Chapter 4 Exercises

Adrien Osakwe

Conceptual

1)

1) 4.2: $p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$

4.3: $\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$

$p(X) \cdot (1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$

$p(X) \cdot (1 + e^{\beta_0 + \beta_1 X}) = \frac{p(X)}{1 - p(X)}$

$1 + e^{\beta_0 + \beta_1 X} = \frac{1}{1 - p(X)}$

$1 - p(X) = \frac{1}{1 + e^{\beta_0 + \beta_1 X}}$

$-p(X) = \frac{1}{1 + e^{\beta_0 + \beta_1 X}} - 1$

$-p(X) = -\frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$

$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$

2)

2) $\delta_k(x)$ is a definition of $\log(p_k x)$

$$p_k(x) = \frac{\pi_k \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}}{\sum_{l=1}^{K} \pi_l \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}}$$

$$\log(p_k(x)) = \log(\pi_k) + \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2\sigma^2}(x-\mu_k)^2 - \log\left(\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}\right)$$

~~All the expression~~ As we are looking for $p_k(x)$ that is largest for a given k, we only care about the k-dependent terms.
The highlighted expressions are k-independent constants we can ignore

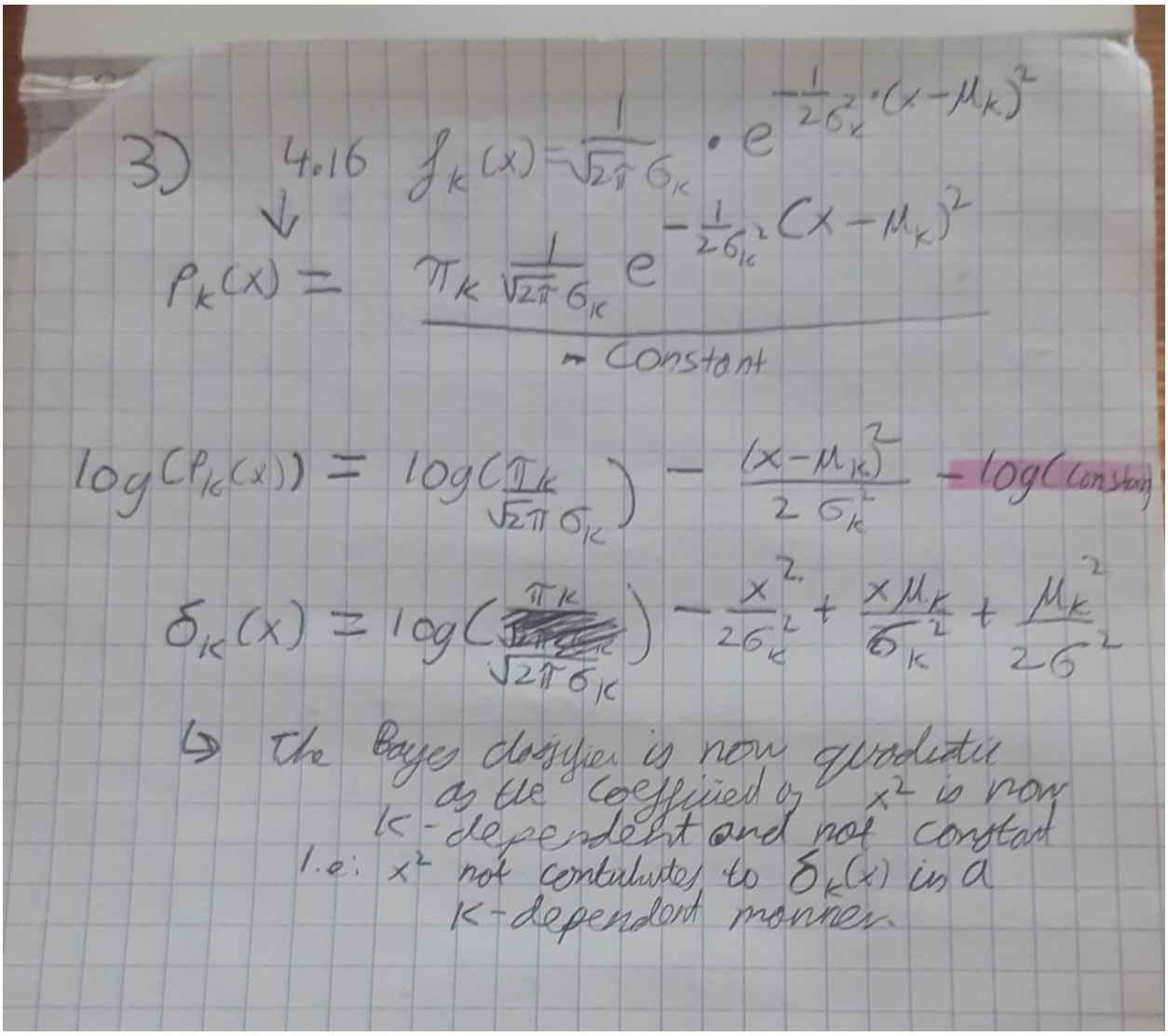
↳ $$\delta_k(x) = \log(\pi_k) - \frac{1}{2\sigma^2}(x^2 - 2\mu_k x + \mu_k^2)$$

$$= \log(\pi_k) - \frac{x^2}{2\sigma^2} + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}$$

$$= \log(\pi_k) + x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}$$

As $\delta_k(x)$ is entirely k-dependent, its maximum value will occur at the same k as $p_k(x)$.
This is because $\log()$ is a monotonic function.

3)



4)

5)

a) we expect QDA to perform better on the training set and LDA on the test set. This is because QDA is more flexible and will better fit the noise of the training set. However, this will cause overfitting, leading the LDA generating a better result in the test set.

b) If non-linear, we expect QDA to outperform LDA in both cases as it will not be disadvantaged by the bias LDA will have given its assumption of linearity. The increased flexibility of QDA will therefore allow a better fit and generate a decision boundary closest to the BDB.

c) We expect this to improve QDA as the increased number of samples will reduce the contribution of noise to the model fitting and reduce the risk of overfitting.

d) False. This may only work if there is a sufficient number of observations to prevent overfitting with the QDA approach. If this is not the case, QDA will most likely start fitting the noise of the data whereas LDA will not have this issue due to its reduced flexibility.

6)

