# Intro to Machine learning

## Problem Set 2

*Jingming Wei(Mark)*

*2/3/2020*

```
# set working directory
setwd("/Users/Mark/Desktop/Intro to Machine Learning/HWK2")

# import the dataset and attach for easy call
NES <- read.csv("/Users/Mark/Documents/GitHub/problem-set-2/nes2008.csv")
attach(NES)
names(NES)
```

```
## [1] "biden"  "female" "age"    "educ"   "dem"    "rep"
```

**Q1 Full Sample Fit**

```
#Linear fit full sample
full.sample.fit <- lm(biden~female+age+educ+dem+rep, data = NES)
summary(full.sample.fit)
```

```
##
## Call:
## lm(formula = biden ~ female + age + educ + dem + rep, data = NES)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -75.546 -11.295   1.018  12.776  53.977
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.81126    3.12444  18.823  < 2e-16 ***
## female        4.10323    0.94823   4.327 1.59e-05 ***
## age           0.04826    0.02825   1.708   0.0877 .
## educ         -0.34533    0.19478  -1.773   0.0764 .
## dem          15.42426    1.06803  14.442  < 2e-16 ***
```

```
## rep         -15.84951    1.31136 -12.086  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.91 on 1801 degrees of freedom
## Multiple R-squared:  0.2815, Adjusted R-squared:  0.2795
## F-statistic: 141.1 on 5 and 1801 DF,  p-value: < 2.2e-16
```

```
#Calculate the MSE
biden.true <- biden
biden.pred <- predict(full.sample.fit)
residual.squared <- (biden.true - biden.pred)^2
full.sample.MSE <- mean(residual.squared)
full.sample.MSE
```

```
## [1] 395.2702
```

As we can see the MSE is farily large, suggesting that the data doesn't fit the data very well. All the estimates are statistically siginicant at least at 0.1. As we see from the above results, females are on average have more feeling of warmth towards Joe Biden. As expected, democratic people on average have more feeling of warmth towards Joe Biden than independents but republican people on average have less towards Joe Biden then independents. Additionally, older people tend to have more feeling of warmth towards Joe Biden while there is a negative effects of higher education on the support of Joe Biden.

### Q2 Hold-out validation approach

a. We first 50-50 split the sample set into a training set and a testing set.

```
> #Set random seed to guarantee reproducibility
> set.seed(1)
> train.index <- sample(1:nrow(NES), size=nrow(NES)*0.5, replace=FALSE)
> train <- NES[train.index, ]
> test <- NES[-train.index,]
```

b. Fit the linear regression using only the training observations.

```
> #Lienar fit training set
> train.fit <- lm(biden~female+age+educ+dem+rep, data=train)
> summary(train.fit)

Call:
lm(formula = biden ~ female + age + educ + dem + rep, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-75.155 -11.442   1.849  11.971  51.869

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  60.71194    4.17436  14.544  < 2e-16 ***
female        4.09297    1.29976   3.149  0.00169 **
```

```
age           0.04792    0.03866   1.240  0.21547
educ         -0.40908    0.26467  -1.546  0.12255
dem          15.99795    1.45584  10.989  < 2e-16 ***
rep         -14.72866    1.78150  -8.268  4.9e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.21 on 897 degrees of freedom
Multiple R-squared:  0.2905,    Adjusted R-squared:  0.2866
F-statistic: 73.46 on 5 and 897 DF,  p-value: < 2.2e-16
```

