HW 4

Zhaoyang Chen

2/10/2020

```
library(tidyverse)
library(glmnet)
library(pls)
library(leaps)
library(caret)
library(mltools)
library(DAAG)
library(splines)
library(ggeffects)
```

Load data

```
train = read.csv('gss_train.csv')
test = read.csv('gss_test.csv')

data_cv = createFolds(as.integer(rownames(train)), k = 10, list = T, returnTrain = F) #create 10-folds
```

Egalitarianism and income

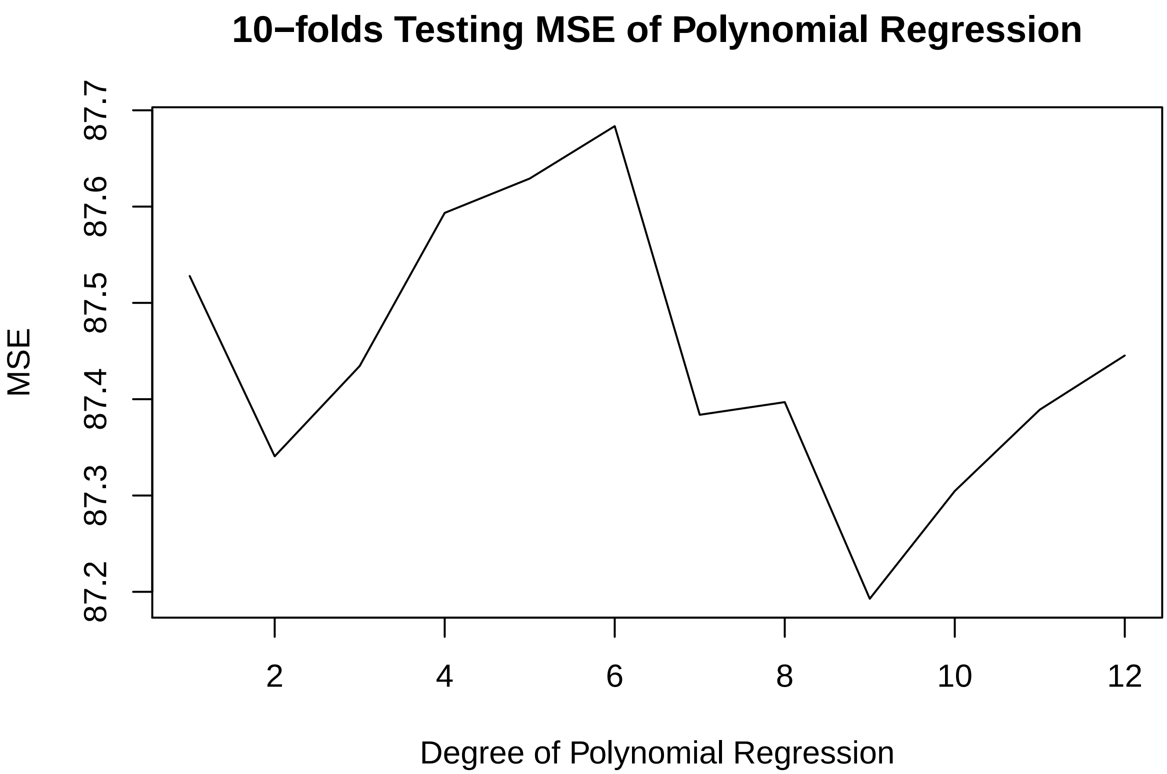
Question one

```
set.seed(1234)# set a random seed
train_poly = train[,c('egalit_scale','income06')] # select two target variables
test_poly_X = test[,'income06']
test_poly_y = test[,'egalit_scale']
models = list()
mses = c()
for(d in 1:12){
  mse = c()
