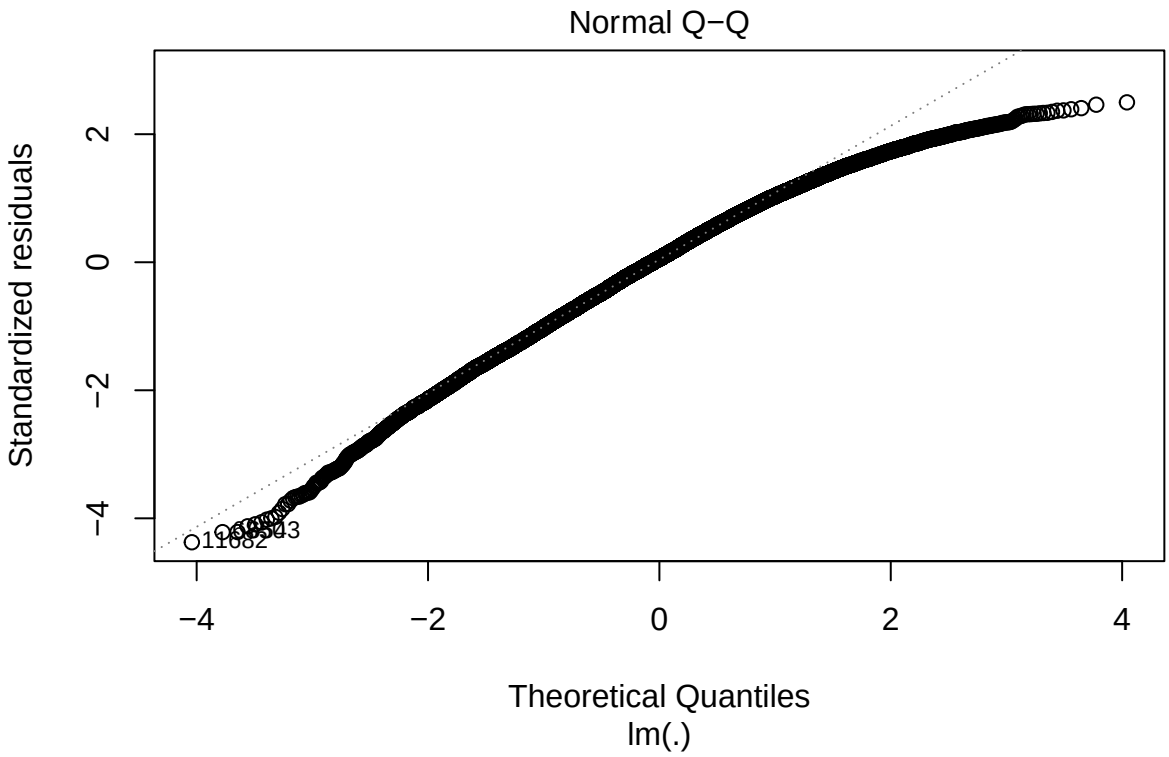
```
# Show model summary
fit_lm %>% summary()
```

