lab8_2024_v1

April 11, 2024

1 Lab 8: Bias in Algorithms

1.1 Methods/concepts: algorithmic bias; choice of "labels" vs. "predictors"

Name: Shreya Chaturvedi

Email: shreyachaturvedi@hks.harvard.edu

HUID: 31575036

Lab: Thursday 3pm at HKS

Date: April 10th, 2024

LAB DESCRIPTION

In this lab, we will dive deeper into bias in algorithms, following Obermeyer, Powers, Vogeli, and Mullainathan (2019). We will train several prediction algorithms, some including the patient's race and others explicitly leaving out the patient's race. We will see how the choice of "label" – either patient costs or patient health – affects the performance of the models. Finally, we will examine the racial composition of patients predicted to have high risk according to the algorithms. For more details on the variables included in these data, see Table 1.

A list and description of each of the R commands needed for questions 6 through 9 on this lab are contained in Table 2.

1.2 QUESTIONS

1. Start by randomly splitting the 48,748 patients included in the **health.dta** data set into a **10%** training data set and a **90%** test data set. Table 1 describes the data. There are two reasons we are using such a small fraction of the data to train the models. First, estimating random forests on a larger fraction of the data would be prohibitively time consuming. Second, we require a large number of observations in the test data set so that we can study differences in risk score by race.

[1]:
```
# #  The following program starts by loading in the health data and dividing it
# #  into a 10% training sample and a 90% test sample.  It then estimates
# #  four random forests models that differ in the predictor variables includes
# #  and the "label" or outcome variable that the models predict.  The next part
# #  of the code computes RMSPEs for the four models.  The last part of the code
# #  exports the predictions of the four models for the test sample as␣
 ↪"lab8_2024_results.dta"
```

```
# #
# #  You will analyze the resulting "lab8_2024_results.dta" data set to answer␣
 ↪questions
# #  6-9 on the lab.  The starter code helps out for questions 1-5.
# #
# #  The code may have some typos -- please be on the look out for them -- and␣
 ↪to
# #  receive credit for the lab you have to make edits to estimate your own
# #  random forests.  These are simply examples of what you might
# #  want to  do in your analysis, but you are expected to make an effort to
# #  understand what you are doing with the code.
# #
# #  Inputs:  health.dta (download from canvas)
# #           randomForest to estimate random forest models
# #           tidyverse library for data manipulations
# #           haven library to load stata data sets into R
# #
# #  Outputs: mod1_importance.png
# #           mod2_importance.png
# #           mod3_importance.png
# #           mod4_importance.png
# #           lab8_2024_results.dta

# # Question 1 example code
# rm(list=ls()) # removes all objects from the environment

# # Install packages (if necessary) and load required libraries
# if (!require(haven)) install.packages("haven"); library(haven)
# if (!require(randomForest)) install.packages("randomForest");␣
 ↪library(randomForest)
# if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)

# #Set seed for cross validation and random forests
# HUID <- 31575036 #Replace with your HUID
# set.seed(HUID)

#␣
 ↪#------------------------------------------------------------------------------
# # Data set up
#␣
 ↪#------------------------------------------------------------------------------

# #Open stata data set
# download.file("https://raw.githubusercontent.com/ekassos/ec50_s24/main/health.
 ↪dta", "health.dta", mode = "wb")
# dat <- read_dta("health.dta")
```

```
# head(dat)

# #Store health variables from time t-1 which all end with _tm
# all_predictors <- colnames(dat[,grep("^[tm1]", names(dat))])
# #all_predictors

# #Store predictor variables which all start with P_*, but EXCLUDE race
# race <- c("tm1_dem_black")
# exclude_race <- setdiff(all_predictors,race)
# #exclude_race

# #Define training and test data sets
# #Use a uniformly distributed random number between 0 and 1
# dat$random_number <- runif(length(dat$patient_id))

# ## Generate a training flag for 10% of the sample
# dat$train_flag <- ifelse(dat$random_number<= 0.1, 1, 0)

# #Report number of observations in training and test samples
# sum(dat$train_flag)
# sum(1-dat$train_flag)

# ## Create some data frames that just contain the training and test data

# #Data frame with training data (randomly selected 10% of the data)
# training <- subset(dat, train_flag == 1)
# summary(training)

# #Data frame with test data (remaining 90% of the data)
# test <- subset(dat, train_flag == 0)
# summary(test)
```

[2]:
```
## YOUR QUESTION 1 CODE GOES HERE
# Remove all existing objects from the workspace for a clean start
rm(list = ls())

# Load required packages, installing them if they are not already installed
if (!require("haven")) install.packages("haven")
library(haven)
if (!require("randomForest")) install.packages("randomForest")
library(randomForest)
if (!require("dplyr")) install.packages("dplyr")
library(dplyr)
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)

# Set a seed for reproducibility
```

```r
set.seed(31575036)  # Use your HUID or any unique ID

# Load the dataset
download.file("https://raw.githubusercontent.com/ekassos/ec50_s24/main/health.
 ↪dta", "health.dta", mode = "wb")
health_data <- read_dta("health.dta")

# Divide data into a 10% training sample and a 90% test sample
health_data$random_number <- runif(n = nrow(health_data))
train_flag <- ifelse(health_data$random_number <= 0.1, TRUE, FALSE)
training_data <- health_data[train_flag, ]
test_data <- health_data[!train_flag, ]

# Identify predictor variables excluding patient's race
exclude_race <- names(health_data)[grepl("^tm1", names(health_data)) & !
 ↪grepl("tm1_dem_black", names(health_data))]
all_predictors <- names(health_data)[grepl("^tm1", names(health_data))]

# Define outcome variable
outcome_var <- "cost_t"  # For models predicting costs
health_outcome_var <- "gagne_sum_t"  # For models predicting health
```

```
Loading required package: haven

Loading required package: randomForest

randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.

Loading required package: dplyr


Attaching package: ‘dplyr’


The following object is masked from ‘package:randomForest’:

    combine


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:
```

```
    intersect, setdiff, setequal, union


Loading required package: ggplot2


Attaching package: 'ggplot2'


The following object is masked from 'package:randomForest':

    margin
```

Question 1 Answer

Example code commented out and modified code given above.

2. Estimate the following statistical models using the training data set:
    1. Random forest to predict the "label" of patient costs (cost_t) using the full set of predictors consisting of all variables starting with tm1_, but *excluding* the patient's race
    2. Random forest to predict the "label" of patient costs (cost_t) using the full predictor set, now *including* the patient's race
    3. Random forest to predict the "label" of patient health (gagne_sum_t) using the full predictor set, *excluding* the patient's race
    4. Random forest to predict the "label" of patient health (gagne_sum_t) using the full predictor set, *including* the patient's race

    Note that random forests with lots of observations and predictors (150) will take a long time to run. You should therefore only use around 100 trees in your forests.

[3]:
```
#␣
 ↪#------------------------------------------------------------------------------
# # Model 1: Random forest trained to predict costs, using all predictors,
# # EXCLUDING patient's race
#␣
 ↪#------------------------------------------------------------------------------

# #Reformulate allows us to write yvar ~ xvar1 + xvar2 + ... using a list of all
# #the xvar1, xvar2, etc. variables without actually writing them out
# mod1 <- randomForest(reformulate(exclude_race, "cost_t"),
#                      ntree=100,
#                      mtry=149,
#                      importance=TRUE, ## add importance=TRUE so that we store␣
 ↪the variable importance information
#                      data=training)
```

```
# #Tuning parameters are ntree and mtry
# #ntree is number of trees in your forest
# #mtry is the number of predictors considered at each split (default is number␣
 ↪of predictors divided by 3)

# ### Try changing mtry and ntree

# mod1 #Review the Random Forest Results

# ### generate predictions for all observations in test and training samples
# y_test_predictions_mod1 <- predict(mod1, newdata=test)
# y_train_predictions_mod1 <- predict(mod1, newdata=training)

# #Variable importance
# importance(mod1)
# varImpPlot(mod1, type=1) #Plot the Random Forest Results
# dev.copy(png,'mod1_importance.png')
# dev.off()

# #type        is either 1 or 2, specifying the type of importance measure
# #(1=mean decrease in accuracy, 2=mean decrease in node impurity)




#␣
 ↪#-------------------------------------------------------------------------------
# # Model 2: Random forest trained to predict costs, using all predictors,
# # INCLUDING patient's race
#␣
 ↪#-------------------------------------------------------------------------------

# #Reformulate allows us to write yvar ~ xvar1 + xvar2 + ... using a list of all
# #the xvar1, xvar2, etc. variables without actually writing them out
# mod2 <- randomForest(reformulate(all_predictors, "cost_t"),
#                      ntree=100,
#                      mtry=150,
#                      importance=TRUE, ## add importance=TRUE so that we store␣
 ↪the variable importance information
#                      data=training)

# #Tuning parameters are ntree and mtry
# #ntree is number of trees in your forest
# #mtry is the number of predictors considered at each split (default is number␣
 ↪of predictors divided by 3)
```

```
# ### Try changing mtry and ntree

# mod2 #Review the Random Forest Results

# ### generate predictions for all observations in test and training samples
# y_train_predictions_mod2 <- predict(mod2, newdata=training)
# y_test_predictions_mod2 <- predict(mod2, newdata=test)

# #Variable importance
# importance(mod2)
# varImpPlot(mod2, type=1) #Plot the Random Forest Results
# dev.copy(png,'mod2_importance.png')
# dev.off()


# #type        is either 1 or 2, specifying the type of importance measure
# #(1=mean decrease in accuracy, 2=mean decrease in node impurity)

#␣
 ↪#------------------------------------------------------------------------------
# # Model 3: Random forest trained to predict health, using all predictors,
# # EXCLUDING patient's race
#␣
 ↪#------------------------------------------------------------------------------

# #Reformulate allows us to write yvar ~ xvar1 + xvar2 + ... using a list of all
# #the xvar1, xvar2, etc. variables without actually writing them out
# mod3 <- randomForest(reformulate(exclude_race, "gagne_sum_t"),
#                          ntree=100,
#                          mtry=149,
#                          importance=TRUE, ## add importance=TRUE so that we␣
 ↪store the variable importance information
#                          data=training)

# #Tuning parameters are ntree and mtry
# #ntree is number of trees in your forest
# #mtry is the number of predictors considered at each split (default is number␣
 ↪of predictors divided by 3)

# ### Try changing mtry and ntree

# mod3 #Review the Random Forest Results

# ### generate predictions for all observations in test and training samples
# y_test_predictions_mod3 <- predict(mod3, newdata=test)
# y_train_predictions_mod3 <- predict(mod3, newdata=training)
```

