Machine Learning for Epi: Assignment 5

2023-02-21

# Predicting Current Alcohol Consumption from Behavioral Scores

## Description of Data

The data we will be using are from the online survey related to drug and alcohol use and personality traits. These data were collected as part of an online survey related to drug and alcohol use and personality traits from the UCI Machine Learning Repository. *We will be using this dataset to try to identify the most important behavioral predictors of alcohol consumption.* We have restricted the dataset to 7 features and an outcome which distinguishes those who reported current alcohol use (defined as alcohol use in the past month or more frequently) vs no current use.

### Step 1: Load data and prepare for analysis

The code chunk below loads the Alcohol Consumption survey data and strips the id variable, omits missing observations, and converts the outcome variable, alc\_consumption to a factor variable.

alcohol\_use = readr::read\_csv("./alcohol\_use.csv") %>%   
 mutate(alc\_consumption = as.factor(alc\_consumption),   
 alc\_consumption = fct\_relevel(alc\_consumption, c("NotCurrentUse", "CurrentUse"))) %>%   
 dplyr::select(-`...1`) %>%   
 drop\_na()   
   
summary(alcohol\_use) %>% knitr::kable(digits = 2)

|  | neurotocism\_score | extroversion\_score | openness\_score | agreeableness\_score | conscientiousness\_score | impulsiveness\_score | sens\_seeking\_score | alc\_consumption |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Min. :-3.464360 | Min. :-3.273930 | Min. :-3.273930 | Min. :-3.464360 | Min. :-3.464360 | Min. :-2.555240 | Min. :-2.078480 | NotCurrentUse: 881 |
|  | 1st Qu.:-0.678250 | 1st Qu.:-0.695090 | 1st Qu.:-0.717270 | 1st Qu.:-0.606330 | 1st Qu.:-0.652530 | 1st Qu.:-0.711260 | 1st Qu.:-0.525930 | CurrentUse :1004 |
|  | Median : 0.042570 | Median : 0.003320 | Median :-0.019280 | Median :-0.017290 | Median :-0.006650 | Median :-0.217120 | Median : 0.079870 | NA |
|  | Mean : 0.000047 | Mean :-0.000163 | Mean :-0.000534 | Mean :-0.000245 | Mean :-0.000386 | Mean : 0.007216 | Mean :-0.003292 | NA |
|  | 3rd Qu.: 0.629670 | 3rd Qu.: 0.637790 | 3rd Qu.: 0.723300 | 3rd Qu.: 0.760960 | 3rd Qu.: 0.584890 | 3rd Qu.: 0.529750 | 3rd Qu.: 0.765400 | NA |
|  | Max. : 3.273930 | Max. : 3.273930 | Max. : 2.901610 | Max. : 3.464360 | Max. : 3.464360 | Max. : 2.901610 | Max. : 1.921730 | NA |

Our resulting variables include 7 behavioral/personality scores, which are numeric, and our binary outcome variable, alc\_consumption. Based on the summary, we can see that the distribution is similar across the numeric variables, and appears to be already scaled and centered. Therefore, will skip the centering and scaling steps in pre-processing.

### Step 2: Partition the data

The code chunk below partitions the data into training and testing sets, using a 70/30 split.

set.seed(123)  
  
#Creating balanced partitions in the data  
train\_index = createDataPartition(alcohol\_use$alc\_consumption, p = 0.7, list = FALSE)  
  
alc\_train = alcohol\_use[train\_index,]  
alc\_test = alcohol\_use[-train\_index,]  
  
#Check distribution of the outcome between train and test data  
summary(alc\_train$alc\_consumption)

## NotCurrentUse CurrentUse   
## 617 703

summary(alc\_test$alc\_consumption)

## NotCurrentUse CurrentUse   
## 264 301

We can see that there are similar distributions of the variable alc\_consumption, with approximately 53% of cases of alcohol use across both the training and testing sets, indicating that the data were successfully partitioned.

### Step 4: Construct logistic regression models to predict healthy days

We will fit 3 regularized and traditional logistic models to predict current alcohol consumption. (feature name: alc\_consumption).

* Model 1 (mod\_elastic): An elastic net model based on all features, using cross-validation to choose alpha and lambda.
* Model 2 (mod\_baseline): An traditional logistic model based on all features, serving as our ‘baseline model’.
* Model 3 (mod\_lasso): A LASSO logistic model based on all features.