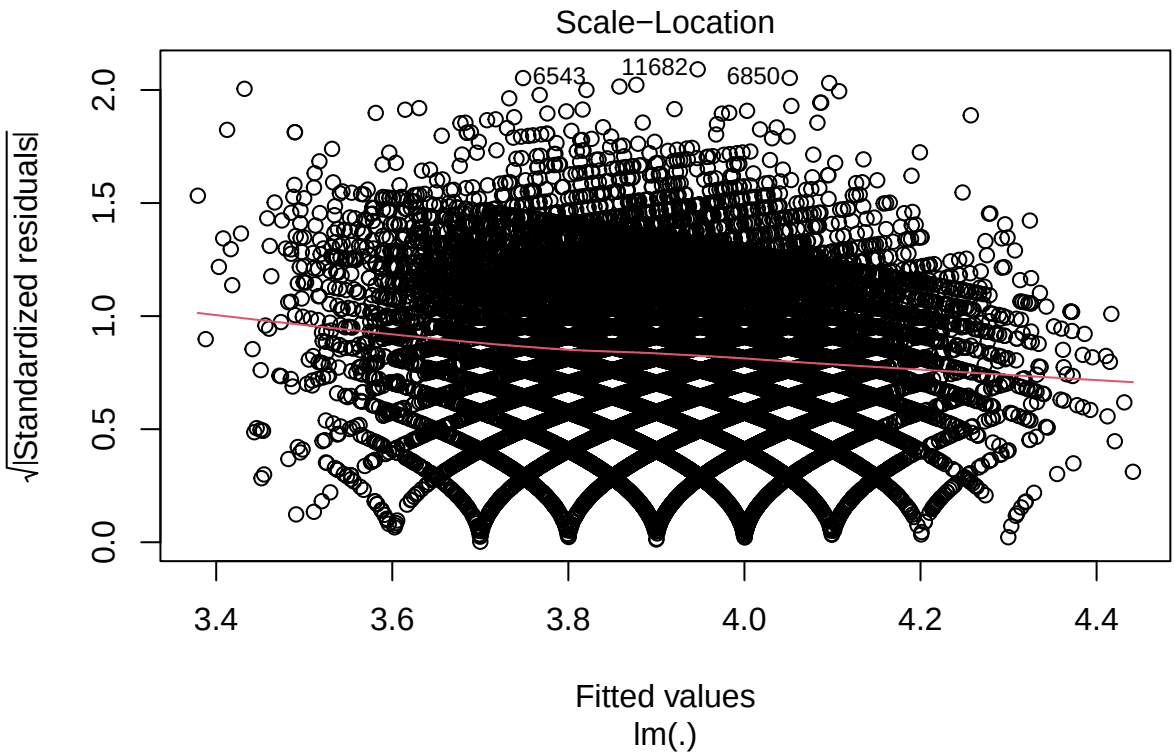
```
##
## Call:
## lm(formula = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.64690 -0.40035  0.03498  0.45182  1.51116
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.3476679  0.0400808  83.523  < 2e-16 ***
## Extrav        0.0898844  0.0052319  17.180  < 2e-16 ***
## Neurot       -0.0092399  0.0056025  -1.649   0.0991 .
## Agree         0.0689122  0.0068283  10.092  < 2e-16 ***
## Consc         0.0433337  0.0064187   6.751 1.51e-11 ***
## age           0.0019151  0.0004006   4.780 1.76e-06 ***
```

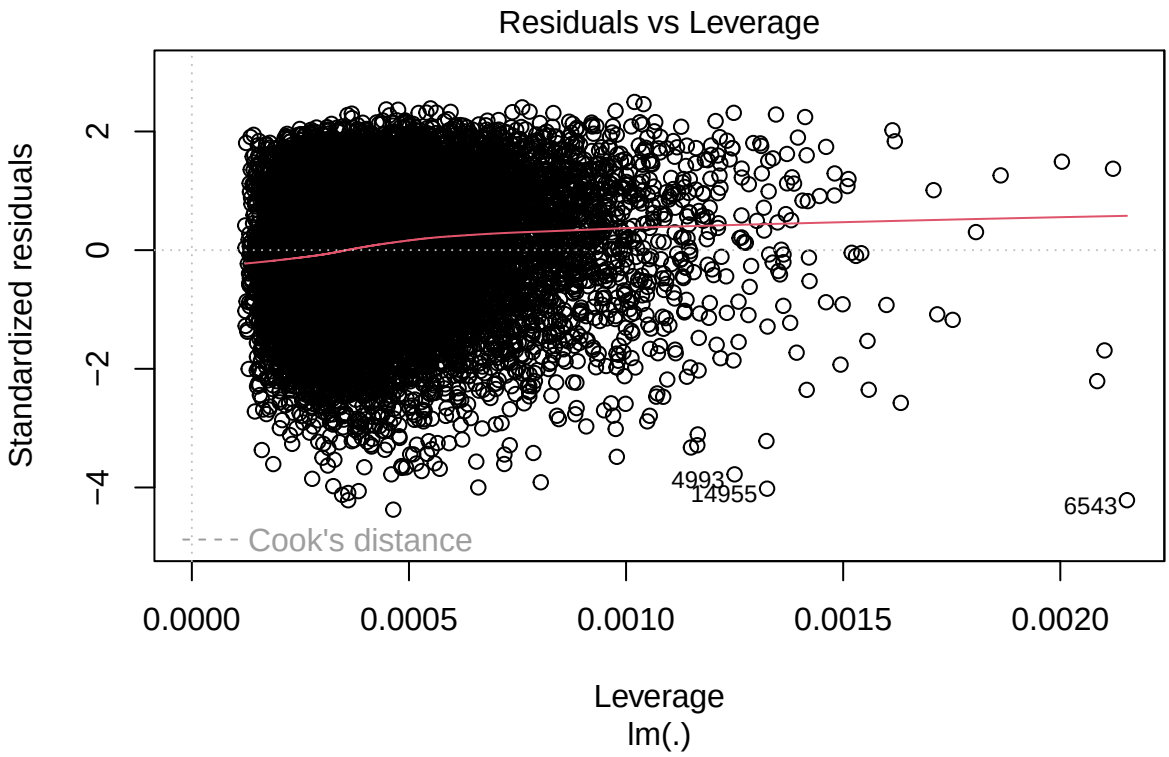
```
## genderFemale -0.1666066  0.0094945 -17.548  < 2e-16 ***
## engnatNo     -0.1055003  0.0092208 -11.442  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6054 on 18829 degrees of freedom
## Multiple R-squared:  0.06057,    Adjusted R-squared:  0.06022
## F-statistic: 173.4 on 7 and 18829 DF,  p-value: < 2.2e-16
```

```
# Diagnostic plots; run command then hit return in Console
# 1: Residuals vs. Fitted
# 2: Normal Q-Q
# 3: Scale-Location
# 4: Residuals vs. Leverage
fit_lm %>% plot()
```




