Assignment 1

Chris Rodgers

07/03/2021

```
knitr::opts_chunk$set(echo = TRUE)
setwd("~/Data Mining\\data-mining-assignment-01")
```

Assignment 1

Name: Chris Rodgers **Student** ID: 81011955

Question 1

Describe one advantage and one disadvantage of flexible (versus a less flexible) approaches for regression. Under what conditions might a less flexible approach be preferred?

Advantage: Flexible models allow for more parameters that can more accurately reflect the true nature of f.

Disadvantage: Flexible models can overfit the data. This means that the model can follow the errors in our taining data and as a consequence not reflect the true nature of f.

Conditions to prefer a less flexible approach: We might prefer a less flexible approach when we are mainly interested in inference where the less flexible model will be easier to interpret. It is much easier to understand the link between X and Y when the model is less flexible. This is not the case with less flexible models which are harder to interpret.

Question 2

Question 3

In this question, you will fit kNN regression models to the Auto data set to predict Y = mpg using X = horsepower. This data has been divided into training and testing sets: AutoTrain.csv and AutoTest.csv (download these sets from Learn). The kNN() R function on Learn should be used to answer this question (you need to run the kNN code before calling the function). (a) Perform kNN regression with k = 2, 5, 10, 20, 30, 50 and 100, (learning from the training data) and compute the training and testing MSE for each value of k. (b) Which value of k performed best? Explain. (c) Plot the training data, testing data and the best kNN model in the same figure. (The points() function is useful to plot the kNN model because it is discontinuous.) (d) Describe the bias-variance trade-off for kNN regression

```
## STAT318/462 kNN regression function

kNN <- function(k,x.train,y.train,x.pred) {
#
```

```r
## This is kNN regression function for problems with
## 1 predictor
#
## INPUTS
#
# k       = number of observations in nieghbourhood
# x.train = vector of training predictor values
# y.train = vector of training response values
# x.pred  = vector of predictor inputs with unknown
#           response values
#
## OUTPUT
#
# y.pred  = predicted response values for x.pred

## Initialize:
n.pred <- length(x.pred);       y.pred <- numeric(n.pred)

## Main Loop
for (i in 1:n.pred){
  d <- abs(x.train - x.pred[i])
  dstar = d[order(d)[k]]
  y.pred[i] <- mean(y.train[d <= dstar])
}
## Return the vector of predictions
invisible(y.pred)
}
```

```r
train = read.csv("~/Data Mining/data-mining-assignment-01/AutoTrain.csv")
test = read.csv("~/Data Mining/data-mining-assignment-01/AutoTest.csv")

knn2 = kNN(2, train$horsepower, train$mpg, test$horsepower)
knn5 = kNN(5, train$horsepower, train$mpg, test$horsepower)
knn10 = kNN(10, train$horsepower, train$mpg, test$horsepower)
knn20 = kNN(20, train$horsepower, train$mpg, test$horsepower)
knn30 = kNN(30, train$horsepower, train$mpg, test$horsepower)
knn50 = kNN(50, train$horsepower, train$mpg, test$horsepower)
knn100 = kNN(100, train$horsepower, train$mpg, test$horsepower)

knn2train = kNN(2, train$horsepower, train$mpg, train$horsepower)
knn5train = kNN(5, train$horsepower, train$mpg, train$horsepower)
knn10train = kNN(10, train$horsepower, train$mpg, train$horsepower)
knn20train = kNN(20, train$horsepower, train$mpg, train$horsepower)
knn30train = kNN(30, train$horsepower, train$mpg, train$horsepower)
knn50train = kNN(50, train$horsepower, train$mpg, train$horsepower)
knn100train = kNN(100, train$horsepower, train$mpg, train$horsepower)


knn2MSE = (test$mpg - knn2)^2
knn5MSE = (test$mpg - knn5)^2
knn10MSE = (test$mpg - knn10)^2
knn20MSE = (test$mpg - knn20)^2
knn30MSE = (test$mpg - knn30)^2
```

```r
knn50MSE = (test$mpg - knn50)^2
knn100MSE = (test$mpg - knn100)^2

knn2MSEtrain = (test$mpg - knn2train)^2
knn5MSEtrain = (test$mpg - knn5train)^2
knn10MSEtrain = (test$mpg - knn10train)^2
knn20MSEtrain = (test$mpg - knn20train)^2
knn30MSEtrain = (test$mpg - knn30train)^2
knn50MSEtrain = (test$mpg - knn50train)^2
knn100MSEtrain = (test$mpg - knn100train)^2


print(sum(knn2MSEtrain))
```

```
## [1] 22561.57
```

```r
print(sum(knn5MSEtrain))
```

```
## [1] 22221.57
```

```r
print(sum(knn10MSEtrain))
```

```
## [1] 22260.9
```

```r
print(sum(knn20MSEtrain))
```

```
## [1] 21897.52
```

```r
print(sum(knn30MSEtrain))
```

```
## [1] 21182.39
```

```r
print(sum(knn50MSEtrain))
```

```
## [1] 19841.95
```

```r
print(sum(knn100MSEtrain))
```

```
## [1] 15876.68
```

```r
print(sum(knn2MSE))
```

```
## [1] 4481.244
```

```r
print(sum(knn5MSE))
```

```
## [1] 3834.391
```

```
print(sum(knn10MSE))
```

```
## [1] 3651.312
```

```
print(sum(knn20MSE))
```

```
## [1] 3394.442
```

```
print(sum(knn30MSE))
```

```
## [1] 3514.315
```

```
print(sum(knn50MSE))
```

```
## [1] 3836.452
```

```
print(sum(knn100MSE))
```

```
## [1] 5157.823
```