These models will be trained on the training dataset using 10-fold cross validation.

set.seed(123)  
  
mod\_elastic = train(alc\_consumption ~ ., data = alc\_train, method = "glmnet",   
 trControl = trainControl("cv", number = 10),   
 tuneLength = 10)  
#Print the values of alpha and lambda that gave best prediction  
mod\_elastic$bestTune

## alpha lambda  
## 54 0.6 0.263144

#Print all of the options examined  
mod\_elastic$results

## alpha lambda Accuracy Kappa AccuracySD KappaSD  
## 1 0.1 0.0003244162 0.8016075 0.6018740 0.04278081 0.08383610  
## 2 0.1 0.0007494434 0.8016075 0.6018740 0.04278081 0.08383610  
## 3 0.1 0.0017313116 0.8015960 0.6018556 0.04210644 0.08243270  
## 4 0.1 0.0039995543 0.8008384 0.6002209 0.04168952 0.08158922  
## 5 0.1 0.0092394891 0.8031112 0.6047407 0.04153189 0.08129602  
## 6 0.1 0.0213444181 0.8023422 0.6029959 0.03765752 0.07382899  
## 7 0.1 0.0493083742 0.8000293 0.5981281 0.03546432 0.06945564  
## 8 0.1 0.1139087396 0.8022849 0.6022622 0.03287346 0.06458279  
## 9 0.1 0.2631439626 0.8075996 0.6119584 0.03214767 0.06370839  
## 10 0.2 0.0003244162 0.8008499 0.6003803 0.04281987 0.08397306  
## 11 0.2 0.0007494434 0.8008499 0.6003803 0.04281987 0.08397306  
## 12 0.2 0.0017313116 0.8000865 0.5989021 0.04302522 0.08432765  
## 13 0.2 0.0039995543 0.7993233 0.5972250 0.04063941 0.07954951  
## 14 0.2 0.0092394891 0.7978023 0.5941654 0.04063619 0.07946837  
## 15 0.2 0.0213444181 0.8023365 0.6030604 0.03633647 0.07122141  
## 16 0.2 0.0493083742 0.8022906 0.6026368 0.03452859 0.06773963  
## 17 0.2 0.1139087396 0.8091033 0.6157212 0.02927197 0.05715894  
## 18 0.2 0.2631439626 0.8159275 0.6286947 0.02970017 0.05876404  
## 19 0.3 0.0003244162 0.8008499 0.6004394 0.04130380 0.08089515  
## 20 0.3 0.0007494434 0.8008499 0.6004394 0.04130380 0.08089515  
## 21 0.3 0.0017313116 0.7993290 0.5973999 0.04198424 0.08225306  
## 22 0.3 0.0039995543 0.7978138 0.5943415 0.03961200 0.07741842  
## 23 0.3 0.0092394891 0.7978023 0.5941654 0.04063619 0.07946837  
## 24 0.3 0.0213444181 0.8023365 0.6030672 0.03771431 0.07396782  
## 25 0.3 0.0493083742 0.8068592 0.6116271 0.03525671 0.06954890  
## 26 0.3 0.1139087396 0.8113818 0.6202613 0.03064663 0.06015933  
## 27 0.3 0.2631439626 0.8219997 0.6401519 0.04067580 0.08069768  
## 28 0.4 0.0003244162 0.8008499 0.6004394 0.04130380 0.08089515  
## 29 0.4 0.0007494434 0.8008499 0.6004394 0.04130380 0.08089515  
## 30 0.4 0.0017313116 0.8008499 0.6004579 0.04177142 0.08175156  
## 31 0.4 0.0039995543 0.7985772 0.5958382 0.03992725 0.07802819  
## 32 0.4 0.0092394891 0.7985657 0.5957407 0.04015751 0.07841550  
## 33 0.4 0.0213444181 0.8023536 0.6032820 0.04249304 0.08306868  
## 34 0.4 0.0493083742 0.8091263 0.6162111 0.03238241 0.06368116  
## 35 0.4 0.1139087396 0.8136775 0.6247093 0.03731296 0.07311969  
## 36 0.4 0.2631439626 0.8356537 0.6666939 0.03097882 0.06226861  
## 37 0.5 0.0003244162 0.8008499 0.6004394 0.04130380 0.08089515  
## 38 0.5 0.0007494434 0.8008499 0.6004394 0.04130380 0.08089515  
## 39 0.5 0.0017313116 0.8008499 0.6004579 0.04177142 0.08175156  
## 40 0.5 0.0039995543 0.7985772 0.5958382 0.03992725 0.07802819  
## 41 0.5 0.0092394891 0.7978195 0.5943344 0.04325059 0.08439964  
## 42 0.5 0.0213444181 