Big 5 Gender Predictor

```r
library('mda')
library('MASS')
library('klaR')
library('nnet')
library('kernlab')
library('caret')
library('e1071')
library("tidyverse")
library("keras")
```

```r
set.seed(5)
data <- read.csv("./data/data.csv") %>%
  filter(gender %in% c(1,2)) # General dataset
c_data <- read.csv("./data/custom_data.csv") # Custom user dataset
num_cust_data <- nrow(c_data) # Track number of custom user entries in the dataset
cust_preds <- data.frame(name = c_data[1:num_cust_data,"name"]) # Placeholder dataframe for later-gener
n <-nrow(data)
data <- rbind(c_data,data[sample(nrow(data)),]) # Shuffle dataset, prepend custom users
```

Only select rows with predictable genders. Non-specified could also be considered if there were more samples, but at the time of writing this only **102** observations are present in the data set. In the original question set several questions were posed such that they measured the low-end of the trait rather than the high end. These values have been reversed.

Trait Definitions:
O: Low Openness -> High Openness
C: Low Conscientiousness -> High Conscientiousness
E: Intraversion -> Extraversion
A: Disagreeableness -> Agreeableness
N: Low Neuroticism -> High Neuroticism

Define R function to correct for question

```r
qCorrect <- function(n){
  switch(as.character(n),
         '1' = 5,
         '2' = 4,
         '3' = 3,
         '4' = 2,
         '5' = 1, 0)
}
qCorrect <- Vectorize(qCorrect) # Vectorize function to allow interoperation with dplyr::mutate
```

```r
# Flip certain answer using qCorrect to align with traits as defined previously
data_qc <- data %>%
              mutate_at(
                .vars = c("E2", "E4", "E6", "E8", "E10", "N2",
                          "N4", "A1", "A3", "A5", "A7", "C2",
                          "C4", "C6", "C8", "O2", "O4", "O6"), funs(qCorrect)) %>%
              mutate(sex = ifelse(gender == 1, 0, 1))
```

Calculate score based on 10 relevant questions to metric

```r
data_rs <- data_qc %>%
            mutate(o_sum = rowSums(.[48:57])) %>%
            mutate(c_sum = rowSums(.[38:47])) %>%
            mutate(e_sum = rowSums(.[ 8:17])) %>%
            mutate(a_sum = rowSums(.[28:37])) %>%
            mutate(n_sum = rowSums(.[18:27]))
str(data_rs[58:64])
```

```
## 'data.frame':    19595 obs. of  7 variables:
##  $ name : Factor w/ 2 levels "Christopher Sparling",..: 1 2 NA NA NA NA NA NA NA NA ...
##  $ sex  : num  0 1 1 1 0 1 1 1 1 1 ...
##  $ o_sum: num  42 24 34 41 45 42 38 42 44 29 ...
##  $ c_sum: num  41 31 33 46 40 29 37 41 36 37 ...
##  $ e_sum: num  35 42 33 44 27 36 36 47 29 24 ...
##  $ a_sum: num  33 23 41 35 27 41 48 49 43 42 ...
##  $ n_sum: num  19 36 34 45 40 35 37 17 16 44 ...
```

Calculate percentile scoring

```r
data_ps <- data_rs %>%
            mutate(o_ps = percent_rank(o_sum)) %>%
            mutate(c_ps = percent_rank(c_sum)) %>%
            mutate(e_ps = percent_rank(e_sum)) %>%
            mutate(a_ps = percent_rank(a_sum)) %>%
            mutate(n_ps = percent_rank(n_sum))

# Select useful columns
big5 <- data_ps[,c(58,59,65:69)]
str(big5)
```

```
## 'data.frame':    19595 obs. of  7 variables:
##  $ name: Factor w/ 2 levels "Christopher Sparling",..: 1 2 NA NA NA NA NA NA NA NA ...
##  $ sex : num  0 1 1 1 0 1 1 1 1 1 ...
##  $ o_ps: num  0.6172 0.0117 0.1878 0.558 0.785 ...
##  $ c_ps: num  0.82 0.348 0.445 0.949 0.784 ...
##  $ e_ps: num  0.664 0.875 0.589 0.921 0.361 ...
##  $ a_ps: num  0.1853 0.031 0.5682 0.2581 0.0651 ...
##  $ n_ps: num  0.0829 0.6795 0.6016 0.9426 0.8214 ...
```

```r
train_indices <- (num_cust_data + 1):round(0.7*n) # ~70% of dataset
test_indices <- c(1:num_cust_data,(round(0.7*n)+1):n) # ~30% of dataset
train <- big5[train_indices,2:7]
test  <- big5[test_indices,2:7]
str(train)
```

```
## 'data.frame':    13713 obs. of  6 variables:
##  $ sex : num  1 1 0 1 1 1 1 1 1 0 ...
##  $ o_ps: num  0.188 0.558 0.785 0.617 0.38 ...
##  $ c_ps: num  0.445 0.949 0.784 0.256 0.65 ...
##  $ e_ps: num  0.589 0.921 0.361 0.698 0.698 ...
##  $ a_ps: num  0.5682 0.2581 0.0651 0.5682 0.9196 ...
##  $ n_ps: num  0.602 0.943 0.821 0.641 0.717 ...
```

```r
str(test)
```

```
## 'data.frame':    5880 obs. of  6 variables:
##  $ sex : num  0 1 1 0 0 1 0 0 1 1 ...
```

```
##  $ o_ps: num  0.6172 0.0117 0.2292 0.9749 0.0742 ...
##  $ c_ps: num  0.82 0.348 0.348 0.55 0.65 ...
##  $ e_ps: num  0.664 0.875 0.472 0.969 0.253 ...
##  $ a_ps: num  0.185 0.031 0.977 0.786 0.112 ...
##  $ n_ps: num  0.0829 0.6795 0.8796 0.039 0.0829 ...
```