3(a)

3(b) The best performing K was 20. This is evidence by it having the lowest MSE and is likely because this value strikes the best balance between bias and variance in the model.

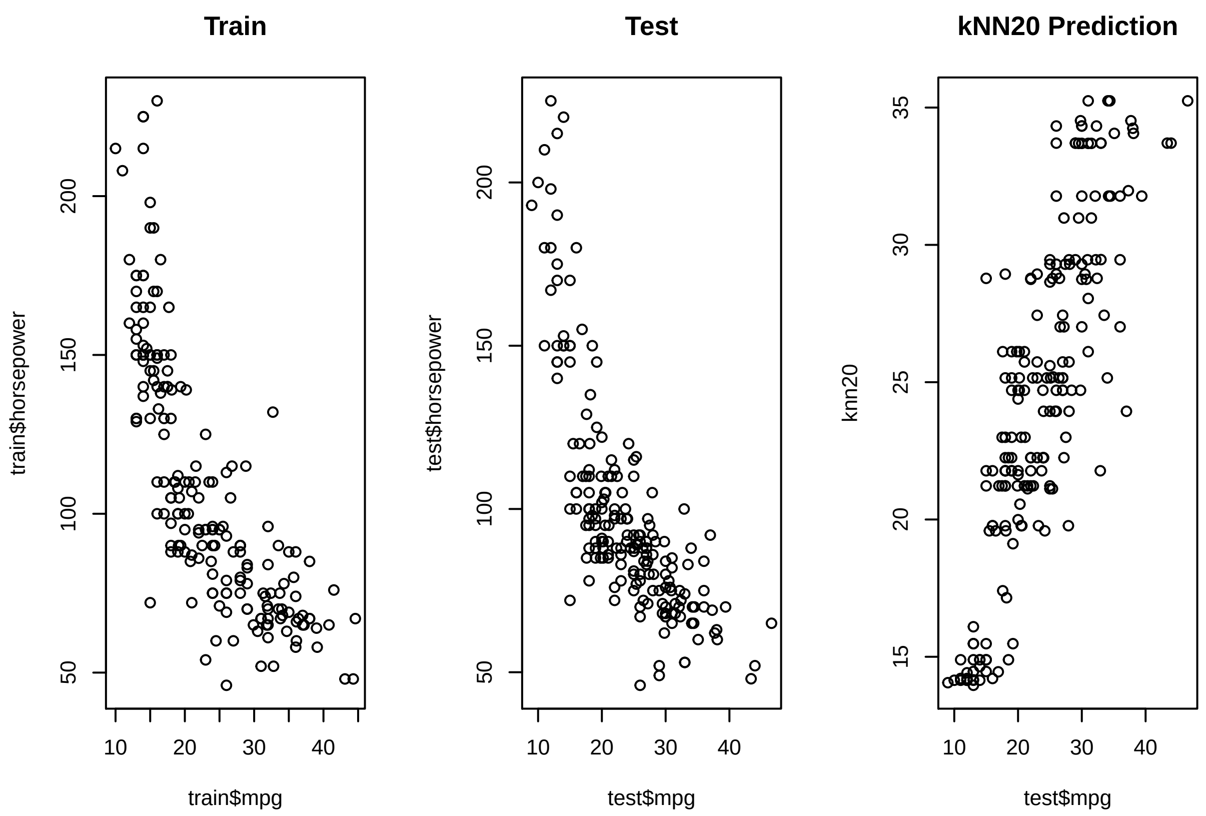
3(c) Plot of training data, test data and the best kNN model.

```
par(mfrow=c(1,3))

plot(train$mpg, train$horsepower, main="Train")

plot(test$mpg, test$horsepower, main="Test")

plot(test$mpg, knn20, main="kNN20 Prediction")
```



3(d) Bias and variance are two competing properties of statistical learning methods. Variance refers to the amount a model will change if we had different training data. Bias refers to the error we introduce by approximating real world complexity into a simpler model. In general, the more flexible a model is the more variance we will have but we will have less bias (and vice-versa for less flexible models).

In the case of kNN.