0.8008384 0.6002155 0.04213457 0.08243496  
## 43 0.5 0.0493083742 0.8046037 0.6073607 0.03484850 0.06837477  
## 44 0.5 0.1139087396 0.8174769 0.6322476 0.03711151 0.07282530  
## 45 0.5 0.2631439626 0.8394472 0.6738069 0.03603032 0.07254530  
## 46 0.6 0.0003244162 0.8023708 0.6034217 0.04164196 0.08154400  
## 47 0.6 0.0007494434 0.8023708 0.6034217 0.04164196 0.08154400  
## 48 0.6 0.0017313116 0.8008499 0.6004579 0.04177142 0.08175156  
## 49 0.6 0.0039995543 0.7993404 0.5975217 0.04205695 0.08211022  
## 50 0.6 0.0092394891 0.7985771 0.5958905 0.04429049 0.08656047  
## 51 0.6 0.0213444181 0.7962929 0.5911070 0.04439497 0.08687512  
## 52 0.6 0.0493083742 0.8038288 0.6058119 0.03359566 0.06584030  
## 53 0.6 0.1139087396 0.8167136 0.6308317 0.03788489 0.07430556  
## 54 0.6 0.2631439626 0.8621812 0.7180342 0.02533697 0.05243557  
## 55 0.7 0.0003244162 0.8031284 0.6049892 0.04157101 0.08133799  
## 56 0.7 0.0007494434 0.8031284 0.6049892 0.04157101 0.08133799  
## 57 0.7 0.0017313116 0.8016075 0.6019436 0.04188390 0.08197007  
## 58 0.7 0.0039995543 0.8000980 0.5990120 0.04279898 0.08357238  
## 59 0.7 0.0092394891 0.7985771 0.5959744 0.04429049 0.08650471  
## 60 0.7 0.0213444181 0.7985542 0.5957236 0.04207805 0.08229265  
## 61 0.7 0.0493083742 0.8068649 0.6118223 0.03537604 0.06962481  
## 62 0.7 0.1139087396 0.8060958 0.6102246 0.03668184 0.07203653  
## 63 0.7 0.2631439626 0.8614236 0.7164602 0.02540180 0.05255109  
## 64 0.8 0.0003244162 0.8031284 0.6049892 0.04157101 0.08133799  
## 65 0.8 0.0007494434 0.8031284 0.6049892 0.04157101 0.08133799  
## 66 0.8 0.0017313116 0.8000980 0.5990074 0.04219879 0.08238663  
## 67 0.8 0.0039995543 0.8000922 0.5990226 0.04327835 0.08445849  
## 68 0.8 0.0092394891 0.7978194 0.5945037 0.04496887 0.08788095  
## 69 0.8 0.0213444181 0.7970562 0.5928388 0.04251158 0.08331792  
## 70 0.8 0.0493083742 0.8053382 0.6088447 0.03577617 0.07039588  
## 71 0.8 0.1139087396 0.8038115 0.6058734 0.03609751 0.07091970  
## 72 0.8 0.2631439626 0.8614236 0.7164602 0.02540180 0.05255109  
## 73 0.9 0.0003244162 0.8031284 0.6049892 0.04157101 0.08133799  
## 74 0.9 0.0007494434 0.8031284 0.6049892 0.04157101 0.08133799  
## 75 0.9 0.0017313116 0.8000980 0.5990074 0.04219879 0.08238663  
## 76 0.9 0.0039995543 0.8000922 0.5990226 0.04327835 0.08445849  
## 77 0.9 0.0092394891 0.7948004 0.5885485 0.04899312 0.09595596  
## 78 0.9 0.0213444181 0.7955582 0.5900000 0.04227690 0.08275209  
## 79 0.9 0.0493083742 0.7932338 0.5852528 0.03688104 0.07274780  
## 80 0.9 0.1139087396 0.7743112 0.5496789 0.03867606 0.07536309  
## 81 0.9 0.2631439626 0.8614236 0.7164602 0.02540180 0.05255109  
## 82 1.0 0.0003244162 0.8023765 0.6035553 0.04309430 0.08417968  
## 83 1.0 0.0007494434 0.8023765 0.6035553 0.04309430 0.08417968  
## 84 1.0 0.0017313116 0.8008556 0.6004977 0.04292352 0.08381462  
## 85 1.0 0.0039995543 0.8000922 0.5990226 0.04327835 0.08445849  
## 86 1.0 0.0092394891 0.7948119 0.5885727 0.04982453 0.09763335  
## 87 1.0 0.0213444181 0.7978368 0.5946788 0.04373777 0.08579727  
## 88 1.0 0.0493083742 0.7819100 0.5635869 0.04332378 0.08494978  
## 89 1.0 0.1139087396 0.7720384 0.5454357 0.03744922 0.07273810  
## 90 1.0 0.2631439626 0.6909366 0.3530459 0.01434547 0.03074794

