projectC

Jiayin Qu

5/4/2020

Libraries

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(stringr)
library(readxl)
library(glmnet)
library(caret)
library(klaR)
library(parallel)
library(doParallel)
```

Data import

- The dataset is extracted from https://openpsychometrics.org/_rawdata/, containing data from a study that examine the Firstborn Personality Scale. The information of the study can be found through https://openpsychometrics.org/tests/birthorder/development/. The variables we utilize in this case are personality items as predictors and firstborn (or not) in multi-children family as outcome.
- I chose this data because it has a clear outcome variable and gathered relatively comprehensive personality data using a reasonable measure.
- Research background: some evidence suggests that birth order can have large practical effects. For example, some groups, such as professors, have many more firstborns than would be expected by chance. However, Rohrer, Egloff, and Schmukle (2015) analyzed birth order effects on the big five personality traits and found a meaningful difference between 1st and 2nd born children on the "Intellect" facet of the personality trait "Openness to Experience", but no differences on the other traits. In their study, they only used ordinary least-squares regression to conduct the analysis. Therefore, this current analysis aims to examine whether other models can provide better prediction results.
- Research question: can personality variables (along with other variables like age, gender, and English as native language) predict whether a person is the firstborn in his/her family?
- Why machine learning: machine learning is proper here as the main purpose is to predict instead of explain in this case. What's more, the dataset is large enough to conduct this type of analysis.

```
dataset <- read_excel("../data/firstborn.xlsx", col_names = TRUE)
# clean the data
firstborn <- dataset %>%
  dplyr::select(-starts_with("Q")) %>%
  filter(birthn > 1 & birthpos !=0 & birthn >= birthpos) %>%
  filter(!is.na(birthpos)) %>%
  na_if(., 0) %>%
  mutate(firstborn = as.factor(ifelse(birthpos == 1, "yes", "no"))) %>%
  dplyr::select(-birthpos, -birthn, -country, -source, -screensize, -introelapse, -testelapse, -endelaps
```

```
missing_value_tbl <- firstborn %>%
  mutate_all(~ ifelse(is.na(.), 1, 0))
firstborn$missings <- rowSums(missing_value_tbl)

firstborn <- firstborn %>%
  filter(missings <= 5) %>%
  dplyr::select(-missings)

holdout_indices <- createDataPartition(firstborn$firstborn, p=0.6, list=FALSE)
train_tbl <- firstborn[holdout_indices, ]
test_tbl <- firstborn[-holdout_indices, ]
```

Data analysis

- We chose three machine learning: glm (binomial), naive bayes, and knn. The reason to choose these three methods is because that we have a categorical outcome and the main purpose here is to do classficiation. Therefore, we have chosen three methods that are commonly used and whose purpose is classification.

```
fold_indices <- createFolds(holdout_indices, 10)
no_of_cores <- detectCores()
cl <- makeCluster(no_of_cores-1)
registerDoParallel(cl)

glm_model <- train(
  firstborn ~ .,
  train_tbl,
  method = "glm",
  family = "binomial",
  preProcess = c("center", "scale", "zv", "knnImpute", "pca"),
  na.action = na.pass,
  trControl = trainControl(method = "cv", number = 10, verboseIter = T, index = fold_indices)
)
```

```
## Aggregating results
## Fitting final model on full training set
```

```
glm_model
```

```
## Generalized Linear Model
##
## 21381 samples
##    53 predictor
##     2 classes: 'no', 'yes'
##
## Pre-processing: centered (53), scaled (53), nearest neighbor
##  imputation (53), principal component signal extraction (53)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2138, 2137, 2139, 2139, 2137, 2138, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5645979  0.05840415
```

```r
confusionMatrix(predict(glm_model, test_tbl, na.action = na.pass), test_tbl$firstborn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7058 5008
##        yes 1062 1125
##
##                Accuracy : 0.5741
##                  95% CI : (0.566, 0.5823)
##     No Information Rate : 0.5697
##     P-Value [Acc > NIR] : 0.1452
##
##                   Kappa : 0.0571
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8692
##             Specificity : 0.1834
##          Pos Pred Value : 0.5849
##          Neg Pred Value : 0.5144
##              Prevalence : 0.5697
##          Detection Rate : 0.4952
##    Detection Prevalence : 0.8466
##       Balanced Accuracy : 0.5263
##
##        'Positive' Class : no
##
```

```r
nb_model <- train(
  firstborn ~ .,
  train_tbl,
  method = "nb",
  metric = "Kappa",
  tuneLength = 2,
  preProcess = c("knnImpute", "zv","center", "scale", "pca"),
  trControl = trainControl(method = "cv", number = 10, verboseIter = T, index = fold_indices),
  na.action = na.pass
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting fL = 0, usekernel = FALSE, adjust = 1 on full training set
```