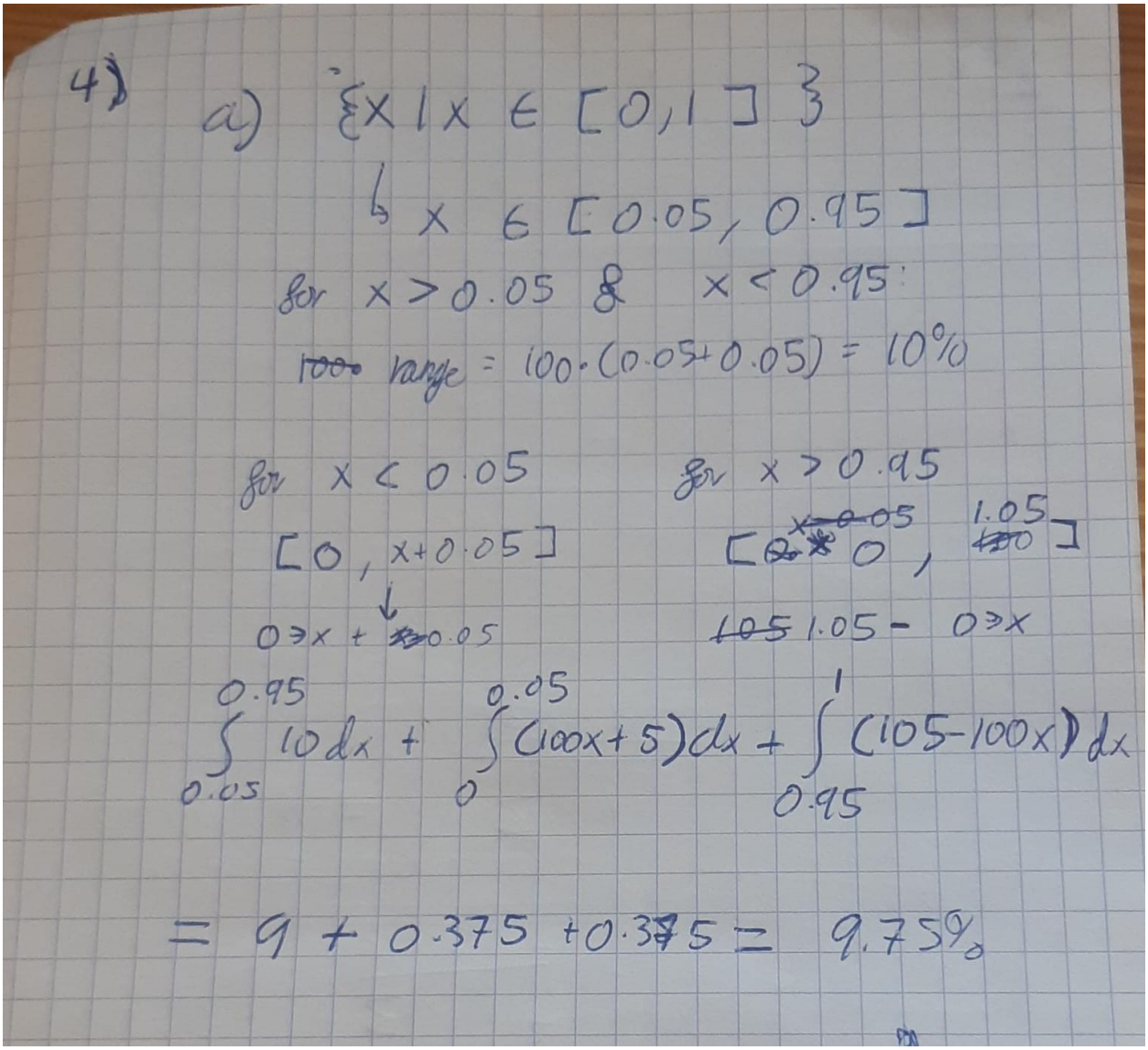

4)
a) {x | x ∈ [0,1]}
↳ x ∈ [0.05, 0.95]
for x > 0.05 & x < 0.95:
range = 100·(0.05+0.05) = 10%
for x < 0.05
[0, x+0.05]
0 ⇒ x + 0.05
for x > 0.95
[0, 1.05]
1.05 − 0 ⇒ x
∫ 10 dx + ∫ (100x+5) dx + ∫ (105−100x) dx
= 9 + 0.375 + 0.375 = 9.75%

Figure 1: Q4_1

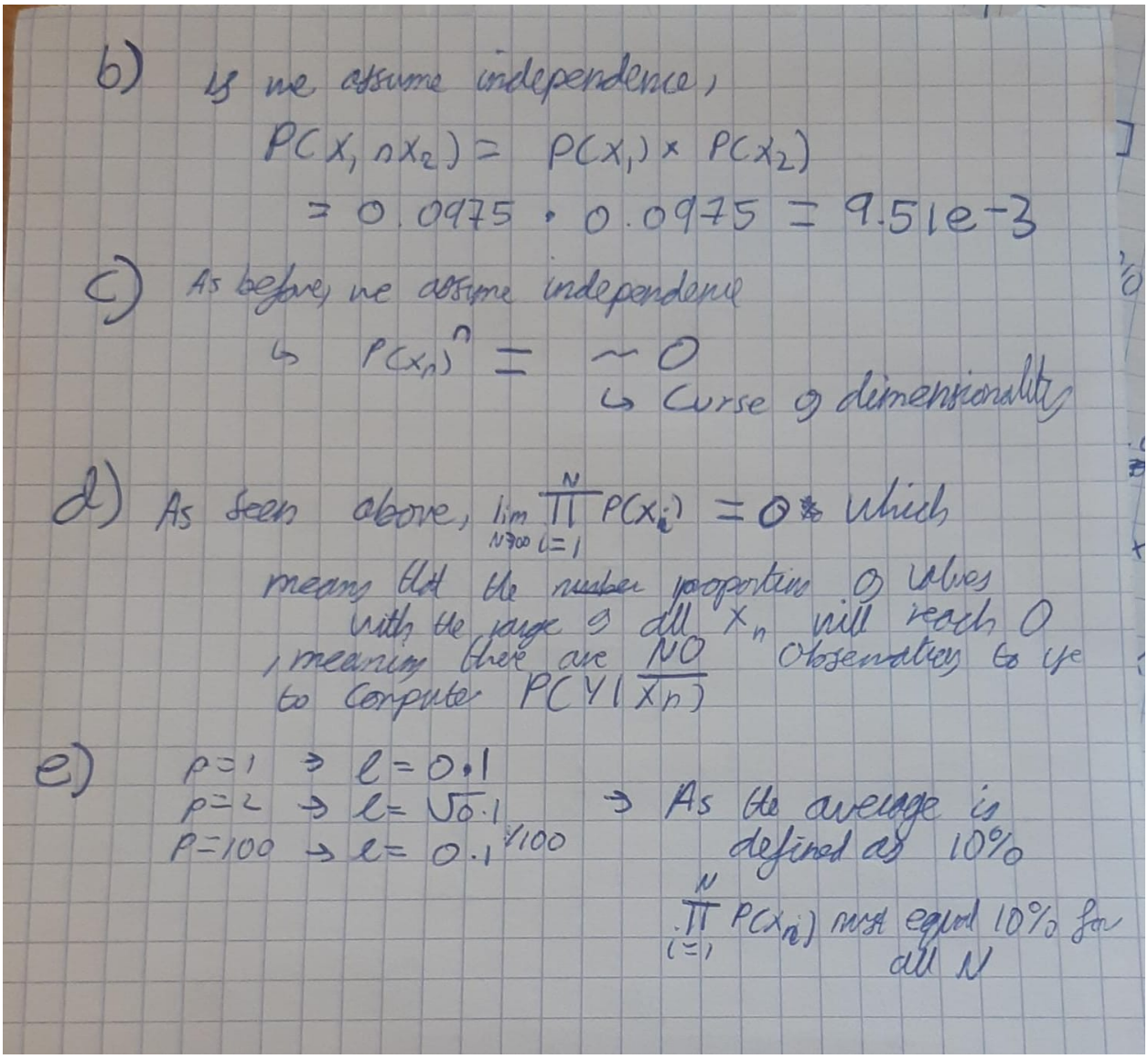

b) If we assume independence,
$P(X_1 \cap X_2) = P(X_1) \times P(X_2)$
$= 0.0975 \cdot 0.0975 = 9.51e-3$
c) As before, we assume independence
↳ $P(X_n)^{\wedge} = \sim 0$
↳ Curse of dimensionality
d) As seen above, $\lim_{N \to \infty} \prod_{i=1}^{N} P(X_i) = 0$ * which
means that the number proportion of values
with the range of all X_n will reach 0
, meaning there are NO observations to use
to compute $P(Y | X_n)$
e) $p=1 \rightarrow \ell = 0.1$
$p=2 \rightarrow \ell = \sqrt{0.1}$
$p=100 \rightarrow \ell = 0.1^{1/100}$
→ As the average is
defined as 10%
$\prod_{(=1}^{N} P(X_n)$ must equal 10% for
all N

