Lab10

Jeffrey Wong

1/16/2020

```r
setwd("/Users/jw-mba/Desktop/r-projects")
#source("GCP_local_IP_connection_setup.R")
#assignment10 <- dbGetQuery(con, "SELECT * FROM econ_621.test_scores
#                                  WHERE gradelevel = 4
#                                  AND academic_year = 2015;")
#write.csv(assignment10, "assignment10.csv", row.names = F)
```

```r
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(varhandle)
library(reshape2)
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(Metrics)
test_scores <- read.csv("assignment10.csv", stringsAsFactors = F)
```

Build a predictive model for whether a student achieves proficiency on the SBAC Math test. A student is considered proficient if they receive 'Standard Met' or 'Standard Achieved' for the exam.

Your process should include these steps:

1. Clean and reformat your data, and describe your reasoning along the way. Hint: we did much of this reformatting in lecture 10.

```
#remove irrelevant columns and duplicates
test_scores <- test_scores[, -which(colnames(test_scores) %in% c("gradelevel", "academic_year", "percen
test_scores <- test_scores[!duplicated(test_scores[, -which(colnames(test_scores) %in% c("testscore", "
test_scores <- test_scores[test_scores$student_id != 1031535,]

test_scores$testname[test_scores$testname == "SBAC Preliminary"] <- "SBAC"
test_scores$testname[test_scores$testname == "NWEA MAP"] <- "MAP"

math_SBAC_prof <- test_scores[test_scores$subject == "Math" &
                                test_scores$testname == "SBAC",
                              c(1,9)]

math_SBAC_prof$proficiency<- as.factor(
  ifelse(math_SBAC_prof$proficiency == "Standard Met" |
           math_SBAC_prof$proficiency == "Standard Exceeded", 1, 0))

test_scores_wide <- dcast(test_scores, student_id + gender + race_ethnicity + school ~
                            testname + subject + testperiod, value.var = "testscore")
test_scores_wide <- merge(test_scores_wide, math_SBAC_prof, by = "student_id", all.x = T)
#test_scores_wide <- test_scores_wide[!is.na(test_scores_wide$proficiency),]

# Convert category variables to factors
factor_vars <- c("gender", "school", "race_ethnicity")
test_scores_wide[, factor_vars] <- data.frame(sapply(test_scores_wide[, factor_vars],  as.factor))
levels(test_scores_wide$race_ethnicity) <- c("AI", "AS", "AA", "FI", "HI", "PI", "MR", "WH")

# For each category variable, create a set of dummies and append to the data set
for (i in 1:length(factor_vars)) {
  dummies <- to.dummy(test_scores_wide[, factor_vars[i]], factor_vars[i])
test_scores_wide <- cbind(test_scores_wide, dummies)
}

#define the independent variables
model_formula <- as.formula("proficiency ~ gender.F +
race_ethnicity.AI + race_ethnicity.AS + race_ethnicity.AA +
race_ethnicity.HI + race_ethnicity.PI + race_ethnicity.MR + race_ethnicity.WH +
school.Charles_Middle + school.Indian_Ridge + school.Nolan_Richardson +
school.Parkland +
Benchmark_ELA_2 + Benchmark_ELA_4 + Benchmark_ELA_5 +
Benchmark_Math_2 + Benchmark_Math_5 + MAP_English_1 + MAP_English_3 + MAP_Math_1 + MAP_Math_3")

#remove all the observations with NAs
```

```
test_scores_wide <-
  test_scores_wide[which(complete.cases(test_scores_wide[,all.vars(model_formula)])),]

sapply(test_scores_wide, function(x) {sum(is.na(x))})
```

```
##              student_id                  gender          race_ethnicity
##                       0                       0                       0
##                  school          Benchmark_ELA_2         Benchmark_ELA_4
##                       0                       0                       0
##         Benchmark_ELA_5        Benchmark_Math_2        Benchmark_Math_4
##                       0                       0                      69
##        Benchmark_Math_5           MAP_English_1           MAP_English_3
##                       0                       0                       0
##              MAP_Math_1              MAP_Math_3              SBAC_ELA_5
##                       0                       0                       0
##             SBAC_Math_5             proficiency                gender.F
##                       0                       0                       0
##                gender.M   school.Charles_Middle     school.Indian_Ridge
##                       0                       0                       0
## school.Nolan_Richardson         school.Parkland            school.Wiggs
##                       0                       0                       0
##       race_ethnicity.AI       race_ethnicity.AS       race_ethnicity.AA
##                       0                       0                       0
##       race_ethnicity.FI       race_ethnicity.HI       race_ethnicity.PI
##                       0                       0                       0
##       race_ethnicity.MR       race_ethnicity.WH
##                       0                       0
```