```r
nb_model
```

```
## Naive Bayes
##
## 21381 samples
##    53 predictor
##     2 classes: 'no', 'yes'
##
## Pre-processing: nearest neighbor imputation (53), centered (53),
##  scaled (53), principal component signal extraction (53)
```

```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2138, 2137, 2139, 2139, 2137, 2138, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.5473448  0.03958094
##    TRUE      0.5338852  0.02885353
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE
##  and adjust = 1.
```

```
confusionMatrix(predict(nb_model, test_tbl, na.action = na.pass), test_tbl$firstborn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7000 5033
##        yes 1120 1100
##
##                Accuracy : 0.5683
##                  95% CI : (0.5601, 0.5765)
##     No Information Rate : 0.5697
##     P-Value [Acc > NIR] : 0.6358
##
##                   Kappa : 0.0449
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8621
##             Specificity : 0.1794
##          Pos Pred Value : 0.5817
##          Neg Pred Value : 0.4955
##              Prevalence : 0.5697
##          Detection Rate : 0.4911
##    Detection Prevalence : 0.8442
##       Balanced Accuracy : 0.5207
##
##        'Positive' Class : no
##
```

```
knn_model <- train(
  firstborn ~ .,
  train_tbl,
  method = "knn",
  metric = "Kappa",
  tuneLength = 2,
  preProcess = c("knnImpute", "zv","center", "scale", "pca"),
  trControl = trainControl(method = "cv", number = 10, verboseIter = T, index = fold_indices),
  na.action = na.pass
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting k = 7 on full training set
```

```
knn_model
```

```
## k-Nearest Neighbors
##
## 21381 samples
##    53 predictor
##     2 classes: 'no', 'yes'
##
## Pre-processing: nearest neighbor imputation (53), centered (53),
##  scaled (53), principal component signal extraction (53)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2138, 2137, 2139, 2139, 2137, 2138, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.5255289  0.01901592
##   7  0.5284754  0.01994088
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
confusionMatrix(predict(knn_model, test_tbl, na.action = na.pass), test_tbl$firstborn)
```

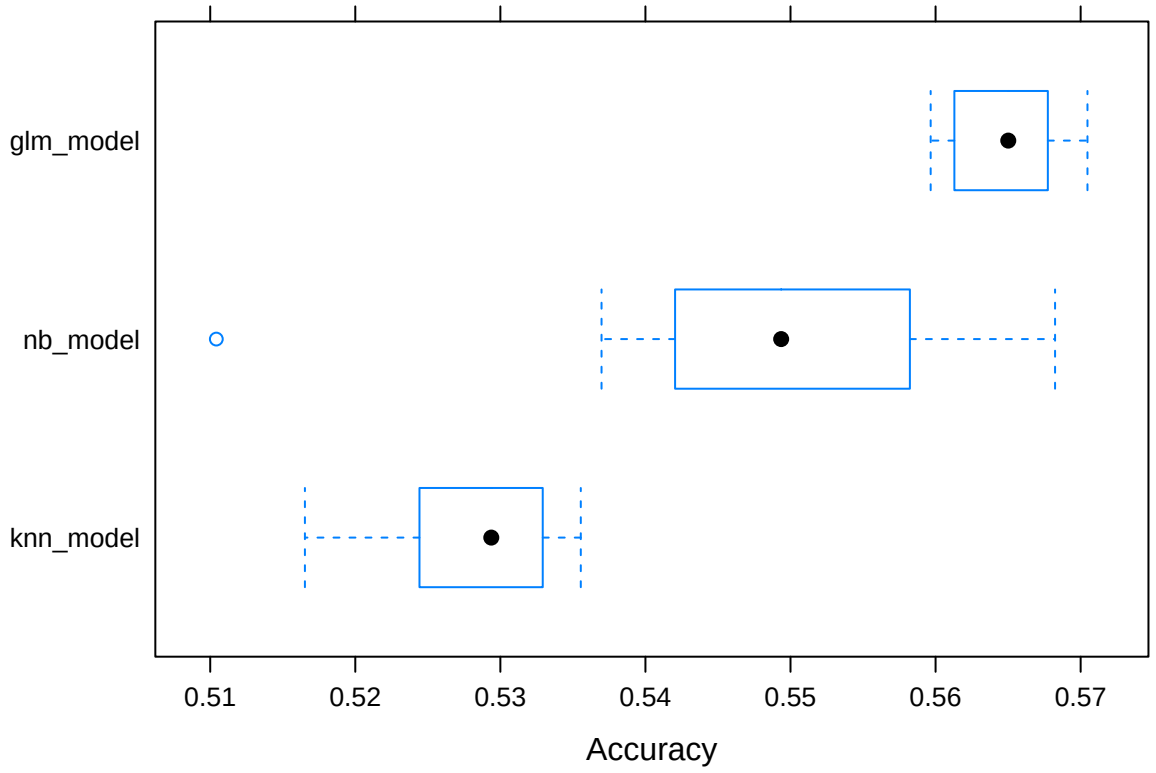
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5290 3800
##        yes 2830 2333
##
##                Accuracy : 0.5348
##                  95% CI : (0.5266, 0.5431)
##     No Information Rate : 0.5697
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0325
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6515
##             Specificity : 0.3804
##          Pos Pred Value : 0.5820
##          Neg Pred Value : 0.4519
##              Prevalence : 0.5697
##          Detection Rate : 0.3711
##    Detection Prevalence : 0.6378
##       Balanced Accuracy : 0.5159
##
##        'Positive' Class : no
##
```

```
stopCluster(cl)
```

