Chase Moreno

@2078503

MATH 4323

Homework 2

1. The table below provides a training data set containing six observations, three predictors, and one qualitative response variable (Y = color ”Blue” or color ”Red”).

Obs.	X_1	X_2	X_3	Y
1	1	2	-1	Blue
2	1	0	3	Blue
3	0	3	0	Red
4	0	2	-1	Red
5	2	0	-1	Blue
6	1	4	1	Red

Suppose we wish to use K-nearest neighbors to work with this data set.

a) Use K = 1 to predict the color (either ”Blue” or ”Red”), using the K-nearest neighbor predictions of color for each of the training observations. What is the average misclassification training error?

Obs.	**X1**	**X2**	**X3**	**Y**
1	1	2	-1	Blue
2	1	0	3	Blue
3	0	3	0	Red
4	0	2	-1	Red
5	2	0	-1	Blue
6	1	4	1	Red

Observation 4 and 5 have the nearest neighbor based on X1, X2, and X3.
The average misclassification training error is 0 because prediction is either blue or red.

b) Use K = 3 to calculate the K-nearest neighbor predictions for all the training observations. What is the average misclassification training error?

Obs.	**X1**	**X2**	**X3**	**Y**
1	1	2	-1	Blue
2	1	0	3	Blue
3	0	3	0	Red
4	0	2	-1	Red
5	2	0	-1	Blue
6	1	4	1	Red

Observation 1, 4, and 5 have the nearest neighbor based on X1, X2, and X3. Now, we have color Blue, Red, and Blue which blue color wins. So, we have 1 misclassification which is observation 4 because of color red and predicted as blue.

Using the formula to find the misclassification training error is by sum up the observations that is misclassified divide by the total number of observations.
1 / 6 = 0.1666667
The misclassification training error is 0.1666667.

c) Suppose we wish to use this data set to make a prediction for Y when x0 = (X1, X2, X3), with X1 = X2 = X3 = 0, using K-nearest neighbors.

i. Compute the Euclidean distance between each observation and the test point, X1 = X2 = X3 = 0.

$$\sqrt{(x_1-0)^2+(x_2-0)^2+(x_3-0)^2}$$

Obs. 1: $\sqrt{(1-0)^2+(2-0)^2+(-1-0)^2}=2.4495$

Obs. 2: $\sqrt{(1-0)^2+(0-0)^2+(3-0)^2}=3.1623$

Obs. 3: $\sqrt{(0-0)^2+(3-0)^2+(0-0)^2}=3$

Obs. 4: $\sqrt{(0-0)^2+(2-0)^2+(-1-0)^2}=2.2361$

Obs. 5: $\sqrt{(2-0)^2+(0-0)^2+(-1-0)^2}=2.2361$

Obs. 6: $\sqrt{(1-0)^2+(4-0)^2+(1-0)^2}=4.2426$

ii. What is our prediction with K = 1? Why?
Our prediction is either red or blue when k = 1 because obs. 4 and 5 has the same distance value which is 2.2361.

iii. What is our prediction with K = 3? Why?
Our prediction is blue when k = 3 because obs. 1, 4 and 5 have the nearest distance value which are 2.4496, 2.2361, and 2.2361. The colors for theses observation are blue, red, and blue. So, we have 2 blues and 1 red which we most likely to predict blue.

iv. What is our prediction with K = 5? Why?
Our prediction is blue when k = 5 because all observations except obs. 6 have the nearest distance from each other. So, we have total of 3 blues and 2 reds. Therefore, blue is our prediction when k = 5.

2. We now review the topic of model validation:

a) Explain how validation set approach is implemented. What are its main disadvantages?
According to lecture material 5, the first step for validation set approach is to randomly divide the data sets into two parts which is Training set and Validation set.

Next, we are using the training data to fit the model, then use the fitted model to predict outcomes for validation data. At the last step, we determine the validation set error rate which is also a test error rate. However, the validation set approach allows us to choose best *K*. The main disadvantages from the validation set approach that some variables can appear to be an expendable. Non-numeric values will give you an error if we do not convert it into a numerical dummy variable. Also, cannot rely on results based on one validation set if we are resulting test error with large data sets.

b) Explain how the leave-one-out cross-validation is implemented (LOOCV). How does it improve on validation set approach?
According to lecture material 5, the first step for LOOCV approach is to split the data into two subsets which is training set and test set of just one observation. Next, we are using the training set to fit the model and produce prediction of $\hat{y}_i$. At the last step, we calculate misclassification error. The LOOCV improve on validation set approach by much more stable with test error estimate and uses nearly the whole data set at each step.

