Question 1 (3 marks) Suppose we collect data for a group of students that have taken STAT318 with variables X1 = hours spent studying per week, X2 = number of classes attended and Y = (1 if the student received a GPA value ≥ 7 in STAT318, 0 otherwise. We fit a logistic regression model and find the estimated coefficients to be $\hat{\beta}_0 = -16$, $\hat{\beta}_1 = 1.6$and $\hat{\beta}_2 = 0.23$.

(a) Write explicitly the equation for (1) the probability of receiving a GPA ≥ 7 and (2) the log-odds of receiving a GPA ≥ 7 as a function of the model variables. Explain what the model coefficient tell about the relationship between variables and GPA.

a.) (1) $Pr(GPA \geq 7|x) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)} = p(x)$

(2) $\log\left(\frac{p(x)}{1-p(x)}\right) = \log\left(\frac{Pr(GPA \geq 7|x)}{1 - Pr(GPA \geq 7|x)}\right) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$

$\hat{\beta}_0$: log odds of student getting a GPA ≥ 7 when X_1 and $X_2 = 0$.

$\hat{\beta}_1$: every 1 hour additional spent studying per week increases the odds of getting a GPA ≥ 7 by $\exp(\hat{\beta}_1)$, keeping $\hat{\beta}_0$ constant.

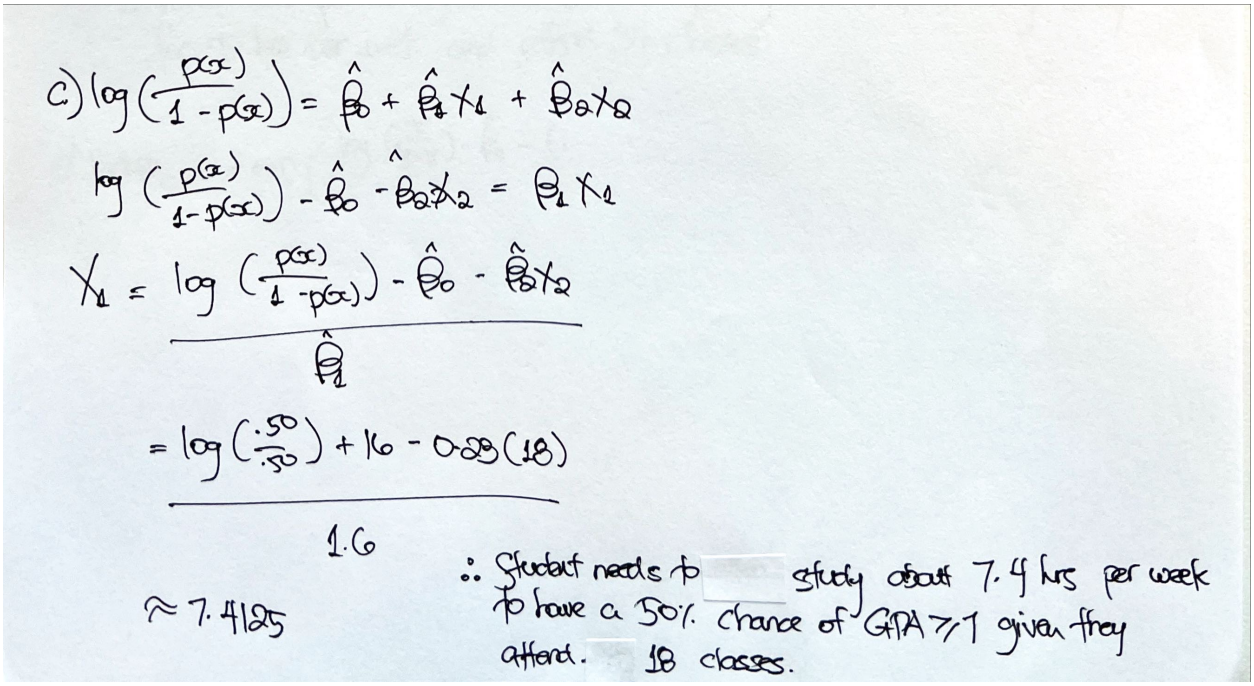
$\hat{\beta}_2$: every 1 additional class attended increases the odds of getting GPA ≥ 7 by $\exp(\hat{\beta}_2)$. keeping $\hat{\beta}_1$ constant.

