STATS 790 Assignment 1

Yiran Zhang
400119421

20 January, 2023

Question 1

Breiman (2001) criticized how the statisticians made assumptions on certain data models when solving data problems. In my opinion, making assumptions itself is a dangerous act in the field of science, because these assumptions are made for simpler analysis but the majority of the time in real world, none of these assumptions holds. For example, when I studied physics earlier in my undergrad, most of the problems referred to "negligible friction" or "negligible air resistance", which is never negligible in real life. In statistics, indeed, most of the data models have assumptions like linear relationship or a certain distribution, and we should all be aware that those assumptions are not reliable at first place. However, I am also aware that some algorithmic models also have assumptions, for example, Naive Bayes Classifier assumes the features are independent. Does it mean that algorithmic models may also be unreliable in some sense?

It's interesting that Breiman mentioned that algorithmic models were popular in industries in 2001. From my work experience, the industry still relies on algorithms for data analysis today. I'm also glad to see that schools are starting to offer more opportunities to learn about algorithmic models so that more statistical problems can be solved from a different perspective.

Question 2

(I discussed this question with Amandeep Sidhu.)

I chose to reproduce Figure 2.1 in ESL. Below is my R Code:

```r
library(MASS) # multivariate normal random sample generator
library(dplyr)

set.seed(400119421)
# Generate 10 means from multivariate normal distribution
# for blue and orange respectively
blue <- mvrnorm(n=10,mu=c(1,0),Sigma=matrix(c(1, 0, 0, 1), ncol=2))
orange <- mvrnorm(n=10,mu=c(0,1),Sigma=matrix(c(1, 0, 0, 1), ncol=2))

# Draw 100 samples from the list above with probability 1/10
# The idea here is to draw 100 samples from the index, and obtain samples from
# blue/orange using the 100 index.
blue_sample <- sample(nrow(blue), 100, replace=TRUE, prob=rep(0.1, 10))
blue_means <- blue[blue_sample, ]

orange_sample <- sample(nrow(orange), 100, replace=TRUE, prob=rep(0.1, 10))
orange_means <- orange[orange_sample, ]

# Generate observations for blue and orange using the means sampled above
blue_obs <- matrix(ncol=2, nrow=100) # to store the observations from blue means
for (i in 1:100){
  blue_obs[i,] <- mvrnorm(n=1, mu=blue_means[i,],
                          Sigma=matrix(c(1/5, 0, 0, 1/5), ncol=2))
} # for each pair of means, generate one pair of multivariate normal random sample

# same idea for orange observations
orange_obs <- matrix(ncol=2, nrow=100)
for (i in 1:100){
  orange_obs[i,] <- mvrnorm(n=1, mu=orange_means[i,],
                            Sigma=matrix(c(1/5, 0, 0, 1/5), ncol=2))
}
```

```r
# Create dataset by combining blue and orange observations
# Transfer into data frame type to fit linear model
blue_orange <- as.data.frame(rbind(blue_obs, orange_obs))

# Linear regression
linear_model <- lm(V2 ~ V1, data=blue_orange)

b0 <- linear_model$coefficients[1] # the coefficient beta_0
b1 <- linear_model$coefficients[2] # the coefficient beta_1

# Fitted linear regression function
linear <- function(x) b0+b1*x

# Generate the background dots in the figure
x <- seq(from = -1.5, to = 4, by = 0.05)
y <- seq(from = -2, to = 3.5, by = 0.05)
background <- expand.grid(x,y) # entire background
y_hat <- linear(x)
# y_hat is the filter for the linear regression line
# that separate blue and orange dots on the background

blue_pt <- (background %>% filter(Var2 < y_hat)) # lower part is blue dots
orange_pt <- (background %>% filter(Var2 > y_hat)) # upper part is orange dots

# Figure reproduction
plot(blue_obs
     , col='deepskyblue' # set point color to blue
     , xlim = c(-1.5,4) # expand x axis to fit orange observations later
     , ylim = c(-2,3.5) # expand y axis to fit orange observations later
     , xlab='', ylab='' # clear x and y labels
     , lwd=3, cex = 1.5 # increase size and thickness of the observations
```

```
     , xaxt='n', yaxt='n') # clear the x and y axis number labels
points(orange_obs, col='orange', lwd = 3, cex = 1.5) # add orange observations
abline(linear_model, col = 'black', lwd = 3) # add the decision boundary
points(orange_pt, col='orange', pch=20, cex = 0.3) # add orange background dots
points(blue_pt, col='deepskyblue', pch=20, cex = 0.3) # add blue background dots
```