  for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    model1 = lm(egalit_scale ~ poly(income06, d, raw = T), data = tra)
    pred1 = predict(model1, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred1, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
  }
  mses = c(mses, mean(mse))
}
```

```
## Warning in predict.lm(model1, newdata = data.frame(income06 =
```

```
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model1, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

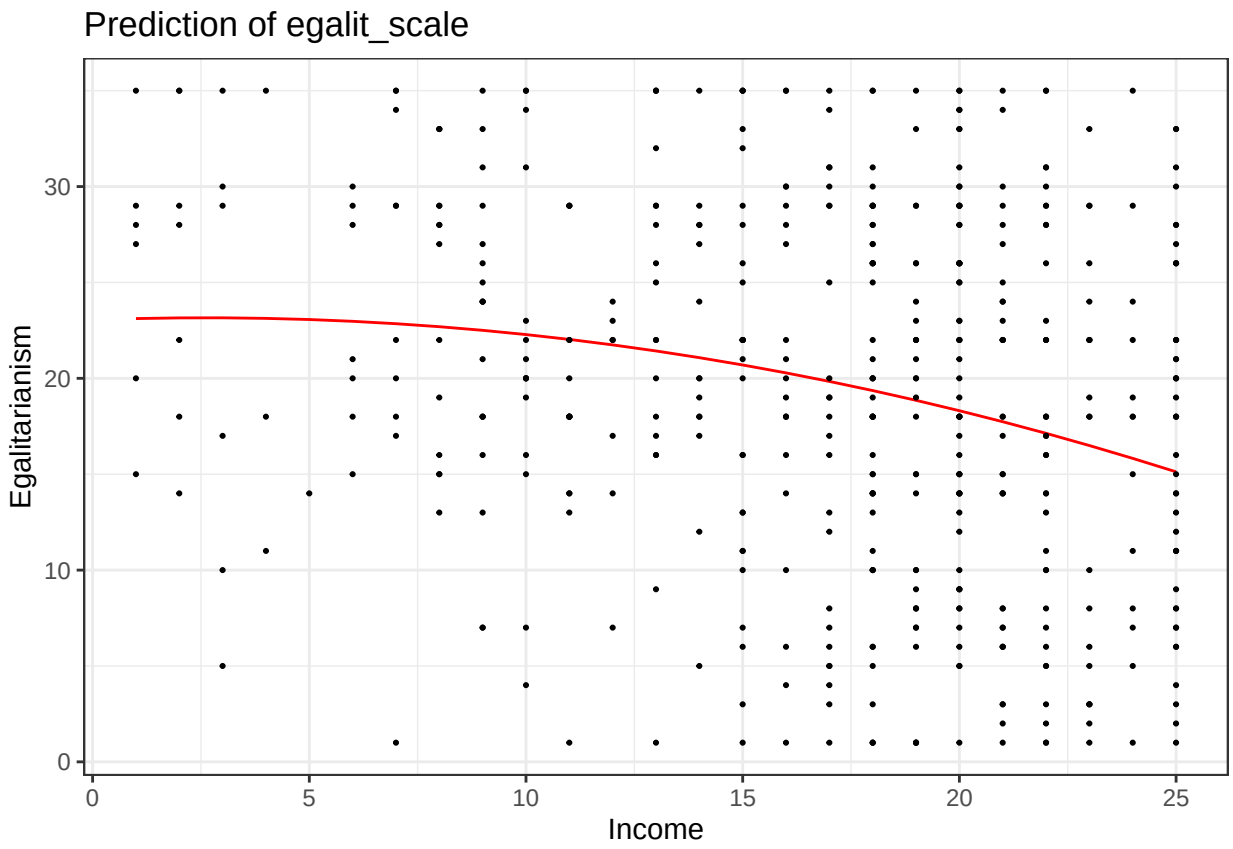
```
plot(mses, type = 'l', xlab = 'Degree of Polynomial Regression', ylab = 'MSE', main = '10-folds Testing
```



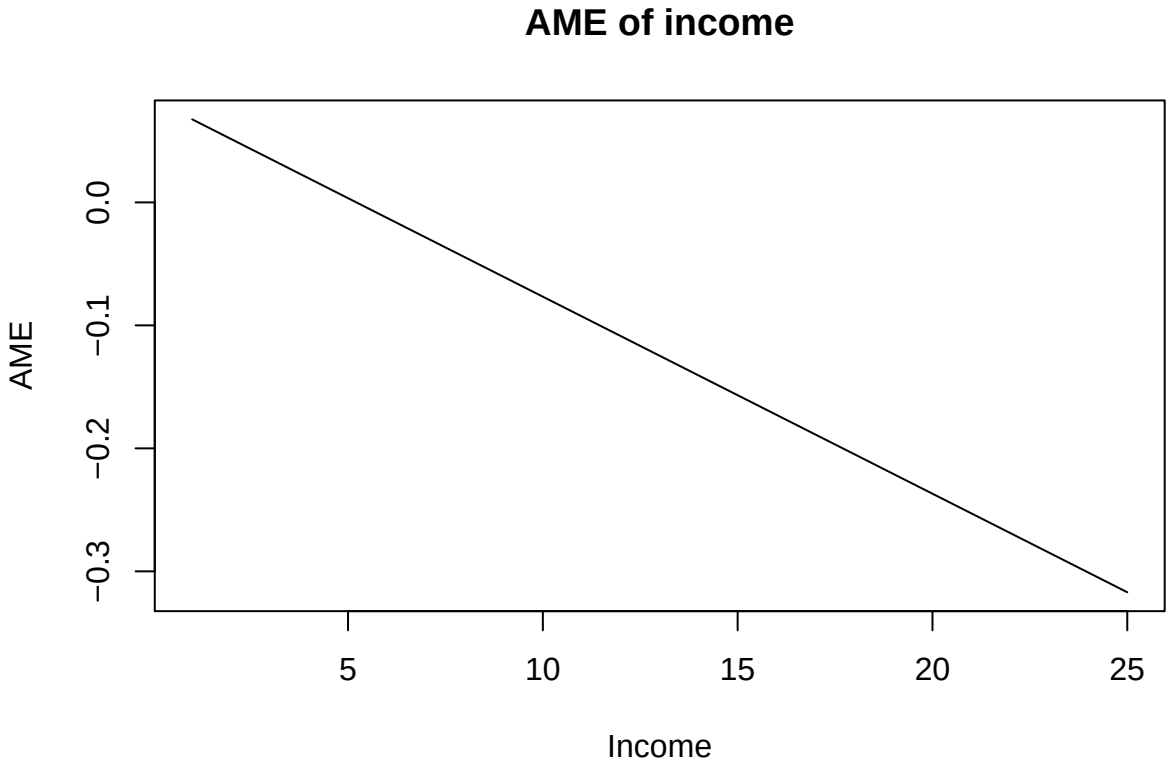

