Homework 01

STAT 430, Fall 2017

Due: Friday, September 15, 11:59 PM

Please see the homework instructions document for detailed instructions and some grading notes. Failure to follow instructions will result in point reductions.

Exercise 1

[**10 points**] This question will use data in a file called `hw01-data.csv`. The data contains four predictors: `a`, `b`, `c`, `d`, and a response `y`.

After reading in the data as `hw01_data`, use the following code to test-train split the data.

```
hw01_data = read.csv("hw01-data.csv")
```

```
set.seed(42)
train_index = sample(1:nrow(hw01_data), size = round(0.5 * nrow(hw01_data)))
train_data = hw01_data[train_index, ]
test_data = hw01_data[-train_index, ]
```

Next, fit four linear models using the training data:

- Model 1: `y ~ .`
- Model 2: `y ~ . + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 3: `y ~ . ^ 2 + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 4: `y ~ a * b * c * d + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`

```
model_1 = lm(y ~ ., data = train_data)
model_2 = lm(y ~ . + I(a ^ 2) + I(b ^ 2) + I(c ^ 2),
             data = train_data)
model_3 = lm(y ~ . ^ 2 + I(a ^ 2) + I(b ^ 2) + I(c ^ 2),
             data = train_data)
model_4 = lm(y ~ a * b * c * d + I(a ^ 2) + I(b ^ 2) + I(c ^ 2),
             data = train_data)
```

For each of the models above, report:

- Train RMSE
- Test RMSE
- Number of Parameters, Excluding the Variance For model 1

```
rmse_train1 = sqrt(mean((train_data$y -
                           predict(model_1,train_data))^2))
rmse_test1 = sqrt(mean((test_data$y -
                          predict(model_1, test_data)) ^ 2))
c1 = length(coef(model_1))
```

For model 2

```
rmse_train2 = sqrt(mean((train_data$y - predict(model_2,train_data))^2))
rmse_test2 = sqrt(mean((test_data$y -
```

```r
                          predict(model_2, test_data)) ^ 2))
c2 = length(coef(model_2))
```

For model 3

```r
rmse_train3 = sqrt(mean((train_data$y - predict(model_3,train_data))^2))
rmse_test3 = sqrt(mean((test_data$y -
                          predict(model_3, test_data)) ^ 2))
c3 = length(coef(model_3))
```

For model 4

```r
rmse_train4 = sqrt(mean((train_data$y - predict(model_4,train_data))^2))
rmse_test4 = sqrt(mean((test_data$y -
                          predict(model_4, test_data)) ^ 2))
c4 = length(coef(model_4))
```

To receive full marks, arrange this information in a well formatted table. Also note which model is best for making predictions.

```r
result = data.frame(model =
                      c("model_1","model_2","model_3","model_4"),
                    Train_RMSE =
                c(rmse_train1,rmse_train2,rmse_train3,rmse_train4),
                Test_RMSE =
                  c(rmse_test1, rmse_test2,rmse_test3,rmse_test4),
                Num_of_Param = c(c1,c2,c3,c4))
knitr::kable(result)
```

model	Train_RMSE	Test_RMSE	Num_of_Param
model_1	1.4381782	1.4286911	5
model_2	1.1242482	1.1526319	8
model_3	0.5105619	0.5206716	14
model_4	0.5082713	0.5211251	19

```
Model_3 is best for making prediction since the test RMSE is the smallest.
```

[**Not Graded**] For fun, find a model that outperforms each of the models above. *Hint:* Consider some exploratory data analysis. *Hint:* Your instructor's solution uses a model with only seven parameters. Yours may have more.

```r
model = lm(y ~. + I(a^2) + a*c*d,
           data = train_data)
rmse_test = sqrt(mean((test_data$y -
                         predict(model, test_data)) ^ 2))
rmse_test
```

```
## [1] 0.5167411
```

```r
length(model$coefficients)
```

```
## [1] 10
```

Exercise 2

[**10 points**] For this question we will use the `Boston` data from the `MASS` package. Use `?Boston` to learn more about the data.

```r
library(tibble)
library(readr)
library(MASS)
data(Boston)
Boston = as_tibble(Boston)
```

Use the following code to test-train split the data.

```r
set.seed(42)
boston_index = sample(1:nrow(Boston), size = 400)
train_boston = Boston[boston_index, ]
test_boston  = Boston[-boston_index, ]
```

Fit the following linear model that uses `medv` as the response.

```r
fit = lm(medv ~ . ^ 2, data = train_boston)
```

