ML PS2

borui sun

1/28/2020

The Questions

1. (10 points) Estimate the MSE of the model using the traditional approach. That is, fit the linear regression model using the *entire* dataset and calculate the mean squared error for the *entire* dataset. Present and discuss your results at a simple, high level.

```r
ols <- lm(biden~ ., data = nes2008)

ols %>% summary() %>% tidy() %>% kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	58.8112590	3.1244366	18.822996	0.0000000
female	4.1032301	0.9482286	4.327258	0.0000159
age	0.0482589	0.0282474	1.708438	0.0877274
educ	-0.3453348	0.1947796	-1.772952	0.0764057
dem	15.4242556	1.0680327	14.441745	0.0000000
rep	-15.8495061	1.3113624	-12.086290	0.0000000

```r
cat(paste0("TRAIN MSE: ", (mse_entire <- summary(ols) %>% {.$residual^2} %>% mean())))
```

```
## TRAIN MSE: 395.270169278649
```

All of our predictors are statistically significant at 0.1 confidence level. The magnitude of coefficients on age and education are relatively small comparing to the other and have relatively larger p-values. The simple linear regression model has a mean squared error of 395.2701 (the loser to zero, the better our model perfoms), indicating that our estimates on average are about 20 points away of the true population value. Since we only have five predictors and a simple linear regression model, only 27.95% variance in the dependent variables are explained by our model. Therefore, it is normal to expect a large MSE.

2. (30 points) Calculate the test MSE of the model using the simple holdout validation approach.
   - (5 points) Split the sample set into a training set (50%) and a holdout set (50%). **Be sure to set your seed prior to this part of your code to guarantee reproducibility of results.**
   - (5 points) *Fit* the linear regression model using *only* the *training* observations.
   - (10 points) Calculate the *MSE* using *only* the *test* set observations.
   - (10 points) How does this value compare to the training MSE from question 1? Present numeric comparison and discuss a bit.

```r
set.seed(123)

split <- initial_split(nes2008, prop = .5)
train <- training(split)
test <- testing(split)
```

```r
ols_train <- lm(biden~., data = train)
mse_test <- mean((test$biden - predict(ols_train, test))^2)
cat(paste0("TEST MSE: ", mse_test))
```

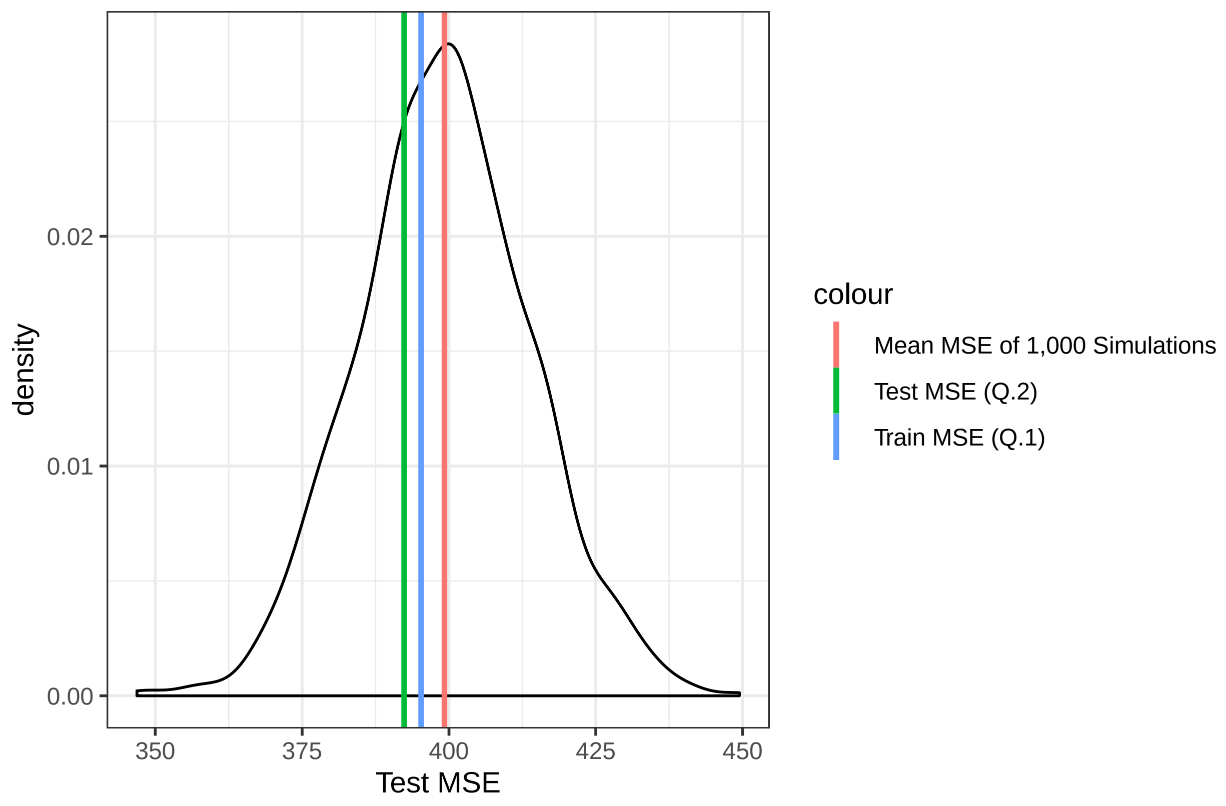
```
## TEST MSE: 392.38103730762
```

The test MSE is 392.3810. It is 2.8891 smaller than the MSE obtained in Q.1 (when the entire sample is used). Generally, we would expect the test-MSE to be larger because our original training sample is splited into training and test data set and hence has less number of observations. As the training data set reduced to half, less information is provided to train our model. However, this phenomenon can be explained by this specific seed of random split of train and test data set. Getting a smaller MSE is possible if we are lucky.

3. (30 points) Repeat the simple validation set approach from the previous question 1000 times, using 1000 different splits of the observations into a training set and a test/validation set. Visualize your results as a sampling distribution (hint: think histogram or density plots). Comment on the results obtained.