```
mses
```

```
##  [1] 87.52793 87.34068 87.43445 87.59346 87.62913 87.68352 87.38382
##  [8] 87.39695 87.19281 87.30466 87.38911 87.44536
```

```r
best_model_1 = lm(egalit_scale ~ poly(income06, 2, raw = T), data = train_poly)
pred_1 =  predict(best_model_1, newdata = data.frame(income06=test_poly_X))
dt = data.frame(pred = as.data.frame(pred_1)[,1], income = test$income06)
ggplot(dt, aes(x = income, y = pred_1)) +
  theme_bw() +
  geom_line(color = 2) +
  geom_point(data =test, aes(x = income06, y = egalit_scale), size = 0.4) +
  labs(title = 'Prediction of egalit_scale', x = 'Income', y = 'Egalitarianism')
```




```r
beta = best_model_1$coefficients
y = beta[2] + beta[3]*sort(train_poly$income06)
plot(sort(train_poly$income06), y, type = 'l', main = 'AME of income', xlab = 'Income', ylab = 'AME')
```



According to the result of 10-fold cross validation, we choose d as 2. However, the best model does not fit the test data well because there does not exist a clear trend on the scatter plot between income and Egalitarianism. Also, from the AME of income across its potential values, we can see that the mean average effect is decreasing, which means that one unit change in income will lead to more reduction in egalitarianism as income becomes larger.

Question two

```
model_list = list()
mses = c()

mse = c()
n = 3
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06), data =
    pred = predict(model, newdata = data.frame(income06=val$income06))
```