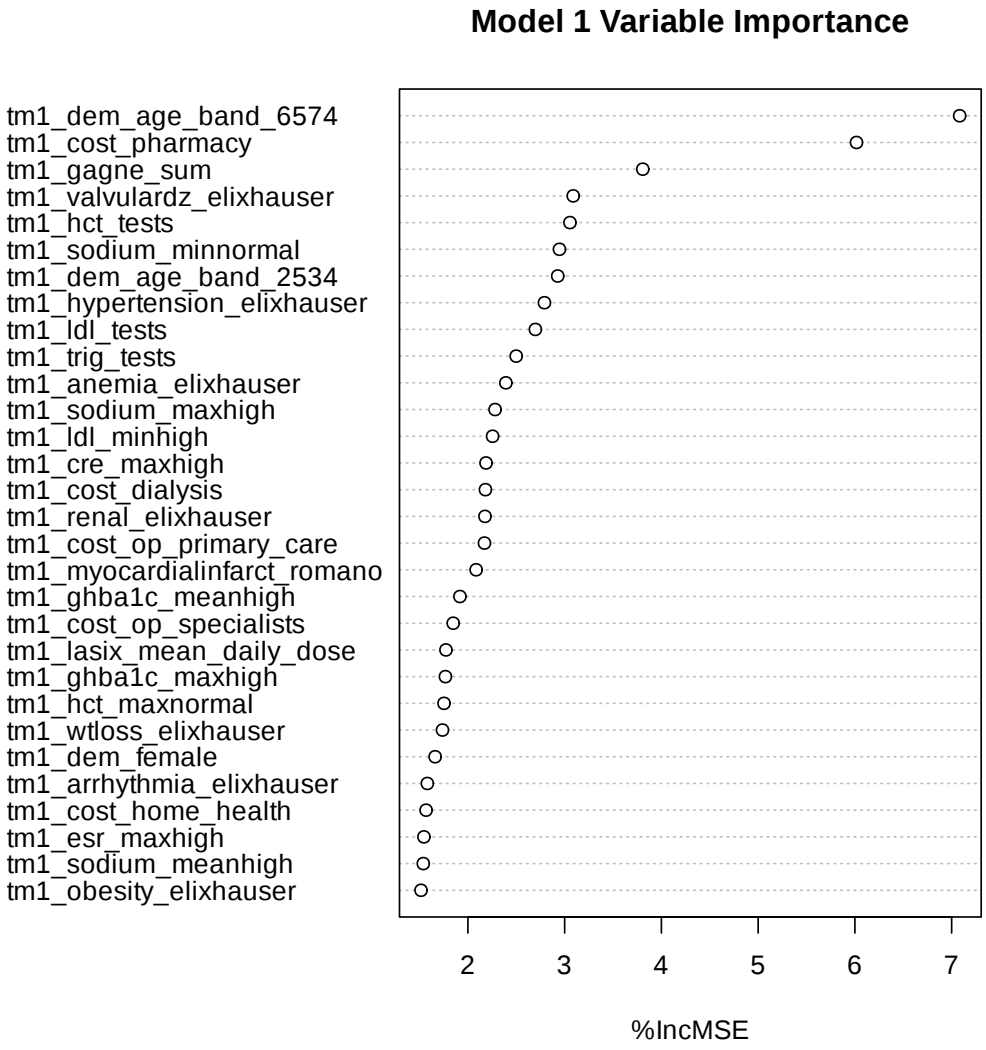
```
# #Variable importance
# importance(mod3)
# varImpPlot(mod3, type=1) #Plot the Random Forest Results
# dev.copy(png,'mod3_importance.png')
# dev.off()

# #type        is either 1 or 2, specifying the type of importance measure
# #(1=mean decrease in accuracy, 2=mean decrease in node impurity)

#␣
 ↪#------------------------------------------------------------------------------
# # Model 4: Random forest trained to predict health, using all predictors,
# # INCLUDING patient's race
#␣
 ↪#------------------------------------------------------------------------------

# #Reformulate allows us to write yvar ~ xvar1 + xvar2 + ... using a list of all
# #the xvar1, xvar2, etc. variables without actually writing them out
# mod4 <- randomForest(reformulate(all_predictors, "gagne_sum_t"),
#                      ntree=100,
#                      mtry=150,
#                      importance=TRUE, ## add importance=TRUE so that we store␣
 ↪the variable importance information
#                      data=training)

# #Tuning parameters are ntree and mtry
# #ntree is number of trees in your forest
# #mtry is the number of predictors considered at each split (default is number␣
 ↪of predictors divided by 3)

# ### Try changing mtry and ntree

# mod4 #Review the Random Forest Results

# ### generate predictions for all observations in test and training samples
# y_train_predictions_mod4 <- predict(mod4, newdata=training)
# y_test_predictions_mod4 <- predict(mod4, newdata=test)

# #Variable importance
# importance(mod4)
# varImpPlot(mod4, type=1) #Plot the Random Forest Results
# dev.copy(png,'mod4_importance.png')
# dev.off()


# #type        is either 1 or 2, specifying the type of importance measure
```

```
# #(1=mean decrease in accuracy, 2=mean decrease in node impurity)
```

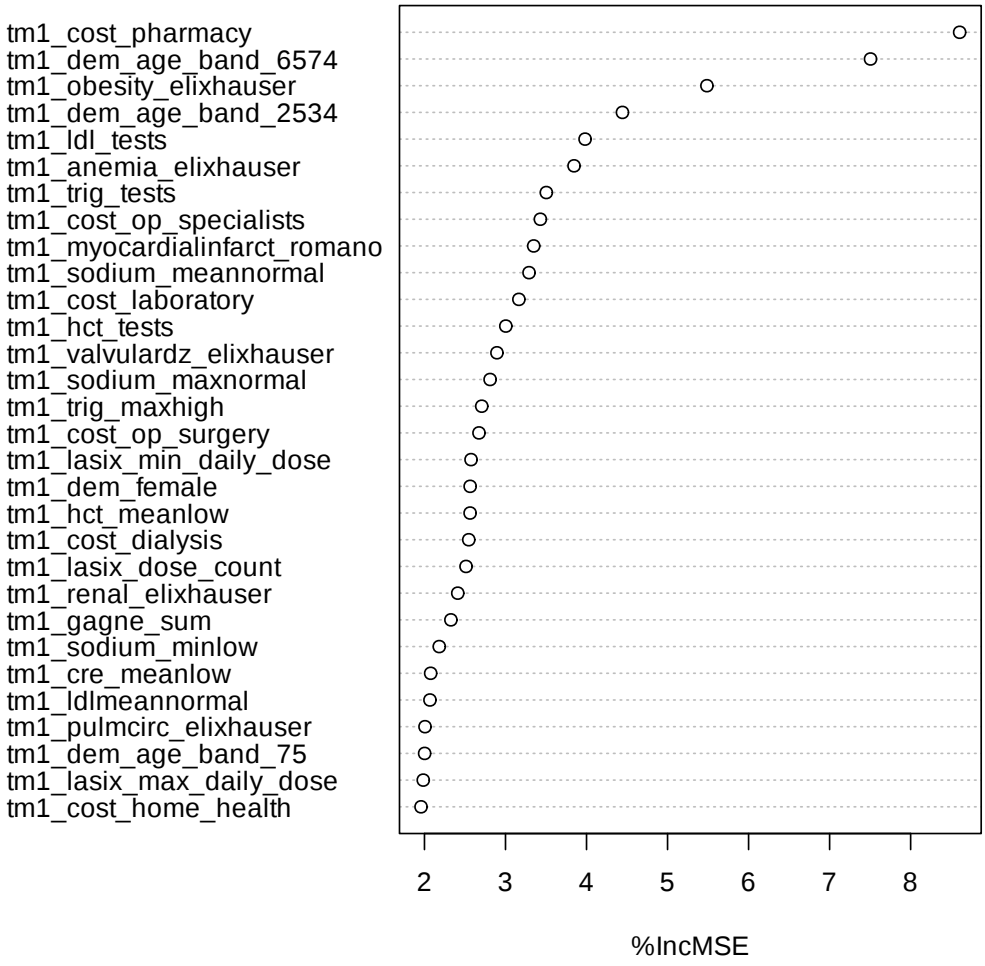
[4]:
```
## YOUR QUESTION 2 CODE GOES HERE
# Model 1: Predicting costs excluding race
mod1 <- randomForest(as.formula(paste(outcome_var, "~", paste(exclude_race,␣
 ↪collapse="+"))), data=training_data, ntree=100,␣
 ↪mtry=round(length(exclude_race)/3), importance=TRUE)

# Model 2: Predicting costs including race
mod2 <- randomForest(as.formula(paste(outcome_var, "~", paste(all_predictors,␣
 ↪collapse="+"))), data=training_data, ntree=100,␣
 ↪mtry=round(length(all_predictors)/3), importance=TRUE)

# Model 3: Predicting health excluding race
mod3 <- randomForest(as.formula(paste(health_outcome_var, "~",␣
 ↪paste(exclude_race, collapse="+"))), data=training_data, ntree=100,␣
 ↪mtry=round(length(exclude_race)/3), importance=TRUE)

# Model 4: Predicting health including race
mod4 <- randomForest(as.formula(paste(health_outcome_var, "~",␣
 ↪paste(all_predictors, collapse="+"))), data=training_data, ntree=100,␣
 ↪mtry=round(length(all_predictors)/3), importance=TRUE)

# Variable importance plots
varImpPlot(mod1, type=1, main="Model 1 Variable Importance")
ggsave("mod1_importance.png")

varImpPlot(mod2, type=1, main="Model 2 Variable Importance")
ggsave("mod2_importance.png")

varImpPlot(mod3, type=1, main="Model 3 Variable Importance")
ggsave("mod3_importance.png")

varImpPlot(mod4, type=1, main="Model 4 Variable Importance")
ggsave("mod4_importance.png")

# Export predictions for the train and test sample
y_train_predictions_mod1 <- predict(mod1, newdata=training_data)
y_test_predictions_mod1 <- predict(mod1, newdata=test_data)

y_train_predictions_mod2 <- predict(mod2, newdata=training_data)
y_test_predictions_mod2 <- predict(mod2, newdata=test_data)

y_train_predictions_mod3 <- predict(mod3, newdata=training_data)
y_test_predictions_mod3 <- predict(mod3, newdata=test_data)

y_train_predictions_mod4 <- predict(mod4, newdata=training_data)
```

```r
y_test_predictions_mod4 <- predict(mod4, newdata=test_data)

# Save the results
write_dta(test_data, "lab8_2024_results.dta")
```

