BA810 Machine Learning

Fernanda Lin, Kyle Blackburn, Mansi Tolla, Lyufan Pan, Honyang Liu

10/16/2019

Import data

```
d <- read_csv('adult.csv')
```

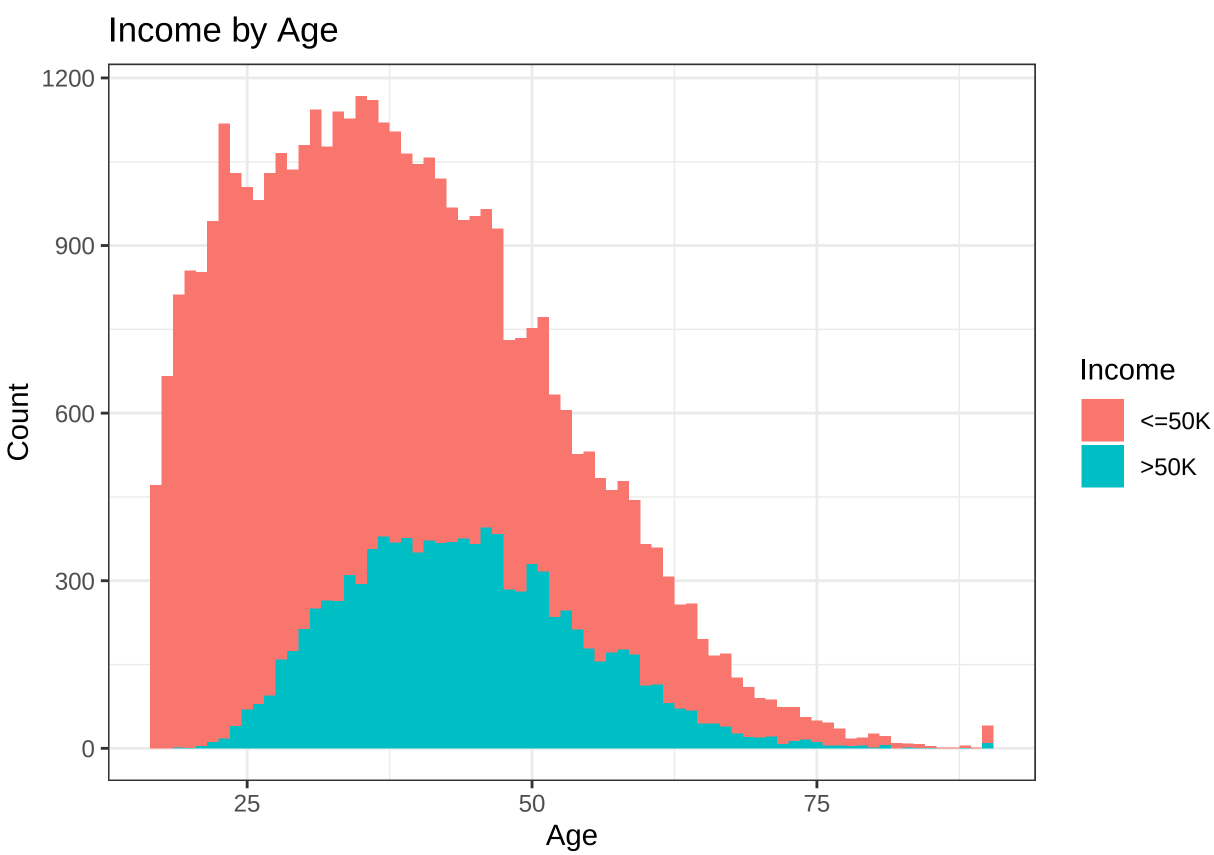
Clean data

```
#remove fnlwgt and educational-num --> not needed
d <- d %>%
  select(-fnlwgt, -`educational-num`, -relationship) %>%
  filter(`native-country` == "United-States") %>%
  select(-`native-country`)

#rename
names(d) <- c('age', 'workClass', 'education', 'maritalStatus', 'occupation', 'race', 'gender', 'capita

#convert ? to NA
d$workClass[which(d$workClass == '?')] <- NA
d$occupation[which(d$occupation == '?')] <- NA

#Ad ID number
ID <- seq(1:nrow(d))
d <- cbind(ID, d)

#Combine all self-employed
d$workClass[d$workClass == "Self-emp-inc" |
            d$workClass == "Self-emp-not-inc"] <- 'Self_employed'

#Married, Not-married, Never-married
d$maritalStatus[d$maritalStatus == "Married-AF-spouse" |
                d$maritalStatus == "Married-civ-spouse" |
                d$maritalStatus == "Married-spouse-absent"] <- "Married"

d$maritalStatus[d$maritalStatus == "Divorced" |
                d$maritalStatus == "Separated" |
                d$maritalStatus == "Widowed"] <- "Not-Married"

#Net capital change
d <- d %>%
  mutate(capChange = capitalGain-capitalLoss) %>%
  select(-capitalGain, -capitalLoss)

d$capChange[d$capChange == 99999] <- NA

#Education levels
d$education[d$education %in% c('Preschool', '1st-4th', '5th-6th', '7th-8th')] <- 'Primary'
```

```
d$education[d$education %in% c('9th', '10th', '11th', '12th', 'HS-grad') ] <- 'Secondary'
d$education[d$education %in% c('Some-college') ] <- 'Partial_uni'
d$education[d$education %in% c('Bachelors') ] <- 'Full_uni'
d$education[d$education %in% c('Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Doctorate', 'Masters') ] <- 'I

#complete cases only
d <- na.omit(d)
```

