P8451 Machine Learning in Public Health - Assignment 5

2023-2-21

In preparation for all the analyses below, we will load the following libraries:

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2  
## ──

## ✔ tibble 3.1.8 ✔ dplyr 1.1.0  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.4 ✔ forcats 1.0.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::lift() masks caret::lift()

library(dplyr)  
library(stats)  
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(cluster)  
library(ggpubr)  
library(pscl)

## Classes and Methods for R developed in the  
## Political Science Computational Laboratory  
## Department of Political Science  
## Stanford University  
## Simon Jackman  
## hurdle and zeroinfl functions by Achim Zeileis

# Part 0: Data Preprocessing

We will begin by importing the drug use and personality trait survey data using the read\_csv function. Next, we will clean the data by first applying the clean\_names function, then applying the mutate function to rename the ID variable and to convert the alc\_consumption variable from a character to a 2-level factor variable. The following 8 features are included in this data set:

* alc\_consumption
* neurotocism\_score
* extroversion\_score
* openness\_score
* agreeableness\_score
* conscientiousness\_score
* impulsiveness\_score
* sens\_seeking\_score

Finally, we remove entries with NA using na.omit, any duplicate ID entries using the distinct function, and finally drop the unwanted ID entry using select.

alcohol\_use = read\_csv("./alcohol\_use.csv") %>%   
 janitor::clean\_names() %>%   
 mutate(id = x1,   
 alc\_consumption = factor(alc\_consumption,   
 labels = c("Current User", "Not Current User"))) %>%   
 na.omit() %>%   
 distinct(id, .keep\_all = TRUE) %>%   
 select(alc\_consumption, neurotocism\_score, extroversion\_score, openness\_score,  
 agreeableness\_score, conscientiousness\_score, impulsiveness\_score,   
 sens\_seeking\_score)

## New names:  
## Rows: 1885 Columns: 9  
## ── Column specification  
## ──────────────────────────────────────────────────────── Delimiter: "," chr  
## (1): alc\_consumption dbl (8): ...1, neurotocism\_score, extroversion\_score,  
## openness\_score, agreea...  
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ  
## Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
## • `` -> `...1`

## Feature Selection: Identifying and Removing Correlated Predictors

Many machine learning algorithms are unable to differentiate between highly correlated features. As such, we want to identify highly correlated features that present the same mathematical information and subsequently remove them, to avoid introducing error in our approach.

To complete this feature selection process, we will first select only the numeric variables in our alcohol\_use data set, since correlations can only be assessed with numeric variables. We will then apply the cor function that will calculate correlations. These calculated correlations will then be fed into the findCorrelation function with a cutoff of **0.4**. The features that correlated at 0.4 and above will be stored in a new objected labeled as high\_correlations.

alcohol\_use\_numeric = alcohol\_use %>%   
 select(where(is.numeric))   
  
correlations = cor(alcohol\_use\_numeric, use = "complete.obs")  
  
high\_correlations = findCorrelation(correlations, cutoff = 0.4)

The high\_correlations object contains the indexes of 2 correlated predictors: 7 and 2. These correspond to the extroversion\_score and sens\_seeking\_score features. In the code chunk below, will remove these highly correlated features.

alcohol\_use\_tidy = alcohol\_use\_numeric[ , -high\_correlations]

## Centering and Scaling

Below, we center and scale these data. In general, it is always good practice to do so!

preprocess\_setup <- preProcess(alcohol\_use\_tidy, method = c("center", "scale"))

## Partitioning Data

For the purposes of this analysis, we will partition the data into training and testing using a 70/30 split. This process involves applying the createDataPartition function to generate a set of training and testing data with equal proportion of individual with the outcome of interest, i.e., alc\_consumption. The new object train\_index contains all the indexes of the rows in the original data set contained in the 70% split. The rows indexed to be in the 70% is assigned to a new training data set, and the remaining 30% is assigned to a new testing data set.