Figure 2: Q4_2

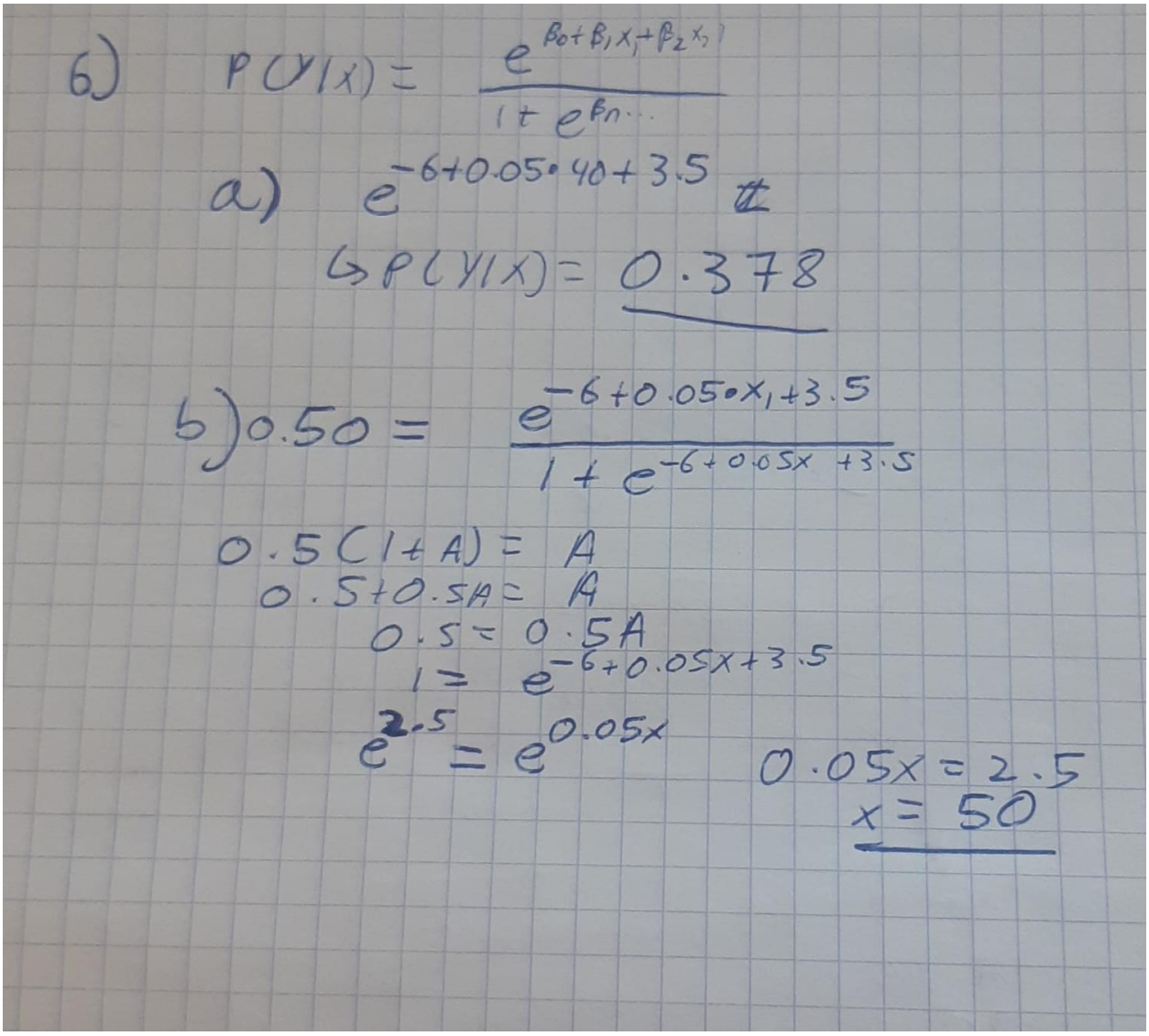


Figure 3: Q6_Image

7)

P(Yes | X = 4)

–> As X is normally distributed, we can use LDA with p = 1 pi_yes = 0.8 mean_yes = 10 mean_no = 0 var = 36

calculate Pk(x = 4) with k = Yes using LDA for p = 1 –> pk(4) = 0.752

8) As the KNN classifier uses k = 1, the training error rate is 0% as the classifier simply returns the same observation as the NN. Therefore, for the average to be 18%, the test error rate must be 36%, meaning the logistic regression model is more accurate.

9)

a) `odds = p(x)/p(!x)`
`0.27`

b) 0.16/0.84 = 0.19

10) Incomplete, need to confirm this answer

11) Need to come back to this

12)

Applied

Q13)

a) `Correlation matrix shows a strong correlation between volume and year in this data set.`
`Box plots of Volume vs. Year seem to agree with this observation. All other correlations are consid`

b) The logistic regression finds that the null hypothesis can only be rejected for the Lag2 parameter.

c) Using 0.50 probability as the threshold, the model has a 56% accuracy rate with the training data. The confusion matrix tells us that our model seems to generate too many false positive (Up predictions) as it is fairly good at identifying Ups as Ups, but bad at identifying the Downs as Downs.

d) Modifying the model to only use Lag2 as a predictor yields an accuracy of 62%. Again, the model still has a hard time correctly identifying Down periods. Although we would be able to predict all the Up days, all the money we make would probably be lost in the FP Down days.

e) LDA Approach yields 62.5% accuracy.

f) QDA simply labels everything as Up...

g) KNN with k = 1 gives 50% accuracy. Good to note that its Down prediction is better than the others.

h) Also labels everything as up..

i) Based on these results, I would go with either the logistic or LDA model. However, I would tweak the thresholds in both to see how much we can minimize the FP rate in both. The model with parameters that can best reduce the FP and maintain a > 50% accuracy would be the best choice.