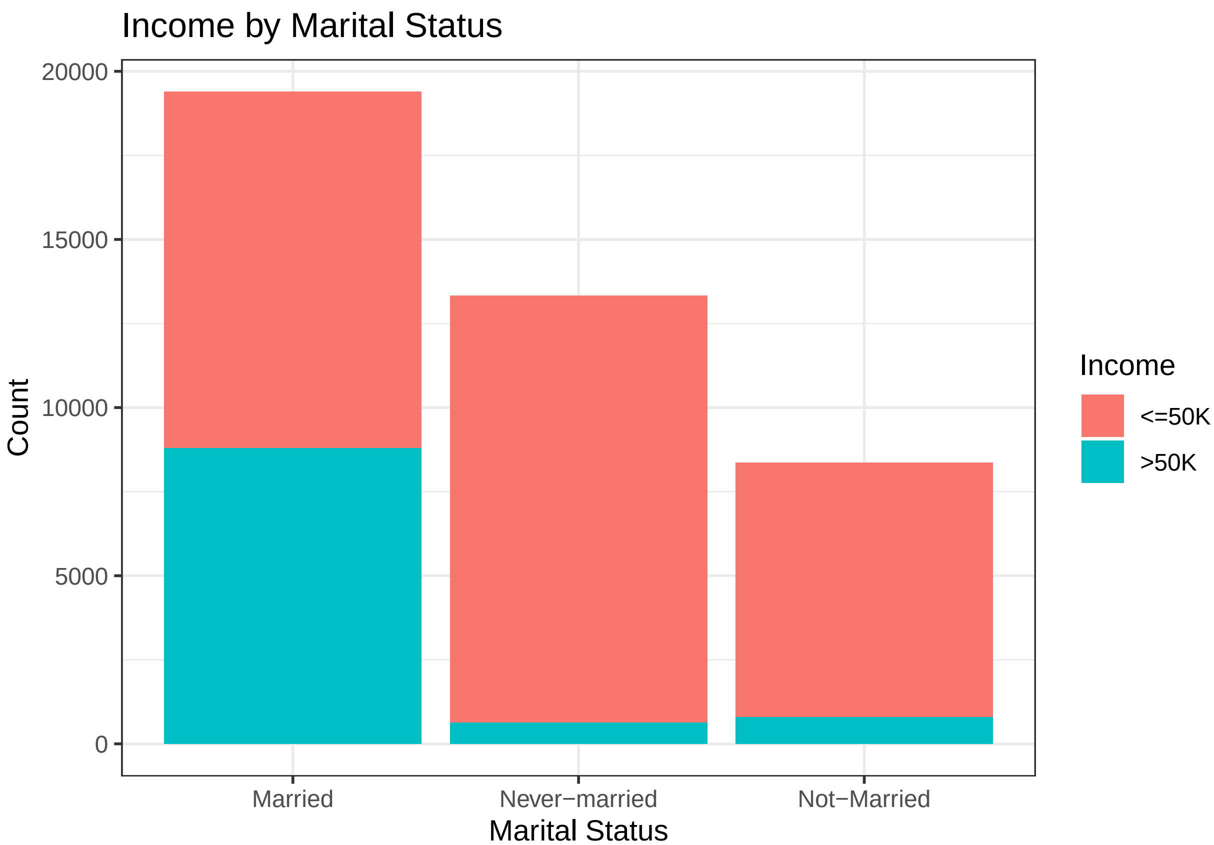
Exploratory Analysis

```
#age
ggplot(d, aes(x = age, fill = income))+
  geom_histogram(binwidth = 1)+
  labs(title = "Income by Age",
       x = "Age",
       y = "Count")+
  scale_fill_discrete(name = "Income")+
  ggsave("Plots/incomeByAge.png")
```

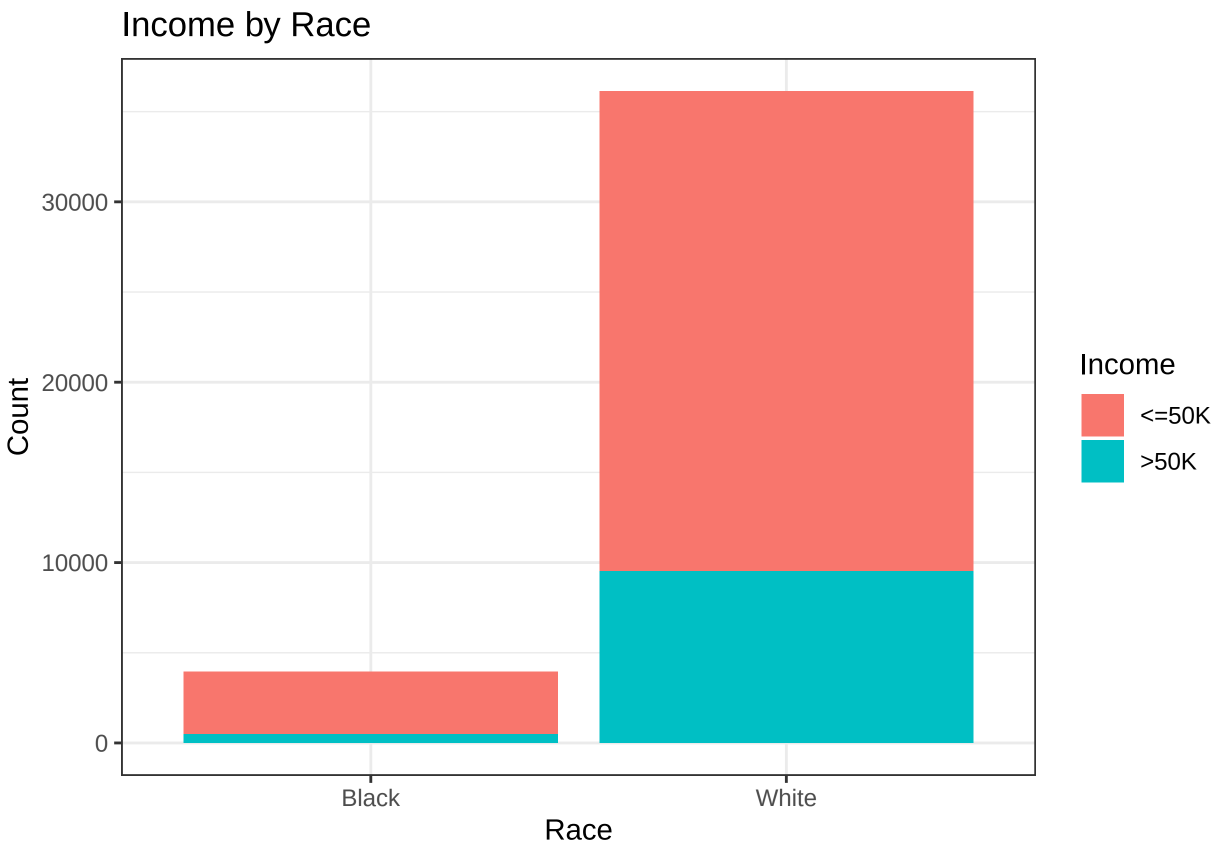



```
#marital status
ggplot(d, aes(x=maritalStatus, fill=income))+
  geom_bar()+
  labs(title = "Income by Marital Status",
       x = "Marital Status",
       y = "Count")+
```