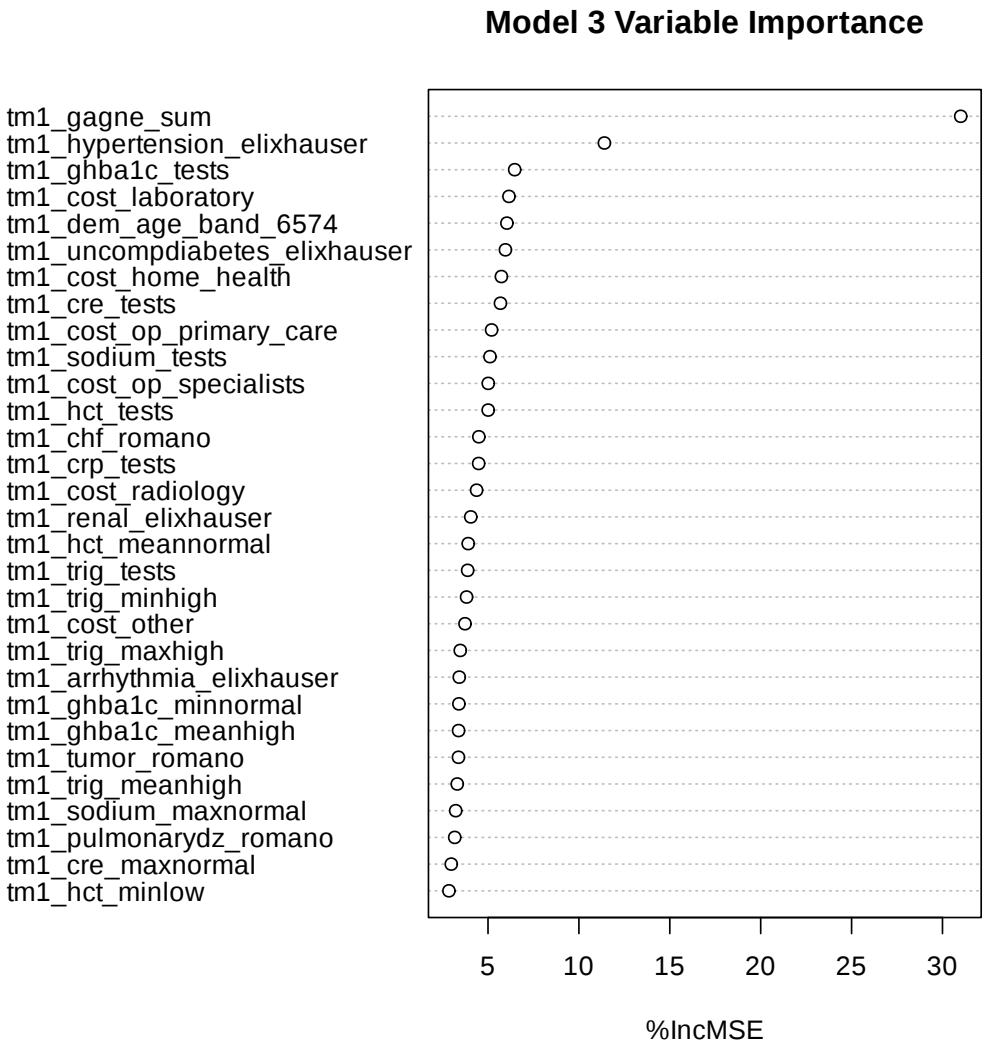
```
Saving 6.67 x 6.67 in image
```



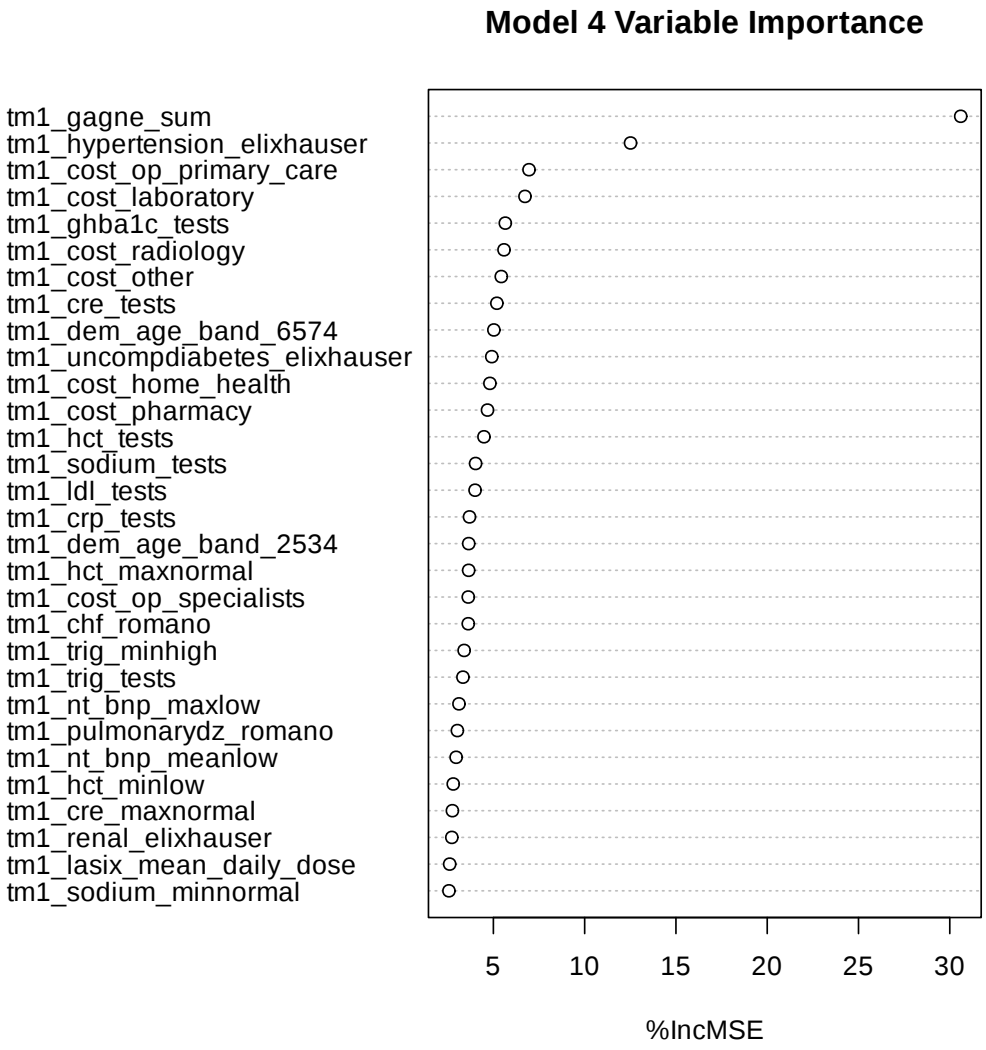

```
Saving 6.67 x 6.67 in image
```




```
Saving 6.67 x 6.67 in image
```




```
Saving 6.67 x 6.67 in image
```



Question 2 Answer

Example code commented out and modified code given above.

3. Calculate and compare the root mean squared prediction error for your models that include patient race vs. those that exclude patient race in the **training sample**.

[5]:
```
#-------------------------------------------------------------------------------
# Compare RMSE for models 1-4
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Calculate and compare the mean squared error in the training sample:
#-------------------------------------------------------------------------------
```

```
## Root mean squared prediction error in  the training sample.
p <- 4
RMSPE <- matrix(0, p, 1)

## Model 1
RMSPE[1] <- sqrt(mean((training_data$cost_t - y_train_predictions_mod1)^2, na.
 ↪rm=TRUE))

## Model 2
RMSPE[2] <- sqrt(mean((training_data$cost_t - y_train_predictions_mod2)^2, na.
 ↪rm=TRUE))

## Model 3
RMSPE[3] <- sqrt(mean((training_data$gagne_sum_t - y_train_predictions_mod3)^2,␣
 ↪na.rm=TRUE))

## Model 4
RMSPE[4] <- sqrt(mean((training_data$gagne_sum_t - y_train_predictions_mod4)^2,␣
 ↪na.rm=TRUE))

#Display a table of the results
data.frame(algorithm = c("Model 1 - Costs (excl. race) ",
                         "Model 2 - Costs (incl. race) ",
                         "Model 3 - Health (excl. race)",
                         "Model 4 - Health (incl. race)"),
           RMSPE)
```

A data.frame: 4 × 2

algorithm <chr>	RMSPE <dbl>
Model 1 - Costs (excl. race)	7884.5601929
Model 2 - Costs (incl. race)	7919.0439181
Model 3 - Health (excl. race)	0.4633335
Model 4 - Health (incl. race)	0.4654095

Question 3 Answer

RMSPE given above.

4. Calculate and compare the root mean squared prediction error for your models that include patient race vs. those that exclude patient race in the **test sample**.

[6]:
```
#------------------------------------------------------------------------------
# Calculate and compare the mean squared error in  the lock box data
#------------------------------------------------------------------------------

## Root mean squared prediction error in the test sample.
p <- 4
RMSPE_OOS <- matrix(0, p, 1)
```

```
## Model 1
RMSPE_OOS[1] <- sqrt(mean((test_data$cost_t - y_test_predictions_mod1)^2, na.
 ↪rm=TRUE))

## Model 2
RMSPE_OOS[2] <- sqrt(mean((test_data$cost_t - y_test_predictions_mod2)^2, na.
 ↪rm=TRUE))

## Model 3
RMSPE_OOS[3] <- sqrt(mean((test_data$gagne_sum_t - y_test_predictions_mod3)^2,␣
 ↪na.rm=TRUE))

## Model 4
RMSPE_OOS[4] <- sqrt(mean((test_data$gagne_sum_t - y_test_predictions_mod4)^2,␣
 ↪na.rm=TRUE))


#Display a table of the results
data.frame(algorithm = c("Model 1 - Costs (excl. race) ",
                         "Model 2 - Costs (incl. race) ",
                         "Model 3 - Health (excl. race)",
                         "Model 4 - Health (incl. race)"),
           RMSPE_OOS)
```

A data.frame: 4 × 2

algorithm <chr>	RMSPE_OOS <dbl>
Model 1 - Costs (excl. race)	16262.201021
Model 2 - Costs (incl. race)	16268.600934
Model 3 - Health (excl. race)	1.051134
Model 4 - Health (incl. race)	1.051537

Question 4 Answer

RMPSE given above.

5. Export a data set with **the test data** and your predictions as a .dta file. If you are in a Stata lab, you can exit Python and load this file into Stata for further analysis.

[7]:
```
#-------------------------------------------------------------------------------
# Export test data set with predictions
#-------------------------------------------------------------------------------

#Export data set with training data + predictions from the models
lab8 <- test_data

lab8$y_test_predictions_mod1 <- y_test_predictions_mod1
lab8$y_test_predictions_mod2 <- y_test_predictions_mod2
lab8$y_test_predictions_mod3 <- y_test_predictions_mod3
```

```
lab8$y_test_predictions_mod4 <- y_test_predictions_mod4

write_dta(lab8, "lab8_2024_results.dta")
```

Question 5 Answer

(Answer here; include your images if needed.)

6. As in Lab 1 and Lab 2, convert the predictions in the test sample from each of your prediction algorithms into percentile ranks, normalized so that the top rank is equal to 100. The percentile rank is the "risk score" from the algorithm.

[8]:
```
# QUESTION 6 Code
# Calculate percentile ranks for predictions from each model and normalize so␣
 ↪top rank equals 100
test_data <- test_data %>%
  mutate(
    prediction_mod1_rank = ntile(-y_test_predictions_mod1, 100),  # Use␣
 ↪negative to rank highest prediction as 100
    prediction_mod2_rank = ntile(-y_test_predictions_mod2, 100),
    prediction_mod3_rank = ntile(-y_test_predictions_mod3, 100),
    prediction_mod4_rank = ntile(-y_test_predictions_mod4, 100)
  )

# Since ntile ranks from 1 to 100 (with 1 being the lowest), to make 100 the␣
 ↪highest score, we subtract each rank from 101
test_data <- test_data %>%
  mutate(
    prediction_mod1_rank = 101 - prediction_mod1_rank,
    prediction_mod2_rank = 101 - prediction_mod2_rank,
    prediction_mod3_rank = 101 - prediction_mod3_rank,
    prediction_mod4_rank = 101 - prediction_mod4_rank
  )

# View the first few rows to confirm the changes
head(test_data)
summary(test_data$prediction_mod1_rank)
summary(test_data$prediction_mod2_rank)
summary(test_data$prediction_mod3_rank)
summary(test_data$prediction_mod4_rank)
```

A tibble: 6 × 160

patient_id <dbl>	gagne_sum_t <dbl>	cost_t <dbl>	cost_avoidable_t <dbl>	race <chr>	tm1_dem_black <dbl>	tm1_d <dbl>
100001	0	1200	0	white	0	0
100002	3	2600	0	white	0	1
100004	0	1300	0	white	0	1
100005	1	1100	0	white	0	1
100006	1	123700	0	white	0	1
100007	1	12900	0	white	0	1

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   26.00   51.00   50.53   76.00  100.00

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   26.00   51.00   50.53   76.00  100.00

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   26.00   51.00   50.53   76.00  100.00