```
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
}
```

```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

```
mses = c(mses, mean(mse))

mse = c()
n = 4
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
y[4] = 0; y[5] = 1
sf_5 = stepfun(x = cut, y = y)
for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
```

```r
    model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06) + sf_5(i
    pred = predict(model, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
}
```

```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

```r
mses = c(mses, mean(mse))

mse = c()
n = 5
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
y[4] = 0; y[5] = 1
sf_5 = stepfun(x = cut, y = y)
y[5] = 0; y[6] = 1
sf_6 = stepfun(x = cut, y = y)
```

```r
for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06) + sf_5(i
    pred = predict(model, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
}
```

```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

```r
mses = c(mses, mean(mse))

mse = c()
n = 6
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
```

```r
y[4] = 0; y[5] = 1
sf_5 = stepfun(x = cut, y = y)
y[5] = 0; y[6] = 1
sf_6 = stepfun(x = cut, y = y)
y[6] = 0; y[7] = 1
sf_7 = stepfun(x = cut, y = y)
for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06) + sf_5(i
    pred = predict(model, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
}
```

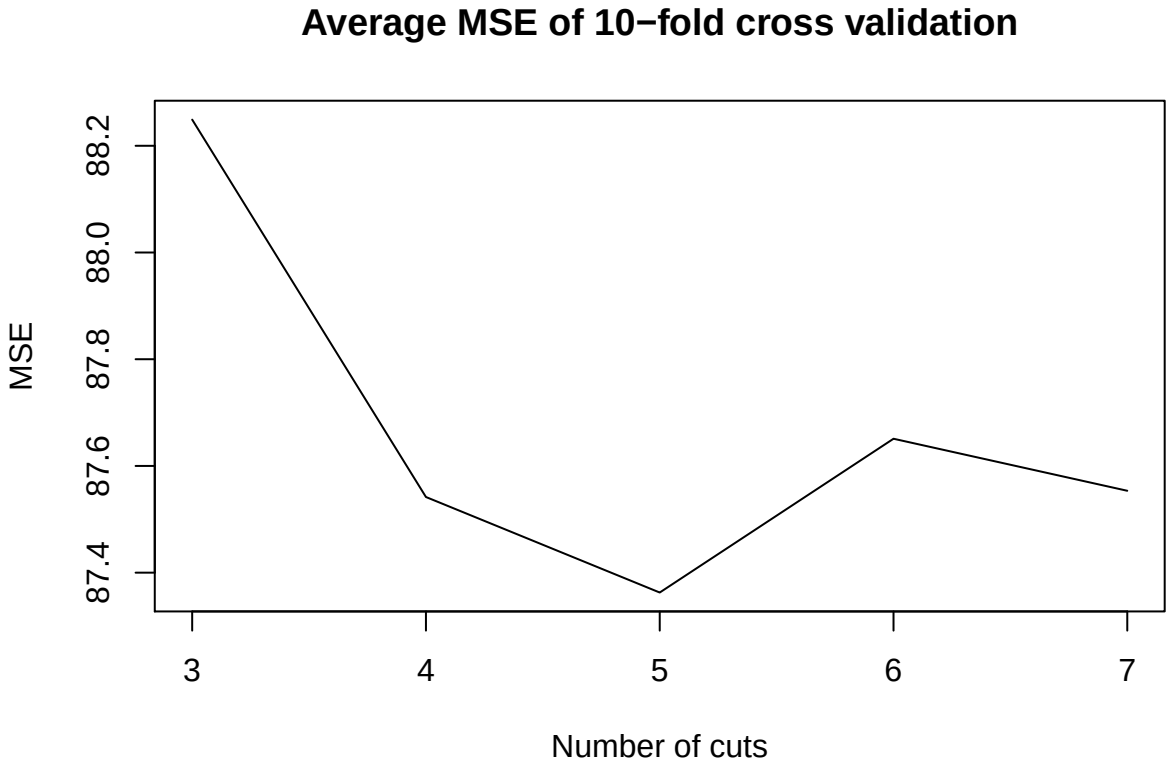
```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