Mixture Discriminant Analysis

```
mda_fit <- mda(sex~., data = train)
mda_predictions <-predict(mda_fit,test[,2:6])
pred_table <- table(mda_predictions,test$sex)
confusionMatrix(as.factor(mda_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  862  601
##          1 1379 3038
##
##                Accuracy : 0.6633
##                  95% CI : (0.651, 0.6753)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 8.823e-13
##
##                   Kappa : 0.2352
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3846
##             Specificity : 0.8348
##          Pos Pred Value : 0.5892
##          Neg Pred Value : 0.6878
##              Prevalence : 0.3811
##          Detection Rate : 0.1466
##    Detection Prevalence : 0.2488
##       Balanced Accuracy : 0.6097
##
##        'Positive' Class : 0
##
```

```
cust_preds$mda <- mda_predictions[1:num_cust_data]
```

Quadratic Discriminant Analysis

```
qda_fit <- qda(sex~., data = train)
qda_predictions <-predict(qda_fit,test[,2:6])$class
confusionMatrix(as.factor(qda_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  842  575
##          1 1399 3064
##
##                Accuracy : 0.6643
##                  95% CI : (0.6521, 0.6764)
```

```
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 2.679e-13
##
##                   Kappa : 0.2343
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3757
##             Specificity : 0.8420
##          Pos Pred Value : 0.5942
##          Neg Pred Value : 0.6865
##              Prevalence : 0.3811
##          Detection Rate : 0.1432
##    Detection Prevalence : 0.2410
##       Balanced Accuracy : 0.6089
##
##        'Positive' Class : 0
##
```

```
cust_preds$qda <- qda_predictions[1:num_cust_data]
```

Regularized Discriminant Analysis

```
rda_fit <- rda(sex~., data = train, gamma = 0.05, lambda = 0.01)
rda_predictions <-predict(rda_fit,test[,2:6])$class
confusionMatrix(as.factor(rda_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  836  571
##          1 1405 3068
##
##                Accuracy : 0.6639
##                  95% CI : (0.6517, 0.676)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 3.998e-13
##
##                   Kappa : 0.2328
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3730
##             Specificity : 0.8431
##          Pos Pred Value : 0.5942
##          Neg Pred Value : 0.6859
##              Prevalence : 0.3811
##          Detection Rate : 0.1422
##    Detection Prevalence : 0.2393
##       Balanced Accuracy : 0.6081
##
##        'Positive' Class : 0
##
```

```
cust_preds$rda <- rda_predictions[1:num_cust_data]
```

Neural Net

```r
nnet_fit <- nnet(as.factor(sex)~., data=train, size=4, decay=0.0001, maxit=500)
```

```
## # weights:  29
## initial  value 9997.473477
## iter  10 value 8476.823886
## iter  20 value 8439.569659
## iter  30 value 8417.923971
## iter  40 value 8399.988666
## iter  50 value 8391.719568
## iter  60 value 8389.188520
## iter  70 value 8387.977525
## iter  80 value 8387.504403
## iter  90 value 8386.898515
## iter 100 value 8386.225527
## iter 110 value 8384.410270
## iter 120 value 8383.675260
## iter 130 value 8383.406930
## iter 140 value 8383.156801
## iter 150 value 8382.379652
## iter 160 value 8381.970109
## iter 170 value 8381.783453
## iter 180 value 8381.707506
## final  value 8381.691473
## converged
```

```r
nnet_predictions <-predict(nnet_fit,test[,2:6], type='class')
confusionMatrix(as.factor(nnet_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  867  612
##          1 1374 3027
##
##                Accuracy : 0.6622
##                  95% CI : (0.65, 0.6743)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 2.828e-12
##
##                   Kappa : 0.234
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3869
##             Specificity : 0.8318
##          Pos Pred Value : 0.5862
##          Neg Pred Value : 0.6878
##              Prevalence : 0.3811
##          Detection Rate : 0.1474
##    Detection Prevalence : 0.2515
##       Balanced Accuracy : 0.6094
##
##        'Positive' Class : 0
##
```

```
cust_preds$nnet <- nnet_predictions[1:num_cust_data]
```