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   26.00   51.00   50.53   76.00  100.00
```

Question 6 Answer

Code given above.

7. Now consider a program that makes patients eligible for extra resources if their "risk score" is above the 55th percentile. (This corresponds to the top 45 percent of risk scores).
   1. As on lab 1 and 2, start by defining four new indicator variables corresponding to whether the risk score from each model is strictly greater than 55.
   2. What fraction of all Black patients would be eligible for the program using each of the four algorithms? To answer this question, report the means of the indicator variables you created in (a) after subsetting the data to Black patients (i.e., `tm1_dem_black == 1`).
   3. Among patients eligible for the program, what fraction are Black? To answer this question, report the mean of the indicator variable `tm1_dem_black` after subseting the data to patients eligible for the program for model 1. Then repeat for models 2, 3, and 4.

[9]:
```r
# QUESTION 7 Code
test_data <- test_data %>%
  mutate(eligible_model1 = ifelse(prediction_mod1_rank > 55, 1, 0),
         eligible_model2 = ifelse(prediction_mod2_rank > 55, 1, 0),
         eligible_model3 = ifelse(prediction_mod3_rank > 55, 1, 0),
         eligible_model4 = ifelse(prediction_mod4_rank > 55, 1, 0))

df_black <- test_data %>% filter(tm1_dem_black == 1)

mean_eligible_black <- sapply(df_black[, c("eligible_model1",␣
 ↪"eligible_model2", "eligible_model3", "eligible_model4")], mean)
mean_eligible_black

fraction_black_eligible <- sapply(1:4, function(i) {
  df_eligible <- test_data %>% filter(.[[paste0("eligible_model", i)]] == 1)
  mean(df_eligible$tm1_dem_black, na.rm = TRUE)
})
fraction_black_eligible
```

eligible__model1 0.51726875744343 **eligible__model2** 0.526796347757046
eligible__model3 0.588725684795554 **eligible__model4** 0.616117506947201

1. 0.131616161616162 2. 0.134040404040404 3. 0.14979797979798 4. 0.156767676767677

Question 7 Answer

A. Code given above.

B. Share of eligible black patients as given: Model 1: 0.517 Model 2: 0.526 Model 3: 0.588 Model 4: 0.616

C. Fraction of eligible that are black as given: Model 1: 0.131 Model 2: 0.134 Model 3: 0.149 Model 4: 0.156

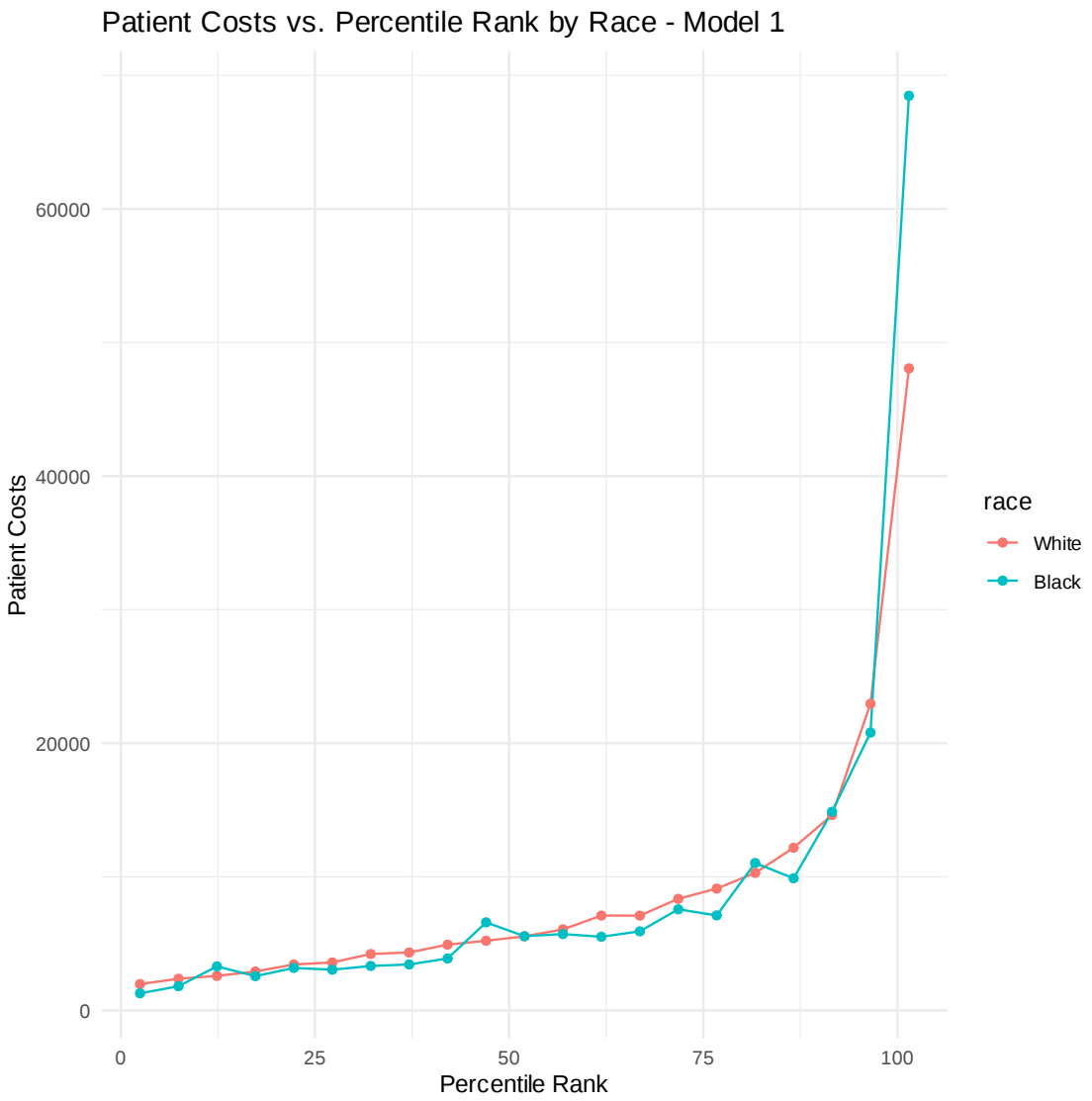
8. Now we will replicate the key figures from Obermeyer, Powers, Vogeli, and Mullainathan (2019). Produce binned scatter plots of patient costs and patient health vs. the percentile rank "risk score" from each algorithm, with White and Black patients plotted separately. This is a total of 8 graphs: 4 models x 2 outcomes.

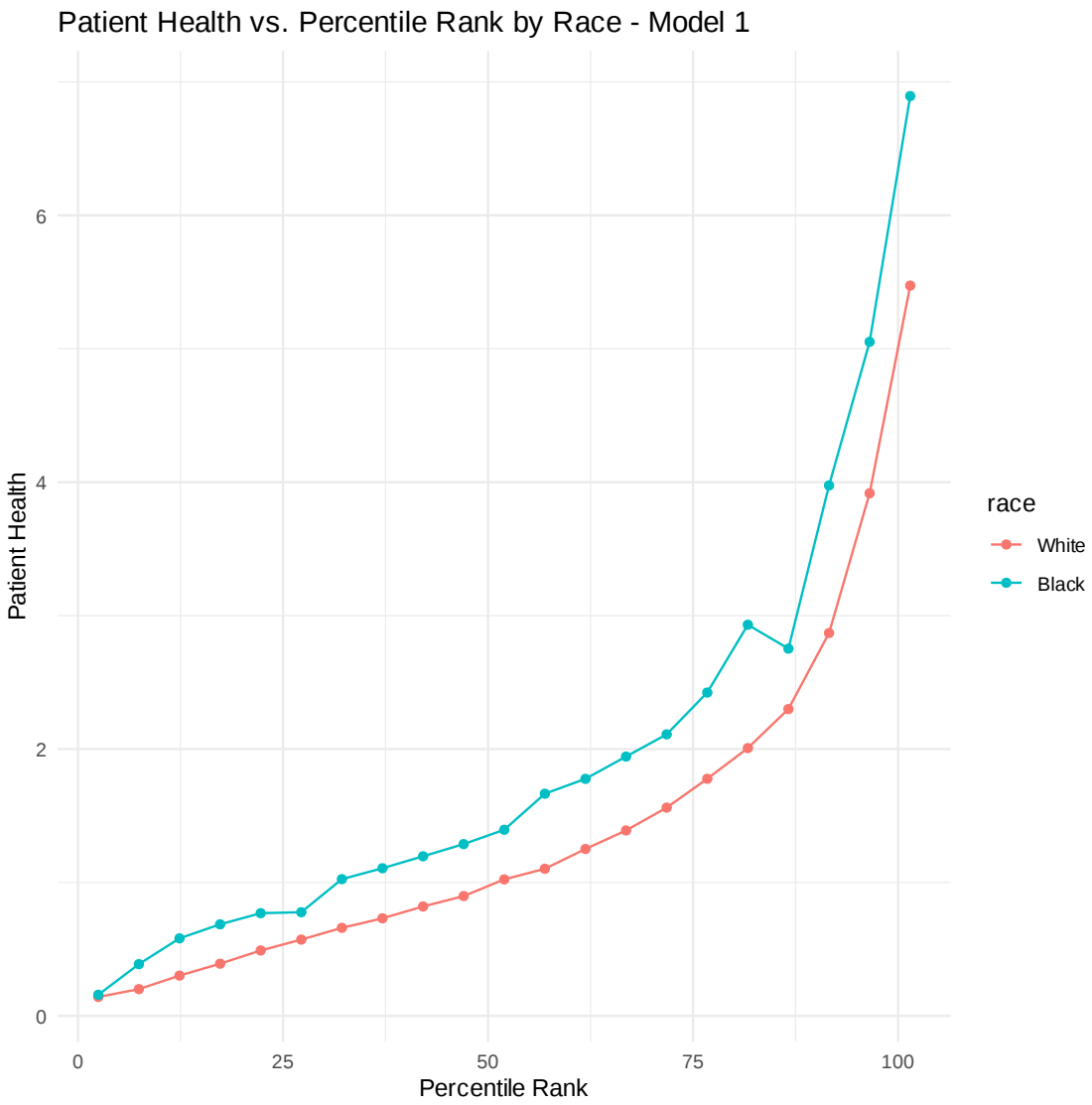
   In R, you could do the same in ggplot by using the geom="line" option and geom="point" option in stat_binmean from the statar package, and set the color option to the race variable to plot Black and White patients separately:

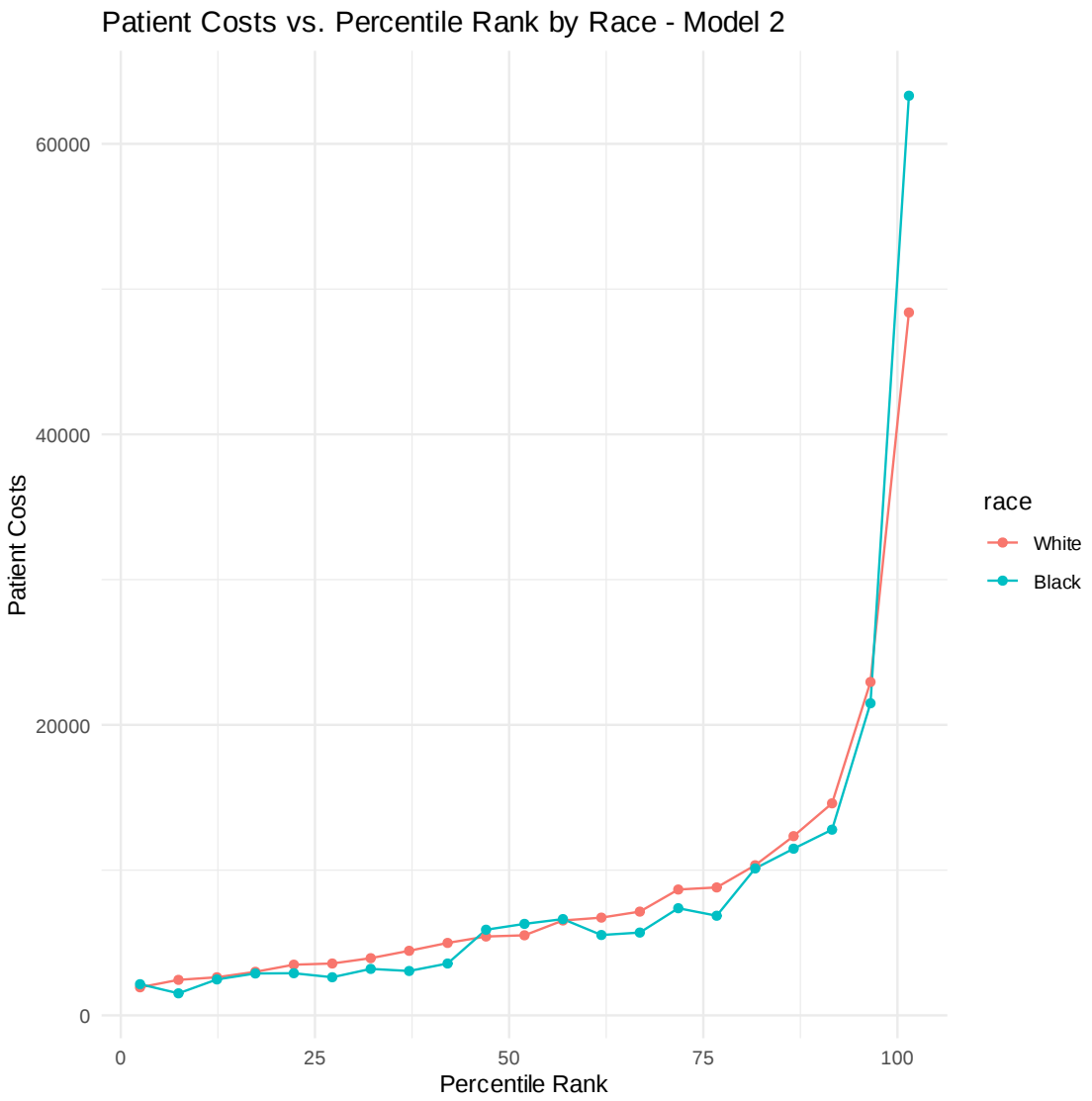
   ```
   ggplot(dat, aes(x = percentile_rank , y = outcome_variable, color = race)) +
   stat_binmean(n = 20, geom = "line") +
   stat_binmean(n = 20, geom = "point")
   ```

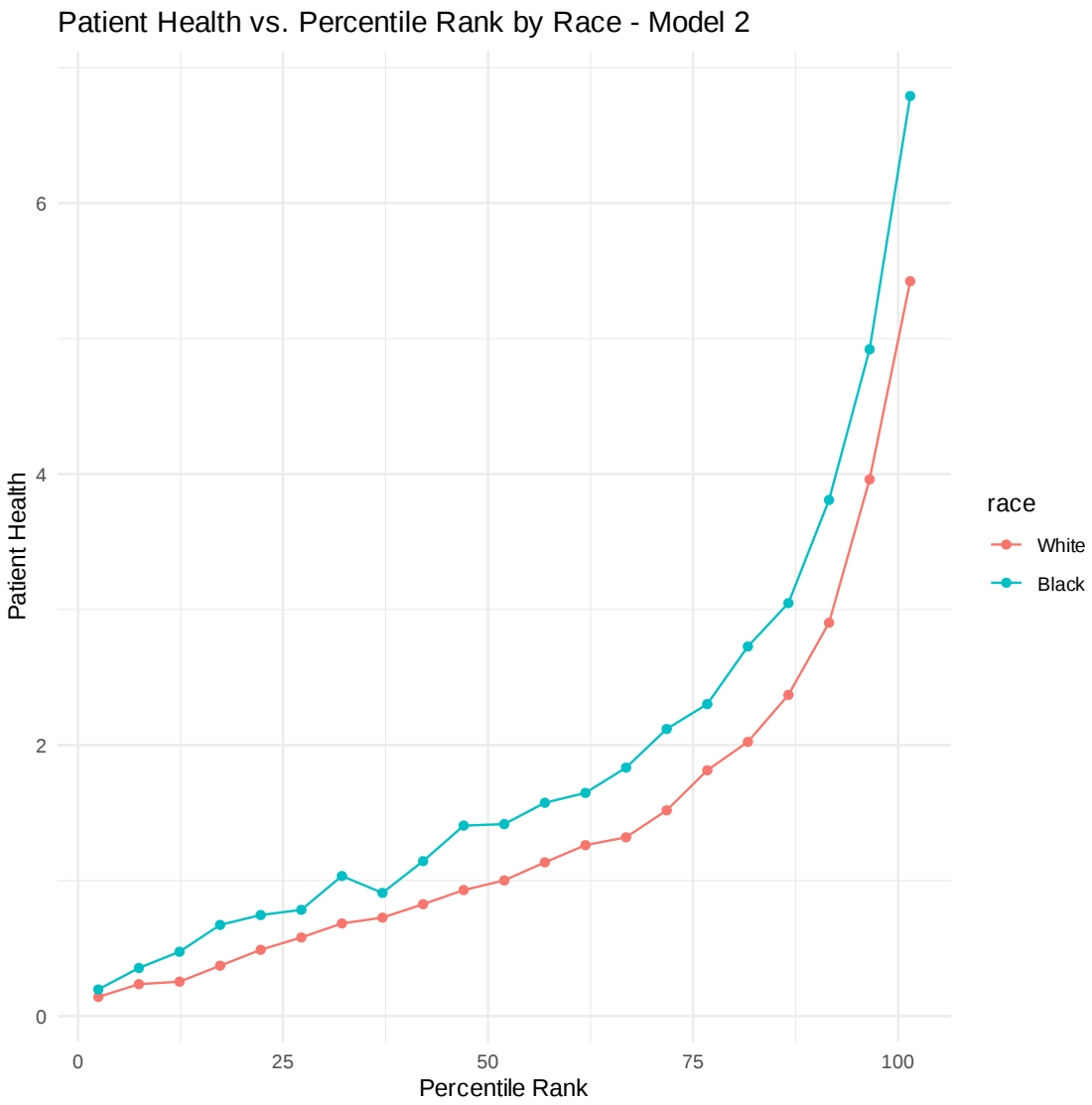
[10]:
```
# QUESTION 8 Code
# Convert the race variable to a factor with descriptive labels in test_data
test_data$race <- factor(test_data$tm1_dem_black, labels = c("White", "Black"))