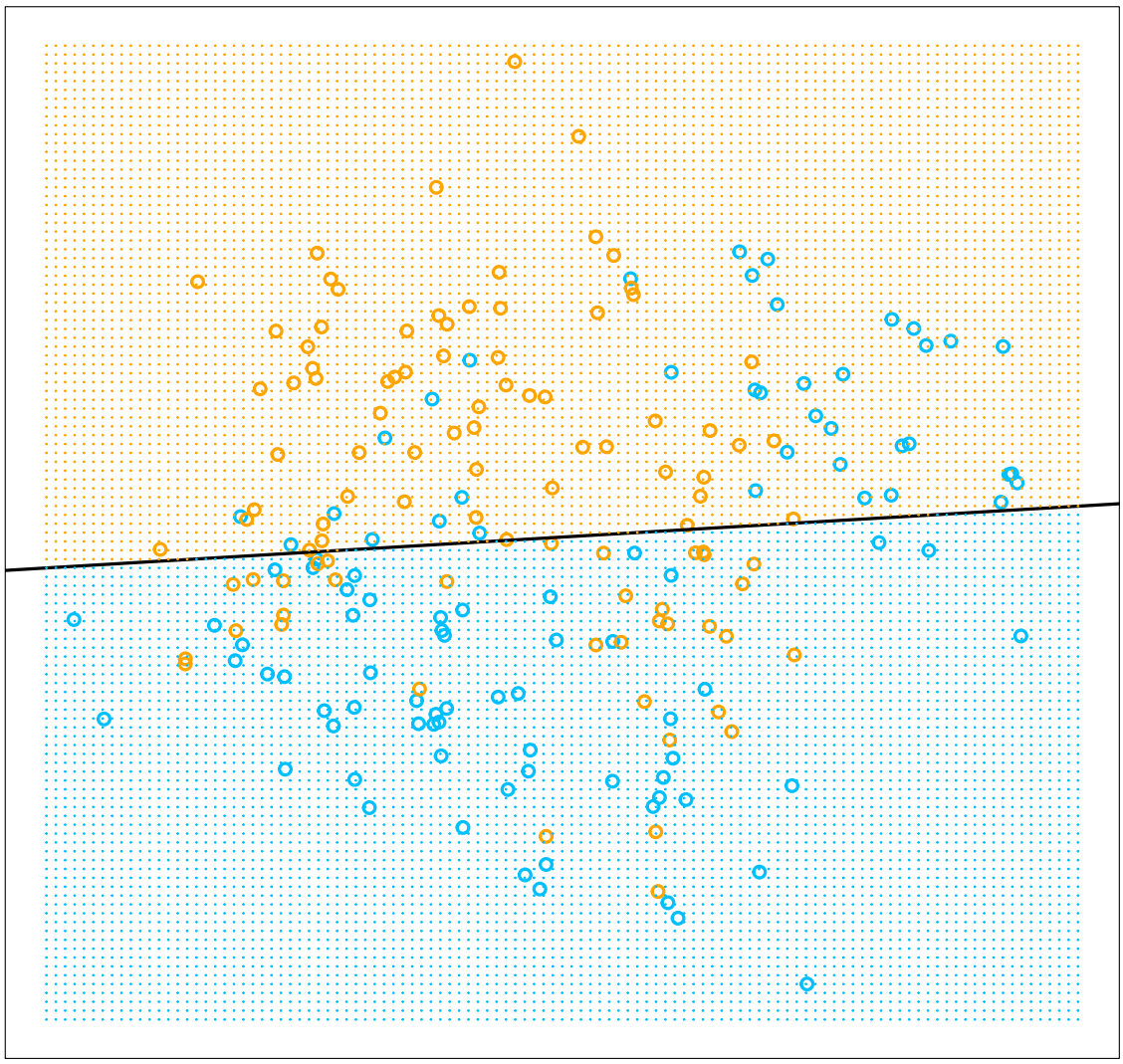

Figure 1: Replication of figure 2.1 in ESL

Question 3

$$\begin{aligned}
\mathrm{E}[|Y-m|] &= \mathrm{E}[Y-m \mid m \le Y < \infty] + \mathrm{E}[m-Y \mid -\infty < Y \le m] \\
&= \int_m^{\infty} (Y-m) f_Y(y) dy + \int_{-\infty}^{m} (m-Y) f_Y(y) dy \\
&= \int_m^{\infty} Y f_Y(y) dy + \int_m^{\infty} (-m) f_Y(y) dy + \int_{-\infty}^{m} m f_Y(y) dy + \int_{-\infty}^{m} (-Y) f_Y(y) dy
\end{aligned}$$

Using the similar idea as MSE example in ADA textbook, we take the first derivative of the above formula with respect to m, set it to 0 and find the optimal value that minimizes MAE.

$$\begin{aligned}
\frac{\partial \mathrm{E}[|Y-m|]}{\partial m} &= \frac{\partial}{\partial m} \int_m^{\infty} Y f_Y(y) dy + \frac{\partial}{\partial m} \int_m^{\infty} (-m) f_Y(y) dy + \frac{\partial}{\partial m} \int_{-\infty}^{m} m f_Y(y) dy + \frac{\partial}{\partial m} \int_{-\infty}^{m} (-Y) f_Y(y) dy \\
&= \int_m^{\infty} \frac{\partial}{\partial m} Y f_Y(y) dy + \int_m^{\infty} \frac{\partial}{\partial m} (-m) f_Y(y) dy + \int_{-\infty}^{m} \frac{\partial}{\partial m} m f_Y(y) dy + \int_{-\infty}^{m} \frac{\partial}{\partial m} (-Y) f_Y(y) dy \\
&= 0 + \int_m^{\infty} (-1) f_Y(y) dy + \int_{-\infty}^{m} 1 * f_Y(y) dy + 0 \\