```r
mses = c(mses, mean(mse))

mse = c()
n = 7
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
```

```r
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
y[4] = 0; y[5] = 1
sf_5 = stepfun(x = cut, y = y)
y[5] = 0; y[6] = 1
sf_6 = stepfun(x = cut, y = y)
y[6] = 0; y[7] = 1
sf_7 = stepfun(x = cut, y = y)
y[7] = 0; y[8] = 1
sf_8 = stepfun(x = cut, y = y)
for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06) + sf_5(i
    pred = predict(model, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
}
```

```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = data.frame(income06 =
## val$income06)): prediction from a rank-deficient fit may be misleading
```

```
mses = c(mses, mean(mse))

plot(x = 3:7, y = mses, type = 'l', main = 'Average MSE of 10-fold cross validation', xlab = 'Number of
```



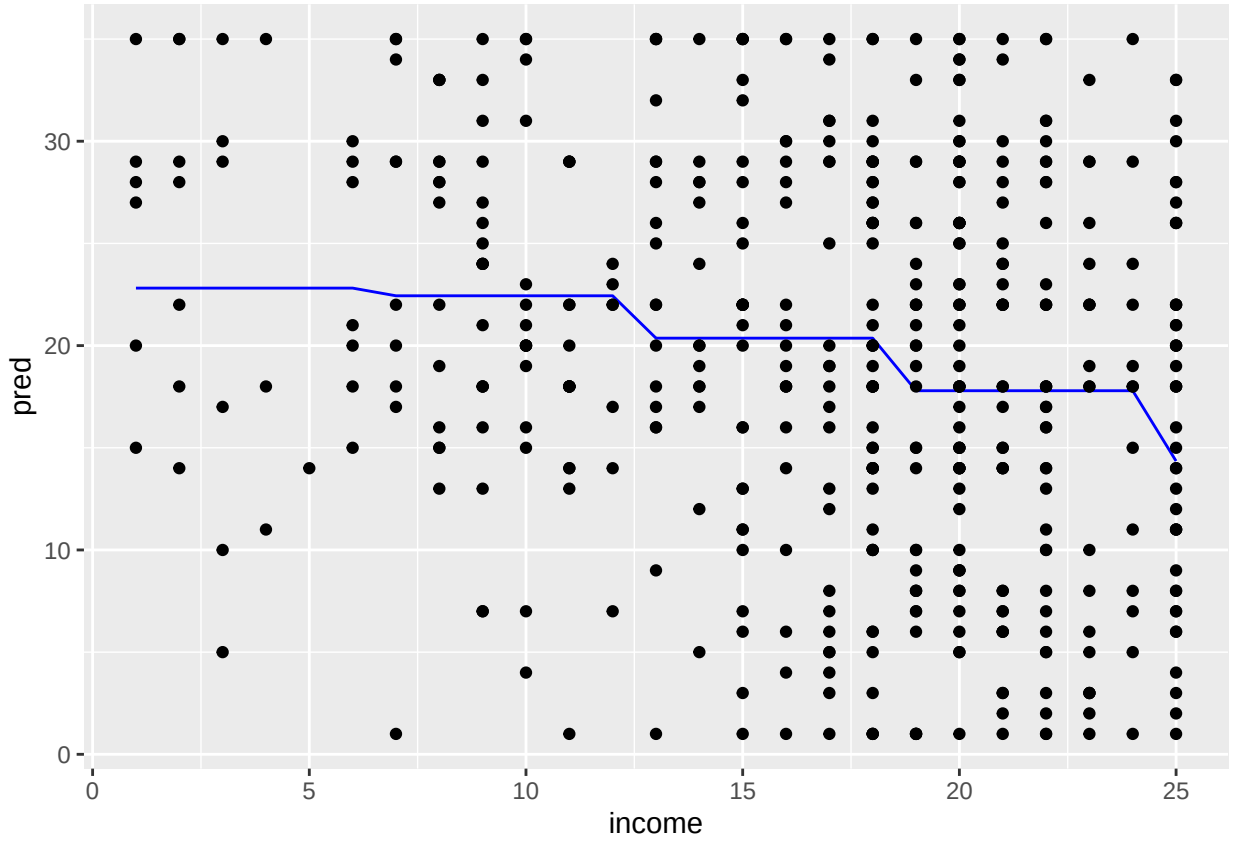