# Plot for Model 1 - Patient Costs
ggplot(test_data, aes(x = prediction_mod1_rank, y = cost_t, color = race)) +
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Costs vs. Percentile Rank by Race - Model 1",
       x = "Percentile Rank", y = "Patient Costs") +
  theme_minimal()

# Plot for Model 1 - Patient Health
ggplot(test_data, aes(x = prediction_mod1_rank, y = gagne_sum_t, color = race))␣
 ↪+
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Health vs. Percentile Rank by Race - Model 1",
       x = "Percentile Rank", y = "Patient Health") +
  theme_minimal()

# Plot for Model 2 - Patient Costs
ggplot(test_data, aes(x = prediction_mod2_rank, y = cost_t, color = race)) +
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Costs vs. Percentile Rank by Race - Model 2",
```

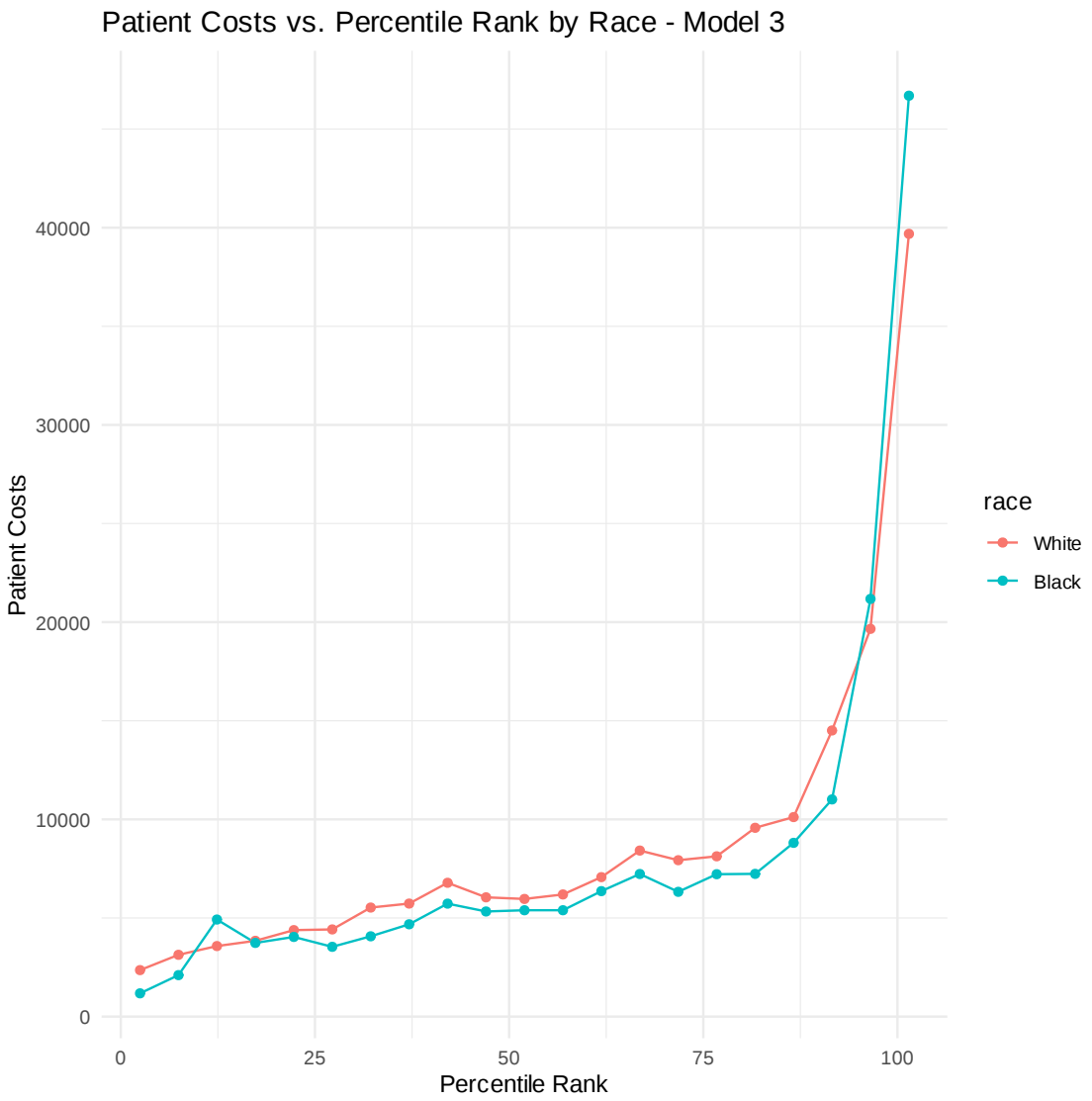
```
       x = "Percentile Rank", y = "Patient Costs") +
  theme_minimal()

# Plot for Model 2 - Patient Health
ggplot(test_data, aes(x = prediction_mod2_rank, y = gagne_sum_t, color = race))␣
 ↪+
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Health vs. Percentile Rank by Race - Model 2",
       x = "Percentile Rank", y = "Patient Health") +
  theme_minimal()

# Plot for Model 3 - Patient Costs
ggplot(test_data, aes(x = prediction_mod3_rank, y = cost_t, color = race)) +
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Costs vs. Percentile Rank by Race - Model 3",
       x = "Percentile Rank", y = "Patient Costs") +
  theme_minimal()

# Plot for Model 3 - Patient Health
ggplot(test_data, aes(x = prediction_mod3_rank, y = gagne_sum_t, color = race))␣
 ↪+
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Health vs. Percentile Rank by Race - Model 3",
       x = "Percentile Rank", y = "Patient Health") +
  theme_minimal()