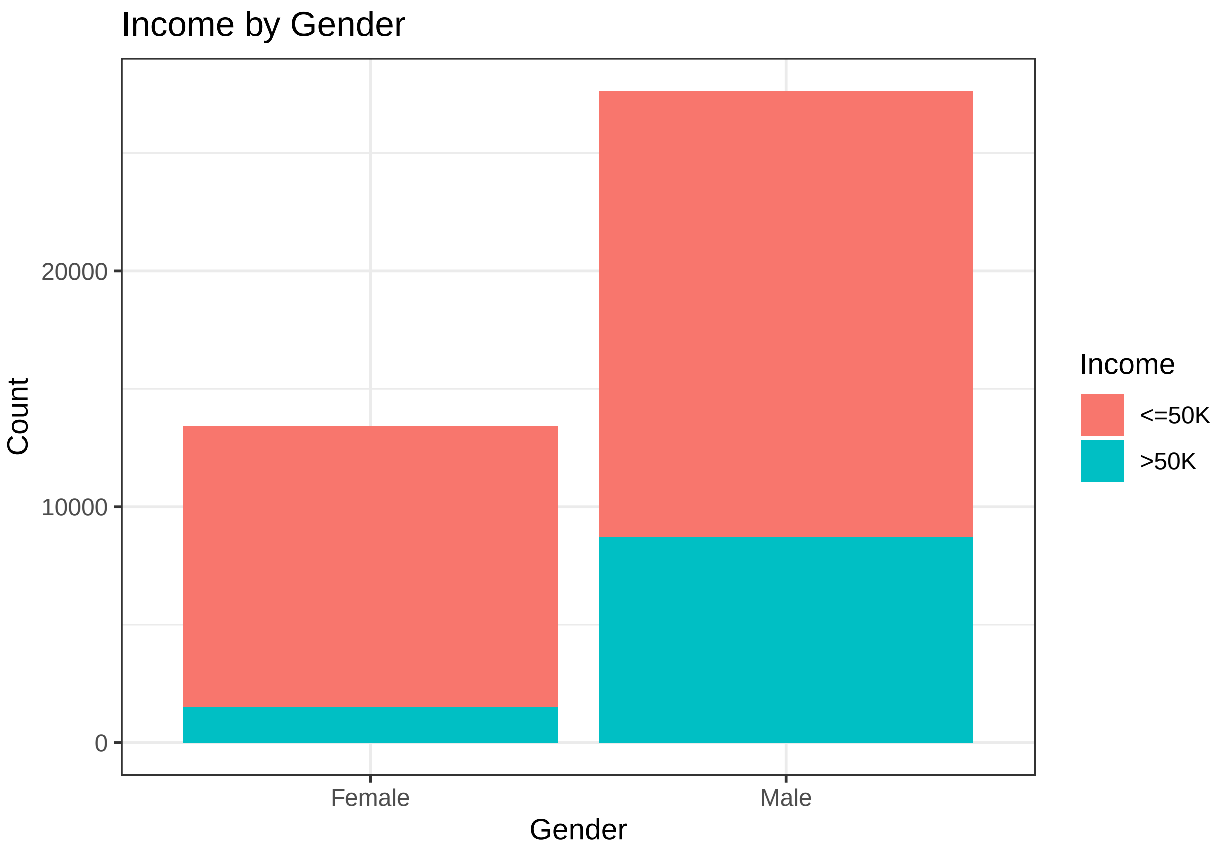
```r
  scale_fill_discrete(name = "Income")+
  ggsave("Plots/incomeByMaritalStatus.png")
```




```r
#race
ggplot(d[d$race == 'White' | d$race == 'Black', ], aes(x= race, fill=income))+
  geom_bar()+
  labs(title = "Income by Race",
       x = "Race",
       y = "Count")+
  scale_fill_discrete(name = "Income")+
  ggsave("Plots/incomeByRace.png")
```




```r
## gender
ggplot(d, aes(x = gender, fill = income)) +
  geom_bar()+
  labs(title = "Income by Gender",
       x = "Gender",
       y = "Count")+
  scale_fill_discrete(name = "Income")+
  ggsave("Plots/incomeByGender.png")
```



Make columns binary & finish cleaning

```
#Gender to binary 0 = Male, 1 = Female
d$gender[d$gender == 'Male'] <- 0
d$gender[d$gender == 'Female'] <- 1

#Income to binary; >50k = 1, <=50k = 0
d$income[d$income == '>50K'] <- 1
d$income[d$income == '<=50K'] <- 0

#Categorize occupation
d <- d %>%
  mutate(skilled = ifelse(occupation %in% c('Craft-repair', 'Machine-op-inspct',
                                            'Exec-managerial', 'Tech-support', 'Prof-specialty'), 1, 0))
  select(-occupation)
```

Create dummy variables

```
#Select only relevant variables
d_char <- d %>%
  select(-ID, -age,  -income, -capChange, -gender, -hoursPerWeek, -skilled)

#separate out into dummy variables
d_dummy <- dummy(d_char)
```

```r
#create final df with everything separated and
d_new <- d %>%
  select(ID, age, income, capChange, gender, hoursPerWeek, skilled) %>%
  cbind(d_dummy)

# change character to numeric
d_new$income <- as.numeric(d_new$income)
d_new$gender <- as.numeric(d_new$gender)
```