# Model coefficients  
coef(mod\_elastic$finalModel, mod\_elastic$bestTune$lambda)

## 8 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 0.14186305  
## neurotocism\_score .   
## extroversion\_score .   
## openness\_score .   
## agreeableness\_score .   
## conscientiousness\_score .   
## impulsiveness\_score 0.44092293  
## sens\_seeking\_score 0.01380889

mod\_baseline = train(alc\_consumption ~ ., data = alc\_train, method = "glm",   
 trControl = trainControl("cv", number = 10))  
  
coef(mod\_baseline$finalModel)

## (Intercept) neurotocism\_score extroversion\_score   
## 0.256267278 0.076800543 0.207811698   
## openness\_score agreeableness\_score conscientiousness\_score   
## 0.070023981 0.109476065 0.009103264   
## impulsiveness\_score sens\_seeking\_score   
## 1.862708564 0.304877917

#Create grid to search lambda  
lambda = 10^seq(-3, 3, length = 100)  
  
#Fit model with tuneGrid  
mod\_lasso = train(alc\_consumption ~ ., data = alc\_train, method = "glmnet",   
 trControl = trainControl("cv", number = 10), tuneGrid = expand.grid(alpha = 1, lambda = lambda))  
  
#Print the values of alpha and lambda that gave best prediction  
mod\_lasso$bestTune