Normal Q−Q
Standardized residuals
2
0
−2
−4
11082
−4
−2
0
2
4
Theoretical Quantiles
lm(.)


Scale−Location
6543
11682
6850
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
3.4
3.6
3.8
4.0
4.2
4.4
Fitted values
lm(.)



SPLIT DATA FOR MACHINE LEARNING

```
# Set random seed
set.seed(333)
```

```
# Take random subsample to save time (total n = 18,837)
# df %<>% sample_n(1000)

# Split data into train and test sets
train <- df %>% sample_frac(.70)  # Sample 70% of data
test  <- anti_join(df, train)     # Use remaining cases
```

```
## Joining, by = c("age", "gender", "engnat", "Extrav", "Neurot", "Agree",
## "Consc", "Open", "open_t")
```

KNN ON TRAINING DATA

```
# Define parameters
statctrl <- trainControl(
  method = "repeatedcv",  # Repeated cross-validation
  number = 10,            # Number of folds
```

```
  repeats = 3              # Number of complete sets of folds
)
```

```
# Define and save model
fit_knn <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,           # Use training data
  method = "knn",
  trControl = statctrl,
  tuneLength = 20,        # 20 dif values for k
  na.action = "na.omit"
)
```

```
# Apply model to training data (takes a moment)
tic("k-NN")
fit_knn
```

```
## k-Nearest Neighbors
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 11867, 11867, 11868, 11867, 11868, 11867, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.5464878  0.09303443
##    7  0.5516698  0.10341346
##    9  0.5592035  0.11849353
##   11  0.5635015  0.12710126
##   13  0.5686085  0.13732431
##   15  0.5703782  0.14087382
##   17  0.5717690  0.14366493
##   19  0.5737396  0.14760974
##   21  0.5739415  0.14801979
##   23  0.5729309  0.14600902
##   25  0.5753328  0.15081889
##   27  0.5757371  0.15163171
##   29  0.5771280  0.15442140
##   31  0.5792010  0.15857497
##   33  0.5770520  0.15428226
##   35  0.5768744  0.15393276
##   37  0.5765203  0.15322771
##   39  0.5771526  0.15449302
##   41  0.5764955  0.15318687
##   43  0.5754086  0.15101927
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 31.
```

```
toc() # 0.04 sec for n = 700; 0.01 for full data (13k)??
```

```
## k-NN: 0.003 sec elapsed
```

```
# Predict training set
open_p <- fit_knn %>%  # "predicted"
  predict(newdata = train)
```

```
# Accuracy of model on training data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # .6180 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 3528 3072
##        Low  1877 4709
##
##                Accuracy : 0.6247
##                  95% CI : (0.6163, 0.633)
##     No Information Rate : 0.5901
##     P-Value [Acc > NIR] : 2.675e-16
##
##                   Kappa : 0.2495
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6527
##             Specificity : 0.6052
##          Pos Pred Value : 0.5345
##          Neg Pred Value : 0.7150
##              Prevalence : 0.4099
##          Detection Rate : 0.2676
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.6290
##
##        'Positive' Class : High
##
```