Split train and test 80/20

```r
set.seed(1234)

#split
d_new$train <- sample(c(0, 1), nrow(d), replace = TRUE, prob = c(0.2, 0.8))
d_test <- d_new %>% filter(train == 0)
d_train <- d_new %>% filter(train == 1)

#xnames
xnames <- colnames(d_new)
xnames <- xnames[! xnames %in% c("ID", "income", "train")]

#get rid of `train` column
d_train <-  d_train %>%
  select(-train)

d_test <-  d_test %>%
  select(-train)
```

Forward Stepwise

```r
#Intercept only
fit_fw <- lm(income ~ 1, data = d_train)

## calculate MSE train and MSE test
yhat_train <- predict(fit_fw, d_train)
mse_train <- mean((d_train$income - yhat_train)^2)

yhat_test <- predict(fit_fw, d_test)
mse_test <- mean((d_test$income - yhat_test)^2)

log_fw <-
  tibble(
    xname = "intercept",
    model = deparse(fit_fw$call),
    mse_train = mse_train,
    mse_test = mse_test
  )
```

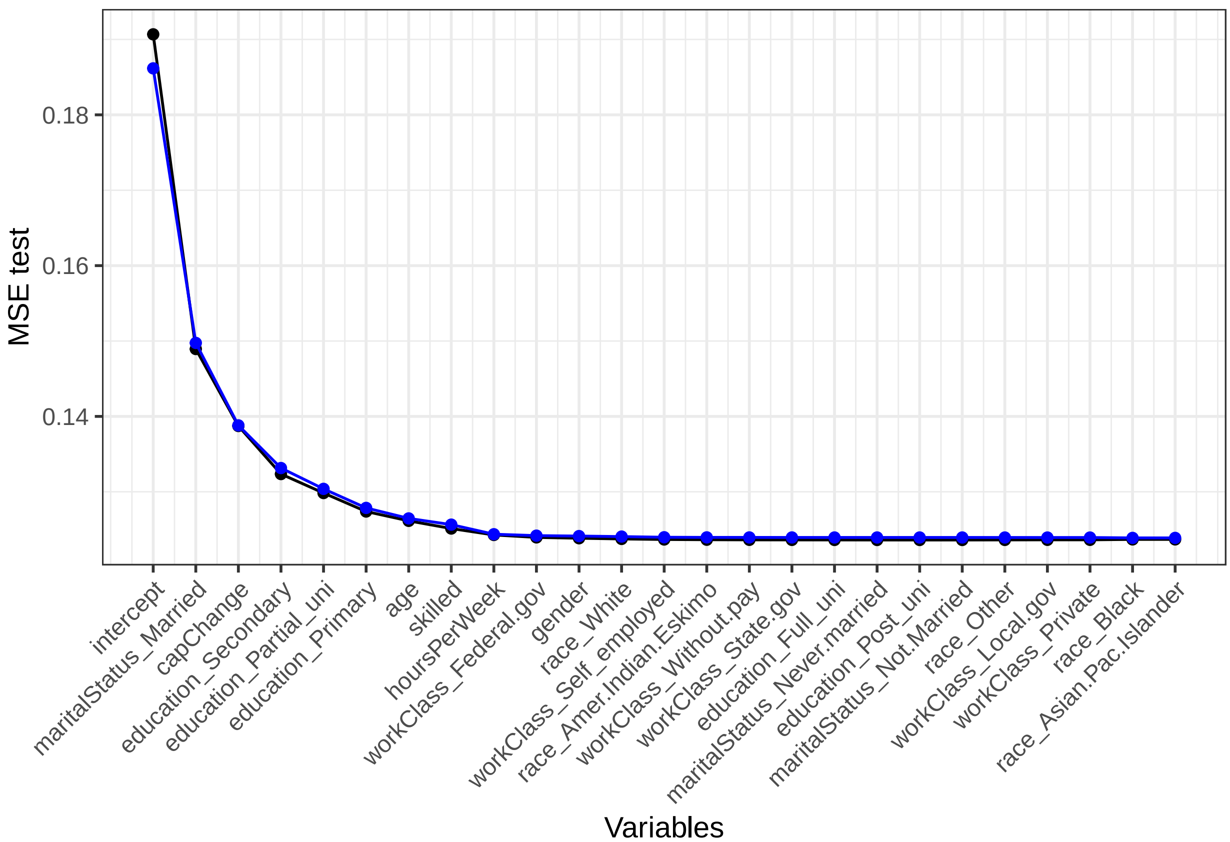
Add variables to the forward stepwise

```
xnames_fw <- xnames

while (length(xnames_fw) > 0) {
  ## keep track of which is the next best variable to add
  best_mse_train <- NA
  best_mse_test <- NA
  best_fit_fw <- NA
  best_xname <- NA

  ## select the next best predictor
  for (xname in xnames_fw) {
    ## fit a model that ads the predictor xname to the current best model
    ## to do this you will want to use the update() command which can ad a predictor
    ## to an existing model
    fit_fw_tmp <- update(fit_fw, as.formula(paste0(". ~ . +", xname)))

    ## compute MSE train
    yhat_train_tmp <- predict(fit_fw_tmp, d_train)
    mse_train_tmp <- mean((d_train$income - yhat_train_tmp)^2, na.rm = TRUE)

    ## compute MSE test
    yhat_test_tmp <- predict(fit_fw_tmp, d_test)
    mse_test_tmp <- mean((d_test$income - yhat_test_tmp)^2, na.rm = TRUE)

    ## if this is the first predictor to be examined
    ## or if this predictors yields a lower MSE than the current best
    ## then store this predictor as the current best predictor
    if(is.na(best_mse_test) | mse_test_tmp < best_mse_test) {
      best_xname <- xname
      best_fit_fw <- fit_fw_tmp
      best_mse_train <- mse_train_tmp
      best_mse_test <- mse_test_tmp
    }
  }
  ## update the log
  log_fw <- log_fw %>%
    add_row(
      xname = best_xname,
      model = paste0(deparse(best_fit_fw$call), collapse = ""),
      mse_train = best_mse_train,
      mse_test = best_mse_test
    )

  ## adopt the best model for the next iteraction
  fit_fw <- best_fit_fw

  ## remove the current best predictor from the list of predictors
  xnames_fw <- xnames_fw[xnames_fw!=best_xname]
}
```