(b) Estimate the probability of a student getting a GPA value ≥ 7 in STAT318 if they study for 5 hours per week and attend all 34 classes.

b.) $Pr(GPA \geq 7|X=x) = p(x) = \frac{\exp(-16 + 1.6(5) + 0.23(34))}{1 + \exp(-16 + 1.6(5) + 0.23(34))} \approx 0.4551$

∴ About 46% probability of a student getting GPA ≥ 7 if they study for 5 hrs per week and attend 34 classes.

(c) If a student attends 18 classes, how many hours do they need to study per week to have a 50% chance of getting a GPA value ≥ 7 in STAT318?



Question 2. (4 marks) Describe one pro and one con of flexible (versus a less flexible) approach for regression. Under what conditions might a less flexible approach be preferred?

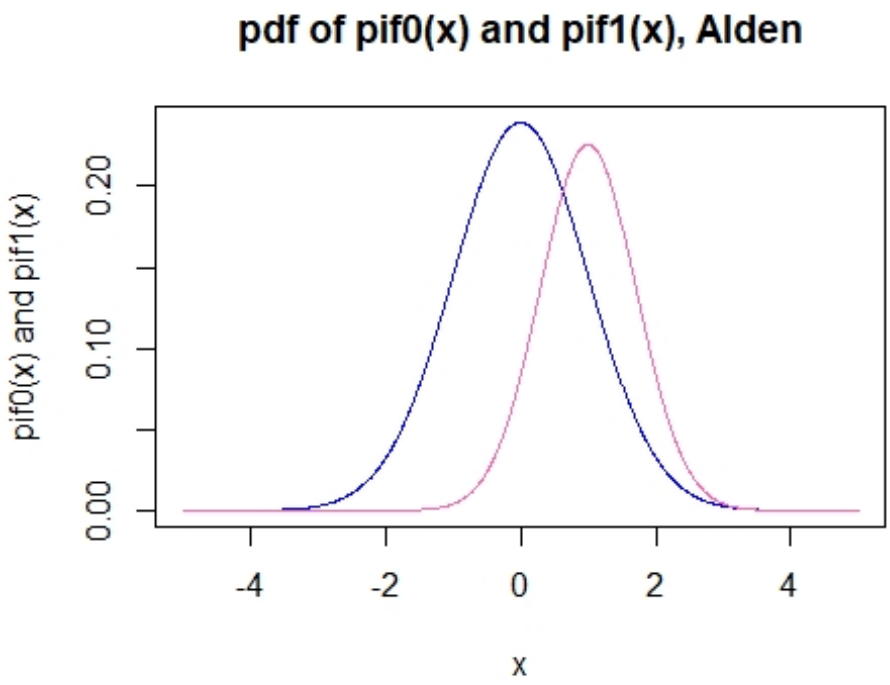
A pro of flexible approach for regression is it may give a good fit for non-linear models which reduces bias. A con of flexible approach for regression is that it needs more parameters to estimate which increases variance.

A less flexible approach is preferred if the sample size is very small and if we are interested in inference of the results.

Question 3. (6 marks) Consider a binary classification problem Y ∈ {0, 1} with one predictor X. The prior probability of being in class 0 is Pr(Y = 0) = π0 = 0.6 and the density function for X in class 0 is a standard normal f0(x) = Normal(0, 1) The density function for X in class 1 is also normal, but with μ = 1 and σ 2 = 0.5 f1(x) = Normal(1, 0.5)

(a) Plot π0f0(x) and π1f1(x) in the same figure.

```
x <- seq(-5,5, by=0.01)
f0 <- 0.6*(1/sqrt(2*pi*1))*exp(-(x^2)/2)
f1 <- 0.4*(1/sqrt(2*pi*0.5))*exp(-x^2+2*x-1)
plot(x,f0, xlim= c(-5,5), type="l", col="blue", main="pdf of pif0(x) and
pif1(x), Alden", ylab="pif0(x) and pif1(x)")
lines(x,f1, col="hotpink")
```


pdf of pif0(x) and pif1(x), Alden
pif0(x) and pif1(x)
0.00
0.10
0.20
-4
-2
0
2
4
x

(b) Find the Bayes decision boundary (Hint: you will need to solve the equation π0f0(x) = π1f1(x), which defines the boundary. Preferibly, do this by using some mathematics, rather than by numerical approximation).