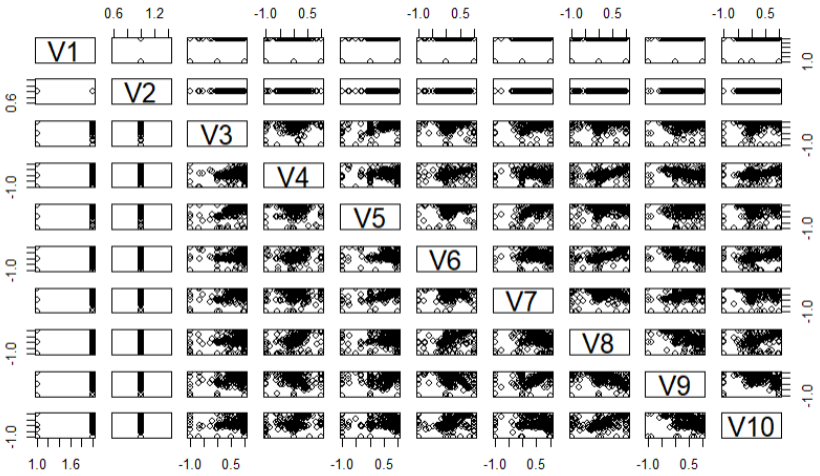
c) What is the main disadvantage of LOOCV approach? Hint: Imagine we have 10000 observations and 20 variables.
The main disadvantage of LOOCV approach would be computationally demanding because it will be expensive and can lead to over fitting for large data set. For example, we have 10000 observations and 20 variables. So, we need to train 10000 times which is expensive and fitting 20 variables can lead to overfitting.

d) Could we use validation set & LOOCV for any supervised learning approach? Or only for KNN?
Yes, we can use both the validation set and LOOCV for any supervised learning.

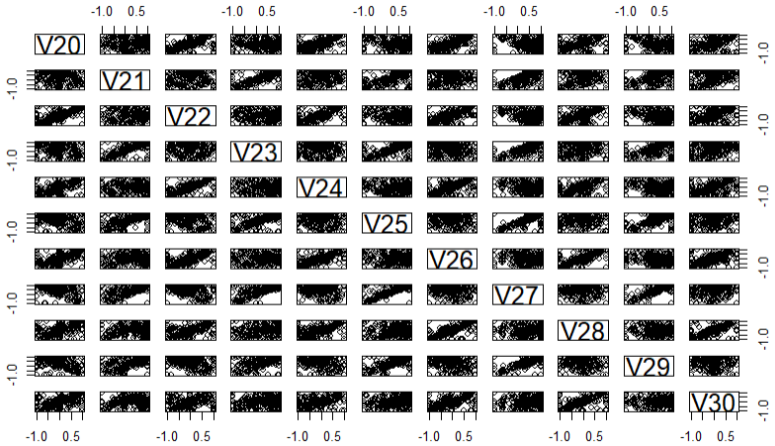
3. This question should be answered using the Ionosphere data set, which is part of the mlbench package. This radar data was collected by a system in Goose Bay, Labrador. The data frame consists of 351 observations on 35 independent variables. The last column in the dataframe is a categorical variable, ”Class”, defining the free electrons in the ionosphere: ”good” radar returns are those showing evidence of some type of structure in the ionosphere. ”bad” returns are those that do not.

a) Produce some numerical and graphical summaries of the Ionosphere data. Do there appear to be any patterns?
My R code:

```
> plot(Ionosphere[1:10])
```




```
> plot(Ionosphere[20:30])
```




```
> summary(Ionosphere)
```

I do not see any patterns. However, from looking at the plot and the summary(), I se e variable 2 to 35 have the min number of -1 and the max number of 1.

b) Notice that the second the column contains only one single value, so remove that column and work on the rest of the questions using the new dataset. Perform a K-Nearest Neighbors (KNN) algorithm with K = 1, where Class is the response, and the rest columns in the dataset as predictors.

My R code to drop the second column and store in the new dataset:

```
> new_df <- Ionosphere[,-2]
> dim(new_df)
[1] 351  34
```

Perform a KNN algorithm with k = 1

```
> library(class)
> train.X <- new_df[,-34]
> test.X <- new_df[,-34]
> train.y <- new_df$class
> test.y <- new_df$class
> set.seed(1)
> knn.predict <- knn(train = train.X,
```

```
+                     test = test.X,
+                     cl = train.y,
+                     k = 1)
> mean(knn.predict != test.y)
[1] 0
```

The test error is 0 with k = 1.

c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by KNN algorithm.

My R code:

```
> table(knn.predict, test.y)
           test.y
knn.predict bad good
       bad  126    0
       good   0  225
```

The confusion matrix is telling me that there are 126 are bad and 225 are good. It does not show any "bad-good" values and "good-bad" values, so all predict seem correct.

d) Split the data randomly into a training set (70%) and a test set (30%). Make sure to use set.seed(4323), for reproducible results. Fit the KNN model (K = 3).