```
n = 5
cut = seq(min(train_poly$income06), max(train_poly$income06), length.out = n)
y = rep(0, n+1)
y[1] = 1
sf_1 = stepfun(x = cut, y = y)
y[1] = 0; y[2] = 1
sf_2 = stepfun(x = cut, y = y)
y[2] = 0; y[3] = 1
sf_3 = stepfun(x = cut, y = y)
y[3] = 0; y[4] = 1
sf_4 = stepfun(x = cut, y = y)
y[4] = 0; y[5] = 1
sf_5 = stepfun(x = cut, y = y)
y[5] = 0; y[6] = 1
sf_6 = stepfun(x = cut, y = y)

model = lm(egalit_scale ~ sf_1(income06) + sf_2(income06) + sf_3(income06) + sf_4(income06) +
             sf_5(income06) + sf_6(income06), data = train)
pred = predict(model, newdata = data.frame(income06=test$income06))
```

```
## Warning in predict.lm(model, newdata = data.frame(income06 =
## test$income06)): prediction from a rank-deficient fit may be misleading
```

```
temp = data.frame(pre = pred, act = test$egalit_scale)
mean((temp$pre - temp$act)^2)
```

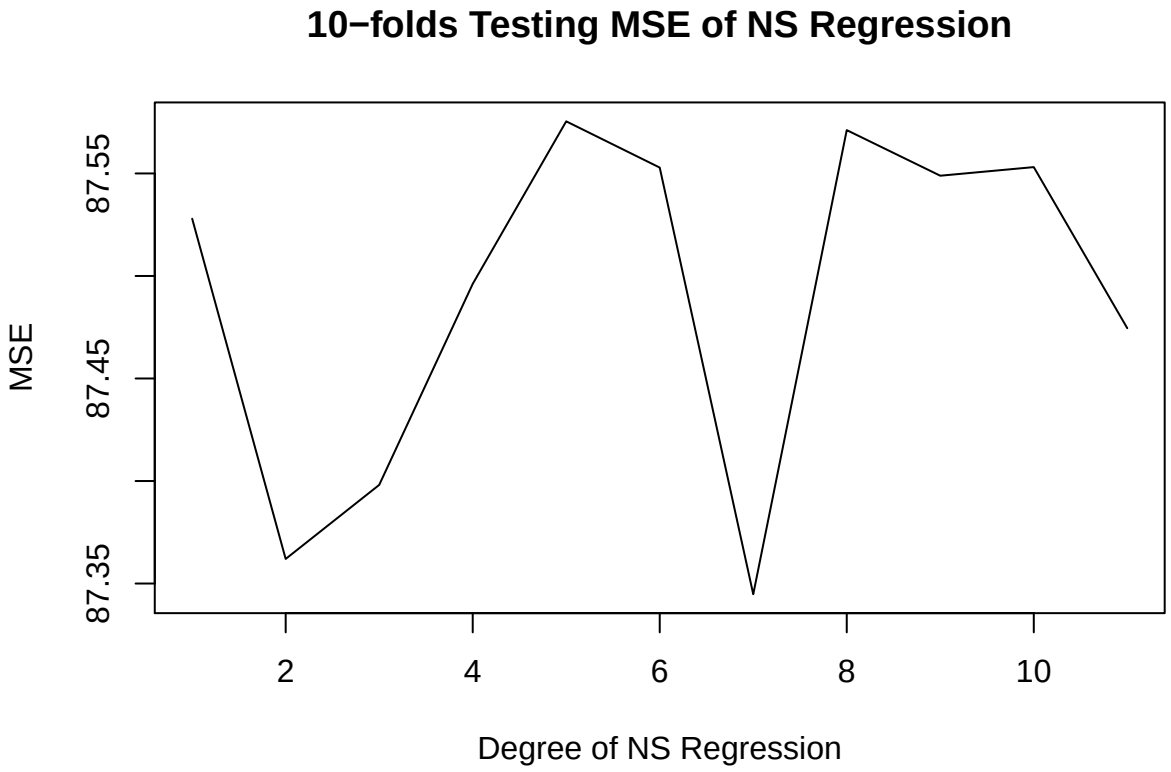
```
## [1] 87.58953
```