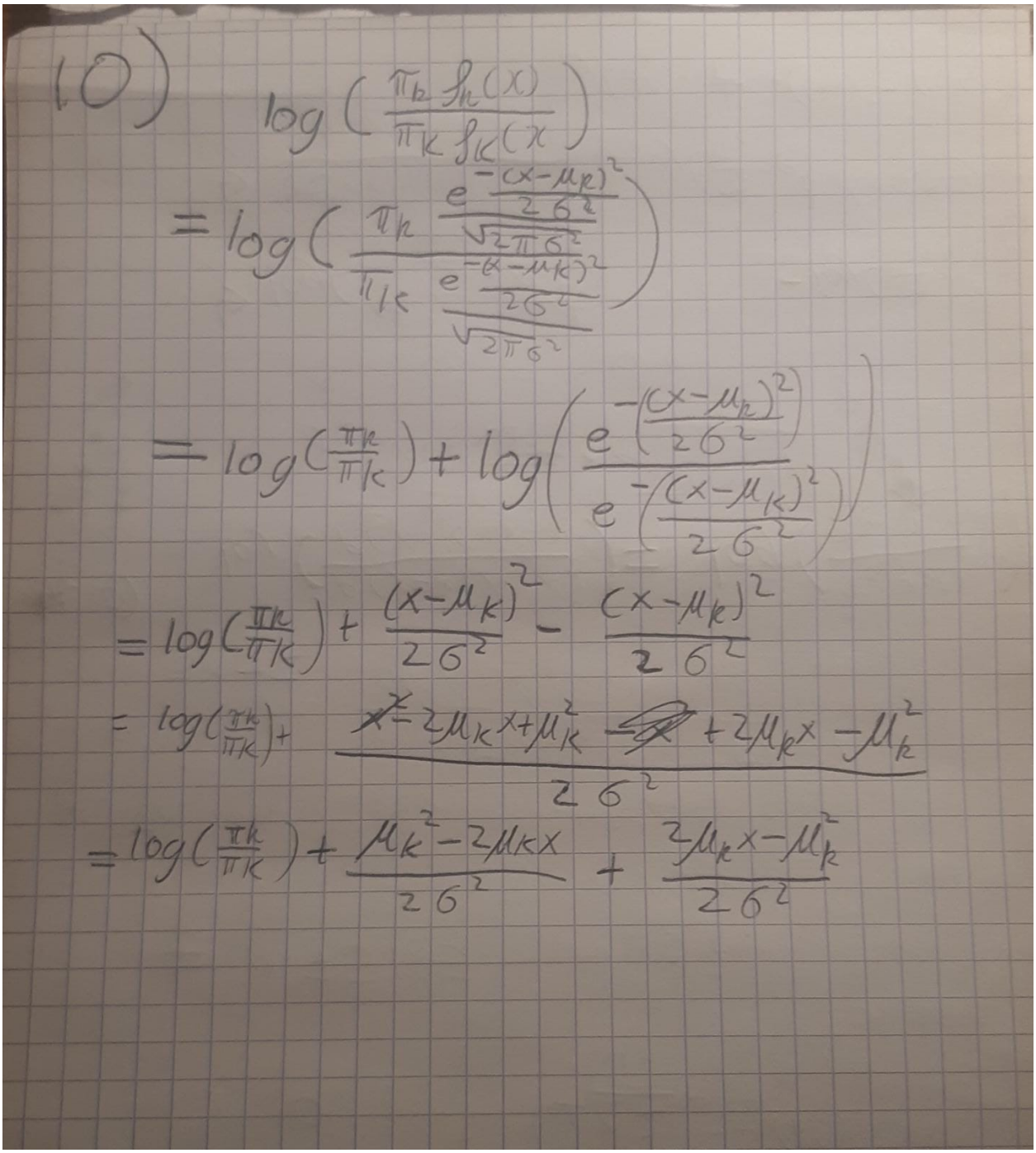


Figure 4: Partial Q10

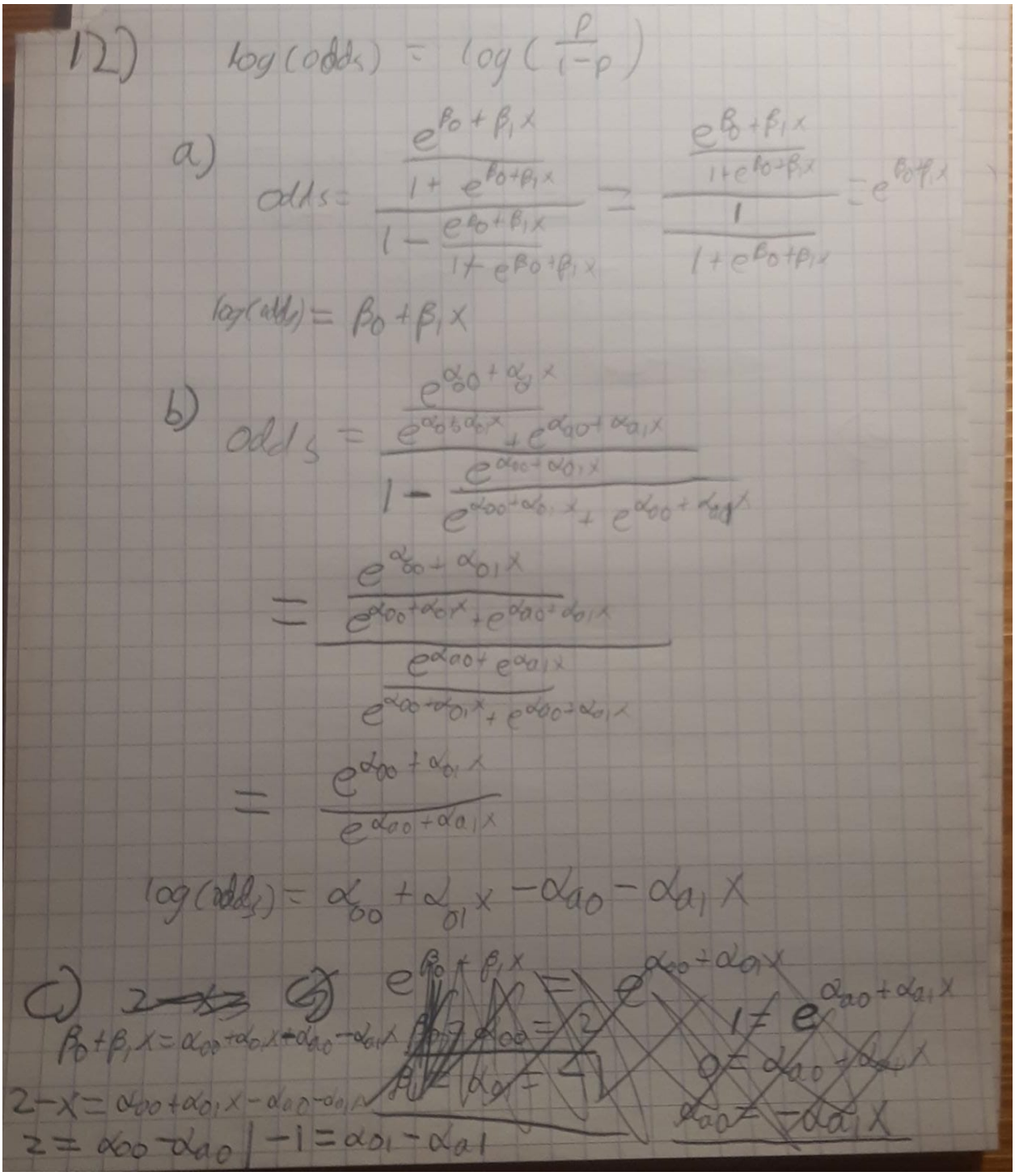


Figure 5: Q12 a-c

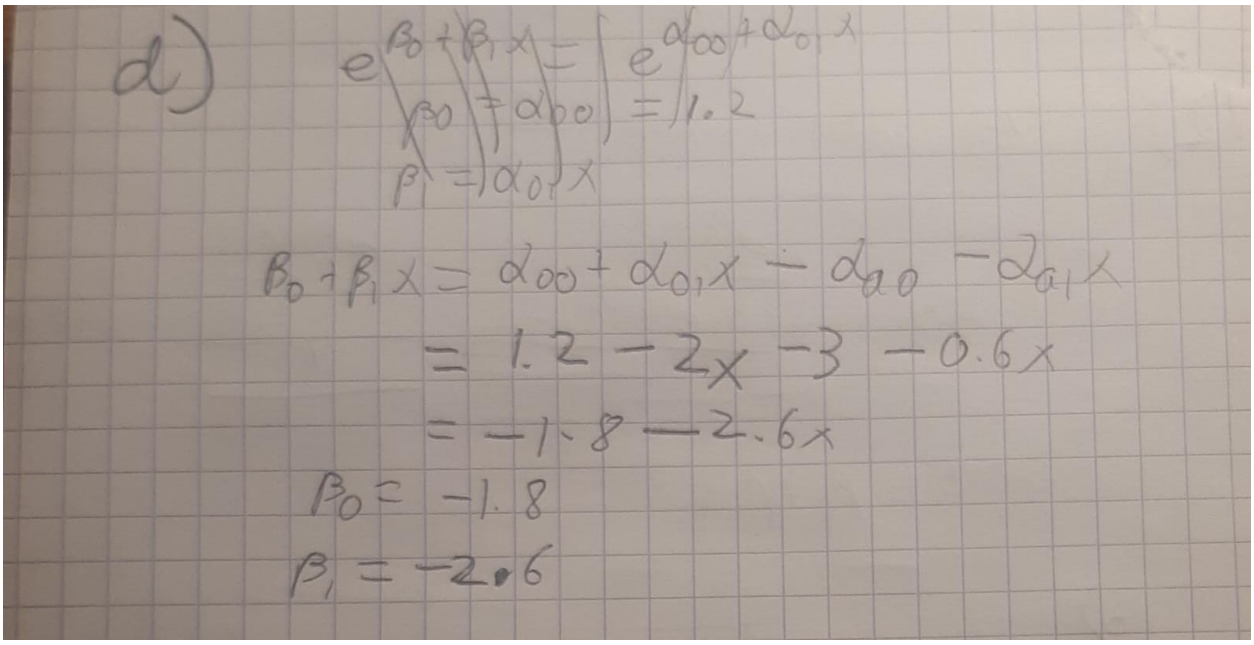


Figure 6: Q12 d

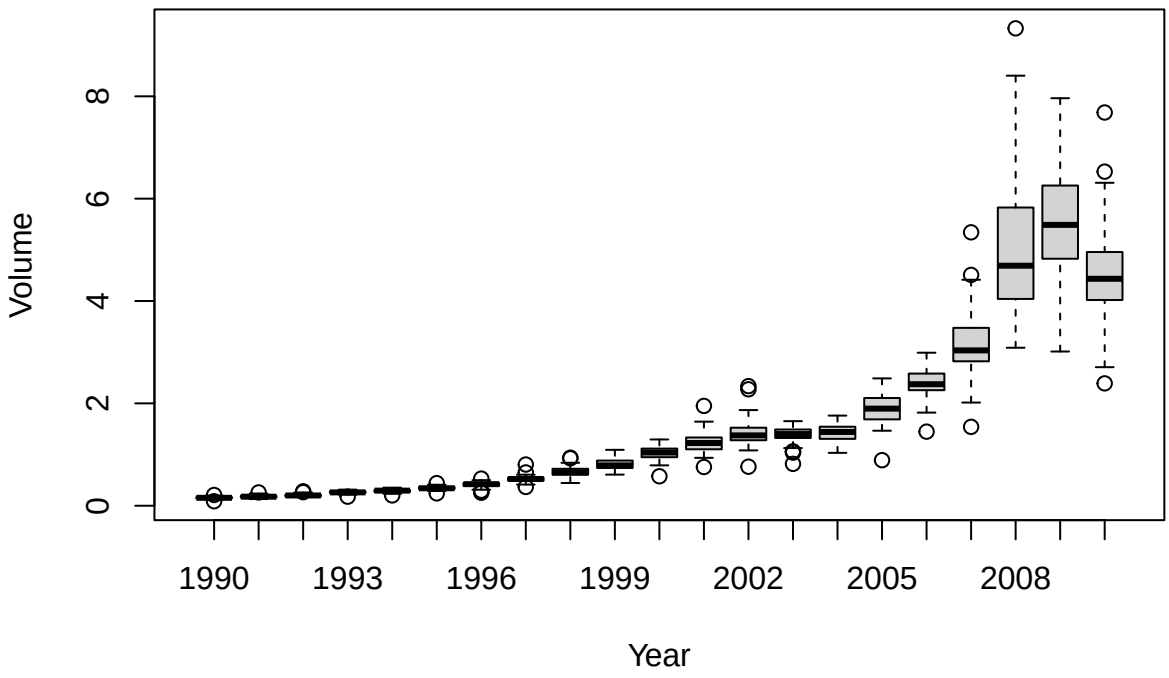
```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
cor(Weekly[,-9])
```

```
##                Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

```
attach(Weekly)
boxplot(Volume ~ Year)
```




```
#Log regression

log.fit1 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data =
Weekly, family = binomial)
summary(log.fit1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

```
#Confusion matrix
log.prob1 <- predict(log.fit1, type = "response")
log.pred1 <- rep("Down",length(log.prob1))
log.pred1[log.prob1 > 0.5] <- "Up"
table(log.pred1,Direction)
```

```
##          Direction
## log.pred1 Down  Up
##      Down   54  48
##      Up    430 557
```

```
mean(log.pred1 == Direction)
```

```
## [1] 0.5610652
```

```
#Create training data set and limit model to Lag2
train <- Weekly$Year %in% c(1990:2008)
test.weekly <- Weekly[!train,]
log.fit2 <- glm(Direction ~ Lag2, data =
Weekly, family = binomial,subset = train)

log.prob2 <- predict(log.fit2, test.weekly, type = "response")
log.pred2 <- rep("Down",length(log.prob2))
log.pred2[log.prob2 > 0.5] <- "Up"
table(log.pred2,test.weekly$Direction)
```

```
##
## log.pred2 Down Up
##      Down    9  5
##      Up     34 56
```

```
mean(log.pred2 == test.weekly$Direction)
```

```
## [1] 0.625
```

```
##LDA
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
lda.fit <- lda(Direction ~ Lag2, data =
Weekly,subset = train)
lda.pred <- predict(lda.fit,test.weekly)
table(lda.pred$class,test.weekly$Direction)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