&= -\int_m^{\infty} f_Y(y) dy + \int_{-\infty}^{m} f_Y(y) dy \\
&= 0
\end{aligned}$$

The above equations imply:

$-\int_m^{\infty} f_Y(y) dy + \int_{-\infty}^{m} f_Y(y) dy = 0$

$\int_m^{\infty} f_Y(y) dy = \int_{-\infty}^{m} f_Y(y) dy$

which means $P(Y \le m) = P(Y > m)$, and we know that $P(Y \le m) + P(Y > m) = 1$, hence $P(Y \le m) = P(Y > m) = 0.5$, which means m is the median of Y.

To conclude, the MAE is minimized by taking $\tilde{\mu} = median(Y)$, it is no longer the mean value. We learned from previous statistics courses that the median is more robust than the mean value, because it will not affected by the outliers, while the mean is very sensitive to the outliers. We should be using MAE if there are outliers detected, we can be indifferent between MAE and MSE if there is no outliers.

Question 4

According to the textbook ADA, the formula for linear smoother is $\sum_i y_i \hat{w}(x_i, x)$, and the formula for global mean is $\frac{\sum_i y_i}{n} = \sum_i \frac{y_i}{n}$. Match these two formulas:

$\sum_i y_i \hat{w}(x_i, x) = \sum_i \frac{y_i}{n}$,

Intuitively, $\hat{w}(x_i, x) = \frac{1}{n}$. The above formula implies that for each i, the weight is $\frac{1}{n}$ (i.e., each entry of w is $\frac{1}{n}$).

