Homework 03

STAT 430, Fall 2017

Due: Friday, September 29, 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 (Data Scaling?)

[8 points] This exercise will use data in `hw03-train-data.csv` and `hw03-test-data.csv` which are train and test datasets respectively. Both datasets contain multiple predictors and a numeric response `y`.

```
train_data = read.csv('hw03-train-data.csv')
test_data = read.csv('hw03-test-data.csv')
```

Fit a total of six k-nearest neighbors models. Consider three values of k: 1, 5, and 25. To make a total of six models, consider both scaled and unscaled X data. For each model, use all available predictors.

```
library(FNN)
```

```
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
get_rmse_knn = function(x_tst, y_tst, x_trn, y_trn, k) {
  pred = knn.reg(train = x_trn,
                 test = x_tst,
                 y = y_trn,
                 k = k)$pred
  rmse(actual = y_tst, predicted = pred)
}
```

```
#################### unscale  ##########################
train_x = train_data[, -1]
test_x = test_data[, -1]
train_y = train_data$y
test_y = test_data$y
k = c(1, 5, 25)
train_unscale = sapply(k, get_rmse_knn, x_tst = train_x, y_tst = train_y,
                       x_trn = train_x, y_trn = train_y)
test_unscale = sapply(k, get_rmse_knn, x_tst = test_x, y_tst = test_y,
                      x_trn = train_x, y_trn = train_y)
```

```
########################## scale  ##########################
scale_x_train = scale(train_x)
scale_x_test = scale(test_x)
train_scale = sapply(k, get_rmse_knn, x_tst = scale_x_train, y_tst = train_y,
                     x_trn = scale_x_train, y_trn = train_y)
test_scale = sapply(k, get_rmse_knn, x_tst = scale_x_test, y_tst = test_y,
                    x_trn = scale_x_train, y_trn = train_y)
```

Summarize these results using a single well-formatted table which displays test RMSE, `k`, and whether or not scaling was used.

```
df = data.frame(Unscale_RMSE = test_unscale,
                Scale_RMSE = test_scale,
                k = k)
                knitr::kable(df)
```

Unscale_RMSE	Scale_RMSE	k
0.6839884	0.7214409	1
0.5369142	0.5467248	5
0.5157672	0.5081259	25

Exercise 2 (KNN versus Linear Models)

[9 points] Find a k-nearest neighbors model that outperforms an additive linear model for predicting `mpg` in the `Auto` data from the `ISLR` package. Use the following data cleaning and test-train split to perform this analysis. Keep all of the predictor variables as numeric variables. Report the test RMSE for both the additive linear model, as well as your chosen model. For your model, also note what value of k you used, as well as whether or not you scaled the X data.

```
# install.packages("ISLR")
library(ISLR)
library(FNN)
auto = Auto[,!names(Auto) %in% c("name")]
```

```
set.seed(42)
auto_idx = sample(1:nrow(auto), size = round(0.5 * nrow(auto)))
auto_trn = auto[auto_idx,]
auto_tst = auto[-auto_idx,]
```

The additive linear model can be fit using:

```
add_model = lm(mpg ~ ., data = auto_trn)
get_rmse = function(model, data) {
  rmse(actual = data[, 'mpg'],
       predicted = predict(model, data))
}
```

```
########################## unscale  ##########################
k = seq(1, 7)
auto_trn_x = auto_trn[, -1]
auto_tst_x = auto_tst[, -1]
auto_trn_y = auto_trn$mpg
auto_tst_y = auto_tst$mpg
#train_unscale = sapply(k,get_rmse_knn,x_tst = auto_trn_x,y_tst = auto_trn_y,
#                       x_trn = auto_trn_x, y_trn = auto_trn_y)
test_unscale = sapply(k, get_rmse_knn, x_tst = auto_tst_x, y_tst = auto_tst_y,
                      x_trn = auto_trn_x, y_trn = auto_trn_y)
```

```
########################## scale  ##########################
scale_x_trn = scale(auto_trn_x)
```

```
scale_x_tst = scale(auto_tst_x)
#train_scale = sapply(k, get_rmse_knn, x_tst = scale_x_trn, y_tst = auto_trn_y,
#                        x_trn = scale_x_trn, y_trn = auto_trn_y)
test_scale = sapply(k, get_rmse_knn, x_tst = scale_x_tst, y_tst = auto_tst_y,
                       x_trn = scale_x_trn, y_trn = auto_trn_y)
```

```
df = data.frame(RMSE_unscale = test_unscale,
                RMSE_scale = test_scale,
                k = k)
#### choose the model with K = 6
ind = which.min(df$RMSE_scale)
ans = df[ind, ]
```

```
rmse_add = get_rmse(add_model, auto_tst)
df0 = data.frame(additive_rmse = rmse_add,
                 KNN_rmse = ans$RMSE_scale,
                 status = "scale", k = 6)
knitr::kable(df0)
```

additive_rmse	KNN_rmse	status	k
3.068489	2.860031	scale	6

Exercise 3 (Bias-Variance Tradeoff, KNN)