(b.)

$$\pi_0 f_0(x) = \pi_1 f_1(x)$$

$$\frac{0.6}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) = \frac{0.4}{\sqrt{\pi}} \exp\left(-(x-1)^2\right)$$

$$\frac{0.6 \cdot \sqrt{\pi}}{0.4 \cdot \sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) = \exp\left(-(x-1)^2\right)$$

$$\left(\frac{0.6}{0.4}\right)\left(\frac{\sqrt{\pi}}{\sqrt{2\pi}}\right) \exp\left(-x^2/2\right) = \exp\left(-(x-1)^2\right)$$

$$\left(\frac{0.6}{0.4}\right)\left(\frac{1}{2}\right) \exp\left(-x^2/2\right) = \exp\left(-(x-1)^2\right)$$

↓ x by ln to get rid of exp

$$\ln\left(\frac{3}{4}\right) - \frac{x^2}{2} = -x^2 + 2x - 1$$

$$-\frac{1}{2}x^2 + 2x - 1 = \ln(3/4)$$

↓ x-2 to make the x^2 whole

$$x^2 - 4x + 2 = -2\ln(3/4)$$

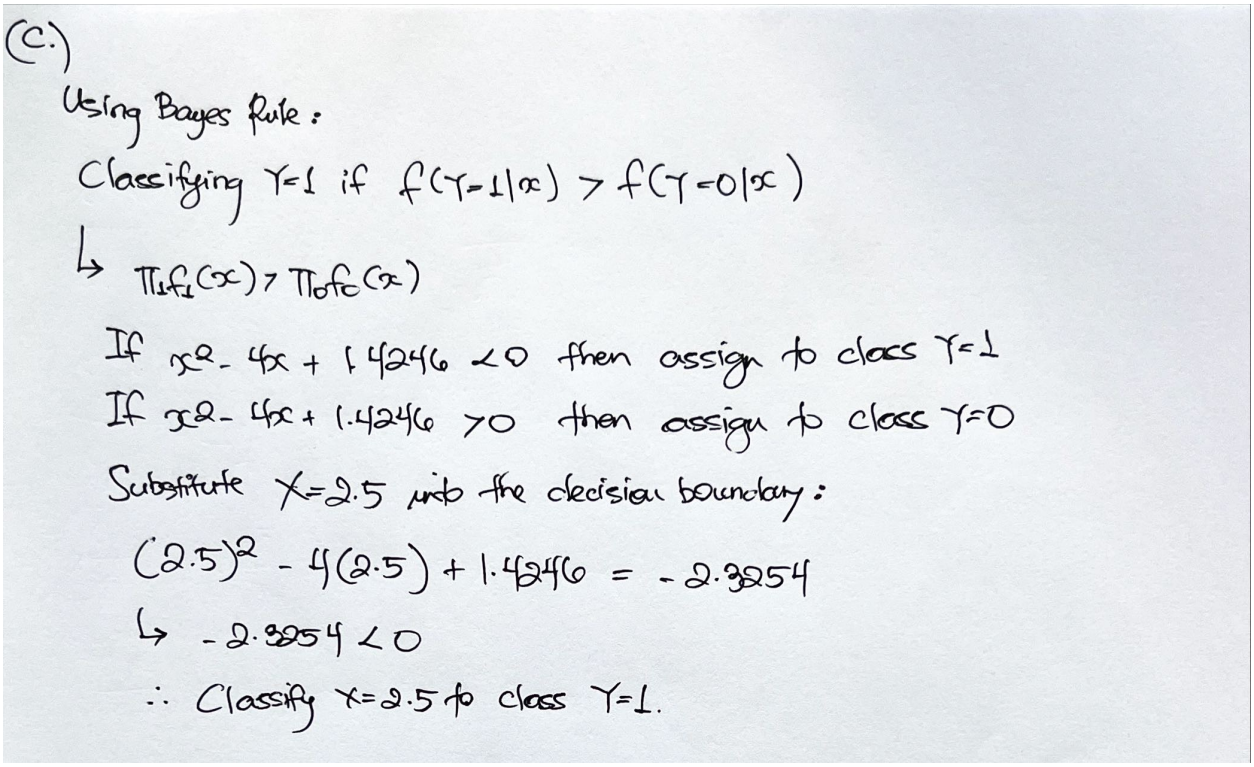
$$x^2 - 4x + 2 + 2\ln(3/4) = 0$$

$$x^2 - 4x + 2 - 0.575364 1449 = 0$$

$$x^2 - 4x + 1.424635855 = 0$$

decision boundary : $x^2 - 4x + 1.4246 = 0$

(c) Using Bayes classifier, classify the observation X = 2.5. Justify your prediction.