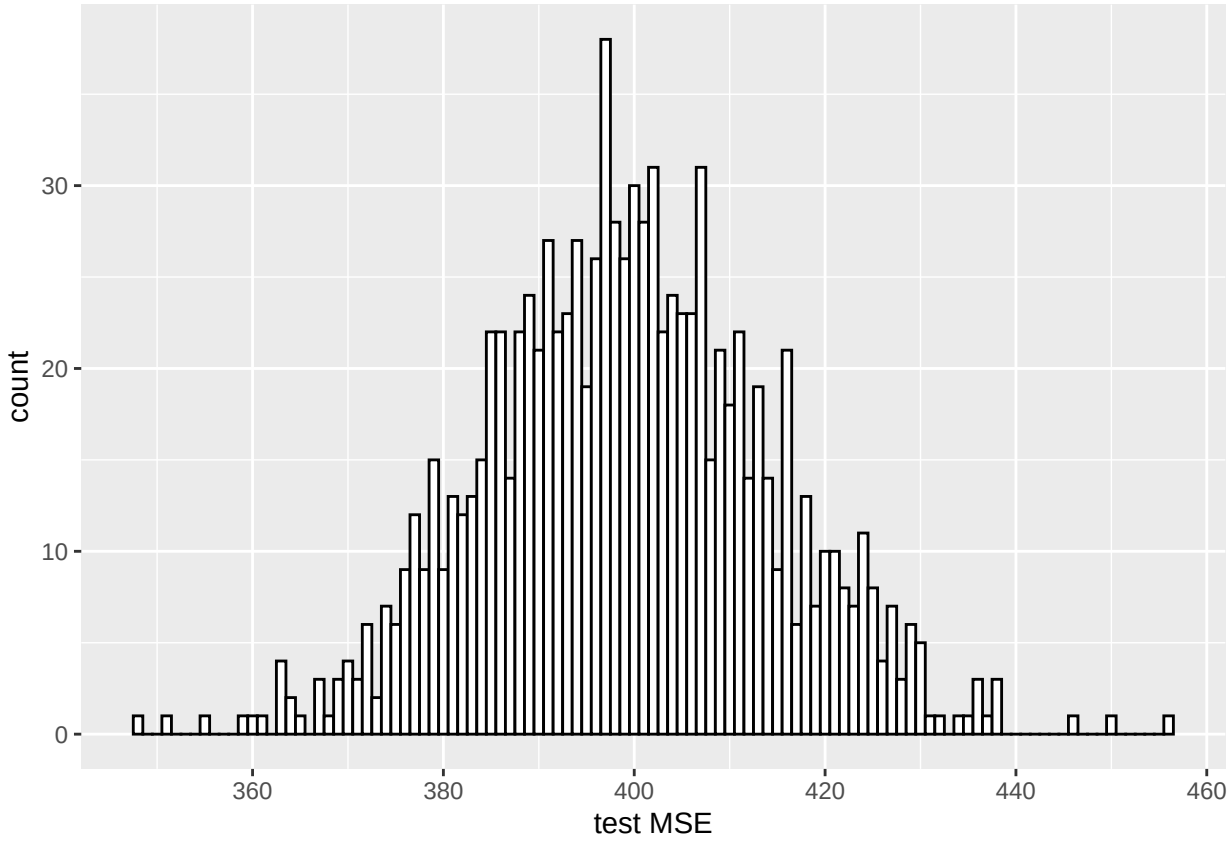
c. Then we calculate the MSE using the testing set.

```
> #Caluculate the MSE using only the testing set
> test.biden.true <- test$biden
> test.biden.pred <- predict(train.fit, test)
> residual.squared <- (test.biden.true - test.biden.pred)^2
> test.MSE <- mean(residual.squared)
> test.MSE
[1] 428.8471
```

d. The MSE using only the test set is 428.8471, which is higher than that using the full sample. Given that the fit using full sample is intrinsically high, we suspect that the model underfits (though strictly, we should compare test MSE with its correpsonding train MSE directly). But we can't rule out the possibility that this result happens at a low chance, since we are using only one possible split. Therefore, we should look at more possible splits to check the variability of the test MSE.

**Q3 Repeat 1000 timess**

```
> #We first calculate 1000 MSE of the testing set using 1000 different splits
> MSE = list()
> for (i in c(1:1000)){
+   set.seed(i)
+   train.index <- sample(1:nrow(NES), size=nrow(NES)*0.5, replace=FALSE)
+   train <- NES[train.index, ]
+   test <- NES[-train.index,]
+   train.fit <- lm(biden~female+age+educ+dem+rep, data=train)
+   test.biden.true <- test$biden
+   test.biden.pred <- predict(train.fit, test)
+   residual.squared <- (test.biden.true - test.biden.pred)^2
+   test.MSE <- mean(residual.squared)
+   MSE[[i]] <- test.MSE
+ }
>
> #We then plot the MSE for 1000 iterations
> library(ggplot2, warn.conflicts = FALSE)
> MSE <- data.frame(mse = unlist(MSE))
> ggplot(MSE, aes(x=mse)) +
+    geom_histogram(binwidth=1, color="black", fill="white") +
+    labs(x="test MSE")
```