# Plot for Model 4 - Patient Costs
ggplot(test_data, aes(x = prediction_mod4_rank, y = cost_t, color = race)) +
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Costs vs. Percentile Rank by Race - Model 4",
       x = "Percentile Rank", y = "Patient Costs") +
  theme_minimal()

# Plot for Model 4 - Patient Health
ggplot(test_data, aes(x = prediction_mod4_rank, y = gagne_sum_t, color = race))␣
 ↪+
  stat_summary_bin(fun = "mean", bins = 20, geom = "line") +
  stat_summary_bin(fun = "mean", bins = 20, geom = "point") +
  labs(title = "Patient Health vs. Percentile Rank by Race - Model 4",
       x = "Percentile Rank", y = "Patient Health") +
  theme_minimal()
```

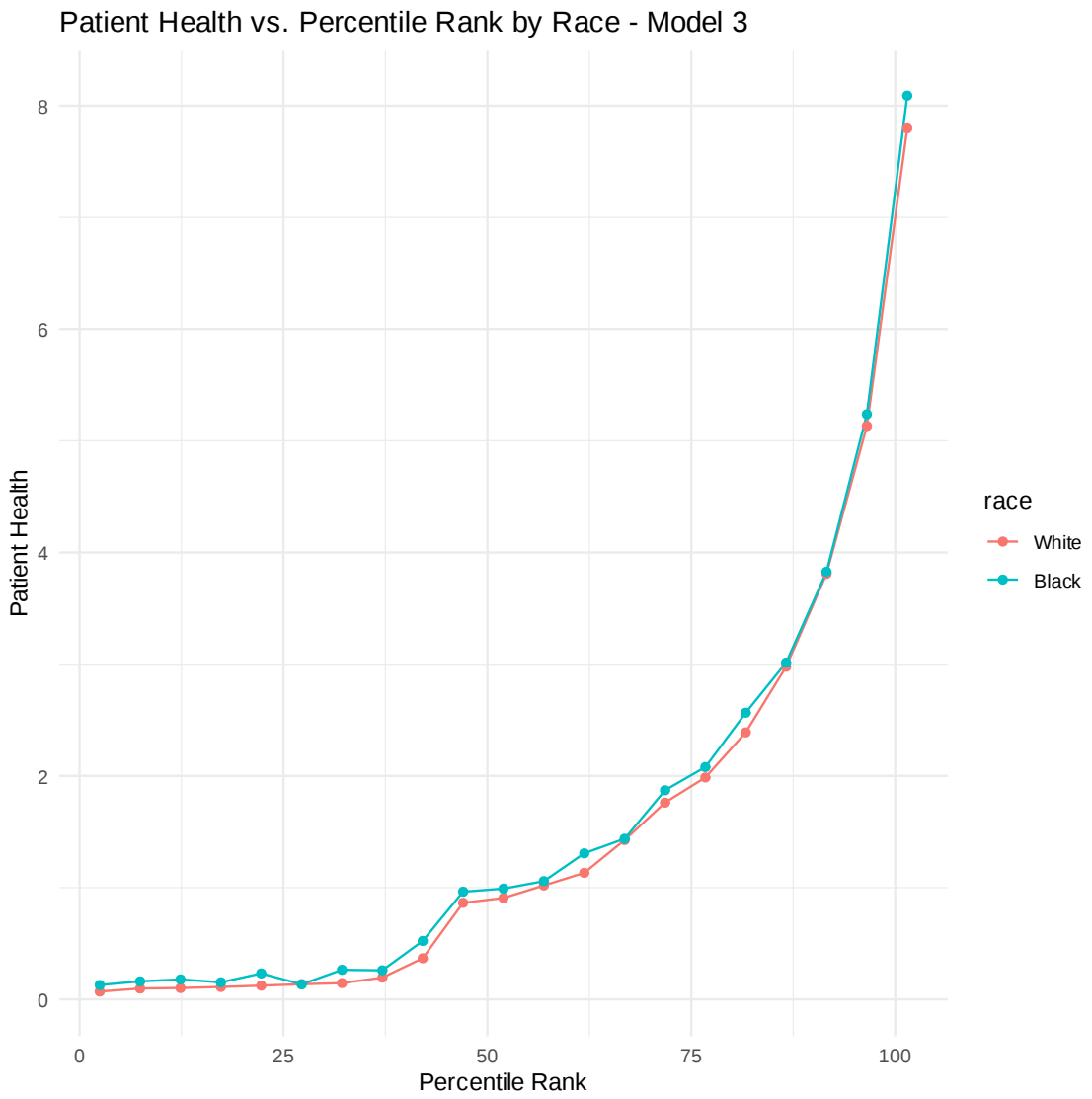

Patient Costs vs. Percentile Rank by Race - Model 1
Patient Costs
60000
40000
20000
0
0
25
50
75
100
Percentile Rank
race
White
Black

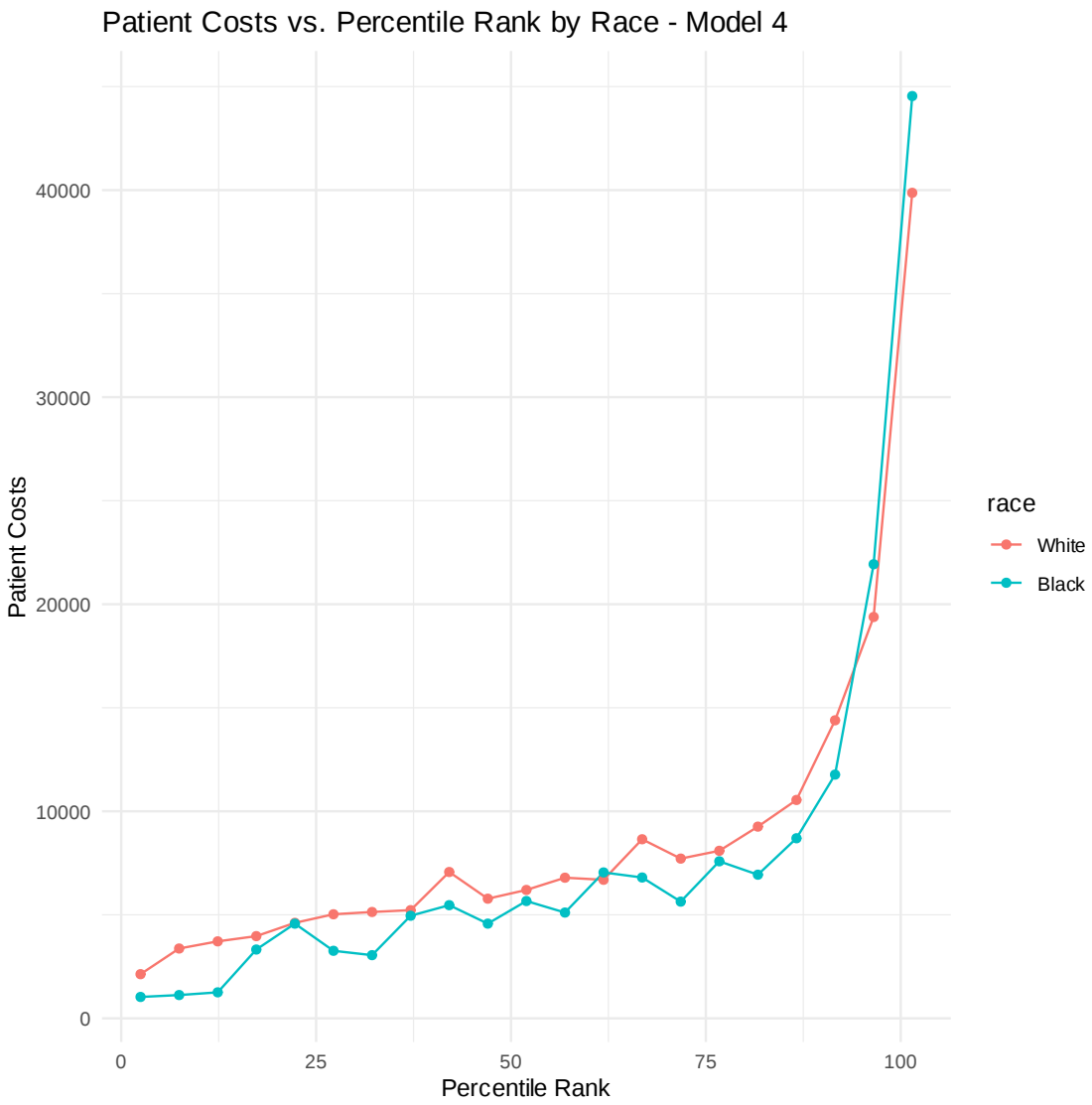

Patient Health vs. Percentile Rank by Race - Model 1
Patient Health
Percentile Rank
0
2
4
6
0
25
50
75
100
race
White
Black

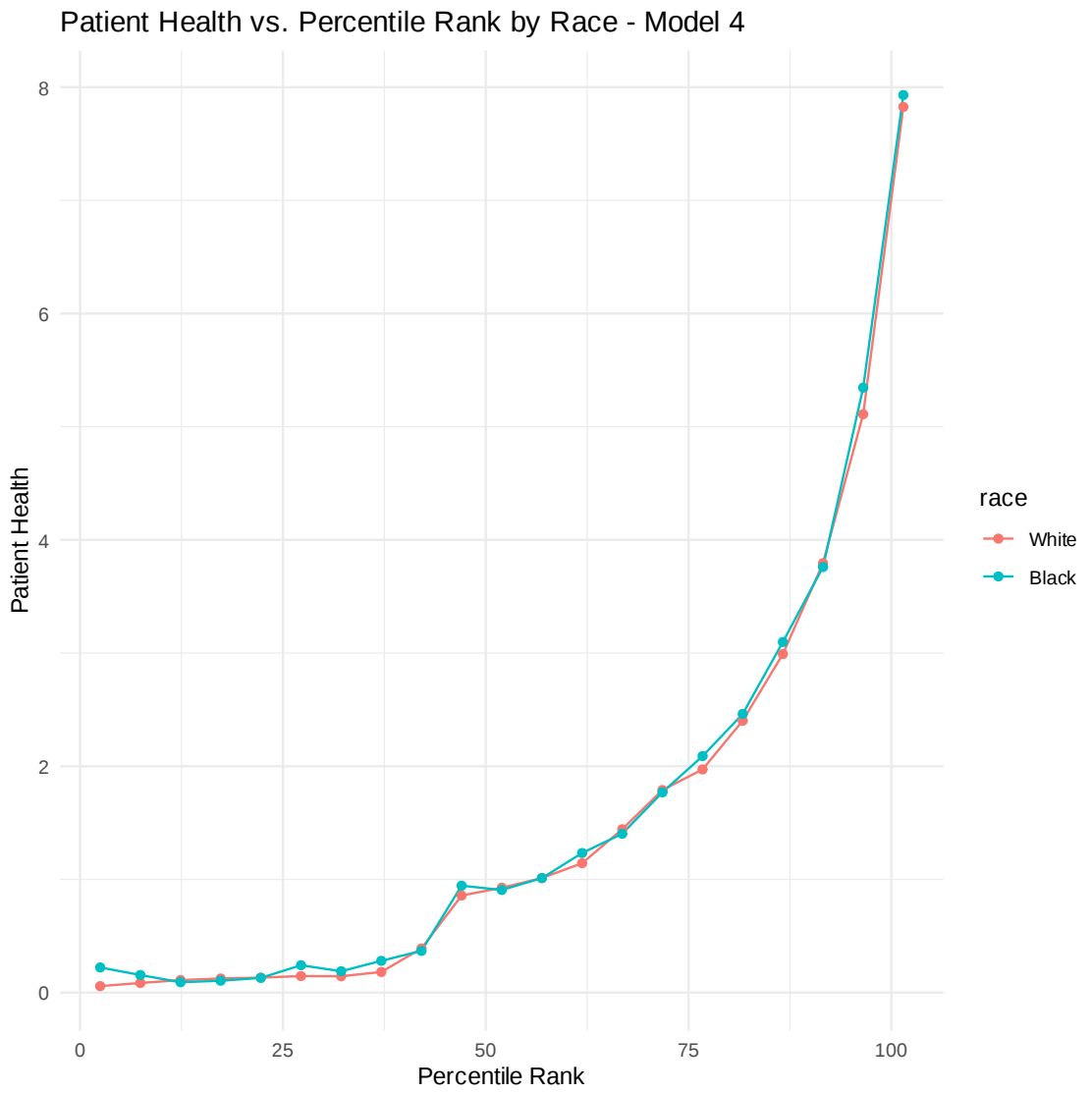

Patient Costs vs. Percentile Rank by Race - Model 2
60000
40000
20000
0
Patient Costs
0
25
50
75
100
Percentile Rank
race
White
Black


Patient Health vs. Percentile Rank by Race - Model 2
Patient Health
Percentile Rank
0
2
4
6
0
25
50
75
100
race
White
Black


Patient Costs vs. Percentile Rank by Race - Model 3
Patient Costs
40000
30000
20000
10000
0
0
25
50
75
100
Percentile Rank
race
White
Black


Patient Health vs. Percentile Rank by Race - Model 3
Patient Health
Percentile Rank
0
2
4
6
8
0
25
50
75
100
race
White
Black


Patient Costs vs. Percentile Rank by Race - Model 4
Patient Costs
40000
30000
20000
10000
0
0
25
50
75
100
Percentile Rank
race
White
Black



Question 8 Answer

(Answer here; include your images if needed.)

9. In the pre-recorded video for this lab, Professor Ziad Obermeyer said that it is the left-hand side variable (i.e., the "label" or target parameter) that is the source of bias in algorithms, not the right-hand side variables (i.e., the predictors). Explain what he meant, and evaluate whether you agree with him using your binned scatters above.

[11]:
```
# QUESTION 9 Code
```

Question 9 Answer

Professor Obermeyer is referring to the idea that bias in predictive algorithms often stems from the target variable used in the model, rather than the predictors. In other words, if the outcome

being predicted is itself a result of biased processes or decisions, then the algorithm, even if it uses unbiased predictors, will perpetuate or even amplify that bias.

From the binned scatter plots above, we can examine this concept. If the outcome variable, such as patient cost or health, has historically been influenced by biased treatment decisions, measurement methods, or systematic inequalities, then models trained to predict these outcomes will reflect those biases. This can happen even if the predictors do not directly include race or other sensitive attributes because the outcome variable effectively "encodes" the bias. In the graphs shown, if there's a consistent discrepancy between the risk scores and actual costs or health outcomes for Black and White patients, especially at similar percentile ranks, this might suggest that the model is reflecting biases in the target parameter.

For instance, if we see that at the same percentile rank, Black patients consistently have higher costs or worse health outcomes compared to White patients, this might indicate that the health outcomes or costs have been influenced by external biased factors which are now being captured by the model. While this holds in some plots (for instance, patient health in model 2), it is not always the case and would require further investigation I think.

10. Create an annotated/commented do-file, .ipynb Jupyter Notebook, or .R file that can replicate all your analyses above. This will be the final code that you submit on Gradescope. The motivation for using do-files and .R files is described on page 4, which has been adapted from training materials used by Innovations for Poverty Action (IPA) and the Abdul Latif Jameel Poverty Action Lab (J-PAL).

Final Submission Checklist for Lab 8

If you're working with R

If you're working with Stata

Lab 8 Write-Up:

PDF of your answers. For graphs, you must save them as images (e.g., .png files) and insert them into the document.

Lab 8 Code:

.R script file, well-annotated replicating all your analyses;OR

.ipynb file and a .PDF version of this file.

Lab 8 Write-Up:

PDF of your answers. For graphs, you must save them as images (e.g., .png files) and insert them into the document.

Lab 8 Code:

do-file, well-annotated replicating all your analyses;AND

log-file, not a .smcl file, with the log showing the output generated by your final do-file.

If you're working with an .ipynb notebook

It is likely that your .ipynb file will be greater than 1 MB in size. Therefore, for this assignment please submit both your *well-annotated* **.ipynb file** and **a .PDF version of this file**. The notebook should replicate all your analyses for Lab 5 (with enough comments that a principal

investigator on a research project would be able to follow and understand what each step of the code is doing).

1.3 How to submit your assignment

Step 1 Access the lab assignment under the "Assignments" tab on Canvas
Step 2 Access Gradescope from Canvas
Step 3 Access the lab assignment on Gradescope
Step 4 Upload your files *Check What files to submit to confirm what files you need to submit.*
Step 5 What you'll see after submitting your lab assignment
Step 6 Check your submitted files
Step 7 You'll receive an email confirmation as well

1.4 What files to submit

If you're using Python Notebook to write your R code, and a document editor to write your answers
If you're using a Python Notebook to write your R code AND to write your answers

1.5 WHAT ARE DO-FILES AND .R FILES AND WHY DO WE NEED ONE?

Let's imagine the following situation - you just found out you have to present your results to a partner– all the averages you produced and comparisons you made. Suppose you also found out that the data you had used to produce all these results was not completely clean, and have only just fixed it. You now have incorrect numbers and need to re-do everything.

How would you go about it? Would you reproduce everything you did for Lab 1 from scratch? Can you do it? How long would it take you to do? Just re-typing all those commands into Stata or R in order and checking them would take an hour.

An important feature of any good research project is that the results should be reproducible. For Stata and R the easiest way to do this is to create a text file that lists all your commands in order, so anyone can re-run all your Stata or R work on a project anytime. Such text files that are produced within Stata or linked to Stata are called do-files, because they have an extension .do (like intro_exercise.do). Similarly, in R, these files are called .R files because they have an extension of .R. These files feed commands directly into Stata or R without you having to type or copy them into the command window.