[8 points] Run a modified version of the simulation study found in Section 8.3 of R4SL. Use the same data generating process to simulate data:

```
f = function(x) {
  x ^ 2
}
```

```
get_sim_data = function(f, sample_size = 100) {
  x = runif(n = sample_size, min = 0, max = 1)
  y = rnorm(n = sample_size, mean = f(x), sd = 0.3)
  data.frame(x, y)
}
```

So, the following generates one simulated dataset according to the data generating process defined above.

```
sim_data = get_sim_data(f)
```

Evaluate predictions of $f(x = 0.90)$ for three models:

- k-nearest neighbors with $k = 1$. $\hat{f}_1(x)$
- k-nearest neighbors with $k = 10$. $\hat{f}_{10}(x)$
- k-nearest neighbors with $k = 100$. $\hat{f}_{100}(x)$

For simplicity, when fitting the k-nearest neighbors models, do not scale X data.

Use 500 simulations to estimate squared bias, variance, and the mean squared error of estimating $f(0.90)$ using $\hat{f}_k(0.90)$ for each k. Report your results using a well formatted table.

At the beginning of your simulation study, run the following code, but with your nine-digit Illinois UIN.

```r
set.seed(650178134)
```

```r
n_sims = 500
x = data.frame(x = 0.90)
n_models = 3
predictions = matrix(0, nrow = n_sims, ncol = n_models)
for(sim in 1:n_sims) {

  # simulate new, random, training data
  # this is the only random portion of the bias, var, and mse calculations
  # this allows us to calculate the expectation over D
  sim_data = get_sim_data(f)

  trn_y = sim_data$y
  trn_x = sim_data["x"]

  # fit models
  knn_1 = knn.reg(train = trn_x, test = x, y = trn_y,k = 1)
  knn_10 = knn.reg(train = trn_x, test = x, y = trn_y,k = 10)
  knn_100 = knn.reg(train = trn_x, test = x, y = trn_y,k = 100)

  # get predictions
  predictions[sim, 1] = knn_1$pred
  predictions[sim, 2] = knn_10$pred
  predictions[sim, 3] = knn_100$pred
}
```

```r
get_bias = function(estimate, truth) {
  mean(estimate) - truth
}

get_var = function(estimate) {
mean((estimate - mean(estimate)) ^ 2)
}
get_mse = function(truth, estimate) {
mean((estimate - truth) ^ 2)
}
```

```r
bias = apply(predictions, 2, get_bias, truth = f(x = 0.90))
Variance = apply(predictions, 2, get_var)
mse = apply(predictions, 2, get_mse, truth = f(x = 0.90))
df1 = data.frame(
  k = c(1, 10, 100),
  bias_sqaured = round(bias ^ 2, 5),
  Variance = round(Variance, 5),
  MSE = round(mse, 5)
  )
knitr::kable(df1)
```

k	bias_sqaured	Variance	MSE
1	0.00017	0.09508	0.09525
10	0.00003	0.01041	0.01045
100	0.22868	0.00177	0.23045

Exercise 4 (Concept Checks)

[**1 point each**] Answer the following questions based on your results from the three exercises.

(a) Based on your results in Exercise 1, which k performed best?

k = 25

(b) Based on your results in Exercise 1, was scaling the data appropriate?

Yes. After scaling, the rmse is not change so much. Scaling might be appropriate to remove the influence of error and outliers.

(c) Based on your results in Exercise 2, why do you think it was so easy to find a k-nearest neighbors model that met this criteria?

KNN model can deal with non-linear data, while linear regression model can only deal with data that are linear distrubuted. Thus, KNN model would be easier to fit in the data.

(d) Based on your results in Exercise 3, which of the three models do you think are providing unbiased predictions?

The second model with k = 10, since the bias of this model is lowest and closest to 0.

(e) Based on your results in Exercise 3, which model is predicting best at $x = 0.90$?

second model with k = 10. Since the MSE of this model is the smallest one.