Fit two additional models, both that perform worse than `fit` with respect to prediction. One should be a smaller model. The other should be a larger mode. Call them `fit_smaller` and `fit_larger` respectively. Any "smaller" model should be nested in any "larger" model.

```r
fit_smaller = lm(medv ~ ., data = train_boston)
fit_larger = lm(medv ~ .^2 + I(crim^2),
                data = train_boston)
```

Report these three models as well as their train RMSE, test RMSE, and number of parameters. Note: you may report the models used using their `R` syntax. To receive full marks, arrange this information in a well formatted table.

```r
# Fit
rmse_train1 = sqrt(mean((train_boston$medv -
                          predict(fit,train_boston))^2))
rmse_test1 = sqrt(mean((test_boston$medv -
                         predict(fit, test_boston)) ^ 2))

c1 = length(coef(fit))

# Fit_smaller
rmse_train2 = sqrt(mean((train_boston$medv -
                          predict(fit_smaller,train_boston))^2))
rmse_test2 = sqrt(mean((test_boston$medv -
                         predict(fit_smaller, test_boston)) ^ 2))

c2 = length(coef(fit_smaller))

# Fit_larger
rmse_train3 = sqrt(mean((train_boston$medv -
                          predict(fit_larger,train_boston))^2))
rmse_test3 = sqrt(mean((test_boston$medv -
                         predict(fit_larger, test_boston)) ^ 2))
c3 = length(coef(fit_larger))
```

```r
result = data.frame(model = c("fit","fit_smaller","fit_larger"),
                    Train_RMSE = c(rmse_train1,rmse_train2,rmse_train3),
                    Test_RMSE = c(rmse_test1, rmse_test2,rmse_test3),
                    Num_of_Param = c(c1,c2,c3))
knitr::kable(result)
```

model	Train_RMSE	Test_RMSE	Num_of_Param
fit	2.623952	3.220200	92
fit_smaller	4.608428	5.053888	14
fit_larger	2.619390	3.258299	93

Exercise 3

[10 points] How do outliers affect prediction? Usually when fitting regression models for explanation, dealing with outliers is a complicated issue. When considering prediction, we can empirically determine what to do.

Continue using the `Boston` data, training split, and models from Exercise 2. Consider the model stored in `fit` from Exercise 2. Obtain the standardized residuals from this fitted model. Refit this model with each of the following modifications:

- Removing observations from the training data with absolute standardized residuals greater than 2.
- Removing observations from the training data with absolute standardized residuals greater than 3.

```r
data = train_boston[abs(rstandard(fit))<=2,]
fit1 = lm(medv ~ . ^ 2, data = data)
data1 = train_boston[abs(rstandard(fit))<=3,]
fit2 = lm(medv ~ . ^ 2, data = data1)
```

(a) Use these three fitted models, including the original model fit to unmodified data, to obtain test RMSE. Summarize these results in a table. Include the number of observations removed for each. Which performs the best? Were you justified modifying the training data?

```r
rmse_test1 = sqrt(mean((test_boston$medv -
                          predict(fit, test_boston)) ^ 2))
rmse_test2 = sqrt(mean((test_boston$medv -
                          predict(fit1, test_boston)) ^ 2))
rmse_test3 = sqrt(mean((test_boston$medv -
                          predict(fit2, test_boston)) ^ 2))
```

```r
d0 = 0
d1 = nrow(train_boston)-nrow(data)
d2 = nrow(train_boston)-nrow(data1)
```

```r
result = data.frame(model = c("fit","fit1","fit2"),
                    Test_MSE = c(rmse_test1,rmse_test2,rmse_test3),
                    Observation_Deleted = c(d0,d1,d2))
knitr::kable(result)
```

model	Test_MSE	Observation_Deleted
fit	3.220200	0
fit1	3.008800	19

model	Test_MSE	Observation_Deleted
fit2	3.056008	5

```
Fit1 is from the model which removes standard residuals greater
than 2, while Fit2 from the model which removes standard residuals
greater than 3.
The model which removes standard residuals greater than two
performs better. Based on the table above, it seems that modifying
data would perform better. But it depends on the case.
```

(b) Using the *best* of these three fitted models, create a 99% **prediction interval** for a new observation with the following values for the predictors:

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0.02763	75.0	3.95	0	0.4280	6.595	22.8	5.4011	3	252	19.3	395.63	4.32

```
newdatas = data.frame(crim = 0.02763,zn = 75,indus = 3.95,
                      chas = 0, nox = 0.4280,rm = 6.595,
                      age= 22.8, dis = 5.4011,rad= 3, tax= 252,
                      ptratio = 19.3,black = 395.63,lstat = 4.32)
names = colnames(test_boston)[-14]
predict(fit1, newdata = newdatas, interval = "prediction", level = 0.99)
```

```
##        fit      lwr      upr
## 1 27.52639 21.03786 34.01491
```