An added bonus is that having do-files and .R files makes it very easy to fix your typos, re-order commands, and create more complicated chains of commands that wouldn't work otherwise. You can now quickly reproduce your work, correct it, adjust it, and build on it.

Finally, do-files and .R files make it possible for multiple people to work on a project, which is necessary for collaborating with others or when you hand off a project to someone else.

1.6 DATA DESCRIPTION, FILE: health.dta

The data consist of 48,784 patient records. Variables that start with tm1_ were measured in the prior year (time $t-1$). Variable that end with _t are measured in the current year. For more details on the construction of the variables included in this data set, please see Obermeyer, Powers, Vogeli, and Mullainathan (2019).

TABLE 1

Variable Definitions

Variable

Description

mean

sd

min

max

(1)

(2)

(3)

(4)

(5)

(6)

patient_id

Patient identification number

n/a

n/a

n/a

n/a

gagne_sum_t

Total number of active chronic illnesses

1.354

1.942

0

17

cost_t

Total medical expenditures, rounded to the nearest 100

7,660

17,990

0

550,500

cost_avoidable_t

Total avoidable (emergency + inpatient) medical expenditures, rounded to nearest

2,435

12,058

0

642,700

race

String variable containing the words “black” and “white”

n/a

n/a

n/a

n/a

tm1_dem_black

1 = Black

0 = White

0.114

0.318

0

1

tm1_dem_female

1 = Female

0 = Male

0.631

0.483

0

1

tm1_dem_age_band_1824

Indicator for patient age between 18-24

0.0369

0.188

0

1

tm1_dem_age_band_2534

Indicator for patient age between 25-34

0.110

0.313

0

1

tm1_dem_age_band_3544

Indicator for patient age between 35-44

0.194

0.396

0

1

tm1_dem_age_band_4554

Indicator for patient age between 45-54

0.239

0.427

0

1

tm1_dem_age_band_5564

Indicator for patient age between 55-64

0.197

0.397

0

1

tm1_dem_age_band_6574

Indicator for patient age between 65-74

0.142

0.349

0

1

tm1_dem_age_band_75

Indicator for patient age 75+

0.0703

0.256

0

1

tm1_alcohol_elixhauser

Indicator for alcohol abuse

0.00892

0.0940

0

1

tm1_anemia_elixhauser

Indicator for deficiency anemia

0.0636

0.244

0

1

tm1_arrhythmia_elixhauser

Indicator for arrhythmia

0.0922

0.289

0

1

tm1_arthritis_elixhauser

Indicator for arthritis

0.0466

0.211

0

1

tm1_bloodlossanemia_elixhauser

Indicator for blood loss anemia

0.00246

0.0495

0

1

tm1_coagulopathy_elixhauser

Indicator for coagulopathy

0.0115

0.107

0

1

tm1_compdiabetes_elixhauser

Indicator for diabetes, complicated

0.0217

0.146

0

1

tm1_depression_elixhauser

Indicator for depression

0.0621

0.241

0

1

tm1_drugabuse_elixhauser

Indicator for drug abuse

0.00623

0.0787

0

1

tm1_electrolytes_elixhauser

Indicator for electrolyte disorder

0.0329

0.178

0

1

tm1_hypertension_elixhauser

Indicator for hypertension

0.332

0.471

0

1

tm1_hypothyroid_elixhauser

Indicator for hypothyroid

0.0938

0.292

0

1

tm1_liver_elixhauser

Indicator for liver disease

0.0159

0.125

0

1

tm1_neurodegen_elixhauser

Indicator for neurodegenerative disease

0.0280

0.165

0

1

tm1_obesity_elixhauser

Indicator for obesity

0.0929

0.290

0

1

tm1_paralysis_elixhauser

Indicator for paralysis

0.000574

0.0240

0

1

tm1_psychosis_elixhauser

Indicator for psychoses

0.0325

0.177

0

1

tm1_pulmcirc_elixhauser

Indicator for pulmonary circulation disorders

0.00558

0.0745

0

1

tm1_pvd_elixhauser

Indicator for peripheral vascular disorders

0.0263

0.160

0

1

tm1_renal_elixhauser

Indicator for renal failure

0.0367

0.188

0

1

tm1_uncompdiabetes_elixhauser

Indicator for diabetes, uncomplicated

0.0987

0.298

0

1

tm1_valvulardz_elixhauser

Indicator for valvular disease

0.0315

0.175

0

1

tm1_wtloss_elixhauser

Indicator for weight loss

0.00139

0.0373

0

1

tm1_cerebrovasculardz_romano

Indicator for cerebrovascular disease

0.0283

0.166

0

1

tm1_chf_romano

Indicator for congestive heart failure

0.0319

0.176

0

1

tm1_dementia_romano

Indicator for dementia

0.00949

0.0970

0

1

tm1_hemiplegia_romano

Indicator for hemiplegia

0.00266

0.0516

0

1

tm1_hivaids_romano

Indicator for HIV/AIDS

0.00305

0.0552

0

1

tm1_metastatic_romano

Indicator for metastasis

0.00613

0.0780

0

1

tm1_myocardialinfarct_romano

Indicator for myocardial infarction

0.0169

0.129

0

1

tm1_pulmonarydz_romano

Indicator for pulmonary disease

0.102

0.302

0

1

tm1_tumor_romano

Indicator for tumor

0.0944

0.292

0

1

tm1_ulcer_romano

Indicator for ulcer

0.00480

0.0691

0

1

tm1_cost_dialysis

Total costs for dialysis, rounded to nearest 10

26.72

976.6

0

63,410

tm1_cost_emergency

Total costs for emergency, rounded to nearest 10

423.7

1,572

0

67,090

tm1_cost_home_health

Total costs for home health, rounded to nearest 10

220.5

1,396

0

56,830

tm1_cost_ip_medical

Total costs for inpatient medical, rounded to nearest 10

638.8

4,570

0

282,300

tm1_cost_ip_surgical

Total costs for inpatient surgical, rounded to nearest 10

978.5

6,575

0

279,930

tm1_cost_laboratory

Total costs for laboratory, rounded to nearest 10

330.9

949.4

-490

62,720

tm1_cost_op_primary_care

Total costs for outpatient primary care, rounded to nearest 10

473.9

1,872

0

240,290

tm1_cost_op_specialists

Total costs for outpatient specialists, rounded to nearest 10

866.2

1,546

0

41,720

tm1_cost_op_surgery

Total costs for outpatient surgery, rounded to nearest 10

846.6

2,659

0

75,790

tm1_cost_other

Total other costs, rounded to nearest 100

1,569

4,639

0

193,200

tm1_cost_pharmacy

Total costs for pharmacy, rounded to nearest 10

342.5

3,995

-10

153,250

tm1_cost_physical_therapy

Total costs for physical therapy, rounded to nearest 10

167.2

534.0

0

10,240

tm1_cost_radiology

Total costs for radiology, rounded to nearest 10

241.1

580.8

0

20,710

tm1_lasix_dose_count

Number of Lasix doses

0.0182

0.228

0

9

tm1_lasix_min_daily_dose

Minimum daily dose of Lasix

0.353

4.370

0

200

tm1_lasix_mean_daily_dose

Mean daily dose of Lasix

0.378

4.535

0

160

tm1_lasix_max_daily_dose

Maximum daily dose of Lasix

0.418

5.247

0

200

tm1_cre_tests

Number of c-reatinine tests

1.237

3.396

0

166

tm1_crp_tests

Number of c-reactive protein tests

0.000471

0.0226

0

2

tm1_esr_tests

Number of erythrocyte sedimentation rate tests

0.113

0.538

0

13

tm1_ghba1c_tests

Number of GHbA1c tests

0.385

0.748

0

9

tm1_hct_tests

Number of hematocrit tests

1.089

3.140

0

164

tm1_ldl_tests

Number of LDL tests

0.520

0.701

0

10

tm1_nt_bnp_tests

Number of BNP tests

0.0305

0.257

0

10

tm1_sodium_tests

Number of sodium tests

1.156

3.237

0

122

tm1_trig_tests

Number of triglycerides tests

0.483

0.681

0

12

tm1_cre_minlow

Indicator for low (< 0.84) minimum creatinine test result

0.222

0.416

0

1

tm1_cre_minhigh

Indicator for high (> 1.21) minimum creatinine test result

0.0391

0.194

0

1

tm1_cre_minnormal

Indicator for normal minimum creatinine test result

0.236

0.424

0

1

tm1_cre_meanlow

Indicator for low (< 0.84) mean creatinine test result

0.200

0.400

0

1

tm1_cre_meanhigh

Indicator for high (> 1.21) mean creatinine test result

0.0512

0.220

0

1

tm1_cre_meannormal

Indicator for normal mean creatinine test result

0.245

0.430

0

1

tm1_cre_maxlow

