CV Homework 5

Code

This is an R script with the purpose ofcomparing kNN performance to linear regression and logistic regression

Ramesh Kanakala

PROBLEM 1

Step 1

Hide

```
#load ISLR and divide Wagee into train and test
library(ISLR)
Wagee <- Wage
Wagee$race <- as.numeric(Wagee$race)
Wagee$jobclass <-as.numeric(Wagee$jobclass)
set.seed(1234)
i <- sample(1:nrow(Wagee), nrow(Wagee)*0.75,
replace=FALSE)
train <- Wagee[i,]
test <- Wagee[-i,]
```

Step 2

Hide

```
#build linear regression, look at summary and residual plots
lm1 <- lm(wage~age+race+jobclass, data=train)
summary(lm1)
par(mfrow=c(2,2))
plot(lm1)
```

Step 3

Hide

```
#evaluate model on test data and output corr and mse
pred <- predict(lm1, newdata=test)
correlation <- cor(pred, test$wage)
print(paste("correlation: ", correlation))
mse <- mean((pred - test$wage)^2)
print(paste("mse: ", mse))
```

Step 4

Hide

```
#kNN regresssion and output corr and mse
library(caret)
summary(train)
fit <- knnreg(train[,c(2,4,7)],train[,11], k=1)
pred1 <- predict(fit, test[,c(2,4,7)])
correlation <- cor(pred1, test$wage)
print(paste("correlation: ", correlation))
mse <- mean((pred1 - test$wage)^2)
print(paste("mse: ", mse))
```

Step 5

a. kNN regression has a barely lower correlation than linear regression when predicting for wage: 0.22002 < 0.22095
b. kNN had a higher mse than linear regression: 1728.40 > 1640.85
c. The mse metric increased meaning there is more error whereas the correlation decreased meaning there is a weaker positive relationship between wage and the predictors here, though the differences are quite minute
d. kNN has low bias with the small value of k whereas linear regression has high bias; linear regression is more likely to underfit and not capture the true shape of the data. On the other hand, kNN regression makes no assumptions about the data as it doesn't focus on building a model rather it looks at the cluster of neighbors around test instances.

PROBLEM 2

Step 1

Hide

```
#load mlbench and divide ablo into train and test
library(AppliedPredictiveModeling)
data(abalone)
ablo <- abalone[abalone$Type != "I", ]
ablo$Diameter.Big <- as.factor(ifelse(ablo$Diameter>0.4600, 1, 0))
ablo$LongShell.Big <- as.factor(ifelse(ablo$LongestShell>0.5850, 1, 0))
set.seed(1234)
i <- sample(1:nrow(ablo), nrow(ablo)*0.75,
replace=FALSE)
train1 <- ablo[i,]
test1 <- ablo[-i,]
train_labels <- ablo[i, 1]
test_labels <- ablo[-i, 1]
```

Step 2

Hide

```
#build logistic regression, look at summary and residual plots
glm1 <- glm(Type~Rings+Diameter.Big+LongShell.Big, family = "binomial", data = train1)
summary(glm1)
```

Step 3

Hide

```
#evaluate model on test data and output accuracy and confusion matrix
library(e1071)
probs <- predict(glm1, newdata=test1, type="response")
pred2 <- ifelse(probs>0.5, 'M', 'F')
acc <- mean(pred2==test1$Type)
acc
library(caret)
confusionMatrix(as.factor(pred2), test1$Type)
```

*Not using class I

Step 4

Hide

```
#use knn() and output accuracy and confusion matrix
library(class)
#using 9 as well to minimize ties
class_pred <- knn(train1[,9:11], test1[,9:11], cl=train_labels, k=1)
acc <- length(which(class_pred == test_labels)) / length(class_pred)
acc
confusionMatrix(as.factor(class_pred), test1$Type)
```

Step 5

Hide

```
#kNN using predictor columns 2-6, 8-10 and output accuracy and a table
library(class)
class_pred1 <- knn(train1[,c(2:6, 8:10)], test1[,c(2:6, 8:10)], cl=train_labels, k=1)
acc <- length(which(class_pred1 == test_labels)) / length(class_pred1)
acc
confusionMatrix(as.factor(class_pred1), test1$Type)
```

Step 6

a. Logistic regression performed better than the first kNN algorithm: 0.5219 > 0.4894. This may be because the abalone types are linearly separable and not veery complex or non-linear. The second kNN algorithm has a greater accuracy, 0.5049, and this is most likely because it uses more predictors.
b. This accuracy for the last kNN algorithm, 0.5049, is less than but more-or-less on-par with what we've seen for the ablo Class classification. Homework 4's glm1 has an accuracy of 0.5179 and the naive bayes has 0.5175. kNN performs worse here for perhaps two reasons. One, the data may not be very complex and the simple algorithms outshine knn here. Second, kNN is a non-parametric approach so it doesn't assume the shape of the distribution like logistic regression or naive-bayes, but that actually might be a disadvantage here.