```
df = data.frame('income' = test$income06, 'pred' = pred)
ggplot(data = df, aes(x = income, y = pred)) +
  geom_line(color = 'blue') +
  geom_point(data = test, aes(x = income06, y = egalit_scale))
```



Question three

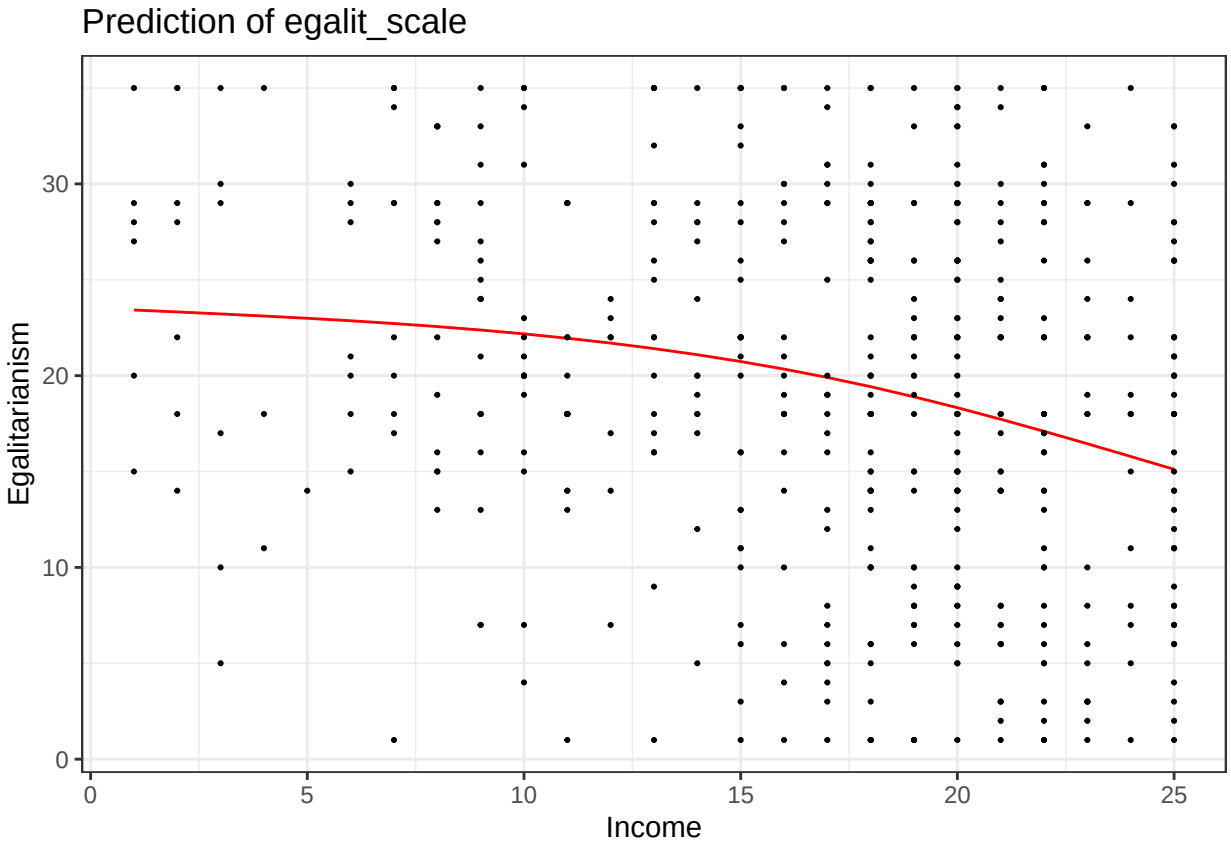
```
models = list()
mses = c()
for(k in 1:11){
  mse = c()
  for(i in 1:10){
    validation_id = data_cv[[i]]
    val = train_poly[validation_id,]
    tra = train_poly[-validation_id,]
    best_model_2 = glm(egalit_scale ~ ns(income06, df = k), data = tra)
    pred = predict(best_model_2, newdata = data.frame(income06=val$income06))
    temp = data.frame(pre = pred, act = val$egalit_scale)
    mse = c(mse, mean((temp$pre - temp$act)^2))
  }
  mses = c(mses, mean(mse))
}
plot(mses, type = 'l', xlab = 'Degree of NS Regression', ylab = 'MSE', main = '10-folds Testing MSE of
```