```r
set.seed(234)
repetition <- 1000

results <- data.frame(matrix(nrow = 0, ncol = 0))

for (i in c(1:repetition)){

  split <- initial_split(nes2008, prop = .5)
  train <- training(split)
  test <- testing(split)

  ols <- lm(biden~., data = train)
  mse <- mean((test[["biden"]] - predict(ols, test))^2)

  id <- i
  simulation_result <- data.frame(id, mse)
  results <- bind_rows(results, simulation_result)

  # print(paste0("-------- Done with simulation ", i, " of ", N))
}

mean_mse <- mean(results$mse)

results %>%
  ggplot(aes(x = mse)) +
  geom_density() +
  geom_vline(aes(xintercept = mean_mse, color = "Mean MSE of 1,000 Simulations"), size = 1) +
  geom_vline(aes(xintercept = mse_test, color = "Test MSE (Q.2)"), size = 1) +
  geom_vline(aes(xintercept = mse_entire, color = "Train MSE (Q.1)"), size = 1) +
  labs(x = "Test MSE", title = "Fig.1 MSE Distribution of 1,000 Simulations")
```

Fig.1 MSE Distribution of 1,000 Simulations



```r
summary(results$mse) %>% tidy() %>% kable()
```

minimum	q1	median	mean	q3	maximum
346.8791	389.8651	399.3784	399.2317	408.6465	449.4437

The MSEs obtained from 1,000 simulations follows a normal distribution with a mean of 399.0932. From Fig.1, we can see that the average MSE of using a 50% holdout validation appraoch is larger than the MSE using the entire data set in Q1. This results support our explanation in Q.2. As sample size decreases, less information is provided to train our model and hence leads to higher MSE. It is possible to obtain a smaller MSE using only half of the data set but it greatly depends the random split. In addition, we also notice that the difference of MSE in Q.1 and the average MSE in 1,000 simulations is not very large. The two MSEs are very close to each other (395 and 399).

4. (30 points) Compare the estimated parameters and standard errors from the original model in question 1 (the model estimated using *all of the available data*) to parameters and standard errors estimated using the bootstrap ($B = 1000$). Comparison should include, at a minimum, both numeric output as well as discussion on differences, similarities, etc. Talk also about the conceptual use and impact of bootstrapping.

```r
set.seed(123)
lm_coefs <- function(splits, ...) {
  mod <- lm(..., data = analysis(splits))
  tidy(mod)
}

 bootstrap_coef <- nes2008 %>%
   bootstraps(1000) %>%
   mutate(coef = map(splits, lm_coefs, as.formula(biden ~ .))) %>%
   unnest(coef) %>%
   group_by(term) %>%
   summarise(b_estimate = mean(estimate),
             b_se = sd(estimate, na = TRUE))

ols %>% summary() %>% tidy() %>%{.[, 1:3]} %>%
  left_join(bootstrap_coef) %>%
  `colnames<-`(c("term", "ols_estimate", "ols_se", "bootstrap_estimate", "bootstrap_se")) %>% kable()
```

term	ols_estimate	ols_se	bootstrap_estimate	bootstrap_se
(Intercept)	60.3993175	4.683819	58.6694834	2.9568005
female	4.1450536	1.398806	4.0912085	0.9550458
age	0.0647333	0.041411	0.0488182	0.0292423
educ	-0.6193741	0.292716	-0.3338699	0.1855375
dem	16.1703558	1.562710	15.3863096	1.0988054
rep	-13.7455192	1.900539	-15.9522726	1.4151573

The bootstrapped estimates of the five parameters and their standard errors are largely identical with minimal differences. The magnitude of traditional estimates are slightly greater than the bootstrapped estimates (except Democrat variable). The bootstrapped standard errors is slightly greater than the standard error of traditional estimates on age, democrat and republican variables. However, the author belives that such differences are negligible. The virtually identical results indicate that our estimates and the standard errors of our estimates in Q.1 are accurate.

The traditional estimates are generated from a linear regression model. The standard errors provide an accurate estimates of the average amount of differences between the estimated parameters $\hat{\beta}$ and the true population parameters β if the assumptions of linear regression holds. If these assumptions are violated (i.e., non-constant error variance, or "heteroskedasticity"), the standard errors will not be accurate. Therefore, bootstrap estimates in general are more robust because they do not rely any distributional assumptions. bootstrapping method is often used to estimate how close our sample answers are to the true population answers, when we are uncertain about the shape of the population or our assumptions about the shape of population are violated. By using bootstrapping, we can learn about the population from the sample we have.