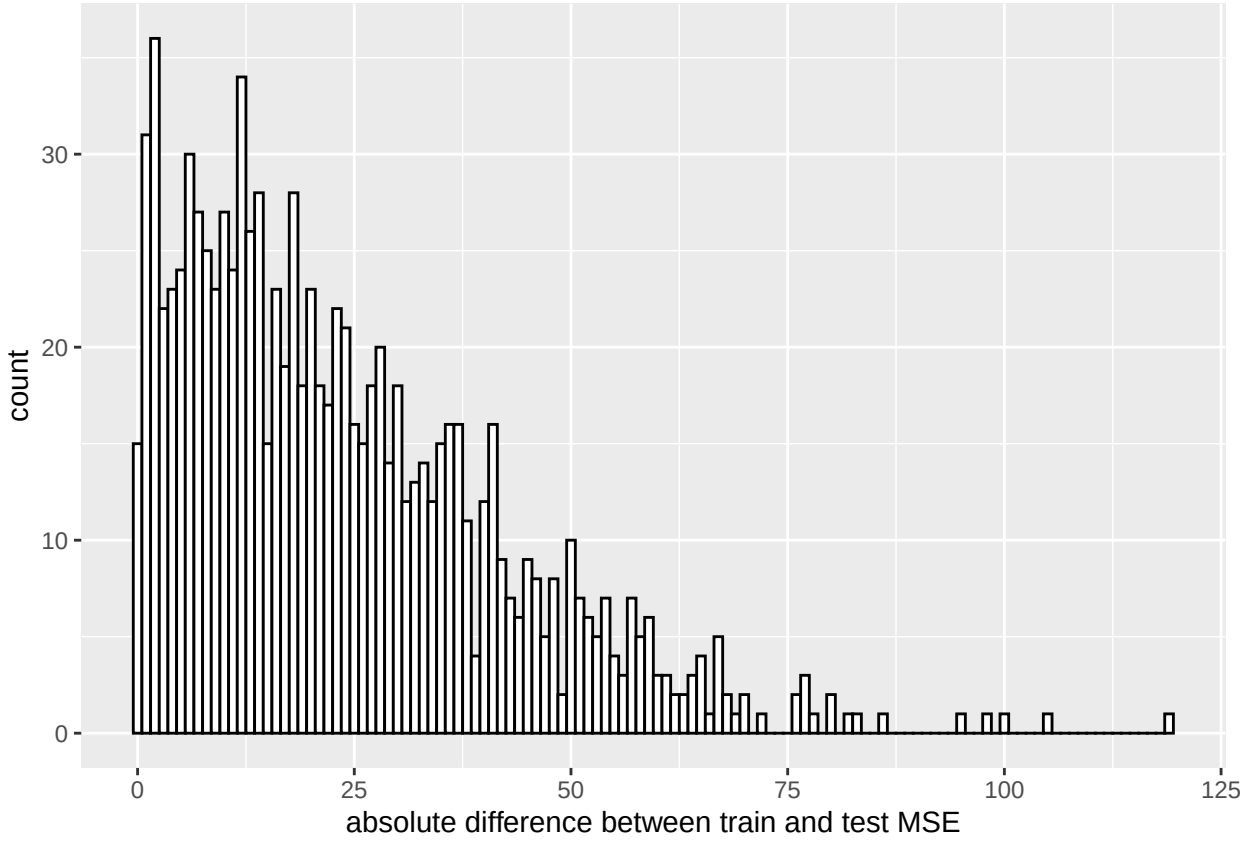



As we can see from the above graph, the distribution of MSE over 1000 different splits is approximately normal with mean centered around 400. Also, most of the values of test MSE span range between 360 and 440, which are fairly close to 400, suggesting that our test MSE is precise at a high value (stable around a large value of test MSE). Therefore, there might be a possiblity of overfitting or underfitting depending upon the train MSE. If the train MSE is also high, then we can conclude that the model underfits. If the train MSE is low, then we can conclude that the model overfits.