Plot Forward Stepwise

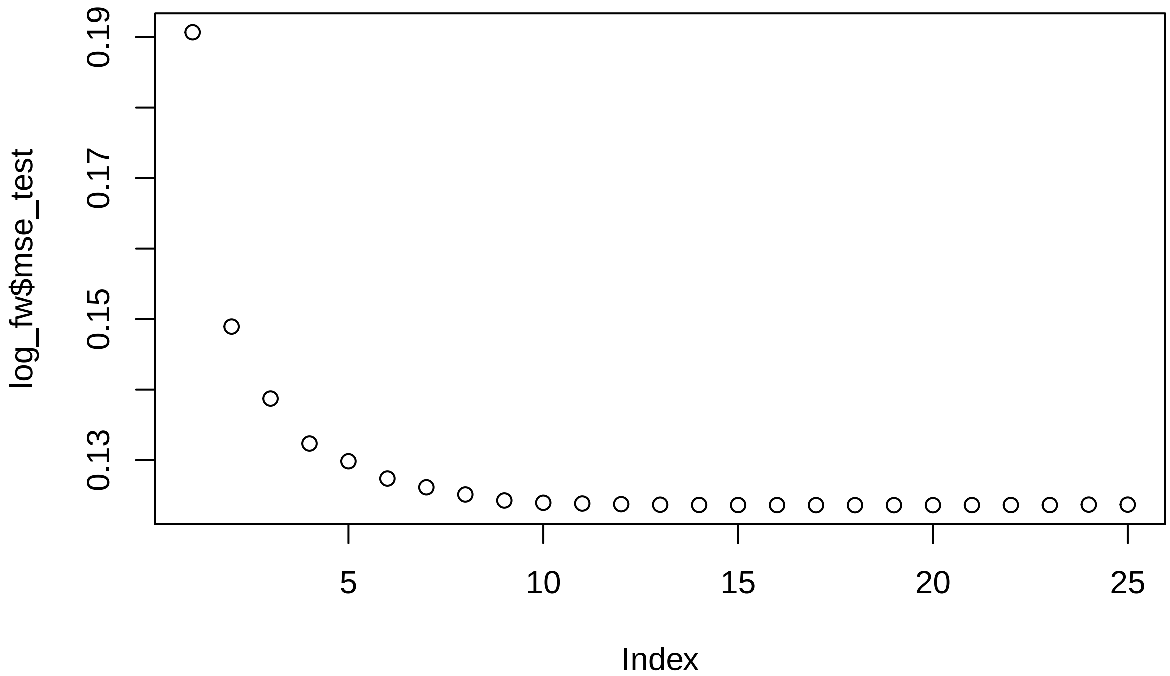
```
ggplot(log_fw, aes(seq_along(xname), mse_test)) +
  geom_point() +
  geom_line() +
  geom_point(aes(y = mse_train), color = "blue") +
  geom_line(aes(y = mse_train), color = "blue") +
  scale_x_continuous("Variables", labels = log_fw$xname, breaks = seq_along(log_fw$xname)) +
  scale_y_continuous("MSE test") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  ggsave("Plots/forwardSelection.png")
```

```
## Saving 6.5 x 4.5 in image
```



Determine Cutoff for Forward Selection

```
tmp <- c()
for (i in 1: nrow(log_fw)){
  tmp <- c(tmp, (log_fw$mse_test[i] - log_fw$mse_test[i+1]))
}

log_fw$change <- tmp
plot(log_fw$mse_test)
```



Backwards Stepwise

```
###REMOVE THE PREDICTOR THAT INCREASES THE MSE THE LEAST

#create own xnames
xnames_bw <- xnames

#create formula with all predictors
bw_formula <- "income ~ ."
for (k in 1:length(xnames_bw)) {
  bw_formula <- paste(bw_formula, "+", xnames_bw[k], collapse = "+")}
bw_f <- as.formula(bw_formula)

#start original fit
fit_bw <- lm(bw_f, data = d_train)

#predictions
yhat_train <- predict(fit_bw, d_train)
yhat_test <- predict(fit_bw, d_test)

#MSE round 1
mse_train <- mean((d_train$income - yhat_train)^2 )
mse_test <- mean((d_test$income - yhat_test)^2 )
xname <- "all"

#update log
log_bw <- tibble(xname = xname,
                model = paste0(deparse(bw_f), collapse = ""),
                mse_train = mse_train,
                mse_test = mse_test)
```