KNN ON TEST DATA

```
# Predict test set
open_p <- predict(  # Create new variable ("predicted")
  fit_knn,          # Apply saved model
  newdata = test    # Use test data
)
```

```r
# Accuracy of model on test data
table(
  actualclass = test$open_t,  # True outcome
  predictedclass = open_p     # Predicted outcome
) %>%
confusionMatrix() %>%         # Accuracy statistics
print()                       # .5772 accuracy on full data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1412 1458
##        Low   938 1843
##
##                Accuracy : 0.576
##                  95% CI : (0.563, 0.5889)
##     No Information Rate : 0.5841
##     P-Value [Acc > NIR] : 0.8952
##
##                   Kappa : 0.1543
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6009
##             Specificity : 0.5583
##          Pos Pred Value : 0.4920
##          Neg Pred Value : 0.6627
##              Prevalence : 0.4159
##          Detection Rate : 0.2499
##    Detection Prevalence : 0.5079
##       Balanced Accuracy : 0.5796
##
##        'Positive' Class : High
##
```

DECISION TREE ON TRAINING DATA

```r
# Train decision tree on training data (takes a moment)
tic("Decision tree")
fit_dt <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,       # Use training data
  method = "rpart",   # Recursive partitioning
  trControl = trainControl(method = "cv")  # Cross-validate
)
toc()  # 1.1 sec for n = 700; 1.8 sec for full data
```

```
## Decision tree: 0.648 sec elapsed
```

```r
# Show processing summary
fit_dt
```

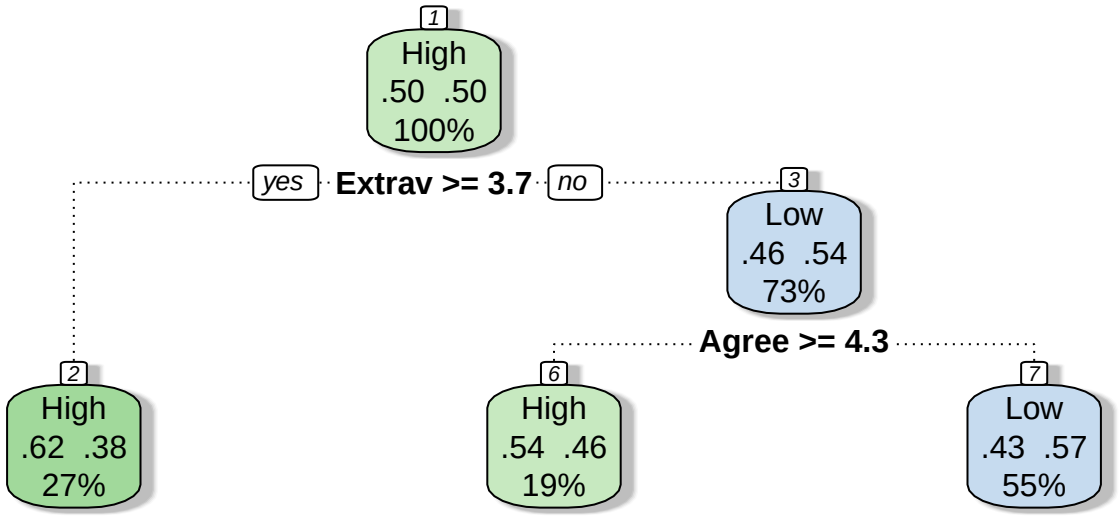
```
## CART
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 11867, 11867, 11868, 11868, 11867, 11867, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.004555117  0.5764478  0.15292936
##   0.032645005  0.5643858  0.12898642
##   0.126935925  0.5232800  0.04603416
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.004555117.
```

```r
# Description of final training model
fit_dt$finalModel
```

```
## n= 13186
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 13186 6586 High (0.5005309 0.4994691)
##   2) Extrav>=3.65 3512 1331 High (0.6210137 0.3789863) *
##   3) Extrav< 3.65 9674 4419 Low (0.4567914 0.5432086)
##     6) Agree>=4.25 2471 1128 High (0.5435047 0.4564953) *
##     7) Agree< 4.25 7203 3076 Low (0.4270443 0.5729557) *
```

```r
# Plot final training model
fit_dt$finalModel %>%
  fancyRpartPlot(
    main = "Predicting Open",
    sub = "Training Data"
  )
```