Indicator for low (< 0.84) maximum creatinine test result

0.178

0.383

0

1

tm1_cre_maxhigh

Indicator for high (> 1.21) maximum creatinine test result

0.0674

0.251

0

1

tm1_cre_maxnormal

Indicator for normal maximum creatinine test result

0.252

0.434

0

1

tm1_crp_minlow

Indicator for low (< 1) minimum c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_minhigh

Indicator for high (> 3) minimum c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_minnormal

Indicator for normal minimum c-reactive protein test result

6.15e-05

0.00784

0

1

tm1_crp_meanlow

Indicator for low (< 1) mean c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_meanhigh

Indicator for high (> 3) mean c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_meannormal

Indicator for normal mean c-reactive protein test result

6.15e-05

0.00784

0

1

tm1_crp_maxlow

Indicator for low (< 1) maximum c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_maxhigh

Indicator for high (> 3) maximum c-reactive protein test result

0.000164

0.0128

0

1

tm1_crp_maxnormal

Indicator for normal maximum c-reactive protein test result

6.15e-05

0.00784

0

1

tm1_esr_minlow

Indicator for low (< 1) minimum erythrocyte sedimentation rate test result

0

0

0

0

tm1_esr_minhigh

Indicator for high (> 20) minimum erythrocyte sedimentation rate test result

0.0218

0.146

0

1

tm1_esr_minnormal

Indicator for normal minimum erythrocyte sedimentation rate test result

0.0514

0.221

0

1

tm1_esr_meanlow

Indicator for low (< 1) mean erythrocyte sedimentation rate test result

0

0

0

0

tm1_esr_meanhigh

Indicator for high (> 20) mean erythrocyte sedimentation rate test result

0.0245

0.155

0

1

tm1_esr_meannormal

Indicator for normal mean erythrocyte sedimentation rate test result

0.0487

0.215

0

1

tm1_esr_maxlow

Indicator for low (< 1) maximum erythrocyte sedimentation rate test result

0

0

0

0

tm1_esr_maxhigh

Indicator for high (> 20) maximum erythrocyte sedimentation rate test result

0.0265

0.161

0

1

tm1_esr_maxnormal

Indicator for normal maximum erythrocyte sedimentation rate test result

0.0470

0.212

0

1

tm1_ghba1c_minlow

Indicator for low (< 4) minimum GHbA1c test result

4.10e-05

0.00640

0

1

tm1_ghba1c_minhigh

Indicator for high (> 5.7) minimum GHbA1c test result

0.123

0.329

0

1

tm1_ghba1c_minnormal

Indicator for normal minimum GHbA1c test result

0.146

0.353

0

1

tm1_ghba1c_meanlow

Indicator for low (< 4) mean GHbA1c test result

4.10e-05

0.00640

0

1

tm1_ghba1c_meanhigh

Indicator for high (> 5.7) mean GHbA1c test result

0.130

0.336

0

1

tm1_ghba1c_meannormal

Indicator for normal mean GHbA1c test result

0.140

0.347

0

1

tm1_ghba1c_maxlow

Indicator for low (< 4) maximum GHbA1c test result

4.10e-05

0.00640

0

1

tm1_ghba1c_maxhigh

Indicator for high (> 5.7) maximum GHbA1c test result

0.133

0.339

0

1

tm1_ghba1c_maxnormal

Indicator for normal maximum GHbA1c test result

0.137

0.344

0

1

tm1_hct_minlow

Indicator for low (< 35.5) minimum hematocrit test result

0.0639

0.245

0

1

tm1_hct_minhigh

Indicator for high (> 48.6) minimum hematocrit test result

0.00679

0.0821

0

1

tm1_hct_minnormal

Indicator for normal minimum hematocrit test result

0.375

0.484

0

1

tm1_hct_meanlow

Indicator for low (< 35.5) mean hematocrit test result

0.0424

0.202

0

1

tm1_hct_meanhigh

Indicator for high (> 48.6) mean hematocrit test result

0.00787

0.0884

0

1

tm1_hct_meannormal

Indicator for normal mean hematocrit test result

0.396

0.489

0

1

tm1_hct_maxlow

Indicator for low (< 35.5) maximum hematocrit test result

0.0242

0.154

0

1

tm1_hct_maxhigh

Indicator for high (> 48.6) maximum hematocrit test result

0.0119

0.109

0

1

tm1_hct_maxnormal

Indicator for normal maximum hematocrit test result

0.410

0.492

0

1

tm1_ldl_minlow

Indicator for low (< 50) minimum LDL test result

0.0155

0.124

0

1

tm1_ldl_minhigh

Indicator for high (> 99) minimum LDL test result

0.204

0.403

0

1

tm1_ldl_minnormal

Indicator for normal minimum LDL test result

0.198

0.398

0

1

tm1_ldlmeanlow

Indicator for low (< 50) mean LDL test result

0.0127

0.112

0

1

tm1_ldlmeanhigh

Indicator for high (> 99) mean LDL test result

0.211

0.408

0

1

tm1_ldlmeannormal

Indicator for normal mean LDL test result

0.134

0.340

0

1

tm1_ldl_maxlow

Indicator for low (< 50) maximum LDL test result

0.0117

0.108

0

1

tm1_ldl_maxhigh

Indicator for high (> 99) maximum LDL test result

0.218

0.413

0

1

tm1_ldl_maxnormal

Indicator for normal maximum LDL test result

0.127

0.333

0

1

tm1_nt_bnp_minlow

Indicator for low (< 100) minimum BNP test result

0.00488

0.0697

0

1

tm1_nt_bnp_minhigh

Indicator for high (> 450) minimum BNP test result

0.00980

0.0985

0

1

tm1_nt_bnp_minnormal

Indicator for normal minimum BNP test result

0.00543

0.0735

0

1

tm1_nt_bnp_meanlow

Indicator for low (< 100) mean BNP test result

0.00668

0.0815

0

1

tm1_nt_bnp_meanhigh

Indicator for high (> 450) mean BNP test result

0.0103

0.101

0

1

tm1_nt_bnp_meannormal

Indicator for normal minimum BNP test result

0.00344

0.0586

0

1

tm1_nt_bnp_maxlow

Indicator for low (< 100) maximum BNP test result

0.00646

0.0801

0

1

tm1_nt_bnp_maxhigh

Indicator for high (> 450) maximum BNP test result

0.0106

0.102

0

1

tm1_nt_bnp_maxnormal

Indicator for normal minimum BNP test result

0.00344

0.0586

0

1

tm1_sodium_minlow

Indicator for low (< 135) minimum sodium test result

0.0403

0.197

0

1

tm1_sodium_minhigh

Indicator for high (> 145) minimum sodium test result

0.000615

0.0248

0

1

tm1_sodium_minnormal

Indicator for normal minimum sodium test result

0.438

0.496

0

1

tm1_sodium_meanlow

Indicator for low (< 135) mean sodium test result

0.0196

0.139

0

1

tm1_sodium_meanhigh

Indicator for high (> 145) mean sodium test result

0.000861

0.0293

0

1

tm1_sodium_meannormal

Indicator for normal mean sodium test result

0.459

0.498

0

1

tm1_sodium_maxlow

Indicator for low (< 135) maximum sodium test result

0.0109

0.104

0

1

tm1_sodium_maxhigh

Indicator for high (> 145) maximum sodium test result

0.00515

0.0715

0

1

tm1_sodium_maxnormal

Indicator for normal maximum sodium test result

0.464

0.499

0

1

tm1_trig_minlow

Indicator for low (< 50) minimum triglycerides test result

0.0318

0.176

0

1

tm1_trig_minhigh

Indicator for high (> 150) minimum triglycerides test result

0.0901

0.286

0

1

tm1_trig_minnormal

Indicator for normal minimum triglycerides test result

0.262

0.440

0

1

tm1_trig_meanlow

Indicator for low (< 50) mean triglycerides test result

0.0289

0.167

0

1

tm1_trig_meanhigh

Indicator for high (> 150) mean triglycerides test result

0.0972

0.296

0

1

tm1_trig_meannormal

Indicator for normal mean triglycerides test result

0.256

0.436

0

1

tm1_trig_maxlow

Indicator for low (< 50) maximum triglycerides test result

0.0279

0.165

0

1

tm1_trig_maxhigh

Indicator for high (> 150) maximum triglycerides test result

0.107

0.309

0

1

tm1_trig_maxnormal

Indicator for normal maximum triglycerides test result

0.251

0.434

0

1

tm1_gagne_sum

Total number of active illnesses

1.443

2.049

0

18

TABLE 2

R Commands

See assignment PDF.