Narrative Code

Learn Against the Machine

November 24, 2018

```
## Set Seed

set.seed(3)

## Load Packages

library(caret)
library(class)
library(dplyr)
library(glmnet)
library(MASS)
library(partykit)
library(pROC)
library(pscl)
library(readr)
library(ROCR)
library(tidyverse)
```

Producing Final Dataset

```
## Reading: county data

County_Drug <- read_csv("County_Drug.csv")

## Reading and Cleaning: presribing behavior data

prescribing_behavior <- read_csv("293 COUNTY DATA/prescribing_behavior.csv") %>%
  mutate(county_id = paste0("05000US", FIPS)) %>%
  subset(select = -c(`State Name`, `State Abbreviation`, `County Name`, `FIPS`))

## Renaming: variable names

colnames(prescribing_behavior) <- c(
  "part_d_prescribers",
  "part_d_opioid_prescribers",
  "opioid_claims",
  "extended_release_opioid_claims",
  "overall_claims",
  "opioid_prescribing_rate",
  "extended_release_prescription_rate",
  "change_in_rate",
  "county_id"
)

## Joining: county info data and prescribing behavior data

working_data <- County_Drug %>%
```

```r
  inner_join(prescribing_behavior, by = "county_id") %>%
  na.omit() %>%
  subset(select = -c(X1))

#View(working_data)
```

```r
## Creating: over vs. under median rate variable = opioid_prescribing_rate

working_data$over_avg_rx_rate <- 0
working_data$over_avg_rx_rate[working_data$opioid_prescribing_rate >
  median(working_data$opioid_prescribing_rate)] <- 1

#View(working_data)
```

Training and Testing Datasets Based on Working Dataset

```r
## Setting Up: training & testing datasets

working_data$id <- 1:nrow(working_data)
train <- working_data %>% dplyr::sample_frac(.75)
test <- dplyr::anti_join(working_data, train, by = "id")
```

Logistic Regression Models

```r
## Logistic Regression: intuitive variables (unemployment rate, HS graduation rate,
## average age, population, proportion of population who is male, number of exchanges,
## poverty rate)

glm_intuitive_fit <- glm(
  over_avg_rx_rate ~
    unemployment_rate +
    hs_graduation_rate +
    average_age +
    population +
    male_proportion +
    num_exchange +
    poverty_rate,
  data = working_data,
  family = "binomial"
)
summary(glm_intuitive_fit)
```

```r
## Logistic Regression: statistically significant variables from intuitive logistic
## regression model, race/ethnicity, and state legislature variables
## (unemployment rate, HS graduation rate, average age, proportion of population
## who is white, proportion of population who is black, proportion of population who
## is american indian, proportion of population who is asian, proportion of
## population who is hawaiian pacific, proportion of population who is interracial,
## proportion of population who is hispanic, state legislature dominant political
## ideology)

glm_intuitive_demographics_fit <- glm(
  over_avg_rx_rate ~
```

```
  unemployment_rate +
    hs_graduation_rate +
    average_age +
    white_proportion +
    black_proportion +
    american_indian_proportion +
    asian_proportion +
    hawaiian_pacific_proportion +
    interracial_proportion +
    hispanic_proportion +
    state_legislature,
  data = working_data,
  family = "binomial"
)

summary(glm_intuitive_demographics_fit)
```

```
## Logistic Regression: statistically significant variables from intuitive and other
## demographics logistic regression model (unemployment rate, HS graduation rate,
## average, poverty rate, state legislature dominant political ideology)

## Important Change: building the model using the training dataset we created, instead of
## the working dataset

glm_significant_fit <- glm(
  over_avg_rx_rate ~
    unemployment_rate +
    hs_graduation_rate +
    average_age +
    poverty_rate +
    state_legislature,
  data = train,
  family = "binomial"
)

summary(glm_significant_fit)
```