Predicting Open



Training Data

```
# Predict training set
open_p <- fit_dt %>%  # "predicted"
  predict(newdata = train)
```

```
# Accuracy of model on training data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # .5802 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 3524 3076
##        Low  2459 4127
##
##                Accuracy : 0.5802
##                  95% CI : (0.5718, 0.5887)
##     No Information Rate : 0.5463
##     P-Value [Acc > NIR] : 2.13e-15
##
##                   Kappa : 0.1606
```

```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5890
##             Specificity : 0.5730
##          Pos Pred Value : 0.5339
##          Neg Pred Value : 0.6266
##              Prevalence : 0.4537
##          Detection Rate : 0.2673
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.5810
##
##        'Positive' Class : High
##
```

DECISION TREE ON TEST DATA

```
# Predict test set
open_p <- fit_dt %>%
  predict(newdata = test)

# Accuracy of model on test data
table(
  actualclass = test$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.5776 on full test data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1544 1326
##        Low  1061 1720
##
##                Accuracy : 0.5776
##                  95% CI : (0.5646, 0.5905)
##     No Information Rate : 0.539
##     P-Value [Acc > NIR] : 2.967e-09
##
##                   Kappa : 0.1562
##
##  Mcnemar's Test P-Value : 6.534e-08
##
##             Sensitivity : 0.5927
##             Specificity : 0.5647
##          Pos Pred Value : 0.5380
##          Neg Pred Value : 0.6185
##              Prevalence : 0.4610
##          Detection Rate : 0.2732
##    Detection Prevalence : 0.5079
```

```
##       Balanced Accuracy : 0.5787
##
##        'Positive' Class : High
##
```

RANDOM FOREST ON TRAINING DATA

```
# Define parameters for the random forest
control <- trainControl(
  method  = "repeatedcv",  # Repeated cross-validation
  number  = 10,            # Number of folds
  repeats = 3,             # Number of sets of folds
  search  = "random",      # Max number of tuning parameters
  allowParallel = TRUE     # Allow parallel processing
)
```

```
# Train random forest on training data (can take a while)
tic("Random forest")
fit_rf <- train(
  open_t ~ age + gender + Extrav + Neurot + Agree + Consc,
  data = train,          # Use training data
  method = "rf",         # Use random forests
  metric = "Accuracy",   # Use accuracy as criterion
  tuneLength = 15,       # Number of levels for parameters
  ntree = 300,           # Number of trees
  trControl = control    # Link to parameters
)
toc()  # 32 sec when n = 700; 635 sec for full data
```