2. Create subsets of your data for training, testing, and validation

```
# Subset the data: 70% training, 20% test, 10% validation
  # Create a dataframe of student ids and random values
sets <- data.frame(student_id = unique(test_scores_wide$student_id),
                   rand = runif(length(unique(test_scores_wide$student_id))))

  # Assign status based on unique values and merge into data
sets$set <- ifelse(sets$rand < 0.7, 'train', ifelse(sets$rand >= 0.9, 'validate', 'test'))
test_scores_wide <- merge(test_scores_wide, sets[, c('student_id', 'set')], by = 'student_id')

  # Subset by status
train <- test_scores_wide[test_scores_wide$set == "train",]
test <- test_scores_wide[test_scores_wide$set == "test",]
validate <- test_scores_wide[test_scores_wide$set == "validate",]

# Evaluate distributions of some variables we might want to stratify by
strats <- c("gender", "school", "race_ethnicity")
for (i in 1:length(strats)){
  print(table(test_scores_wide[, strats[i]])/nrow(test_scores_wide))
  print(table(train[, strats[i]])/nrow(train))
  print(table(test[, strats[i]])/nrow(test))
  print(table(validate[, strats[i]])/nrow(validate))
}
```

```
##
##         F         M
## 0.4534884 0.5465116
##
##    F    M
## 0.48 0.52
##
##         F         M
## 0.4038462 0.5961538
##
##         F         M
## 0.3870968 0.6129032
##
##   Charles Middle     Indian Ridge Nolan Richardson         Parkland
##        0.1317829        0.1705426        0.1395349        0.2596899
##            Wiggs
##        0.2984496
##
##   Charles Middle     Indian Ridge Nolan Richardson         Parkland
##        0.1314286        0.1542857        0.1485714        0.2800000
##            Wiggs
##        0.2857143
##
##   Charles Middle     Indian Ridge Nolan Richardson         Parkland
##        0.1538462        0.1730769        0.1346154        0.2307692
##            Wiggs
##        0.3076923
##
##   Charles Middle     Indian Ridge Nolan Richardson         Parkland
##       0.09677419       0.25806452       0.09677419       0.19354839
##            Wiggs
##       0.35483871
##
##         AI         AS         AA         FI         HI         PI         MR
## 0.01162791 0.03100775 0.10465116 0.00000000 0.80232558 0.01550388 0.01550388
##         WH
## 0.01937984
##
##         AI         AS         AA         FI         HI         PI         MR
## 0.01714286 0.02285714 0.12000000 0.00000000 0.79428571 0.02285714 0.01142857
##         WH
## 0.01142857
##
##         AI         AS         AA         FI         HI         PI         MR
## 0.00000000 0.05769231 0.07692308 0.00000000 0.78846154 0.00000000 0.01923077
##         WH
## 0.05769231
##
##         AI         AS         AA         FI         HI         PI         MR
## 0.00000000 0.03225806 0.06451613 0.00000000 0.87096774 0.00000000 0.03225806
##         WH
## 0.00000000
```