To further verify it, according to equation 1.69 on page 38 from ADA, the linear smoother can be transformed into a matrix form of: $\hat{\mu} = wy$, where w is n by n matrix, and y is n by 1 vector. Then the linear smoother is of the form:

$$\begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \vdots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \frac{y_1}{n} + \frac{y_2}{n} + \cdots + \frac{y_n}{n} \\ \frac{y_1}{n} + \frac{y_2}{n} + \cdots + \frac{y_n}{n} \\ \vdots \\ \frac{y_1}{n} + \frac{y_2}{n} + \cdots + \frac{y_n}{n} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n \frac{y_i}{n} \\ \sum_{i=1}^n \frac{y_i}{n} \\ \vdots \\ \sum_{i=1}^n \frac{y_i}{n} \end{bmatrix}$$

The final vector at the right hand side is the same formula as global mean. Hence we can conclude that the influence matrix is a n by n matrix with all $\frac{1}{n}$.

Using the formula 1.70 in ADA, the degree of freedom = tr(w) = $\frac{1}{n} + \cdots + \frac{1}{n}$ = n*$\frac{1}{n}$ = 1, as desired.

Question 5

According to the textbook ESL Chapter 2, the formula for a KNN regression is $\hat{\mu}(x) = \frac{1}{k} \sum_{x_i \in N_k} y_i$, where N_k is the area defined by the k nearest neighbors of x. Fit this formula into the linear smoother formula, we get that $w(x_i, x) = \frac{1}{k}$, implying that the weight is $\frac{1}{k}$ if x_i is within the neighborhood, and also note that the weight will be 0 if it's not inside the neighborhood. In matrix, the influence matrix w = (w_{ij}), 1 ≤ i,j ≤ n, where $w_{ii} = \frac{1}{k}$ for all i (because x_i is definitely inside its own neighborhood), and for $i \neq j$, $w_{ij} = \frac{1}{k}$ if the observation is inside the neighborhood of x_i, 0 if not. In other words, the influence matrix w is a matrix with diagonal of all $\frac{1}{k}$'s, and the off-diagonal entries are either 0 or $\frac{1}{k}$ depending if the observation is inside the neighborhood. Also note that, there will be only k numbers of $\frac{1}{k}$ each row (so n-k 0's), so each row adds up to 1.

To verify using the hint, we reduce to the case where k = n (i.e., consider all observations as the neighbor), then $a_{ij} = 1$ for all i and j, every entry takes value 1, so the influence matrix w is an n by n matrix with all $\frac{1}{k}$ (or $\frac{1}{n}$ since k = n). This is the same formula as the global mean in the above question, and this is reasonable because considering all the observations and put equal weights on them is exactly the idea of global mean, hence it is correct.

Using the same definition of 1.70 in ADA, we get that the degree of freedom = tr(w) = $\frac{1}{k} + \ldots + \frac{1}{k} = \frac{n}{k}$ because there is n $\frac{1}{k}$'s on the diagonal of that influence matrix w.