```
mean(lda.pred$class == test.weekly$Direction)
```

```
## [1] 0.625
```

```
##QDA
qda.fit <- qda(Direction ~ Lag2, data =
Weekly,subset = train)
qda.pred <- predict(qda.fit,test.weekly)
table(qda.pred$class,test.weekly$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

```
mean(qda.pred$class == test.weekly$Direction)
```

```
## [1] 0.5865385
```

```
## KNN K = 1
library(class)
train.lag <- as.matrix( Lag2[train])
test.lag <- as.matrix(Lag2[!train])
set.seed(1)
knn.pred <- knn(train.lag,test.lag,Direction[train], k = 1)
table(knn.pred,Direction[!train])
```

```
##
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
mean(knn.pred == Direction[!train])
```

```
## [1] 0.5
```

```
##Naive Bayes
library(e1071)

nb.fit <- naiveBayes(Direction ~ Lag2, data =
Weekly,subset = train)
nb.pred <- predict(nb.fit, Weekly[!train,])
table(nb.pred, Direction[!train])
```

```
##
## nb.pred Down Up
##    Down    0  0
##    Up     43 61
```

```
#Modifying parameters
#LR  with p > 0.6
log.pred3 <- rep("Down",length(log.prob2))
log.pred3[log.prob2 > 0.4] <- "Up"
table(log.pred3,Direction[!train])
```

```
##
## log.pred3 Down Up
##        Up   43 61
```

```
mean(log.pred3 == Direction[!train])
```

```
## [1] 0.5865385
```

```
#KNN
knn.pred2 <- knn(train.lag,test.lag,Direction[train], k =3)
table(knn.pred2,Direction[!train])
```

```
##
## knn.pred2 Down Up
##      Down   16 19
##      Up     27 42
```

```
mean(knn.pred2 == Direction[!train])
```

```
## [1] 0.5576923
```

14)

b) Based on the pairs() plot, horsepower and weight seem to be the best predictors of mpg01.

c)

d) LDA model has an 88% accuracy (12% error rate).

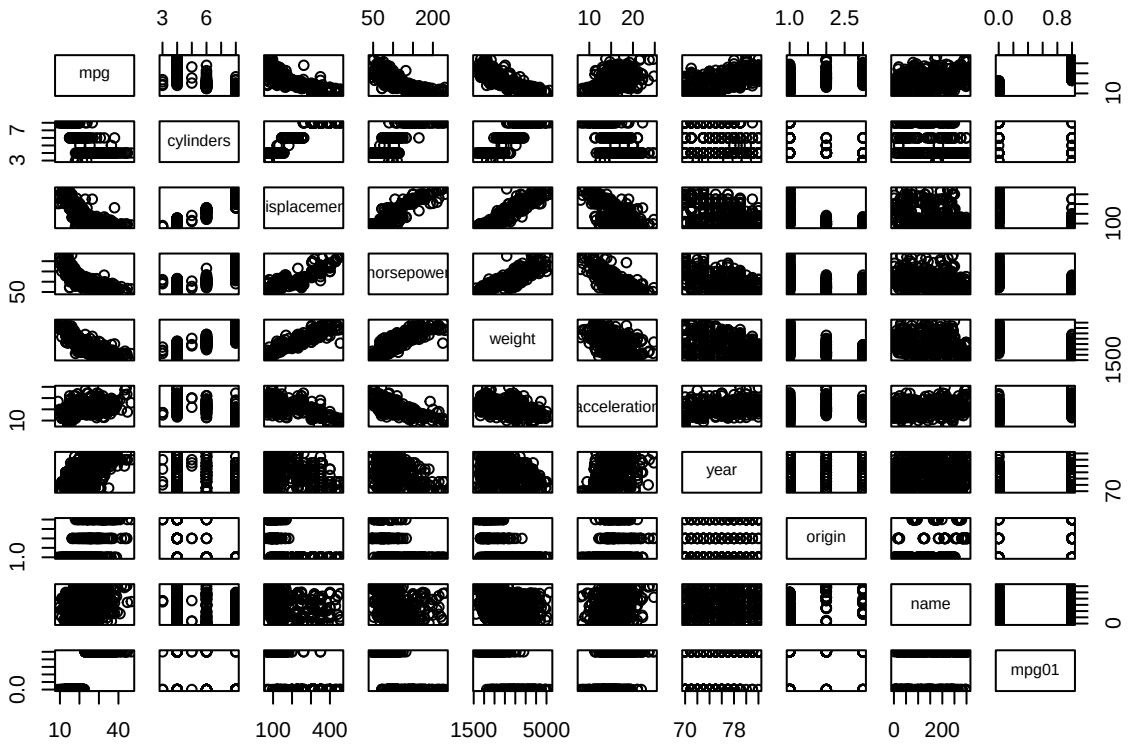
e) QDA model has 87% accuracy (13% error rate).