(c.)

Using Bayes Rule:

Classifying $Y=1$ if $f(Y=1|x) > f(Y=0|x)$

↳ $\pi_1 f_1(x) > \pi_0 f_0(x)$

If $x^2 - 4x + 1.4246 < 0$ then assign to class $Y=1$

If $x^2 - 4x + 1.4246 > 0$ then assign to class $Y=0$

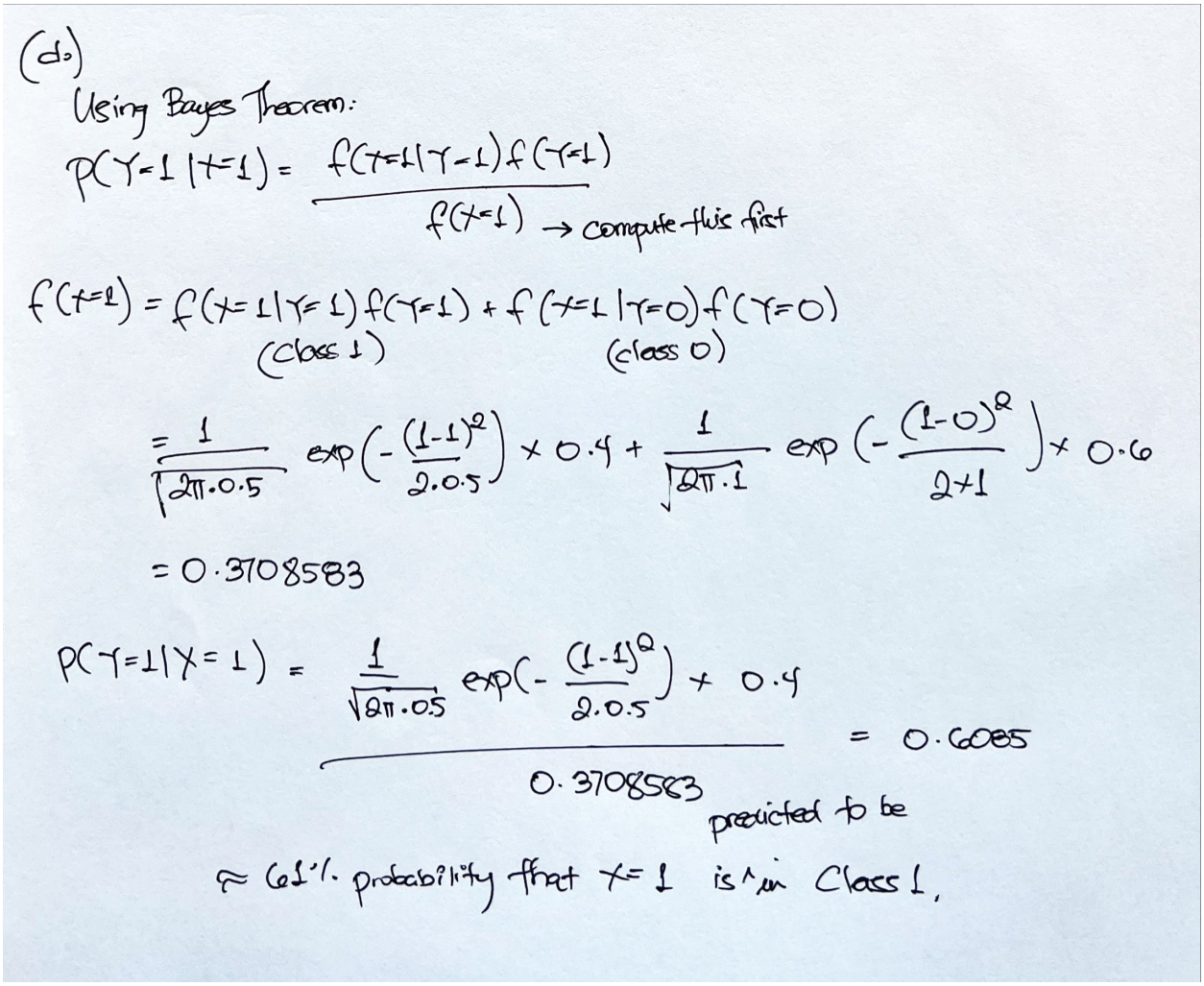
Substitute $X=2.5$ into the decision boundary:

$(2.5)^2 - 4(2.5) + 1.4246 = -2.3254$

↳ $-2.3254 < 0$

∴ Classify $x=2.5$ to class $Y=1$.

