Homework 03

STAT 430, Fall 2017

Due: Friday, September 29, 11:59 PM

Exercise 1 (Data Scaling?)

```
library(tibble)
library(readr)

make_hw03_data = function(n_obs = 1000) {

  x1 = rnorm(n = n_obs, mean = 0 , sd = 1)
  x2 = rnorm(n = n_obs, mean = 0 , sd = 1)
  x3 = rnorm(n = n_obs, mean = 0 , sd = 1)
  x4 = rnorm(n = n_obs, mean = 0 , sd = 1)
  y = rnorm(x1 + x2 + x3, sd = 0.5)
  tibble(y, x1 = x1 / 2, x2 = x2 / 5, x3 = x3 / 5, x4 = x4 * 10)

}

# generate datasets
set.seed(42)
hw03_train_data = make_hw03_data(n_obs = 150)
hw03_test_data =  make_hw03_data(n_obs = 850)

# write to files
write_csv(hw03_train_data, "hw03-train-data.csv")
write_csv(hw03_test_data, "hw03-test-data.csv")

# clean up
rm(hw03_train_data)
rm(hw03_test_data)
```

[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`.

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.

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

Solution:

```
# load necessary libraries
library(readr)
library(FNN)
```

```r
# read in data
trn_data = read_csv("hw03-train-data.csv")
tst_data = read_csv("hw03-test-data.csv")
```

```r
# RMSE helper function
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```r
# split response and predictors for use with knn.reg()
X_trn = trn_data[, !names(trn_data) %in% c("y")]
X_tst = tst_data[, !names(tst_data) %in% c("y")]
y_trn = trn_data$y
y_tst = tst_data$y
```

```r
# get predictions using scaling
sp_01 = knn.reg(train = scale(X_trn), test = scale(X_tst), y = y_trn, k = 1)$pred
sp_05 = knn.reg(train = scale(X_trn), test = scale(X_tst), y = y_trn, k = 5)$pred
sp_25 = knn.reg(train = scale(X_trn), test = scale(X_tst), y = y_trn, k = 25)$pred

# get unscaled predictions
up_01 = knn.reg(train = X_trn, test = X_tst, y = y_trn, k = 1)$pred
up_05 = knn.reg(train = X_trn, test = X_tst, y = y_trn, k = 5)$pred
up_25 = knn.reg(train = X_trn, test = X_tst, y = y_trn, k = 25)$pred

# get test RMSE
predictions = list(sp_01, sp_05, sp_25, up_01, up_05, up_25)
rmses = sapply(predictions, rmse, actual = y_tst)
```

```r
# summarize results
results = data.frame(
  scaling = c(rep("Yes", 3), rep("No", 3)),
  k = rep(c(1, 5, 25), 2),
  rmses
)

colnames(results) = c("Scaling?", "k", "Test RMSE")
knitr::kable(results)
```

Scaling?	k	Test RMSE
Yes	1	0.7214409
Yes	5	0.5467248
Yes	25	0.5081259
No	1	0.6839884
No	5	0.5369142
No	25	0.5157672

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)
auto = Auto[, !names(Auto) %in% c("name")]
```

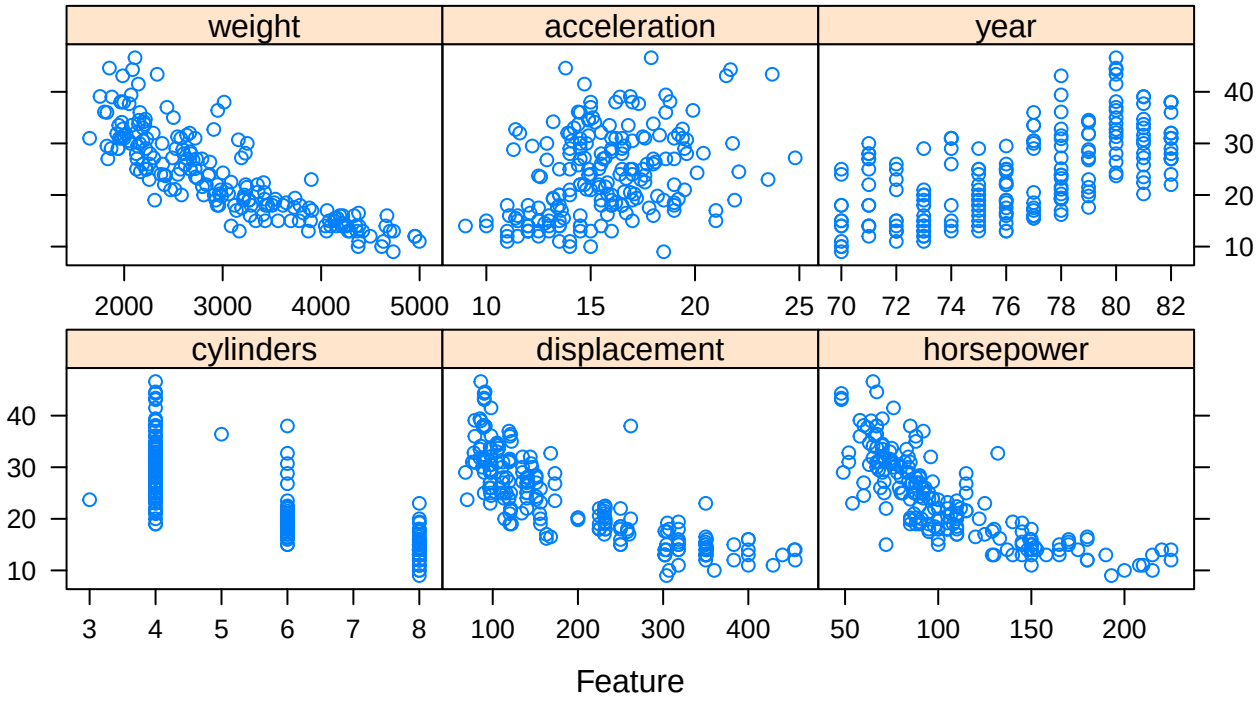
```
set.seed(42)
auto_idx = sample(1:nrow(auto), size = round(0.5 * nrow(auto)))
auto_trn = auto[auto_idx, ]
auto_tst = auto[-auto_idx, ]
```

The additive linear model can be fit using:

```
lm(mpg ~ ., data = auto_trn)
```

Solution:

```
# some exploratory data analysis
caret::featurePlot(
  x = auto_trn[ , c("cylinders", "displacement", "horsepower",
                    "weight", "acceleration", "year")],
  y = auto_trn$mpg)
```




```
# additive model to outperform
fit = lm(mpg ~ ., data = auto_trn)

# split auto (both test and train) into response and predictors
X_trn_auto = auto_trn[, !names(auto_trn) %in% c("mpg")]
X_tst_auto = auto_tst[, !names(auto_tst) %in% c("mpg")]
y_trn_auto = auto_trn$mpg
y_tst_auto = auto_tst$mpg
```

```
# get KNN pred
knn_pred = knn.reg(train = scale(X_trn_auto),
                   test = scale(X_tst_auto),
                   y = y_trn_auto, k = 5)$pred

# verify KNN outperforms lm() in test data
rmse(predicted = predict(fit, auto_tst), actual = auto_tst$mpg)
```

```
## [1] 3.068489
```

```
rmse(predicted = knn_pred , actual = y_tst_auto)
```

```
## [1] 2.897238
```

Here we did scale the data and used $k = 5$.

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.

```
set.seed(123456789)
```

Solution:

```
# setup simulation
n_sims = 500
n_models = 3
x = data.frame(x = 0.90) # fixed point at which we make predictions
predictions = matrix(0, nrow = n_sims, ncol = n_models)
```

```r
# perform simulations
for(sim in 1:n_sims) {

  # generate datasets according to the data generating process
  sim_data = get_sim_data(f)

  # get predictions
  predictions[sim, 1] = knn.reg(train = sim_data["x"], test = x,
                                y = sim_data["y"], k = 1)$pred
  predictions[sim, 2] = knn.reg(train = sim_data["x"], test = x,
                                y = sim_data["y"], k = 10)$pred
  predictions[sim, 3] = knn.reg(train = sim_data["x"], test = x,
                                y = sim_data["y"], k = 100)$pred
}
```

```r
# helper functions from R4SL

get_var = function(estimate) {
  mean((estimate - mean(estimate)) ^ 2)
}

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

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

```r
# calculate bias, variance, and mse of predictions for each k
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))
```

```r
# summarize results
results = data.frame(
  k = c(1, 10, 100),
  round(mse, 5),
  round(bias ^ 2, 5),
  round(variance, 5)
)
colnames(results) = c("k", "Mean Squared Error", "Bias Squared", "Variance")
rownames(results) = NULL
knitr::kable(results)
```

k	Mean Squared Error	Bias Squared	Variance
1	0.09396	0.00004	0.09392
10	0.00896	0.00001	0.00895
100	0.22809	0.22614	0.00195

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?

Solution: Based on this test data, $k = 25$ performed best.

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

Solution: Based on this test data, scaling was appropriate.

(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?

Solution: Based on the exploratory data analysis, we see some **non-linear** relationships that an additive model would not detect, but a non-parametric method like KNN would automatically approximate.

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

Solution: Both $k = 1$ and $k = 10$.

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

Solution: Based on these results, $k = 10$, as it has the lowest MSE.