alcohol\_use\_tidy$alc\_consumption = alcohol\_use$alc\_consumption  
  
train\_index = createDataPartition(alcohol\_use\_tidy$alc\_consumption, p = 0.7, list = FALSE)  
  
alcohol\_use\_train <- alcohol\_use\_tidy[train\_index,]  
alcohol\_use\_test <- alcohol\_use\_tidy[-train\_index,]

# Part I: Creating Three Different Models

For the purposes of this analysis, we will create and compare the following models:

1. Elastic Net Model that chooses alpha and lambda via cross-validation using all features
2. Traditional Logistic Regression Model using all features
3. Lasso Model using all features

## 1.1 Model 1: Elastic Net with All Features

In the code chunk below, we will use the trainControl function to set our validation method. For the purposes of this analysis, we will use the 10-fold cross validation method.

control.settings =   
 trainControl(method = "cv", number = 10)

These control settings can now be applied within the train function, which will be used to implement our algorithms. We also apply the tuneLength function to set the number of combinations of different values of alpha and lambda to compare. In this analysis, we will set tunelength to 10. Finally, we can apply the coef function to model the coefficients at the lambda and alpha values that minimize the RMSE.

set.seed(123)  
alcohol\_use\_model\_1 =   
 train(alc\_consumption ~ neurotocism\_score + openness\_score + agreeableness\_score +   
 conscientiousness\_score + impulsiveness\_score,   
 data = alcohol\_use\_train,   
 method = "glmnet",   
 preProc = c("center", "scale"),   
 trControl = control.settings,   
 tuneLength = 10)  
  
alcohol\_use\_model\_1$bestTune %>%   
 knitr::kable()

|  | alpha | lambda |
| --- | --- | --- |
| 36 | 0.4 | 0.2633636 |

coef(alcohol\_use\_model\_1$finalModel, alcohol\_use\_model\_1$bestTune$lambda)

## 6 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) -0.1391271  
## neurotocism\_score .   
## openness\_score .   
## agreeableness\_score .   
## conscientiousness\_score .   
## impulsiveness\_score -0.5054400

Based on the output above, the alpha and lambda values that minimize the RMSE are 0.4 and 0.26, respectively.

## 1.2 Model 2: Traditional Logistic Regression Model with All Features

Below we construct a logistic regression model using the glm function, with alcohol consumption as the outcome of interest, and neurotocism, openness, agreeableness, conscientiousness, and impulsiveness scores as the independent variables of interest. To do so, we apply the glm function. To assess how the model fits the data, we can compute the McFadden’s R^2.

alcohol\_use\_model\_2 =   
 glm(alc\_consumption ~ neurotocism\_score + openness\_score + agreeableness\_score +   
 conscientiousness\_score + impulsiveness\_score,   
 data = alcohol\_use\_train,   
 family = binomial())  
  
pscl::pR2(alcohol\_use\_model\_2)

## fitting null model for pseudo-r2

## llh llhNull G2 McFadden r2ML r2CU   
## -604.6943628 -912.1507779 614.9128302 0.3370675 0.3723942 0.4972312

The McFadden’s R^2 takes on a value of 0.34. Since 0.34 lies between 0.2 and 0.4, there is evidence of very good model fit.

## 1.3 Model 3: Lasso Model with All Features

To generate a lasso model, we will set alpha equal to 1. We will apply cross validation methods to select the value of lambda that minimizes the RMSE. This process involves using a tuning grid for lambda, and the parameters of the grid search are determined by first running a rough grid search, then narrowing down the range of values to yield more precise values.

lambda = seq(0, 1, length = 20)  
lambda\_grid = expand.grid(alpha = 1, lambda = lambda)  
  
set.seed(123)  
alcohol\_use\_model\_3 =   
 train(alc\_consumption ~ neurotocism\_score + openness\_score + agreeableness\_score +   
 conscientiousness\_score + impulsiveness\_score,   
 data = alcohol\_use\_train,   
 method = "glmnet",   
 preProc = c("center", "scale"),   
 trControl = control.settings,   
 tuneGrid = lambda\_grid)  
  
alcohol\_use\_model\_3$bestTune

## alpha lambda  
## 5 1 0.2105263

Based on the output above, we can see that the

## 1.4 Choosing a Final Model