(d) What is the probability that an observation with X = 1 is in class 1?



Question 4 (8 marks) 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, 4, 8, 16, 32, 64 and 128, (learning from the training data) and compute the training and testing MSE for each value of k. Do take care in using the right observed y-values (testing or training) for estimating testing and training MSE.

```
auto_train <- read.csv("AutoTrain.csv")
auto_test <- read.csv("AutoTest.csv")
x.train <- auto_train$horsepower
y.train <- auto_train$mpg
x.pred <- seq(46,230,length=196)

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

```
## 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)
}

print("training MSE") ## [1] "training MSE"

print('For k=2'## [1] "For k=2"

k = 2
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE

## [1] 141.5267

print("For k=4") ## [1] "For k=4"

k = 4
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE

## [1] 138.9665

print("For k=8") ## [1] "For k=8"

k = 8
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE

## [1] 134.2644

print("For k=16") ## [1] "For k=16"

k = 16
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE

## [1] 133.2156

print("For k=32") ## [1] "For k=32"
```

```r
k = 32
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE
```

```
## [1] 131.0171
```

```r
print("For k=64") ## [1] "For k=64"
```

```r
k = 64
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE
```

```
## [1] 117.7905
```

```r
print("For k=128") ## [1] "For k=128"
```

```r
k = 128
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((y.train - kNN.vals)^2)
MSE
```

```
## [1] 85.5368
```

```r
print("testing MSE") ## [1] "testing MSE"
```

```r
print('For k=2') ## [1] "For k=2"
```

```r
k = 2
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE
```

```
## [1] 174.0914
```

```r
print("For k=4") ## [1] "For k=4"
```

```r
k = 4
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE
```

```
## [1] 170.1967
```

```r
print("For k=8") ## [1] "For k=8"
```

```r
k = 8
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE
```

```
## [1] 165.9041
```

```r
print("For k=16") ## [1] "For k=16"

k = 16
kNN.vals_16 <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE

## [1] 165.9041

print("For k=32") ## [1] "For k=32"

k = 32
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE

## [1] 164.1646

print("For k=64") ## [1] "For k=64"

k = 64
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE

## [1] 146.348

print("For k=128") ## [1] "For k=128"

k = 128
kNN.vals <- kNN(k, x.train, y.train, x.pred)
MSE <- mean((auto_test$mpg - kNN.vals)^2)
MSE