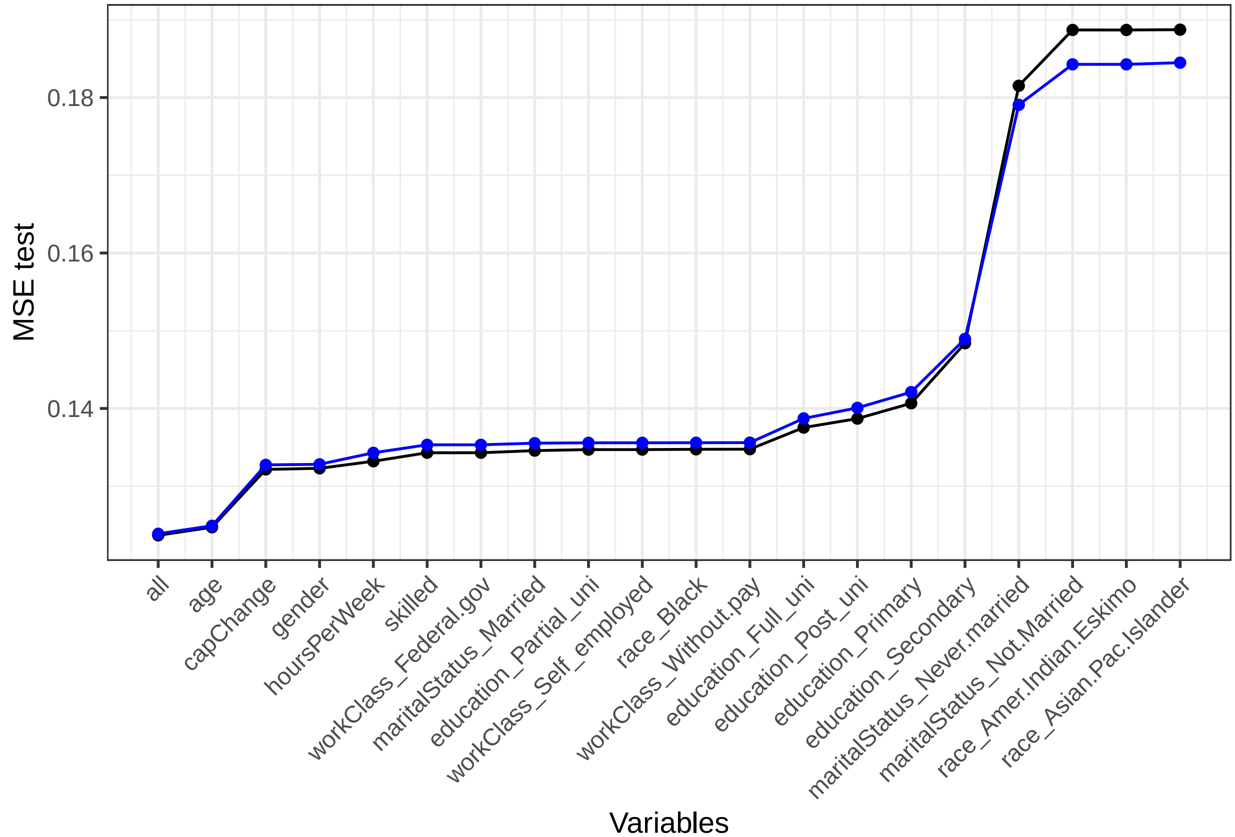
```r
while (length(xnames_bw)>2 ){
  smallest_mse_inc_train <- NA
  smallest_mse_inc_test <- NA
  best_fit_bw <- NA
  best_xname <- NA

  for (xname in xnames_bw){

    #run backwards selection
    xnames_tmp <- xnames_bw[xnames_bw != xname]
    fit_bw_tmp <- update(fit_bw, as.formula( paste0("income~", paste0(xnames_tmp, collapse = "+"))))

    #save yhats
    yhat_train_tmp <- predict(fit_bw_tmp, d_train)
    yhat_test_tmp <- predict(fit_bw_tmp, d_test)

    #save MSEs
    mse_train_tmp <- mean((d_train$income - yhat_train_tmp)^2 )
    mse_test_tmp <- mean((d_test$income - yhat_test_tmp)^2 )

    #keep only the model that has the mse increased the least
    if (is.na(smallest_mse_inc_train) | mse_train_tmp < smallest_mse_inc_train) {
      best_xname <- xname
      best_fit_bw <- fit_bw_tmp
      smallest_mse_inc_train <- mse_train_tmp
      smallest_mse_inc_test <- mse_test_tmp
    }

    # adopt the best model for the next iteration
    fit_bw <- best_fit_bw

    # remove the current best predictor from the list of predictors
    xnames_bw <- xnames_bw[xnames_bw != best_xname]
  }

  #log results
    log_bw <-log_bw %>% add_row(xname = best_xname,
                                model = paste0(deparse(best_fit_bw$call), collapse = ""),
                                mse_train = smallest_mse_inc_train,
                                mse_test = smallest_mse_inc_test)
}
```

Plot Backward Stepwise

```r
ggplot(log_bw, aes(seq_along(xname), mse_test)) +
  geom_point() +
  geom_line() +
  geom_point(aes(y = mse_train), color = "blue") +
  geom_line(aes(y = mse_train), color = "blue") +
  scale_x_continuous("Variables", labels = log_bw$xname, breaks = seq_along(log_bw$xname)) +
  scale_y_continuous("MSE test") +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  ggsave("Plots/backwardSelection.png")
```

```
## Saving 6.5 x 4.5 in image
```



Lasso/Ridge/Elastic Net

```
#Formula
xnames_lasso<-xnames
loopformula <- "income~age"
for (k in 2:24) {
  loopformula <- paste(loopformula, "+",xnames_lasso[k], sep = " ")}
f <- as.formula(loopformula)


##Splitting tables into test and train
x_train <- model.matrix(f, d_train)[ , -1]

#Intercept added by default
x_test <- model.matrix(f, d_test) [ , -1]


#Lasso
##Creating folds for lasso
fit_lasso <- cv.glmnet(x_train, d_train$income, alpha = 1, nfolds = 10)
```

```r
##Calculating Lasso MSE
yhat_train_lasso <- predict(fit_lasso, x_train, s = fit_lasso$lambda.min)
mse_train_lasso <- mean((d_train$income - yhat_train_lasso)^2)
yhat_test_lasso <- predict(fit_lasso, x_test, s = fit_lasso$lambda.min)
mse_test_lasso <- mean((d_test$income - yhat_test_lasso)^2)



#Ridge
##Creating folds for ridge
fit_ridge <- cv.glmnet(x_train, d_train$income, alpha = 0, nfolds = 10)

##Calculating Ridge MSE
yhat_train_ridge <- predict(fit_ridge, x_train, s = fit_ridge$lambda.min)
mse_train_ridge <- mean((d_train$income - yhat_train_ridge)^2)
yhat_test_ridge <- predict(fit_ridge, x_test, s = fit_ridge$lambda.min)
mse_test_ridge <- mean((d_test$income - yhat_test_ridge)^2)



#Elastic Net
##Creating folds for elastic nets
fit_elastic <- cv.glmnet(x_train, d_train$income, alpha = 0.5, nfolds = 10)

##Calculating Elastic Nets MSE
yhat_train_elastic <- predict(fit_elastic, x_train, s = fit_elastic$lambda.min)
mse_train_elastic <- mean((d_train$income - yhat_train_elastic)^2)
yhat_test_elastic <- predict(fit_elastic, x_test, s = fit_elastic$lambda.min)
mse_test_elastic <- mean((d_test$income - yhat_test_elastic)^2)
```

