STA314 Homework 3

student number: 1003942326

Yulin WANG

24/10/2019

Question 1

(a)

$$\begin{aligned} P(X|Dividend = Yes) &= \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(x-\mu_{yes})^2) \\ &= \frac{1}{6\sqrt{2\pi}}\exp(-\frac{1}{2\cdot 6^2}(x-10)^2) \\ &= \frac{1}{6\sqrt{2\pi}}\exp(-\frac{1}{72}(x-10)^2) \end{aligned}$$

(b)

$$\begin{aligned} P(X|Dividend = No) &= \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(x-\mu_{no})^2) \\ &= \frac{1}{6\sqrt{2\pi}}\exp(-\frac{1}{2\cdot 6^2}(x-0)^2) \\ &= \frac{1}{6\sqrt{2\pi}}\exp(-\frac{1}{72}x^2) \end{aligned}$$

(c)

Since we have:

```r
dnorm(4,10,6)
```

```
## [1] 0.04032845
```

```r
dnorm(4,0,6)
```

```
## [1] 0.05324133
```

So we have:

$$P(X=4|Dividend = Yes) = dnorm(4,10,6) = 0.04032845$$
$$P(X=4|Dividend = No) = dnorm(4,0,6) = 0.05324133$$

(d)

$$P(Dividend = Yes) = 0.8$$

(e)

$$P(Dividend = No) = 1 - P(Dividend = Yes) = 0.2$$

(f)

By Bayes' rule, we have:

$$
\begin{aligned}
&P(Dividend = Yes|X = 4) = \frac{P(Dividend = Yes, X = 4)}{P(X = 4)} \\
&= \frac{P(X = 4|Dividend = Yes) \cdot P(Dividend = Yes)}{P(X = 4|Dividend = Yes) \cdot P(Dividend = Yes) + P(X = 4|Dividend = No) \cdot P(Dividend = No)} \\
&= \frac{0.04032845 \cdot 0.8}{0.04032845 \cdot 0.8 + 0.05324133 \cdot 0.2} \\
&= 0.7518524
\end{aligned}
$$

Question 2

(a)

```r
library(ISLR)
#str(Auto) #glimpse the data
#firstly, create a new rv mpg01 only contains 0 with same length as mpg
mpg01 <- rep(0, length(Auto$mpg))
mpg01[Auto$mpg > median(Auto$mpg)] <- 1 # =1 if mpg value above its median
new_df <- data.frame(Auto, mpg01) #new data frame including all variables
```

(b)

Firstly, return the covariance matrix ignoring the discrete variables "name" and "cylinders"

```r
cor(new_df[,-c(2,9)])
```

```
##                     mpg displacement horsepower     weight acceleration
## mpg           1.0000000   -0.8051269 -0.7784268 -0.8322442    0.4233285
## displacement -0.8051269    1.0000000  0.8972570  0.9329944   -0.5438005
## horsepower   -0.7784268    0.8972570  1.0000000  0.8645377   -0.6891955
## weight       -0.8322442    0.9329944  0.8645377  1.0000000   -0.4168392
## acceleration  0.4233285   -0.5438005 -0.6891955 -0.4168392    1.0000000
## year          0.5805410   -0.3698552 -0.4163615 -0.3091199    0.2903161
## origin        0.5652088   -0.6145351 -0.4551715 -0.5850054    0.2127458
## mpg01         0.8369392   -0.7534766 -0.6670526 -0.7577566    0.3468215
##                    year     origin      mpg01
## mpg           0.5805410  0.5652088  0.8369392
## displacement -0.3698552 -0.6145351 -0.7534766
## horsepower   -0.4163615 -0.4551715 -0.6670526
## weight       -0.3091199 -0.5850054 -0.7577566
## acceleration  0.2903161  0.2127458  0.3468215
## year          1.0000000  0.1815277  0.4299042
## origin        0.1815277  1.0000000  0.5136984
## mpg01         0.4299042  0.5136984  1.0000000
```

From the covariance matrix above, we can find that there are three continuous features: displacement, horsepower, and weight seem to be highly correlated with mpg/mpg01(absolute value of correlation coefficients > 0.6), so that they seem most likely to be useful in predicting mpg.

(c)

```r
library(caTools)
set.seed(101)
names <- new_df[,9] # extract labels from the data
hold <- sample.split(names, SplitRatio = 0.7)
train <- new_df[hold,] #training set
test <- new_df[!hold,] #test set
```

(d)

```r
library('MASS')
fit.lda <- lda(mpg01 ~ displacement+horsepower+weight, data=train)
pred.lda <- predict(fit.lda, test)$class
table(pred.lda, test$mpg01)
```

```
##
## pred.lda  0  1
##        0 50  1
##        1 12 55
```

```r
error.lda <- mean(pred.lda != test$mpg01) #test error
error.lda
```

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

(e)

```r
fit.qda <- qda(mpg01 ~ displacement+horsepower+weight, data=train)
pred.qda <- predict(fit.qda, test)$class
table(pred.qda, test$mpg01)
```

```
##
## pred.qda  0  1
##        0 51  4
##        1 11 52
```

```r
error.qda <- mean(pred.qda != test$mpg01) #test error
error.qda
```

```
## [1] 0.1271186
```

(f)

```
fit.log <- glm(mpg01 ~ displacement+horsepower+weight, data=train, family = 'binomial')
probs <- predict(fit.log, test, type = "response")
pred.log <- rep(0, length(probs))
pred.log[probs > 0.5] <- 1
table(pred.log, test$mpg01)
```

```
##
## pred.log  0  1
##        0 53  5
##        1  9 51
```

```
error.log <- mean(pred.log != test$mpg01) #test error
error.log
```

```
## [1] 0.1186441
```

(g)

```
library(class)
train_x <- train[,c("displacement", "horsepower", "weight")]
train_y <- train[, "mpg01"]
test_x <- test[,c("displacement", "horsepower", "weight")]
k_list <- c(1,5,10,15,20,30,50,100,150,200)
error_knn <- rep(0, length(k_list))
for (i in 1:length(k_list)) {
  k_i <- k_list[i]
  pred_knn <- knn(train_x, test_x, train_y, k=k_i)
  error_knn[i] <- mean(pred_knn != test$mpg01)
}
```

test errors correspond to 10 different values of K from K=1 to K=200 in order:

```
error_knn
```

```
##  [1] 0.10169492 0.08474576 0.10169492 0.11864407 0.13559322 0.12711864
##  [7] 0.12711864 0.12711864 0.13559322 0.16101695
```

best perform on this data set with lowest test error rate:

```
K <- k_list[which.min(error_knn)]
K
```

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

Thus, K=5 seems to perform the best on this data set since it has the lowest test error.