## [1] 98.53572
```

(b) Which value of k performed best? Explain.

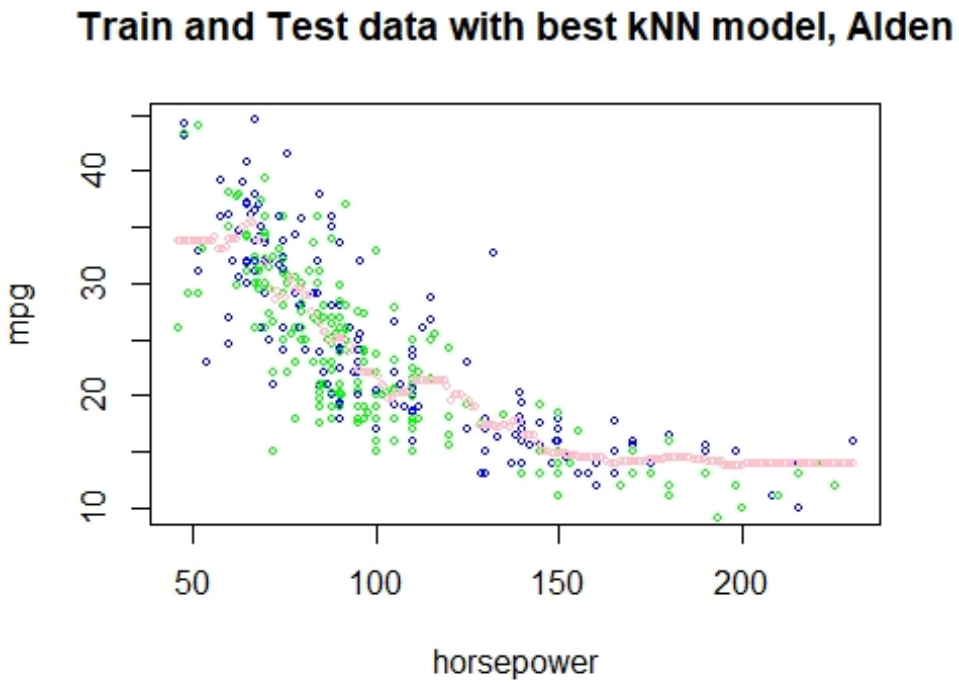
We can't choose a very low value of k (e.g. k=2) as they tend to overfit the structure of the data and usually follow the errors. We can't also choose a high value of k as they tend to underfit the structure of the data losing the overall feature of the data. This implies that we need to choose a middle k value (which minimises the testing error rate).

The k that performed best is the square root of N (196 observations) which sqrt(196) = 14. The k that performed best is k = 16 as it is the closest to 14. Also, its testing MSE is 165.2168 which is in the middle meaning it reduces the risk of overfitting and underfitting.

(c) Plot the training data, testing data and the BEST KNN MODEL in the same figure. Include your name somewhere in the plot. (The points() function is useful to plot the kNN model because it is discontinuous.)

```r
plot(auto_train$horsepower,
    auto_train$mpg,
```

```
    col="blue",
    cex=0.6,
    xlab = "horsepower",
    ylab = "mpg",
    main = "Train and Test data with best kNN model, Alden")
points(auto_test$horsepower,
    auto_test$mpg,
    col="green",
    cex=0.6)
points(x.pred,
       kNN.vals_16,
       col="pink",
       cex=0.6)
```



(d) Describe the bias-variance trade-off for kNN regression and how it is influenced by the choice of k.

The bias-variance trade-off for kNN regression occurs when bias decreases and the variance increases as model flexibility increases. In contrast, bias increases and variance decreases as model flexibility decreases.

The choice of k influences the model flexibility. For example, if k is too small then the model is overfitting as it is following the errors too closely making predictions unstable. This typically means less bias and high variance. If k is too large then it is underfitting as it is accounting for more neighbours and losing the structure of the data. This typically means high bias and less variance.

We would need to find k that is optimal. That is when changes to bias-variance trade-off and the testing error rate is minimised.

Question 5. (10 marks) In this question, you will fit a logistic regression model to predict the probability of a banknote being forged using the Banknote data set. This data has been divided into training and testing sets: BankTrain.csv and BankTest.csv (download these sets from Learn). The response variable is y (the fifth column), where y = 1 denotes a forged banknote and y = 0 denotes a genuine banknote. Although this data set has four predictors, you will be using x1 and x3 to fit your model1.

(a) Perform multiple logistic regression using the training data. Comment on the model obtained.

```
banktrain <- read.csv("BankTrain.csv")
banktest <- read.csv("BankTest.csv")
banktrain$y <- as.factor(banktrain$y)
banktest$y <- as.factor(banktest$y)
head(banktrain)
```

```
##        x1          x2      x3       x4 y
## 1  1.9340 -0.00000928  4.8160 -0.33967 0
## 2  4.4338  9.88700000 -4.6795 -3.74830 0
## 3 -2.9020 -7.65630000 11.8318 -0.84268 1
## 4  3.4776  8.81100000 -3.1886 -0.92285 0
## 5  3.3397 -4.61450000  3.9823 -0.23751 0
## 6  3.2585 -4.46140000  3.8024 -0.15087 0
```

```
dim(banktest)
```

```
## [1] 412   5
```

```
glm.fit = glm(y~x1+x3,data=banktrain,family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x3, family = binomial, data = banktrain)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22041    0.11206   1.967   0.0492 *
## x1          -1.31489    0.08822 -14.905  < 2e-16 ***
## x3          -0.21738    0.02880  -7.548 4.42e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1322.01  on 959  degrees of freedom
```