Flexible Discriminant Analysis

```
fda_fit <- fda(as.factor(sex)~., data=train)
fda_predictions <-predict(fda_fit,test[,2:6])
confusionMatrix(as.factor(fda_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  857  598
##          1 1384 3041
##
##                Accuracy : 0.6629
##                  95% CI : (0.6507, 0.675)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 1.305e-12
##
##                   Kappa : 0.2338
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3824
##             Specificity : 0.8357
##          Pos Pred Value : 0.5890
##          Neg Pred Value : 0.6872
##              Prevalence : 0.3811
##          Detection Rate : 0.1457
##    Detection Prevalence : 0.2474
##       Balanced Accuracy : 0.6090
##
##        'Positive' Class : 0
##
```

```
cust_preds$fda <- fda_predictions[1:num_cust_data]
```

Support Vector Machine

```
svm_fit <- ksvm(as.factor(sex)~., data=train)
svm_predictions <- predict(svm_fit, test[,2:6], type='response')
confusionMatrix(as.factor(svm_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  741  455
##          1 1500 3184
##
##                Accuracy : 0.6675
##                  95% CI : (0.6553, 0.6796)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 5.144e-15
##
##                   Kappa : 0.2259
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3307
##             Specificity : 0.8750
##          Pos Pred Value : 0.6196
##          Neg Pred Value : 0.6798
##              Prevalence : 0.3811
##          Detection Rate : 0.1260
##    Detection Prevalence : 0.2034
##       Balanced Accuracy : 0.6028
##
##        'Positive' Class : 0
##
```

```
cust_preds$svm <- svm_predictions[1:num_cust_data]
```

k-Nearest Neighbours

```
knn_fit <- knn3(as.factor(sex)~., data=train, k = 10)
knn_predictions <- predict(knn_fit, test[,2:6], type='class')
confusionMatrix(as.factor(knn_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  950  867
##          1 1291 2772
##
##                Accuracy : 0.633
##                  95% CI : (0.6205, 0.6453)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 0.01322
##
##                   Kappa : 0.1927
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.4239
##             Specificity : 0.7617
##          Pos Pred Value : 0.5228
##          Neg Pred Value : 0.6823
##              Prevalence : 0.3811
##          Detection Rate : 0.1616
##    Detection Prevalence : 0.3090
##       Balanced Accuracy : 0.5928
##
##        'Positive' Class : 0
##
```

```
cust_preds$knn <- knn_predictions[1:num_cust_data]
```

Naive Bayes

```
nb_fit <- naiveBayes(as.factor(sex)~., data=train)
nb_predictions <- predict(nb_fit, test[,2:6], type='class')
confusionMatrix(as.factor(nb_predictions),as.factor(test$sex))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  770  505
##          1 1471 3134
##
##                Accuracy : 0.6639
##                  95% CI : (0.6517, 0.676)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : 3.998e-13
##
##                   Kappa : 0.2233
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3436
##             Specificity : 0.8612
##          Pos Pred Value : 0.6039
##          Neg Pred Value : 0.6806
##              Prevalence : 0.3811
##          Detection Rate : 0.1310
##    Detection Prevalence : 0.2168
##       Balanced Accuracy : 0.6024
##
##        'Positive' Class : 0
##
```

```
cust_preds$nb <- nb_predictions[1:num_cust_data]
```

Custom User Predictions

```
cust_preds
```

```
##                   name mda qda rda nnet fda svm knn nb
## 1 Christopher Sparling   0   0   0    0   0   0   0  0
## 2        Florence Awde   1   1   1    1   1   1   1  1
```

References:

https://engineering.semantics3.com/debugging-neural-networks-a-checklist-ca52e11151ec https://keras.rstudio.com/articles/functional_api.html#multi-input-and-multi-output-models
https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
https://machinelearningmastery.com/non-linear-classification-in-r