## alpha lambda  
## 40 1 0.231013

#Print all of the options examined  
mod\_lasso$results

## alpha lambda Accuracy Kappa AccuracySD KappaSD  
## 1 1 1.000000e-03 0.7984743 0.5951636 0.035294329 0.07047331  
## 2 1 1.149757e-03 0.7984743 0.5951636 0.035294329 0.07047331  
## 3 1 1.321941e-03 0.7984743 0.5951636 0.035294329 0.07047331  
## 4 1 1.519911e-03 0.7992319 0.5967166 0.036018889 0.07196898  
## 5 1 1.747528e-03 0.7992319 0.5967166 0.036018889 0.07196898  
## 6 1 2.009233e-03 0.7999894 0.5981968 0.036010283 0.07197989  
## 7 1 2.310130e-03 0.7999894 0.5981968 0.036010283 0.07197989  
## 8 1 2.656088e-03 0.8007413 0.5996450 0.034596017 0.06932304  
## 9 1 3.053856e-03 0.8007413 0.5996450 0.034596017 0.06932304  
## 10 1 3.511192e-03 0.8007413 0.5996450 0.034596017 0.06932304  
## 11 1 4.037017e-03 0.8014989 0.6011333 0.034734379 0.06958331  
## 12 1 4.641589e-03 0.7999951 0.5982214 0.036343677 0.07257427  
## 13 1 5.336699e-03 0.7999951 0.5982214 0.036343677 0.07257427  
## 14 1 6.135907e-03 0.7984799 0.5952901 0.034379912 0.06840528  
## 15 1 7.054802e-03 0.7984741 0.5953032 0.035340113 0.07021539  
## 16 1 8.111308e-03 0.7977165 0.5938147 0.034582904 0.06870490  
## 17 1 9.326033e-03 0.7977107 0.5938143 0.034992224 0.06943630  
## 18 1 1.072267e-02 0.7984741 0.5953986 0.034423235 0.06816939  
## 19 1 1.232847e-02 0.7992375 0.5970939 0.034389307 0.06784940  
## 20 1 1.417474e-02 0.7984856 0.5956960 0.032641170 0.06436000  
## 21 1 1.629751e-02 0.7969819 0.5926868 0.031896699 0.06277293  
## 22 1 1.873817e-02 0.7954610 0.5897342 0.033352741 0.06560095  
## 23 1 2.154435e-02 0.7962070 0.5913380 0.035749964 0.07017988  
## 24 1 2.477076e-02 0.7946974 0.5884466 0.037311995 0.07308304  
## 25 1 2.848036e-02 0.7954607 0.5899616 0.037689689 0.07367758  
## 26 1 3.274549e-02 0.7878732 0.5750144 0.038138031 0.07437828  
## 27 1 3.764936e-02 0.7848372 0.5690133 0.033033647 0.06449076  
## 28 1 4.328761e-02 0.7840853 0.5675709 0.034308416 0.06691584  
## 29 1 4.977024e-02 0.7840853 0.5675709 0.034308416 0.06691584  
## 30 1 5.722368e-02 0.7833335 0.5664492 0.038606886 0.07508964  
## 31 1 6.579332e-02 0.7765095 0.5533844 0.036616719 0.07131605  
## 32 1 7.564633e-02 0.7704432 0.5419956 0.036435711 0.07122771  
## 33 1 8.697490e-02 0.7719583 0.5451808 0.035442601 0.06923229  
## 34 1 1.000000e-01 0.7719583 0.5451808 0.035442601 0.06923229  
## 35 1 1.149757e-01 0.7719583 0.5451808 0.035442601 0.06923229  
## 36 1 1.321941e-01 0.7719583 0.5451808 0.035442601 0.06923229  
## 37 1 1.519911e-01 0.8090796 0.6162801 0.066612869 0.12939145  
## 38 1 1.747528e-01 0.8613432 0.7161366 0.024321126 0.05086916  
## 39 1 2.009233e-01 0.8613432 0.7161366 0.024321126 0.05086916  
## 40 1 2.310130e-01 0.8613432 0.7161366 0.024321126 0.05086916  
## 41 1 2.656088e-01 0.6909253 0.3525186 0.037256287 0.08202090  
## 42 1 3.053856e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 43 1 3.511192e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 44 1 4.037017e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 45 1 4.641589e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 46 1 5.336699e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 47 1 6.135907e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 48 1 7.054802e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 49 1 8.111308e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 50 1 9.326033e-01 0.5325765 0.0000000 0.002654595 0.00000000  
## 51 1 1.072267e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 52 1 1.232847e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 53 1 1.417474e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 54 1 1.629751e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 55 1 1.873817e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 56 1 2.154435e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 57 1 2.477076e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 58 1 2.848036e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 59 1 3.274549e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 60 1 3.764936e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 61 1 4.328761e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 62 1 4.977024e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 63 1 5.722368e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 64 1 6.579332e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 65 1 7.564633e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 66 1 8.697490e+00 0.5325765 0.0000000 0.002654595 0.00000000  
## 67 1 1.000000e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 68 1 1.149757e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 69 1 1.321941e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 70 1 1.519911e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 71 1 1.747528e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 72 1 2.009233e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 73 1 2.310130e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 74 1 2.656088e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 75 1 3.053856e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 76 1 3.511192e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 77 1 4.037017e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 78 1 4.641589e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 79 1 5.336699e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 80 1 6.135907e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 81 1 7.054802e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 82 1 8.111308e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 83 1 9.326033e+01 0.5325765 0.0000000 0.002654595 0.00000000  
## 84 1 1.072267e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 85 1 1.232847e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 86 1 1.417474e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 87 1 1.629751e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 88 1 1.873817e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 89 1 2.154435e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 90 1 2.477076e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 91 1 2.848036e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 92 1 3.274549e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 93 1 3.764936e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 94 1 4.328761e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 95 1 4.977024e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 96 1 5.722368e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 97 1 6.579332e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 98 1 7.564633e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 99 1 8.697490e+02 0.5325765 0.0000000 0.002654595 0.00000000  
## 100 1 1.000000e+03 0.5325765 0.0000000 0.002654595 0.00000000

# Model coefficients  
coef(mod\_lasso$finalModel, mod\_lasso$bestTune$lambda)

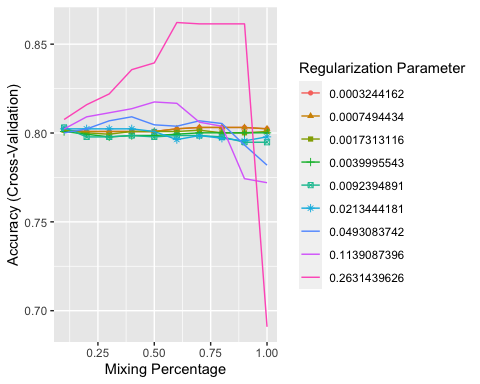
## 8 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 0.1370717  
## neurotocism\_score .   
## extroversion\_score .   
## openness\_score .   
## agreeableness\_score .   
## conscientiousness\_score .   
## impulsiveness\_score 0.3152417  
## sens\_seeking\_score .