```
## Residual deviance:  572.07  on 957  degrees of freedom
## AIC: 578.07
##
## Number of Fisher Scoring iterations: 6
```

All the variables are statistically significant as their p-values are less than 0.05. The intercept is the log odds of a banknote to be forged when x1 and x3 are zero. x1: every one unit of change increases the odds of a note being forged by exp(x1). Keeping x3 constant. x3: every one unit of change increases the odds of a note being forged by exp(x3). Keeping x1 constant.

(b) Suppose we classify observations using f(x) = (forged banknote if Pr(Y = 1|X = x) > θ , genuine banknote otherwise.

i. Plot the training data (using a different symbol for each class) and the decision boundary for θ = 0.5 on the same figure.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.1

## Warning: package 'ggplot2' was built under R version 4.4.1

## Warning: package 'tibble' was built under R version 4.4.1

## Warning: package 'tidyr' was built under R version 4.4.1

## Warning: package 'readr' was built under R version 4.4.1

## Warning: package 'purrr' was built under R version 4.4.1

## Warning: package 'dplyr' was built under R version 4.4.1

## Warning: package 'forcats' was built under R version 4.4.1

## Warning: package 'lubridate' was built under R version 4.4.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── ──
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
b = coef(glm.fit)
slope = -b[2]/b[3]
intercept = -b[1]/b[3]

banktrain %>%
  ggplot(aes(x1, x3, group = y))+
  geom_point(aes(shape = y, color = y))+
  geom_abline(aes(intercept = intercept, slope=slope), color = 'purple')+
  ggtitle("Classification of training data with decision boundary, Alden")
```

ii. Using θ = 0.5, compute the confusion matrix for the testing set and comment on your output.

```
glm.probs=predict(glm.fit,banktest,type="response")
glm.pred=rep("0",412)
glm.pred[glm.probs>.5]="1"
table(glm.pred,banktest$y)

##
## glm.pred   0   1
##        0 204  24
##        1  32 152

print("training error")

## [1] "training error"

(24+32)/(32+152+204+24)
```

```
## [1] 0.1359223
```

```
print("false positive rate")
```

```
## [1] "false positive rate"
```

```
32/(32+204)
```

```
## [1] 0.1355932
```

```
print("true positive rate")
```

```
## [1] "true positive rate"
```

```
152/(24+152)
```

```
## [1] 0.8636364
```

The error is approximately 14% which is great suggesting that the model is wrong only about 14% of the time on unseen data (testing data). The false positive rate is approximately 14% implying that about 14% of genuine banknotes are incorrectly classified as forged banknotes. The true positive rate is approximately 86% which is great. This means that about 86% of forged banknotes are correctly classified as forged banknotes.

iii. Compute confusion matrices for the testing set using θ = 0.6 and compare it with the one for θ = 0.5. Comment on your output. Describe a situation when the θ = 0.6 model may be the preferred model.

```
glm.probs=predict(glm.fit,banktest,type="response")
glm.pred=rep("0",412)
glm.pred[glm.probs>.6]="1"
table(glm.pred,banktest$y)
```

```
##
## glm.pred   0   1
##        0 210  35
##        1  26 141
```

```
print("training error")
```

```
## [1] "training error"
```

```
(26+35)/(32+152+204+24)
```

```
## [1] 0.1480583
```

```
print("false positive rate")
```

```
## [1] "false positive rate"
```

```
26/(26+210)
```

```
## [1] 0.1101695
```

```r
print("true positive rate")
```

```
## [1] "true positive rate"
```

```r
141/(35+141)
```

```
## [1] 0.8011364
```

The error is approximately 15% which is great suggesting that the model is wrong only about 15% of the time on unseen data (testing data). The false positive rate is approximately 11% implying that about 11% of genuine banknotes are incorrectly classified as forged banknotes. The true positive rate is approximately 80% which is great. This means that about 80% of forged banknotes are correctly classified as forged banknotes.

The testing set with 0 = 0.6 has a higher error meaning that it is more likely to classified incorrectly than the previous model. This model has a lower true positive rate meaning that there will be less forged banknotes classified as forged banknotes, which is concerning making this model worse than the previous model.

The situation where this model will be preferred is when classifying genuine banknotes as forged banknotes. This model has a lower false positive rate than the previous model meaning there will be less genuine banknotes classified as forged banknotes. It is better to classify a genuine banknote as forged rather than a forged banknote as genuine, and it is even better to have less of that and this model satisfies that.