My R code:

```
> set.seed(4323)
> n <- nrow(new_df)
> train <- sample(1:n, 0.7*n)
> train.X <- new_df[train, -34]; train.y <- new_df[train, 34]
> test.X <- new_df[-train, -34]; test.y <- new_df[-train, 34]
> set.seed(4323)
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
3)
> mean(knn.predict != test.y)
[1] 0.1886792
```

e) Repeat (d) using K = 5.

```
> set.seed(4323)
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
5)
> mean(knn.predict != test.y)
[1] 0.1792453
```

f) Repeat (d) using K = 7.

```
> set.seed(4323)
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
7)
> mean(knn.predict != test.y)
[1] 0.1981132
```

g) Which of these methods appears to provide the best results on this data?

K = 5 has the best result because it has the lowest error rate.

4. In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

a) Create a binary variable, mpg01 , that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

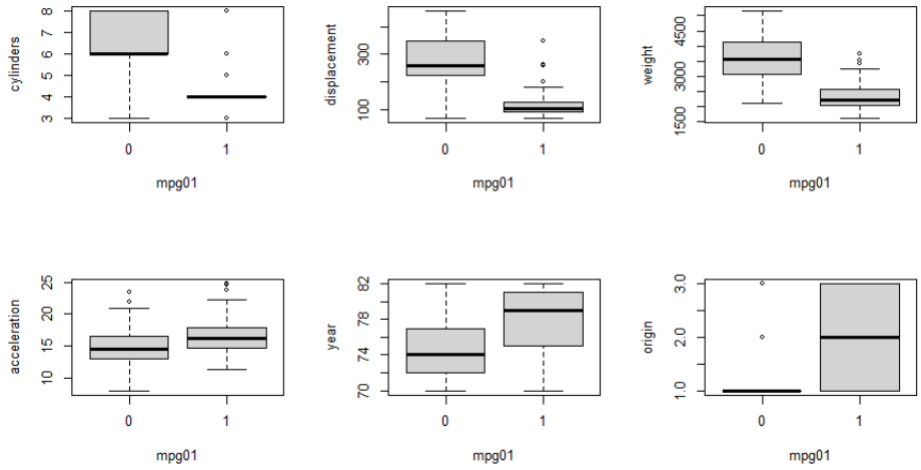
My R code:

```
> library(ISLR)
> View(Auto)
> mpg01 <- rep(0, length(Auto$mpg))
> mpg01[Auto$mpg > median(Auto$mpg)] <- 1
> new_df_auto <- data.frame(Auto, mpg01)
```

b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

My R code:

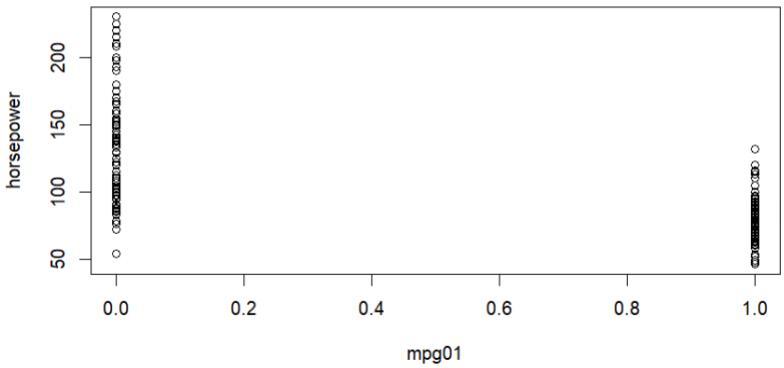
```
> par(mfrow = c(2, 3))
> boxplot(cylinders ~ mpg01, new_df_auto)
> boxplot(displacement ~ mpg01, new_df_auto)
> boxplot(weight ~ mpg01, new_df_auto)
> boxplot(acceleration ~ mpg01, new_df_auto)
> boxplot(year ~ mpg01, new_df_auto)
> boxplot(origin ~ mpg01, new_df_auto)
```



For some reason that "horsepower" column does not work on boxplot, so I do not include it.

Here is a scatterplot of mpg01 vs. horsepower

```
> plot(horsepower ~ mpg01, new_df_auto)
```



From observing these plots, I think displacement, weight, year, and horsepower seem to be useful in predicting mgp01. They all show a significant difference between 0 and 1. So, we can easily predict it to see how high and low from those relationship with mpg01.

c) Split the data randomly into a training set (70%) and a test set (30%). Make sure to use set.seed(1), for reproducible results.