Random Forest

```r
#need to make fields into factors for RF
d_rf <- d
d_rf$gender[d$gender == 0] <- "Male"
d_rf$gender[d$gender == 1] <- "Female"
d_rf$workClass <- as.factor(d_rf$workClass)
d_rf$education <- as.factor(d_rf$education)
d_rf$maritalStatus <- as.factor(d_rf$maritalStatus)
d_rf$race <- as.factor(d_rf$race)
d_rf$gender <- as.factor(d_rf$gender)
d_rf$skilled <- as.factor(d_rf$skilled)
d_rf$income <- as.factor(d_rf$income)
```

Split train and test 80/20 RANDOM FOREST ONLY

```r
set.seed(1234)
d_rf$train <- sample(c(0, 1), nrow(d), replace = TRUE, prob = c(0.2, 0.8))
d_rf_test <- d_rf %>% filter(train == 0)
d_rf_train <- d_rf %>% filter(train == 1)
```

```r
## Cleaning
cat_name <- names(d_rf)
cat_name <- cat_name[!cat_name %in% c("ID", "income", "train")]

loopformula <- "income ~ 1"

for (name in cat_name) {
loopformula <- paste(loopformula, "+", name, sep = "")
}
f_rf <- as.formula(loopformula)
```

Random Forest

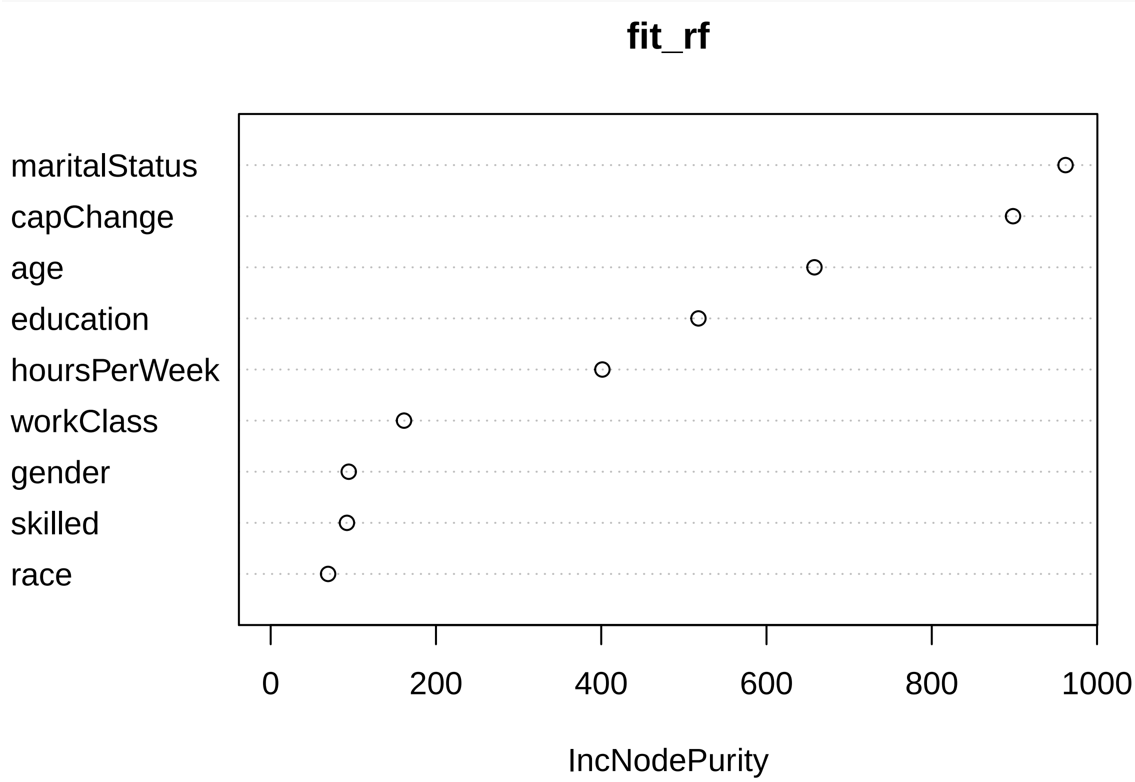
```r
#Train/Test
d_rf_train$income <- as.double(d_rf_train$income)
d_rf_test$income <- as.double(d_rf_test$income)

x1_train <- model.matrix(f_rf, d_rf_train)[, -1]
y_train <- d_rf_train$income
x1_test <- model.matrix(f_rf, d_rf_test)[, -1]
y_test <- d_rf_test$income

#RF fit
fit_rf <- randomForest(f_rf,
                       d_rf_train,
                       ntree=100,
                       do.trace=T)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##    1 |   0.1341    72.06 |
##    2 |   0.1298    69.75 |
##    3 |   0.1274    68.46 |
##    4 |   0.1249    67.11 |
##    5 |   0.1238    66.51 |
##    6 |   0.1207    64.86 |
##    7 |   0.1186    63.73 |
##    8 |   0.1172    62.96 |
##    9 |   0.1154    61.98 |
##   10 |   0.1139    61.18 |
##   11 |   0.1128    60.59 |
##   12 |   0.1121    60.22 |
##   13 |   0.1113    59.80 |
##   14 |   0.1108    59.50 |
##   15 |   0.1104    59.28 |
##   16 |     0.11    59.07 |
##   17 |   0.1097    58.93 |
##   18 |   0.1094    58.79 |
##   19 |   0.1091    58.60 |
```

```
##   20 |    0.1089    58.50 |
##   21 |    0.1085    58.27 |
##   22 |    0.1083    58.15 |
##   23 |     0.108    58.02 |
##   24 |     0.108    57.99 |
##   25 |    0.1077    57.87 |
##   26 |    0.1077    57.83 |
##   27 |    0.1075    57.74 |
##   28 |    0.1074    57.68 |
##   29 |    0.1072    57.59 |
##   30 |     0.107    57.48 |
##   31 |    0.1069    57.41 |
##   32 |    0.1067    57.33 |
##   33 |    0.1067    57.29 |
##   34 |    0.1064    57.17 |
##   35 |    0.1064    57.17 |
##   36 |    0.1064    57.13 |
##   37 |    0.1063    57.11 |
##   38 |    0.1062    57.04 |
##   39 |    0.1061    56.97 |
##   40 |     0.106    56.94 |
##   41 |     0.106    56.92 |
##   42 |    0.1059    56.89 |
##   43 |    0.1058    56.84 |
##   44 |    0.1058    56.81 |
##   45 |    0.1057    56.79 |
##   46 |    0.1057    56.77 |
##   47 |    0.1057    56.75 |
##   48 |    0.1056    56.74 |
##   49 |    0.1056    56.73 |
##   50 |    0.1056    56.70 |
##   51 |    0.1055    56.68 |
##   52 |    0.1055    56.66 |
##   53 |    0.1054    56.63 |
##   54 |    0.1054    56.62 |
##   55 |    0.1054    56.60 |
##   56 |    0.1054    56.62 |
##   57 |    0.1053    56.59 |
##   58 |    0.1053    56.59 |
##   59 |    0.1053    56.59 |
##   60 |    0.1053    56.56 |
##   61 |    0.1052    56.54 |
##   62 |    0.1052    56.53 |
##   63 |    0.1053    56.55 |
##   64 |    0.1053    56.54 |
##   65 |    0.1052    56.52 |
##   66 |    0.1052    56.52 |
##   67 |    0.1052    56.53 |
##   68 |    0.1052    56.52 |
##   69 |    0.1052    56.50 |
##   70 |    0.1052    56.49 |
##   71 |    0.1051    56.47 |
##   72 |    0.1052    56.48 |
##   73 |    0.1051    56.48 |
```

```
##    74 |    0.1051     56.46 |
##    75 |    0.1051     56.47 |
##    76 |    0.1051     56.46 |
##    77 |     0.105     56.43 |
##    78 |     0.105     56.41 |
##    79 |     0.105     56.42 |
##    80 |     0.105     56.42 |
##    81 |     0.105     56.41 |
##    82 |     0.105     56.39 |
##    83 |     0.105     56.38 |
##    84 |    0.1049     56.37 |
##    85 |    0.1049     56.37 |
##    86 |    0.1049     56.36 |
##    87 |    0.1049     56.34 |
##    88 |    0.1049     56.34 |
##    89 |    0.1049     56.35 |
##    90 |    0.1049     56.34 |
##    91 |    0.1049     56.34 |
##    92 |    0.1049     56.33 |
##    93 |    0.1048     56.32 |
##    94 |    0.1048     56.29 |
##    95 |    0.1048     56.28 |
##    96 |    0.1048     56.28 |
##    97 |    0.1048     56.27 |
##    98 |    0.1047     56.26 |
##    99 |    0.1047     56.25 |
##   100 |    0.1047     56.25 |
```

```
#Plot RF fit
varImpPlot(fit_rf)
```



```r
#Predict
yhat_rf_train <- predict(fit_rf, d_rf_train)
mse_rf <- mean((yhat_rf_train -d_rf_train$income) ^ 2)
yhat_rf_test <- predict(fit_rf, d_rf_test)
mse_rf_test <- mean((yhat_rf_test - d_rf_test$income) ^ 2)
```