Question 6

```
library(dplyr)
library(class) # KNN function

# Import dataset
digits_train <- read.table('zip.train.gz') %>%
  filter(V1 == 2 | V1 == 3) # Train set focusing on digits 2 and 3
digits_test <- read.table('zip.test.gz') %>%
  filter(V1 == 2 | V1 == 3) # Test set

train_x <- digits_train[ ,-1] # Predictors for the train set
train_y <- digits_train[ ,1] # Target for the train set

test_x <- digits_test[ ,-1] # Predictors for the test set
test_y <- digits_test[ ,1] # Target for the test set

# Linear regression
digits_linear <- lm(V1 ~., data = digits_train) # Linear regression model

## Training error calculation ##
linear_pred_train <- predict(digits_linear, digits_train) # Prediction on train set
# modify predicted values, if greater than 2.5 then set to 3, else 2.
```

```r
linear_pred_train1 <- ifelse(linear_pred_train < 2.5, 2, 3)
linear_train_error <- mean(linear_pred_train1 != train_y)
linear_train_error # Training error
```

```
## [1] 0.005759539
```

```r
## Testing error calculation ##
linear_pred_test <- predict(digits_linear, digits_test) # Prediction on test set
linear_pred_test1 <- ifelse(linear_pred_test < 2.5, 2, 3) # modify predicted values
linear_test_error <- mean(linear_pred_test1 != test_y)
linear_test_error # Testing error
```

```
## [1] 0.04120879
```

```r
# K-Nearest Neighbor
# K = 1
knn1_pred_train <- knn(train_x, train_x, train_y, k = 1) # Predict on train
knn1_train_error <- mean(knn1_pred_train != train_y)
knn1_train_error # Training error
```

```
## [1] 0
```

```r
knn1_pred_test <- knn(train_x, test_x, train_y, k = 1) # Predict on test
knn1_test_error <- mean(knn1_pred_test != test_y)
knn1_test_error # Testing error
```

```
## [1] 0.02472527
```

```r
# K = 3
knn3_pred_train <- knn(train_x, train_x, train_y, k = 3) # Predict on train
knn3_train_error <- mean(knn3_pred_train != train_y)
knn3_train_error # Training error
```

```
## [1] 0.005039597
```

```
knn3_pred_test <- knn(train_x, test_x, train_y, k = 3) # Predict on test
knn3_test_error <- mean(knn3_pred_test != test_y)
knn3_test_error # Testing error
```

```
## [1] 0.03021978
```

```
# K = 5
knn5_pred_train <- knn(train_x, train_x, train_y, k = 5) # Predict on train
knn5_train_error <- mean(knn5_pred_train != train_y)
knn5_train_error # Training error
```

```
## [1] 0.005759539
```

```
knn5_pred_test <- knn(train_x, test_x, train_y, k = 5) # Predict on test
knn5_test_error <- mean(knn5_pred_test != test_y)
knn5_test_error # Testing error
```

```
## [1] 0.03021978
```

```
# K = 7
knn7_pred_train <- knn(train_x, train_x, train_y, k = 7) # Predict on train
knn7_train_error <- mean(knn7_pred_train != train_y)
knn7_train_error # Training error
```

```
## [1] 0.006479482
```

```
knn7_pred_test <- knn(train_x, test_x, train_y, k = 7) # Predict on test
knn7_test_error <- mean(knn7_pred_test != test_y)
knn7_test_error # Testing error
```

```
## [1] 0.03296703
```

```r
# K = 15
knn15_pred_train <- knn(train_x, train_x, train_y, k = 15) # Predict on train
knn15_train_error <- mean(knn15_pred_train != train_y)
knn15_train_error # Training error
```

```
## [1] 0.009359251
```

```r
knn15_pred_test <- knn(train_x, test_x, train_y, k = 15) # Predict on test
knn15_test_error <- mean(knn15_pred_test != test_y)
knn15_test_error # Testing error
```

```
## [1] 0.03846154
```

```r
# Comparison among all the models
# Build a matrix with training error and testing error on column
# and each method on the row
comparison <- matrix(c(linear_train_error, linear_test_error, knn1_train_error,
                       knn1_test_error, knn3_train_error, knn3_test_error,
                       knn5_train_error, knn5_test_error, knn7_train_error,
                       knn7_test_error, knn15_train_error, knn15_test_error),
                     ncol=2, byrow=TRUE)
# Set column names
colnames(comparison) <- c('Training Error','Testing Error')
# Set row names
rownames(comparison) <- c('Linear Regression','KNN-1','KNN-3','KNN-5','KNN-7','KNN-15')
comparison <- as.table(comparison) # transfer to table format
```

Table 1: Comparison between training error and testing error from each model

	Training Error	Testing Error
Linear Regression	0.0057595	0.0412088
KNN-1	0.0000000	0.0247253
KNN-3	0.0050396	0.0302198
KNN-5	0.0057595	0.0302198
KNN-7	0.0064795	0.0329670
KNN-15	0.0093593	0.0384615

From the comparison table above, we can see that KNN models with relatively small k value will have a better performance than the linear regression model. Among all the KNN models, both the training error rate and testing error rate increases as value of k increases.