3. Estimate a logit model on the training dataset.

```r
logit_model <- glm(model_formula, data = train, family = binomial(link = "logit"))
summary(logit_model)
```

```
##
## Call:
## glm(formula = model_formula, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.57931  -0.06186  -0.00146   0.00000   2.13837
##
## Coefficients: (1 not defined because of singularities)
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -1.312e+02  1.027e+04  -0.013   0.9898
## gender.F               -1.805e-01  1.126e+00  -0.160   0.8727
## race_ethnicity.AI       1.874e+01  1.027e+04   0.002   0.9985
## race_ethnicity.AS       4.003e+01  1.226e+04   0.003   0.9974
## race_ethnicity.AA      -1.251e+00  1.062e+04   0.000   0.9999
## race_ethnicity.HI       1.740e+01  1.027e+04   0.002   0.9986
## race_ethnicity.PI       1.482e+01  1.027e+04   0.001   0.9988
## race_ethnicity.MR      -2.149e+00  1.425e+04   0.000   0.9999
## race_ethnicity.WH              NA         NA      NA       NA
## school.Charles_Middle  -8.030e-01  2.055e+00  -0.391   0.6960
## school.Indian_Ridge    -2.276e+00  2.052e+00  -1.109   0.2674
## school.Nolan_Richardson -1.370e+00  1.834e+00  -0.747   0.4550
## school.Parkland         1.442e+00  1.894e+00   0.761   0.4466
## Benchmark_ELA_2         6.500e-02  4.330e-02   1.501   0.1333
## Benchmark_ELA_4        -1.904e-02  6.294e-02  -0.303   0.7622
## Benchmark_ELA_5        -4.146e-02  3.908e-02  -1.061   0.2887
## Benchmark_Math_2        7.658e-02  3.428e-02   2.234   0.0255 *
## Benchmark_Math_5       -2.884e-02  5.254e-02  -0.549   0.5831
## MAP_English_1          -1.652e-01  1.100e-01  -1.502   0.1330
## MAP_English_3           2.439e-01  1.472e-01   1.658   0.0973 .
## MAP_Math_1              2.094e-01  1.109e-01   1.887   0.0591 .
## MAP_Math_3              2.370e-01  1.338e-01   1.772   0.0765 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 177.878  on 174  degrees of freedom
## Residual deviance:  33.541  on 154  degrees of freedom
## AIC: 75.541
##
## Number of Fisher Scoring iterations: 19
```

4. Constrain the model.

```r
constrained_logit_model <- step(logit_model, direction = 'backward')
```

```
## Start:  AIC=75.54
```

```
## proficiency ~ gender.F + race_ethnicity.AI + race_ethnicity.AS +
##     race_ethnicity.AA + race_ethnicity.HI + race_ethnicity.PI +
##     race_ethnicity.MR + race_ethnicity.WH + school.Charles_Middle +
##     school.Indian_Ridge + school.Nolan_Richardson + school.Parkland +
##     Benchmark_ELA_2 + Benchmark_ELA_4 + Benchmark_ELA_5 + Benchmark_Math_2 +
##     Benchmark_Math_5 + MAP_English_1 + MAP_English_3 + MAP_Math_1 +
##     MAP_Math_3
##
##
## Step:  AIC=80.15
## proficiency ~ gender.F + race_ethnicity.AI + race_ethnicity.AS +
##     race_ethnicity.AA + race_ethnicity.HI + race_ethnicity.PI +
##     race_ethnicity.MR + race_ethnicity.WH + school.Charles_Middle +
##     school.Indian_Ridge + school.Nolan_Richardson + school.Parkland +
##     Benchmark_ELA_2 + Benchmark_ELA_4 + Benchmark_ELA_5 + Benchmark_Math_2 +
##     Benchmark_Math_5 + MAP_English_1 + MAP_English_3 + MAP_Math_3
```

```r
summary(constrained_logit_model)
```

```
##
## Call:
## glm(formula = proficiency ~ gender.F + race_ethnicity.AI + race_ethnicity.AS +
##     race_ethnicity.AA + race_ethnicity.HI + race_ethnicity.PI +
##     race_ethnicity.MR + race_ethnicity.WH + school.Charles_Middle +
##     school.Indian_Ridge + school.Nolan_Richardson + school.Parkland +
##     Benchmark_ELA_2 + Benchmark_ELA_4 + Benchmark_ELA_5 + Benchmark_Math_2 +
##     Benchmark_Math_5 + MAP_English_1 + MAP_English_3 + MAP_Math_3,
##     family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.45272  -0.05378  -0.00222   0.00000   2.03606
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -7.119e+14  2.274e+15  -0.313  0.75419
## gender.F                -8.386e-01  9.809e-01  -0.855  0.39256
## race_ethnicity.AI        7.119e+14  2.274e+15   0.313  0.75419
## race_ethnicity.AS        5.216e+15  2.274e+15   2.294  0.02180 *
## race_ethnicity.AA        7.119e+14  2.274e+15   0.313  0.75419
## race_ethnicity.HI        7.119e+14  2.274e+15   0.313  0.75419
## race_ethnicity.PI        7.119e+14  2.274e+15   0.313  0.75419
## race_ethnicity.MR        7.119e+14  2.274e+15   0.313  0.75419
## race_ethnicity.WH        7.119e+14  2.274e+15   0.313  0.75419
## school.Charles_Middle   -1.054e+00  1.757e+00  -0.600  0.54843
## school.Indian_Ridge     -1.143e+00  1.795e+00  -0.637  0.52436
## school.Nolan_Richardson -1.536e+00  1.693e+00  -0.908  0.36406
## school.Parkland          2.270e+00  1.804e+00   1.258  0.20840
## Benchmark_ELA_2          6.630e-02  4.212e-02   1.574  0.11546
## Benchmark_ELA_4          7.684e-03  5.532e-02   0.139  0.88952
## Benchmark_ELA_5         -5.631e-02  3.247e-02  -1.734  0.08286 .
## Benchmark_Math_2         7.973e-02  3.376e-02   2.362  0.01818 *
## Benchmark_Math_5        -2.984e-02  4.546e-02  -0.656  0.51155
## MAP_English_1           -1.452e-01  1.017e-01  -1.427  0.15354
```

```
## MAP_English_3            3.144e-01  1.385e-01   2.270  0.02321 *
## MAP_Math_3               3.217e-01  1.243e-01   2.589  0.00963 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 177.878  on 174  degrees of freedom
## Residual deviance:  38.153  on 154  degrees of freedom
## AIC: 80.153
##
## Number of Fisher Scoring iterations: 25
```