f) Logistic regression with a threshold of 50% gives an accuracy of 88.5%.

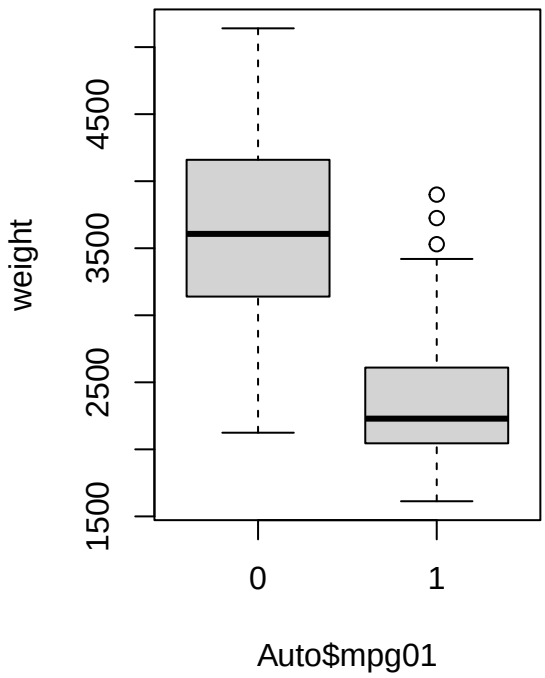
g) Same as above

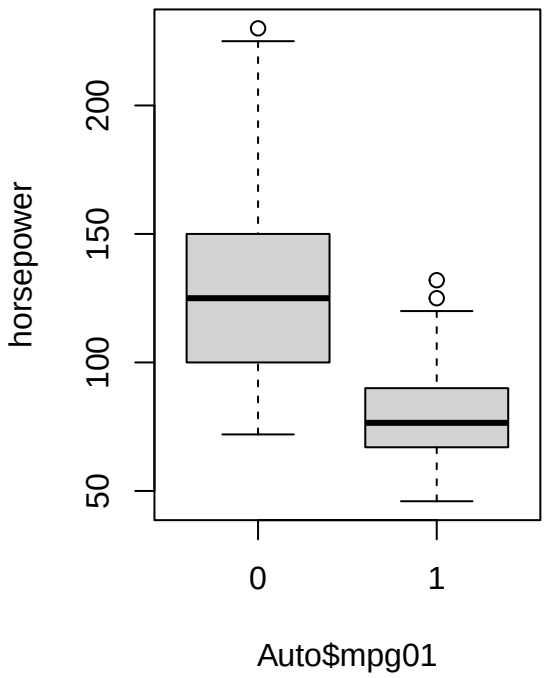
h) K = 10 gives an accuracy of 89%. This tops all the previous models.

```
attach(Auto)
Auto$mpg01 <- ifelse(Auto$mpg > median(Auto$mpg),1,0)
pairs(Auto)
```




```
Auto$train <- sample(c(TRUE,FALSE),nrow(Auto),replace = TRUE)
train <- Auto[Auto$train,]
test <- Auto[!Auto$train,]

par(mfrow = c(1,2))
boxplot(weight ~Auto$mpg01)
boxplot(horsepower ~ Auto$mpg01)
```






```
##LDA
library(MASS)

lda.fit1 <- lda(mpg01 ~ weight + horsepower, data =
Auto,subset = Auto$train)
lda.pred1 <- predict(lda.fit1,test)
table(lda.pred1$class,test$mpg01)
```

```
##
##      0  1
##   0 73  1
##   1 27 90
```

```
mean(lda.pred1$class == test$mpg01)
```

```
## [1] 0.8534031
```

```
##QDA

qda.fit1 <- qda(mpg01 ~ weight + horsepower, data =
Auto,subset = Auto$train)
qda.pred1 <- predict(qda.fit1,test)
table(qda.pred1$class,test$mpg01)
```

```
##
##      0  1
##   0 77  5
##   1 23 86
```

```
mean(qda.pred1$class == test$mpg01)
```

```
## [1] 0.8534031
```

```
##LR

log.fit3 <- glm(mpg01 ~ weight + horsepower, data =
Auto,subset = Auto$train, family = binomial)
log.prob3 <- predict(log.fit3, test, type = "response")
log.pred3 <- ifelse(log.prob3 > 0.5,1,0)
table(log.pred3,test$mpg01)
```

```
##
## log.pred3  0  1
##         0 81  5
##         1 19 86
```

```
mean(log.pred3 == test$mpg01)
```

```
## [1] 0.8743455
```

```
##NB
library(e1071)

nb.fit1 <- naiveBayes(mpg01 ~ weight + horsepower, data =
Auto,subset = Auto$train)
nb.pred1 <- predict(nb.fit1, test)
table(nb.pred1, test$mpg01)
```

```
##
## nb.pred1  0  1
##        0 77  4
##        1 23 87
```

```
mean(nb.pred1 == test$mpg01)
```

```
## [1] 0.8586387
```

```
##KNN
library(class)
knn.pred3 <- knn(train[,c("weight","horsepower")],test[,c("weight","horsepower")],train$mpg01, k = 10)
table(knn.pred3,test$mpg01)
```

```
##
## knn.pred3  0  1
##         0 75  2
##         1 25 89
```

```r
mean(knn.pred3 == test$mpg01)
```

```
## [1] 0.8586387
```

15)

```r
#A
Power <- function(){
  2^3
}
#B
Power2 <- function(a,b){
  a^b
}

Power2(2,3)  #8
```

```
## [1] 8
```

```r
Power2(3,2) # 9
```

```
## [1] 9
```

```r
#C
Power2(10,3)
```

```
## [1] 1000
```