My R code:

```
> set.seed(1)
> N <- nrow(new_df_auto)
> train <- sample(1:N, 0.7*N)
> my_feature <- c("displacement", "horsepower", "weight", "year")
> train.X <- new_df_auto[train, my_feature]
> test.X <- new_df_auto[-train, my_feature]
> train.y <- new_df_auto$mpg01[train]
> test.y <- new_df_auto$mpg01[-train]
```

d) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

My R code:

K = 1

```
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
1)
> mean(knn.predict != test.y)
[1] 0.1610169
```

K = 3

```
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
3)
> mean(knn.predict != test.y)
[1] 0.1440678
```

K = 5

```
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
5)
```

```
> mean(knn.predict != test.y)
[1] 0.1355932
```

K = 7

```
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
7)
> mean(knn.predict != test.y)
  [1] 0.1271186
```

K = 9

```
> knn.predict <- knn(train = train.X, test = test.X, cl = train.y, k =
9)
> mean(knn.predict != test.y)
[1] 0.1355932
```

The test errors that I obtained:

16.1% for k = 1

14.4% for k = 3

13.6% for k = 5

12.7% for k = 7

13.6% for k = 9

K = 7 seems perform the best on this data set because it has the lowest error rate.

e) Are the predictors you included into KNN model on the same scale? Proceed to scale the train & test data from parts (c)-(d) as we did in the lab. Only use the predictors that you claimed to be useful in explaining mpg01. Repeat part (d) for the scaled data.

No, the predictors that I included into KNN model are not the same scale.

My R code:

```
> train.X_scaled <- scale(train.X)
> test.X_scaled <- scale(test.X, center = attr(train.X_scaled, "scaled:
center"),
+                                scale = attr(train.X_scaled, "scaled:
scale"))
```

K = 1

```
> knn.predict <- knn(train = train.X_scaled, test = test.X_scaled, cl =
train.y, k = 1)
> mean(knn.predict != test.y)
[1] 0.04237288
```

K = 3

```
> knn.predict <- knn(train = train.X_scaled, test = test.X_scaled, cl =
train.y, k = 3)
> mean(knn.predict != test.y)
[1] 0.02542373
```

K = 5

```
> knn.predict <- knn(train = train.X_scaled, test = test.X_scaled, cl =
train.y, k = 5)
> mean(knn.predict != test.y)
[1] 0.04237288
```

K = 7

```
> knn.predict <- knn(train = train.X_scaled, test = test.X_scaled, cl =
train.y, k = 7)
> mean(knn.predict != test.y)
[1] 0.03389831
```

K = 9

```
> knn.predict <- knn(train = train.X_scaled, test = test.X_scaled, cl =
train.y, k = 9)
> mean(knn.predict != test.y)
[1] 0.03389831
```

K = 3 has the best test error after scaling the data which is 2.5%

5. For problem 4, instead of implementing validation set approach, proceed to use leave-one-out cross-validation (function knn.cv()). Run it for K = 1, 3, 10 and compare the resulting CV errors. Use all observations of Auto data set for relevant predictors, not just the ”training subset” (as we are not doing any train/test subdivision here).

a) Run set.seed(1) command prior to each cv.knn() call, for uniformity of the results.
My R code:

```
> for(K in c(1, 3, 10)){
+   set.seed(1)
+   knn.predict <- knn.cv(train = new_df_auto[,c("displacement", "horse
power", "weight", "year")],
+                         cl = new_df_auto$mpg01, k = K)
+ print(mean(knn.predict != new_df_auto$mpg01))
+ }
[1] 0.127551
[1] 0.119898
[1] 0.130102
```

b) First do it for data in its original form (as you had in 5(d), no scaling). What are the test errors? Which method wins? Show the code for just one of K values (e.g. K = 3).
My R code:

```
> for(K in c(1, 3, 10)){
+   set.seed(1)
+   knn.predict <- knn.cv(train = new_df_auto[,c("displacement", "horse
power", "weight", "year")],
+                         cl = new_df_auto$mpg01, k = K)
+ print(mean(knn.predict != new_df_auto$mpg01))
+ }
[1] 0.127551
[1] 0.119898
[1] 0.130102
```

12.8% for k = 1, 12% for k = 3, and 13.1% for k = 10
Overall, the LOOCV wins for this case because these test errors are lower than the validation set approach results.

c) Then do it for scaled data (as you had in 5(e)). What are the test errors? Which method wins? Show the code for just one of K values (e.g. K = 3).

My R code:

```
for(K in c(1, 3, 10)){
+   set.seed(1)
+   knn.predict <- knn.cv(train = scale(new_df_auto[,c("displacement",
"horsepower", "weight", "year")]),
+                         cl = new_df_auto$mpg01, k = K)
+   print(mean(knn.predict != new_df_auto$mpg01))
+ }
[1] 0.06122449
[1] 0.05357143
[1] 0.07142857
```

6.1% for k = 1, 5.4% for k = 3, and 7.1% for k = 10

Overall, the LOOCV wins for this case because these test errors are lower than the validation set approach results.

d) Which results should we trust more - validation set approach from problem 5? Or CV results here? Why?

I think we should trust the CV method more because it has lower test error and more stable results.