Results of the cross-validated training shows the model with all variables, lm\_all, has a slightly lower RMSE compared to the model with only health habits. However, we want to confirm this by running each model on the testing data.

### Step 5: Model Evaluation

Next, to determine the preferred prediction model, we will apply both models to the test data, and compare their performance based on the RMSE, which is an evaluation metric used for linear regression models.

## ELASTIC MODEL  
# Make predictions in test set  
en\_pred = mod\_elastic %>% predict(alc\_test)  
alc\_test = alc\_test %>% mutate(en\_pred = as.factor(en\_pred))  
  
# Model prediction performance  
elastic\_perf = postResample(en\_pred, alc\_test$alc\_consumption) %>% as\_tibble\_row()  
  
ggplot(mod\_elastic)



confusionMatrix(data = alc\_test$en\_pred, reference = alc\_test$alc\_consumption)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 169 0  
## CurrentUse 95 301  
##   
## Accuracy : 0.8319   
## 95% CI : (0.7984, 0.8618)  
## No Information Rate : 0.5327   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6546   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6402   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.7601   
## Prevalence : 0.4673   
## Detection Rate : 0.2991   
## Detection Prevalence : 0.2991   
## Balanced Accuracy : 0.8201   
##   
## 'Positive' Class : NotCurrentUse   
##

## LOGISTIC REG MODEL  
# Make predictions in test set  
logmod\_pred = mod\_baseline %>% predict(alc\_test)  
alc\_test = alc\_test %>% mutate(logmod\_pred = as.factor(logmod\_pred))  
  
# Model prediction performance  
baseline\_perf = postResample(logmod\_pred, alc\_test$alc\_consumption) %>% as\_tibble\_row()  
  
confusionMatrix(data = alc\_test$logmod\_pred, reference = alc\_test$alc\_consumption)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 203 50  
## CurrentUse 61 251  
##   
## Accuracy : 0.8035   
## 95% CI : (0.7683, 0.8355)  
## No Information Rate : 0.5327   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.6044   
##   
## Mcnemar's Test P-Value : 0.3425   
##   
## Sensitivity : 0.7689   
## Specificity : 0.8339   
## Pos Pred Value : 0.8024   
## Neg Pred Value : 0.8045   
## Prevalence : 0.4673   
## Detection Rate : 0.3593   
## Detection Prevalence : 0.4478   
## Balanced Accuracy : 0.8014   
##   
## 'Positive' Class : NotCurrentUse   
##

## ELASTIC MODEL  
# Make predictions in test set  
lasso\_pred = mod\_lasso %>% predict(alc\_test)  
alc\_test = alc\_test %>% mutate(lasso\_pred = as.factor(lasso\_pred))  
  
# Model prediction performance  
LASSO\_perf = postResample(lasso\_pred, alc\_test$alc\_consumption) %>% as\_tibble\_row()  
  
confusionMatrix(data = alc\_test$lasso\_pred, reference = alc\_test$alc\_consumption)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 169 0  
## CurrentUse 95 301  
##   
## Accuracy : 0.8319   
## 95% CI : (0.7984, 0.8618)  
## No Information Rate : 0.5327   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6546   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6402   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.7601   
## Prevalence : 0.4673   
## Detection Rate : 0.2991   
## Detection Prevalence : 0.2991   
## Balanced Accuracy : 0.8201   
##   
## 'Positive' Class : NotCurrentUse   
##

rbind(elastic\_perf, baseline\_perf, LASSO\_perf) %>%   
 mutate(Model = c("Elastic Net", "Baseline", "LASSO")) %>%   
 relocate(Model, .before = Accuracy) %>% knitr::kable()

| Model | Accuracy | Kappa |
| --- | --- | --- |
| Elastic Net | 0.8318584 | 0.6546299 |
| Baseline | 0.8035398 | 0.6043742 |
| LASSO | 0.8318584 | 0.6546299 |

The table shows that the lm\_all model performs better on the test data, with an RMSE of 7.172, compared to the lm\_habits model’s RMSE of 7.413. Therefore, if I were only interested in prediction performance, I would select the linear model with all features over the linear model with health habit features to predict the number of healthy days in a month.

### Research Applications

One setting where the lm\_all model would be useful would be in a program evaluation setting. For instance, in a health program that promotes increased physical activity, when enrolling new subjects, we could predict the number of healthy days they are predicted to have in a month at baseline, and then compare the number of observed healthy days after program participation. This provides evidence for evaluating the success of the physical activity program in improving the number of healthy days a person may have.