Narrative Code

Learn Against the Machine

November 24, 2018

```
## reading in data files, joining to access prescribing behavior (response vars)
set.seed(3)
library(caret)
library(class)
library(dplyr)
library(glmnet)
library(MASS)
library(pROC)
library(pscl)
library(ROCR)
library(tidyverse)

County_Drug <- read_csv("County_Drug.csv")
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`))
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 data sets
working_data <- County_Drug %>%
  inner_join(prescribing_behavior, by = "county_id") %>%
  na.omit() %>%
  subset(select = -c(X1))
```

```
## creating over vs. under median rate variable
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

## setting up testing & training 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: intuitive vars
glm_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_fit)

## logistic regression: intuitive vars, race/ethnicity vars, and state legislature var
glm_party_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_party_fit)

## logisticreg regression: vars significant in last model
glm_sig_fit <- glm(
  over_avg_rx_rate ~ unemployment_rate +
    hs_graduation_rate +
    average_age +
    poverty_rate +
    unemployment_rate +
    state_legislature,
  data = train,
  family = "binomial"
)
summary(glm_sig_fit)

## testing prediction accuracy of all three models in turn
## (fill in model name for different calculations)
glm_probs <- data.frame(probs = predict(glm_sig_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)
```

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

```r
## creating data set without county-specific vars (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]

## designating full model with all vars
full_mod <- glm(over_avg_rx_rate ~ ., data = step_train, family = "binomial")

## prediction accuracy of full mod
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 mod
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,
                                             newdata = step_test,
                                             type = "response"))
glm_step_pred <- glm_step_probs %>%
  mutate(pred = ifelse(probs > 0.5, 1, 0))
```

```r
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)
  )
```

```r
## Log regression with only sig vars from stepwise

## can put change_inrate back in this model...
## I was just testing to see what it would do to remove it.
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)
  )
```

```r
## 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)

## ROC curve for step 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 full model
```

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

## 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)
```

```r
## 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
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)

## producing initial lasso mod
lasso_mod <- glmnet(lasso_train_x, lasso_train_y, alpha = 1, lambda = grid)

## using cross val 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

## lasso model with optimal lambda value
lasso_pred <- predict(lasso_mod, s = bestlam, newx = lasso_test_x)
```

```r
## lasso test MSE
mean((lasso_pred - lasso_test_y) ^ 2)
```