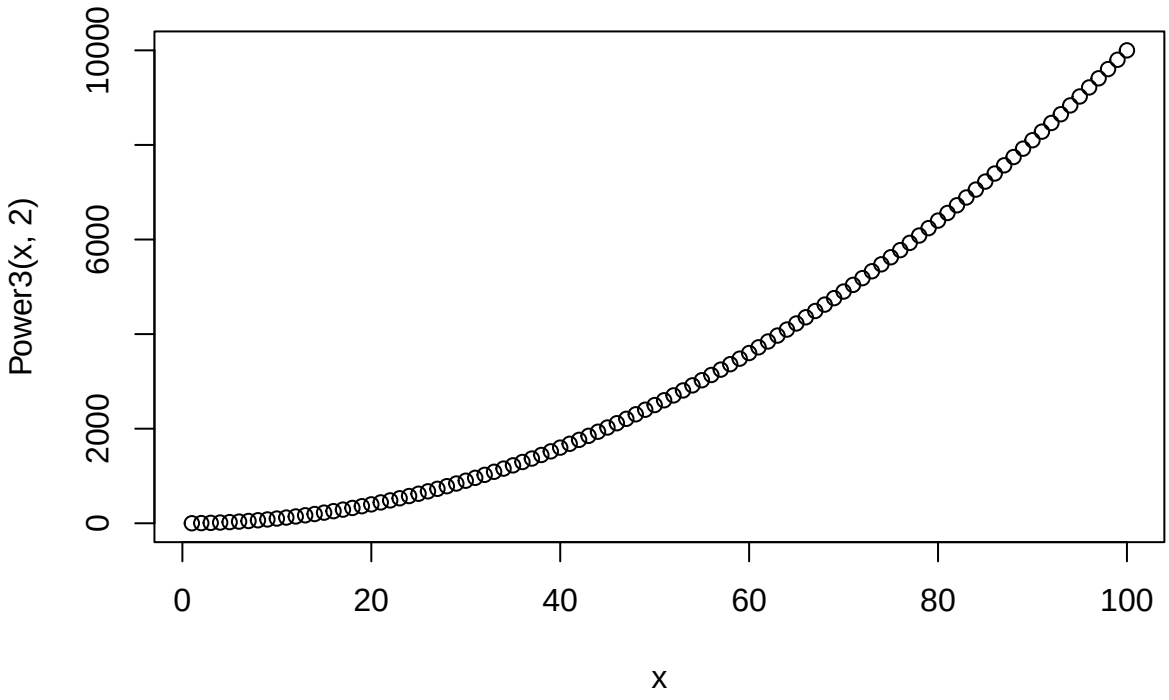
```r
Power2(8,17)
```

```
## [1] 2.2518e+15
```

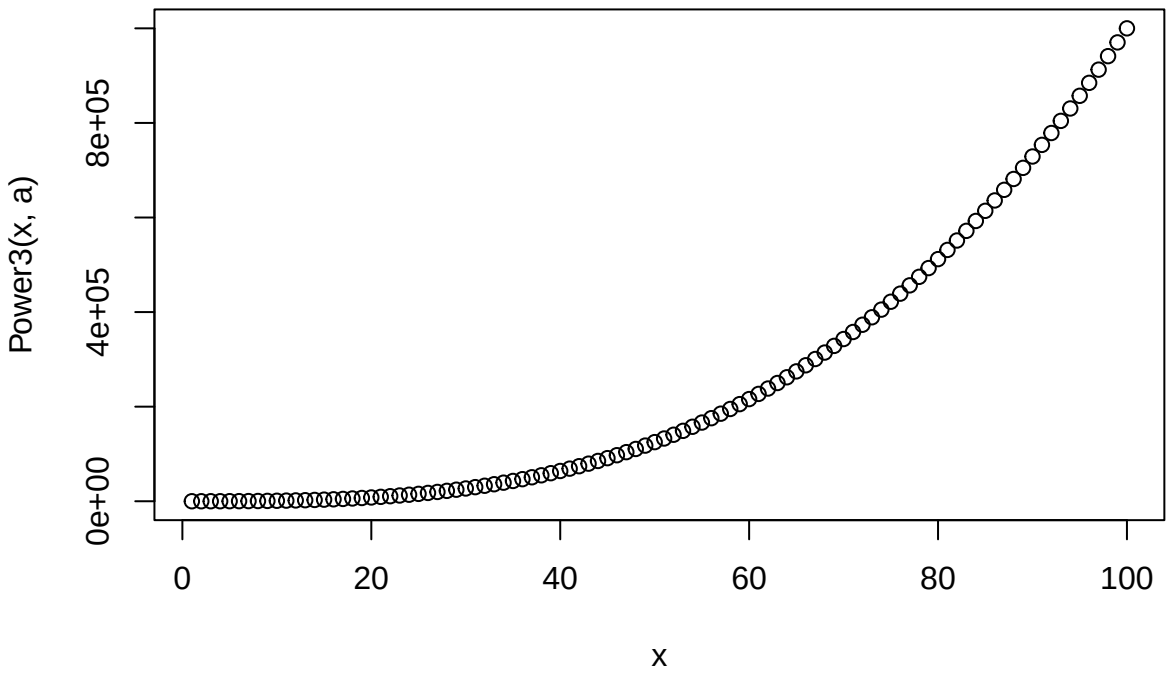
```r
Power2(131,3)
```

```
## [1] 2248091
```

```r
#D

Power3 <- function(a,b){
  result <- a^b
  return(result)
}

#E
x <- 1:100
plot(x,Power3(x,2))
```




```
#F

PlotPower <- function(x,a){
  plot(x,Power3(x,a))
}

PlotPower(1:100,3)
```



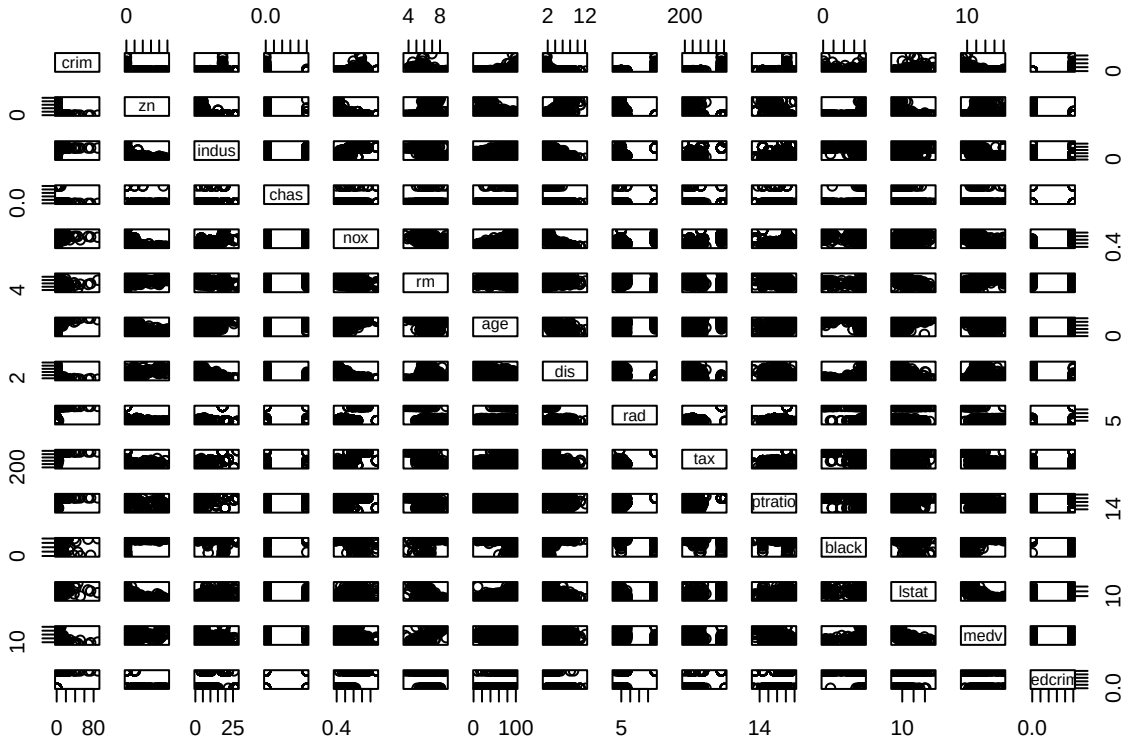
16)

At first glance the nox, dis and medv measurements seem to show the most correlation with median(crim). The predictors don't seem to hae guassian distributions so it may not be worth exploring LDA. Will focus on LR, KNN and QDA. The pairs() results show some correlation between predictors so I don't think we can assume independence as required by naive bayes. There may be some collinearity between these predictors as all three reject their null hypothesis when trained alone but only nox does when all used. This is the case even with interactions included. Will proceed with only nox for this model.