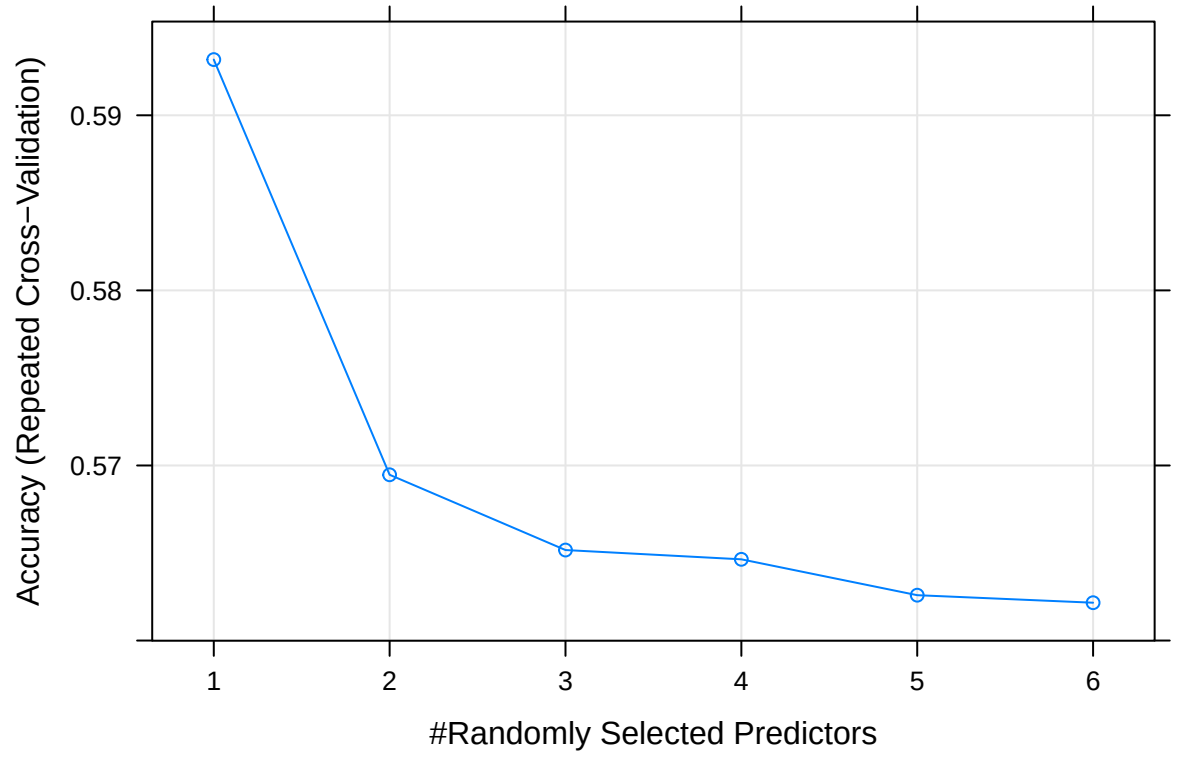
```
## Random forest: 535.279 sec elapsed
```

```
# Show processing summary
fit_rf
```

```
## Random Forest
##
## 13186 samples
##     6 predictor
##     2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 11867, 11867, 11868, 11868, 11867, 11867, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.5931825  0.1864808
##   2     0.5694682  0.1389488
##   3     0.5651699  0.1303523
##   4     0.5646404  0.1292891
##   5     0.5625916  0.1251975
```

```
##   6     0.5621620  0.1243339
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

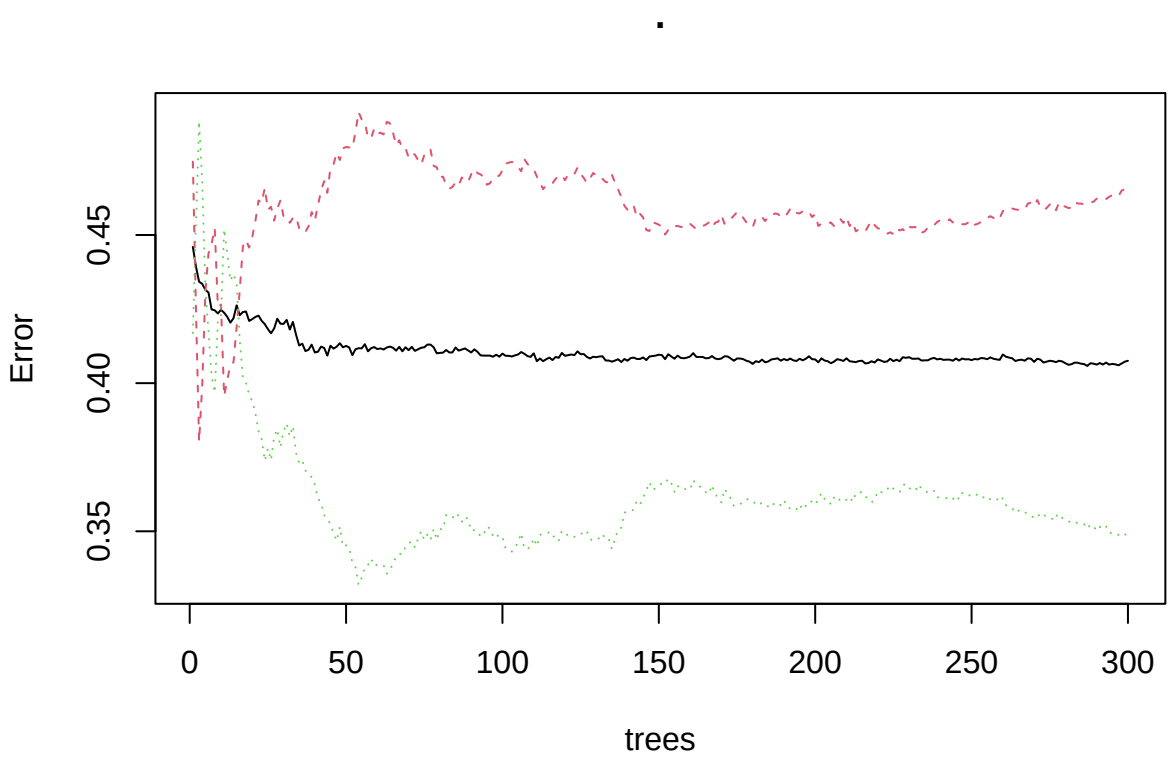
```
# Plot accuracy by number of predictors
fit_rf %>% plot()
```