Testing Significant Model's Prediction Accuracy (Full Model and Forward Stepwise Model)

```
## Testing: prediction accuracy of all three models in turn
## (fill in model name for different calculations)
glm_probs <- data.frame(probs = predict(glm_significant_fit,
                                        newdata = test,
                                        type = "response"))
glm_pred <- glm_probs %>%
  mutate(pred = ifelse(probs > 0.5, 1, 0))

glm_pred <- cbind(test, glm_pred)

glm_pred %>%
  count(pred, over_avg_rx_rate) %>%
  spread(over_avg_rx_rate, n, fill = 0)
```

```
glm_pred %>%
  summarise(
    score = mean(pred == over_avg_rx_rate),
    recip = mean(pred != over_avg_rx_rate)
  )
```

```
## Creating: data set without county-specific variables (i.e. county name, state, etc.)
## Note: Could be interesting to test state as a variable

minus_vector <- c(1, 5, 6, 24, 25, 26, 27, 28, 29, 30, 33)
full_mod_set <- working_data[, -minus_vector]
full_mod_set <- full_mod_set[, -21]
```

```
## Creating: test and training sets for stepwise model selection

full_mod_set$id <- 1:nrow(full_mod_set)
step_train <- full_mod_set %>% dplyr::sample_frac(.75)
step_test <- dplyr::anti_join(full_mod_set, train, by = "id")
step_train <- step_train[, -22]
step_test <- step_test[, -22]

## Creating: full model with all variables

full_mod <- glm(over_avg_rx_rate ~ ., data = step_train, family = "binomial")

## Predicting: accuracy of full model

glm_probs <- data.frame(probs = predict(full_mod, newdata = step_test, type = "response"))
glm_pred <- glm_probs %>%
  mutate(pred = ifelse(probs > 0.5, 1, 0))

glm_pred <- cbind(step_test, glm_pred)
glm_pred %>%
  count(pred, over_avg_rx_rate) %>%
  spread(over_avg_rx_rate, n, fill = 0)
glm_pred %>%
  summarise(
    score = mean(pred == over_avg_rx_rate),
    recip = mean(pred != over_avg_rx_rate)
  )
```

```
## Calculating: stepwise model, forward selection, from full model

step_model <- full_mod %>%
  stepAIC(trace = FALSE, direction = "forward")
coef(step_model)

summary(step_model)
```

```
## Comparing: stepwise and full model for coefficient values (they are the same)

cbind(coef(step_model), coef(full_mod))
```

```
## Checking: prediction accuracy for forward stepwise model selection
glm_step_probs <- data.frame(probs = predict(step_model,
```

```r
                                                  newdata = step_test,
                                                  type = "response"))
glm_step_pred <- glm_step_probs %>%
  mutate(pred = ifelse(probs > 0.5, 1, 0))

glm_step_pred <- cbind(step_test, glm_step_pred)

glm_step_pred %>%
  count(pred, over_avg_rx_rate) %>%
  spread(over_avg_rx_rate, n, fill = 0)

glm_step_pred %>%
  summarise(
    score = mean(pred == over_avg_rx_rate),
    recip = mean(pred != over_avg_rx_rate)
  )
```

Testing Change In Rate Variable

```r
## Logistic regression with only significant variables from forward stepwise model.

## Comparing: how does the variable "change_inrate" impact the effectiveness of the model

sig_step_mod <- glm(over_avg_rx_rate ~ unemployment_rate +
                      hs_graduation_rate +
                      average_age +
                      no_hispanic_proportion +
                      state_legislature +
                      poverty_rate,
                    data = step_test, family = "binomial")

summary(sig_step_mod)

## Checking: prediction accuracy for model with significant vars from stepwise model

new_probs <- data.frame(probs = predict(sig_step_mod,
                                        newdata = step_test,
                                        type = "response"))
new_pred <- new_probs %>%
  mutate(pred = ifelse(probs > 0.5, 1, 0))

new_pred <- cbind(step_test, new_pred)
new_pred %>%
  count(pred, over_avg_rx_rate) %>%
  spread(over_avg_rx_rate, n, fill = 0)
new_pred %>%
  summarise(
    score = mean(pred == over_avg_rx_rate),
    recip = mean(pred != over_avg_rx_rate)
  )
```

Receiver Operating Characteristic Curves

```
## ROC curve for full model

prob <- predict(full_mod, newdata = step_test, type = "response")
pred <- prediction(prob, step_test$over_avg_rx_rate)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf)

## Area under ROC curve for full model

auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## ROC curve for forward stepwise model

step_prob <- predict(step_model, newdata = step_test, type = "response")
step_pred <- prediction(step_prob, step_test$over_avg_rx_rate)
step_perf <- performance(step_pred, measure = "tpr", x.measure = "fpr")
plot(step_perf, colorize = TRUE)
plot(step_perf, add = TRUE)

## area under ROC curve for forward stepwise model

step_auc <- performance(step_pred, measure = "auc")
step_auc <- step_auc@y.values[[1]]
step_auc
```

```
## relative variable importance for forward stepwise model

varImp(step_model)

### pseudo-r-squared

pR2(step_model)
```

Lasso

```
## Troubleshooting: weird errors when running code

## Creating: training and test sets for lasso (splitting up x and y)

x <- model.matrix(over_avg_rx_rate ~., full_mod_set)[, -22]
y <- full_mod_set %>%
  dplyr::select(over_avg_rx_rate) %>%
  unlist() %>%
  as.numeric()

lasso_train <- full_mod_set %>%
  sample_frac(0.75)
lasso_test <- full_mod_set %>%
  setdiff(lasso_train)

## Removing: id variable
```

```r
lasso_train_x <- model.matrix(over_avg_rx_rate~., lasso_train)[, -22]
lasso_test_x <- model.matrix(over_avg_rx_rate~., lasso_test)[, -22]

lasso_train_y <- lasso_train %>%
  dplyr::select(over_avg_rx_rate) %>%
  unlist() %>%
  as.numeric()
lasso_test_y <- lasso_test %>%
  dplyr::select(over_avg_rx_rate) %>%
  unlist() %>%
  as.numeric()

grid <- 10 ^ seq(10, -2, length = 100)
```

```r
## Producing :initial lasso model

lasso_mod <- glmnet(lasso_train_x, lasso_train_y, alpha = 1, lambda = grid)

## Using: cross validation to select optimal lambda
cv_lasso <- cv.glmnet(lasso_train_x, lasso_train_y, alpha = 1, family = "binomial")
plot(cv_lasso)

bestlam <- cv_lasso$lambda.min

## Building: lasso model with optimal lambda value

lasso_pred <- predict(lasso_mod, s = bestlam, newx = lasso_test_x)

coef(cv_lasso)

## lasso test MSE

mean((lasso_pred - lasso_test_y) ^ 2)
```