5. Write a function to create predicted values from the model's parameters, and check the function against model's fitted values.

```
coefficients <- constrained_logit_model$coefficients

pred_probability <- function(data, obs, beta)
  {pred <- rbind.fill(obs[names(data) %in% names(beta)],
                      as.data.frame(t(beta))) %>% t %>% as.data.frame %>% subset(!is.na(V1))
  pred$product <- pred$V1 * pred$V2
  1/(1 + exp(-(sum(pred$product, unname(beta[1])))))
}

#testing data
i <- 5
pred_probability(train, train[i,], coefficients)
```

```
## [1] 0.01590639
```

```
constrained_logit_model$fitted.values[i]
```

```
##         8
## 0.0157397
```

6. Calculate an optimized threshold for model predictions.

```
#create a function to calculate best threshold for different data set and predicted values
thresh_calc <- function(data, fitted){
  thresh <- data.frame(threshold = seq(0, 1, 0.01))
  data1 <- data.frame(data, pred = fitted)
  thresh$precision <- apply(thresh, 1, function(x) {
    sum(data1$pred > x & data1$proficiency == 1)/sum(data1$pred > x)})
  thresh$recall <- apply(thresh, 1, function(x) {
    sum(data1$pred > x & data1$proficiency == 1)/sum(data1$proficiency == 1)})

  thresh$F1 <- 2 * ((thresh$precision * thresh$recall)/(thresh$precision + thresh$recall))
  return(thresh[which.max(thresh$F1), "threshold"])
}

best_thresh1 <- thresh_calc(train, constrained_logit_model$fitted.values)
best_thresh1
```

```
## [1] 0.45
```

7. Evaluate the model's performance on training and testing datasets.

```r
library(cvAUC)
```

```
## Loading required package: ROCR

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:reshape2':
##
##     dcast, melt

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

##

## cvAUC version: 1.1.0

## Notice to cvAUC users: Major speed improvements in version 1.1.0

##
```

```r
AUC(ifelse(constrained_logit_model$fitted.values > best_thresh1, 1, 0), train$proficiency) # AUC of the
```

```
## [1] 0.9439448
```

```r
#calculate TESTING sets predicted values with the constraint model
fitted_values <- function(data){
  test_fitted <- c()
  for (i in 1:length(data$proficiency)) {
  test_fitted[[i]] <- pred_probability(data, data[i,], coefficients)
  }
  return(test_fitted)
}

forecast <- fitted_values(test)
head(forecast)
```

```
## [1] 1.824255e-01 1.177951e-15 9.610242e-05 4.539787e-05 1.098694e-02
## [6] 6.515062e-19
```

```r
length(forecast)
```

```
## [1] 52
```

```r
best_thresh2 <- thresh_calc(test, forecast)
best_thresh2
```

```
## [1] 0.86
```

```r
AUC(ifelse(forecast > best_thresh2, 1, 0), test$proficiency) # AUC of the testing sets
```

```
## [1] 0.7761905
```