```
# Accuracy of model with training data
fit_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 40.76%
## Confusion matrix:
##      High  Low class.error
## High 3524 3076   0.4660606
## Low  2298 4288   0.3489220
```

```
# Plot error by number of trees; Red is error for "High,"
# green is error for "Low," and black is error or "OOB,"
# or "out of bag" (i.e., the probability that any given
# prediction is not correct within the test data, or the
# overall accuracy)
fit_rf$finalModel %>% plot()
```




```
# Predict training data
open_p <- fit_rf %>%        # "predicted"
  predict(newdata = train)  # Use train data
```

```
# Accuracy of model on test data
table(
  actualclass = train$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.7145 accuracy on full training data
```

```
## Confusion Matrix and Statistics
##
##             predictedclass
## actualclass High  Low
##        High 4277 2323
```

```
##        Low  1471 5115
##
##                Accuracy : 0.7123
##                  95% CI : (0.7045, 0.72)
##     No Information Rate : 0.5641
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4246
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7441
##             Specificity : 0.6877
##          Pos Pred Value : 0.6480
##          Neg Pred Value : 0.7766
##              Prevalence : 0.4359
##          Detection Rate : 0.3244
##    Detection Prevalence : 0.5005
##       Balanced Accuracy : 0.7159
##
##        'Positive' Class : High
##
```

RANDOM FOREST ON TEST DATA

```
# Predict test set
open_p <- fit_rf %>%       # "predicted"
  predict(newdata = test)  # Use test data
```

```
# Accuracy of model on test data
table(
  actualclass = test$open_t,
  predictedclass = open_p
) %>%
confusionMatrix() %>%
print()  # 0.5912 accuracy on full test data
```

```
## Confusion Matrix and Statistics
##
##            predictedclass
## actualclass High  Low
##        High 1525 1345
##        Low  1000 1781
##
##                Accuracy : 0.585
##                  95% CI : (0.5721, 0.5979)
##     No Information Rate : 0.5532
##     P-Value [Acc > NIR] : 7.360e-07
##
##                   Kappa : 0.1714
##
```

```
##  Mcnemar's Test P-Value : 1.214e-12
##
##             Sensitivity : 0.6040
##             Specificity : 0.5697
##          Pos Pred Value : 0.5314
##          Neg Pred Value : 0.6404
##              Prevalence : 0.4468
##          Detection Rate : 0.2699
##    Detection Prevalence : 0.5079
##       Balanced Accuracy : 0.5868
##
##        'Positive' Class : High
##
```

SUMMARIZING PREDICTIONS

```
# TOTAL n = 18,837
# TRAIN n = 13,186
# TEST  n =  5,651

# How many cases in "high"?
test %>% pull(open_t) %>% summary()
```

```
## High  Low
## 2870 2781
```

```
# 2870 / 5651 = .5078 (about 51%)

#      TRAIN  TEST
# KNN    62%   58%
# DT     58%   58%
# RF     71%   59%
```

```
# CLEAN UP ##########################################################

# Clear data
# rm(list = ls())  # Removes all objects from environment

# Clear packages
# p_unload(all)  # Remove all contributed packages

# Clear plots
#graphics.off()  # Clears plots, closes all graphics devices

# Clear console
#cat("\014")  # Mimics ctrl+L

# Clear R
#   You may want to use Session > Restart R, as well, which
#   resets changed options, relative paths, dependencies,
#   and so on to let you start with a clean slate
```

```
# Clear mind :)
```