Boosting

```r
#create own xnames
xnames_bt <- xnames

#create formula with all predictors
bt_formula <- "income ~ ."
for (k in 1:length(xnames_bt)) {
  bt_formula <- paste(bt_formula, "+", xnames_bt[k], collapse = "+")}
bt_f <- as.formula(bt_formula)

fit_bt <- gbm(bt_f, data = d_train,
              distribution = "gaussian",
              n.trees = 100,
              interaction.depth = 3,
              shrinkage = 0.01,
              cv.folds = 10)

relative.influence(fit_bt)
```

```
## n.trees not given. Using 100 trees.
```

```
##                        ID                      age
##                    0.0000                 669.9887
##                 capChange                   gender
##                12588.6860                   0.0000
##              hoursPerWeek                  skilled
##                  175.4788                   0.0000
##       workClass_Federal.gov      workClass_Local.gov
##                    0.0000                   0.0000
##         workClass_Private     workClass_Self_employed
##                    0.0000                   0.0000
##        workClass_State.gov     workClass_Without.pay
##                    0.0000                   0.0000
##        education_Full_uni     education_Partial_uni
##                  295.4085                   0.0000
##        education_Post_uni        education_Primary
##                  142.3257                 132.3511
##       education_Secondary     maritalStatus_Married
##                 3901.9499               25490.1172
## maritalStatus_Never.married  maritalStatus_Not.Married
##                    0.0000                   0.0000
##    race_Amer.Indian.Eskimo    race_Asian.Pac.Islander
##                    0.0000                   0.0000
##                race_Black               race_Other
##                    0.0000                   0.0000
##                race_White
```

```
##                          0.0000
```

```r
#train MSe
yhat_train_bt <- predict(fit_bt, d_train, n.trees = 100)
mse_train_bt <- mean((yhat_train_bt - d_train$income) ^ 2)

#test MSE
yhat_test_bt <- predict(fit_bt, d_test, n.trees = 100)
mse_test_bt <- mean((yhat_test_bt - d_test$income) ^ 2)
```

```r
## Log Test MSEs
log_testMSE <-
  tibble(model = "Forward Selection",
                    bestTestMSE = min(log_fw$mse_test[1:18])) %>%
  add_row(model = "Backward Selection",
          bestTestMSE = min(log_bw$mse_test)) %>%
  add_row(model = "Lasso",
          bestTestMSE = mse_test_lasso) %>%
  add_row(model = "Ridge",
          bestTestMSE = mse_test_ridge) %>%
  add_row(model = "Elastic net",
          bestTestMSE = mse_test_elastic) %>%
  add_row(model = "Random Forest",
          bestTestMSE = mse_rf_test) %>%
  add_row(model = "Boosting",
          bestTestMSE = mse_test_bt)

write_csv(log_testMSE, path = "/Users/kyleblackburn1/Census-810/log_testMSE.csv")
```

The Random Forest model had the lowest test MSE and thus is the best model for predicting income above or below $50K/year