- LR: 83.6% accuracy
- QDA: 80.4%
- KNN: 76% (k = 5 and used all three predictors)

Based on these results, I would go with the logistic regression model for this scenario. Would need to consider changing threshold probability to see how it adjusts error rate.

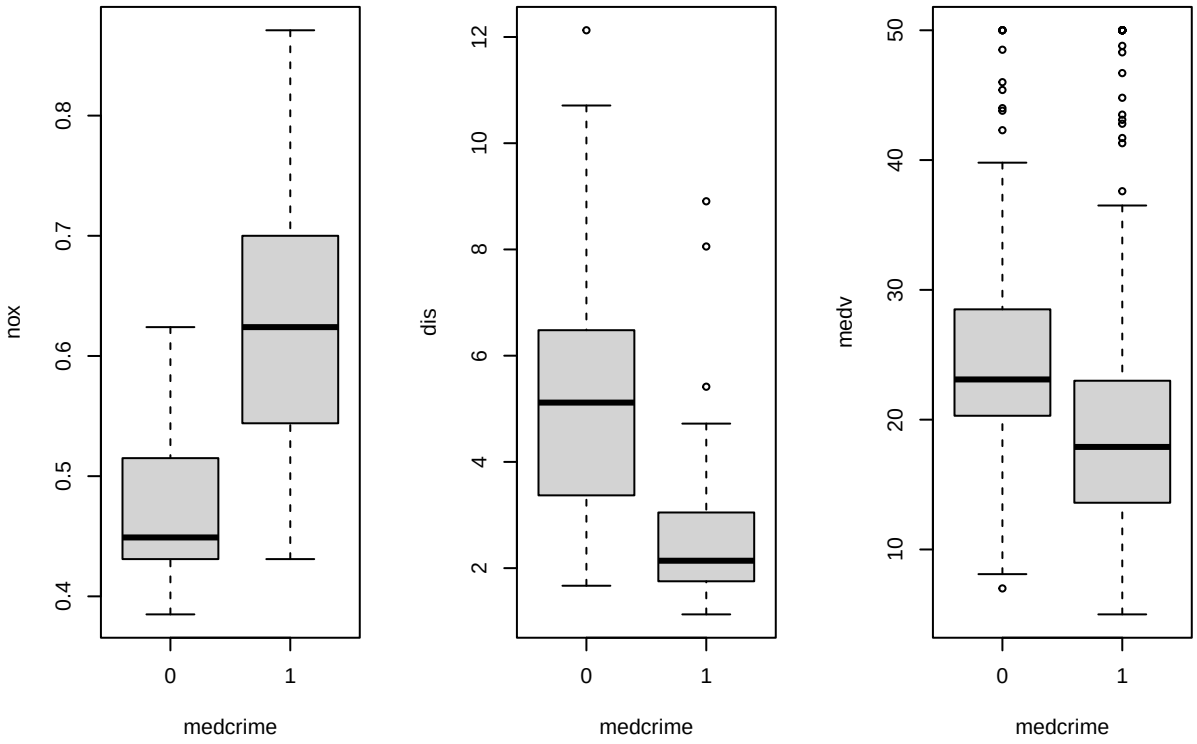
```
attach(Boston)
Boston$medcrime <- ifelse(crim > median(crim),1,0)
pairs(Boston)
```



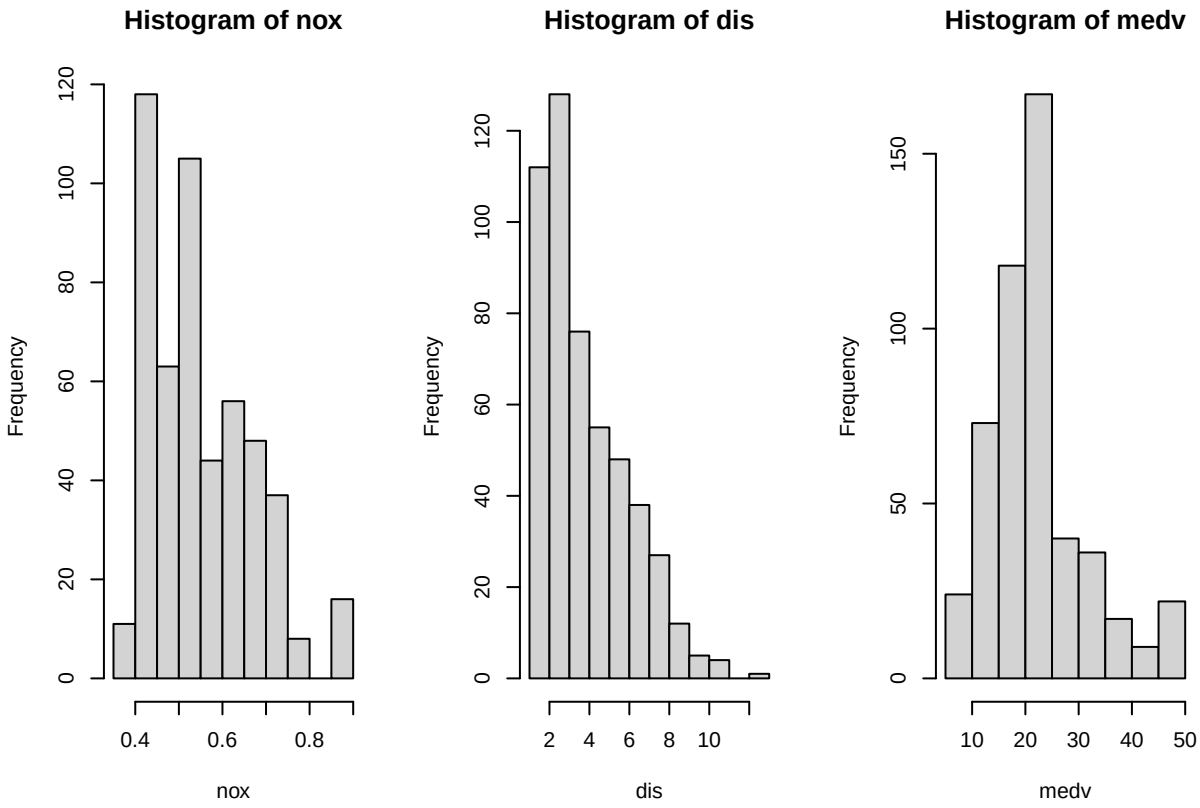

```
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 3):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn
```

```
par(mfrow = c(1,3))
boxplot(nox ~ medcrime)
boxplot(dis  ~ medcrime)
boxplot(medv  ~ medcrime)
```




```
#Distributions
par(mfrow = c(1,3))
hist(nox)
hist(dis)
hist(medv)
```




```
#LR
train <- sample(c(TRUE,FALSE),nrow(Boston),replace = TRUE)
Bos_train <- Boston[train,]
Bos_test <- Boston[!train,]
log.fit4 <- glm(medcrime ~ nox, data = Boston, subset = train, family = binomial)
log.prob4 <- predict(log.fit4, Bos_test,type = "response")
log.pred4 <- ifelse(log.prob4 > 0.5, 1,0)
table(log.pred4,Bos_test$medcrime)
```

```
##
## log.pred4   0   1
##         0 106  32
##         1  14 107
```

```
mean(log.pred4 == Bos_test$medcrime)
```

```
## [1] 0.8223938
```

```
#QDA
qda.fit2 <- qda(medcrime ~ nox, data = Boston, subset = train)
qda.pred2 <- predict(qda.fit2,Bos_test)
mean(qda.pred2$class == Bos_test$medcrime)
```

```
## [1] 0.8532819
```

```r
#KNN
knn.pred4 <- knn(Bos_train[,c("nox","dis","medv")],Bos_test[,c("nox","dis","medv")],Bos_train$medcrime,
mean(knn.pred4 == Bos_test$medcrime)
```

```
## [1] 0.8030888
```