```
mses
```

```
##  [1] 87.52793 87.36199 87.39810 87.49614 87.57542 87.55286 87.34477
##  [8] 87.57115 87.54890 87.55311 87.47451
```

```
best_model_2 = glm(egalit_scale ~ ns(income06, df = 2), data = train_poly)
pred =  predict(best_model_2, newdata = data.frame(income06=test_poly_X))
dt = data.frame(pred = as.data.frame(pred)[,1], income = test$income06)
ggplot(dt, aes(x = income, y = pred)) +
  theme_bw() +
  geom_line(color = 2) +
  geom_point(data =test, aes(x = income06, y = egalit_scale), size = 0.4) +
  labs(title = 'Prediction of egalit_scale', x = 'Income', y = 'Egalitarianism')
```



Egalitarianism and everything

Question four

Linear regression

```
train_cv = createFolds(1:1481, 10)
train_num = train %>% select_if(., is.numeric)
test_num = test %>% select_if(., is.numeric)
mse = c()
for (i in 1:10) {
  val_id = train_cv[[i]]
  val = train_num[val_id,]
  tra = train_num[-val_id,]
  model = lm(egalit_scale~., data = tra)
  pred = predict(model, newdata = val[,-5])
  mse = c(mse, mse(pred, val[,5]))
}
id = which(mse == min(mse))
best_model_1 = lm(egalit_scale~., data = train_num[-train_cv[[id]],])
pred = predict(best_model_1, newdata = test_num[,-5])
mse(pred, test_num[,5])
```

```
## [1] 85.04611
```

Elastic net regression

```
alpha = seq(0, 1, length.out = 15)
cv_mse = c()
for(a in alpha) {
  model = cv.glmnet(x = as.matrix(train_num[,-5]), y = train_num[,5], alpha = a)
  cv_mse = c(cv_mse, min(model$cvm))
}
id = which(cv_mse == min(cv_mse))
best_model_2 = cv.glmnet(x = as.matrix(train_num[,-5]), y = train_num[,5], alpha = alpha[id])
best_lambda = best_model_2$lambda.min
best_alpha = alpha[id]
pred = predict(best_model_2, newx = as.matrix(test_num[,-5]))
mse(pred, test[,15])
```

```
## [1] 84.22531
```

PCR

```
model <- pcr(egalit_scale ~ .,
                data = select_if(train, is.numeric),
                center = TRUE,
                scale = TRUE,
                validation = "CV",
                segments = 10)

mse = as.vector(MSEP(model, estimate = "CV", intercept = FALSE)$val)
id = which(mse == min(mse))
best_model_3 = pcr(egalit_scale ~ ., data = train_num, comps = id)
pred = predict(best_model_3, newdata = as.matrix(test_num[,-5]), comps = id)
mse(pred, test[,15])
```

```
## [1] 89.23381
```

PLS

```
model <- plsr(egalit_scale ~ .,
                data = select_if(train, is.numeric),
                center = TRUE,
                scale = TRUE,
                validation = "CV",
                segments = 10)

mse = as.vector(MSEP(model, estimate = "CV", intercept = FALSE)$val)
id = which(mse == min(mse))
best_model_4 = plsr(egalit_scale ~ ., data = train_num, comps = id)
pred = predict(best_model_4, newdata = as.matrix(test_num[,-5]), comps = id)
mse(pred, test[,15])
```

```
## [1] 90.26552
```

Question five