Further, I plot the difference between train and test MSE for each iteration to see if they closely move together (the difference is low). If this is the case, then we can possibly conclude that the model underfits (both MSE are large). And from the graph below, we can see that most diffences between train MSE and test MSE are below 50, suggesting they closely comove.Therefore, we can possibly (the result might be changed depending on seeds we set for random split) say that the model underfits.

```
#Here, I campare the train MSE and test MSE for each iteration
diff = list()
for (i in c(1:1000)){
  set.seed(i)
  train.index <- sample(1:nrow(NES), size=nrow(NES)*0.5, replace=FALSE)
  train <- NES[train.index, ]
  test <- NES[-train.index,]
  train.fit <- lm(biden~female+age+educ+dem+rep, data=train)

  test.biden.true <- test$biden
  test.biden.pred <- predict(train.fit, test)
  test.mse <- mean((test.biden.true - test.biden.pred)^2)

  train.true <- train$biden
```

```
  train.pred <- predict(train.fit, train)
  train.mse <- mean((train.true - train.pred)^2)


  diff.MSE <- abs(test.mse-train.mse)
  diff[[i]] <- diff.MSE
}

#Plot the difference between the train and test MSE for 1000 iterations
library(ggplot2, warn.conflicts = FALSE)
diff <- data.frame(diff = unlist(diff))
ggplot(diff, aes(x=diff)) +
   geom_histogram(binwidth=1, color="black", fill="white") +
   labs(x="absolute difference between train and test MSE")
```




**Q4 Comparison between full sample and bootstrap (B=1000)**

We write a function estimating coefficient for later use in bootstrapping.

```
biden.fit=function(index){
  result <- lm(biden~female+age+educ+dem+rep, data = NES[index,])
  coef <- t(data.frame(coef(result)))
  rownames(coef) <- NULL
  return(coef)
}
```

Then we bootstrapped 1000 times to get the estimates of each coefficient.

```
boot.estimates <- data.frame()
for (b in c(1:10000)){
  set.seed(b)
  index <- sample(1:1000, size=1000, replace = TRUE)
  boot.estimates <- rbind(boot.estimates, biden.fit(index))
}
```

Then we calculate the mean and standard error of the estimates over a 1000 boostrapped samples. The output is shown in a table.

```
boot.summary <- data.frame(Predictor=numeric(), Estimate=numeric(), Std.Error=numeric())
for (col in c(1:6)){
  current.row <- data.frame(Predictor=names(boot.estimates)[col], Estimate=mean(boot.estimates[,col]), S
  boot.summary <- rbind(boot.summary, current.row)
}

boot.summary
```

```
##     Predictor     Estimate Std.Error
## 1 (Intercept)  57.67517928 3.8312648
## 2      female   4.30437506 1.2628100
## 3         age   0.05921522 0.0371868
## 4        educ  -0.26633967 0.2662260
## 5         dem  15.07530311 1.3787385
## 6         rep -15.94971641 1.9198718
```

We can see from the above table that the bootstrapped estimates of the coefficients are almost the same as those estimated in question 1 and the signs of those coefficients match, implying that the directions of effects of each predictor on feeling therometer are consistent for both sets of estimates. Also, the standard errors are small in both sets of estimates and are pretty close to each other, suggesting that both sets of estimates are fairly precise.

Bootstrapping is a resampling method estimating sampling dsitributions. We can use such method to estimate standard errors of any statistics calculated from a sample (the statistics can be viewed as some function of a sample X). Specifically, suppose we want to estimate a statistic $\alpha = \hat{f}(X)$. Each time, we sample with replacement until the sample size reaches the size of the original sample. Based on the sample, we calculate the statistics using the formula $\hat{f}(X)$. In total, we perform such sampling B times and obtain B values of $\hat{f}(X)$, and then we can get the standard errors of these B values.

Ideally, we can draw many independent samples to estimate a certain statistics. But such methods might not always be pratically viable subject to resource constraints. Therefore, bootstrap is in some sense an economic way of generating new samples (though not indpendent) from the sample we already obtained. Additionally, since distributional assumptions don't affect bootstrapped estimates, they are often times more robust.