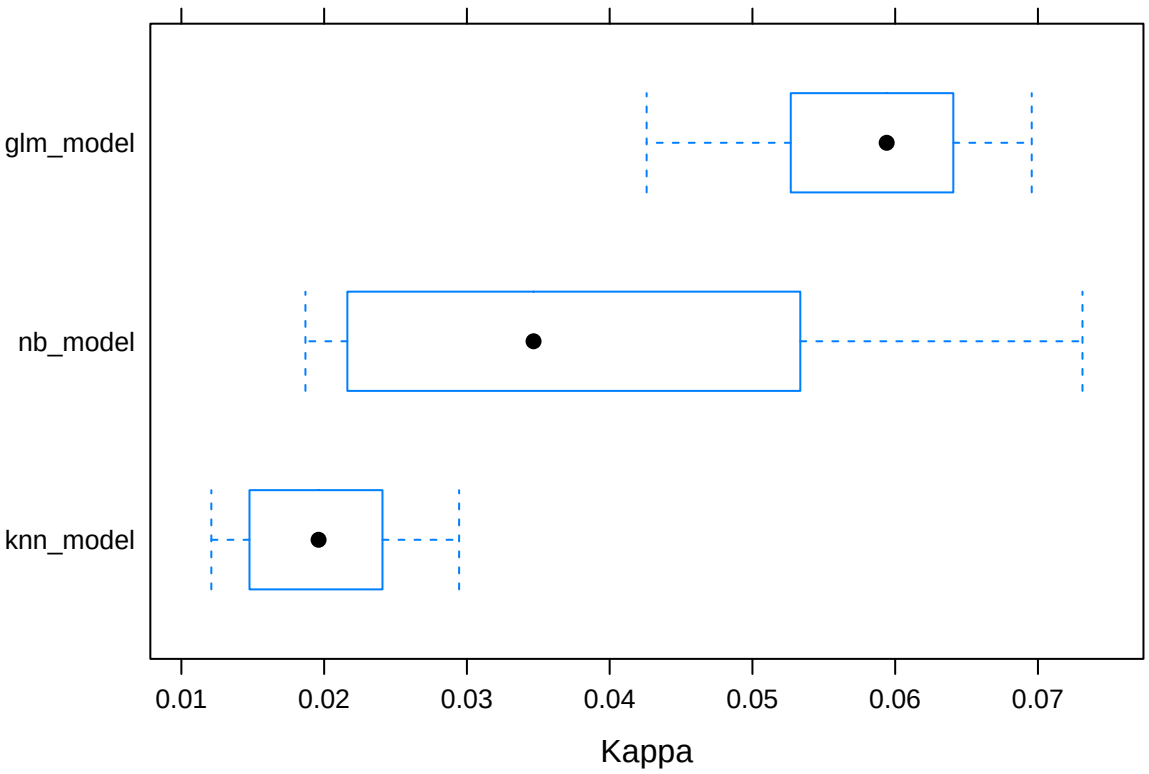
Model comparison

- Based on the comparison, we found that glm model performs the best among the three models. However, based on the Kappa and Accuracy, none of the models turns out to predict firstborn to a reasonable level (Kappas are smaller than 0.01 and Accuracies are about the same as random guessing). The findings further support Rohrer, Egloff, and Schmukle (2015)'s finding in that personalities seem to not have an effects on whether a person is first born or not.

```
model_list <- list(glm_model = glm_model, nb_model = nb_model, knn_model = knn_model)
resamples <- resamples(model_list)
bwplot(resamples, metric = "Accuracy")
```




```
bwplot(resamples, metric = "Kappa")
```


glm_model
nb_model
knn_model
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Kappa