STA 545 Statistical Data Mining I, Fall 2020

Homework 7

Stella Liao

October 28, 2020

Problem 1

(a) With a one-unit increase in X_1(hours studied), the expected change in log odds is 0.05(which is β_1).
(b) When we plug in the equation, we could get that

$$p = \frac{e^{-6+0.05\times40+3.5}}{1+e^{-6+0.05\times40+3.5}}$$

```
exp(-6+0.05*40+3.5)/(1+exp(-6+0.05* 40+3.5))
```

```
## [1] 0.3775407
```

Therefore, the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class is about 37.75%

(c) In this case, we have that

$$p = \frac{e^{-6+0.05X_1+3.5}}{1+e^{-6+0.050.05X_1+3.5}} = 0.8$$

and then,

$$e^{-6+0.05X_1+3.5} = 4$$

```
(log(4)-3.5+6)/0.05
```

```
## [1] 77.72589
```

Hence, the student in part (a) need to study about 77.73 hours to have a 80% chance of getting an A in the class.

Problem 2

(a)

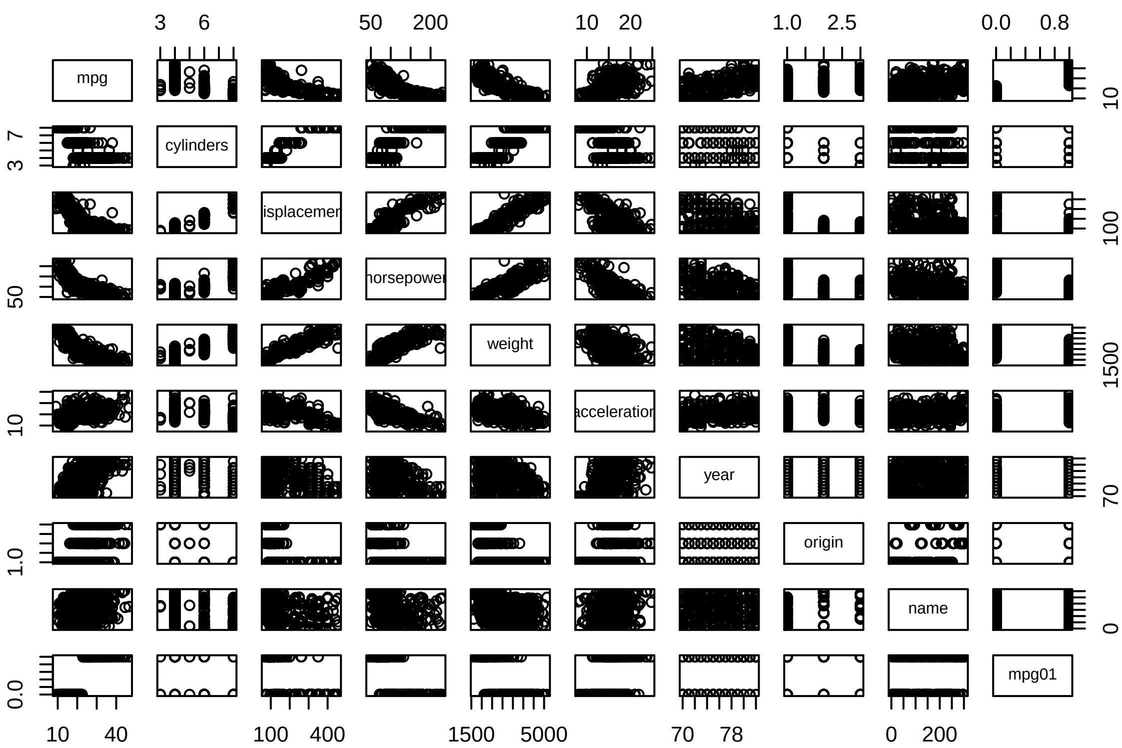
```
library(ISLR)
library(dplyr)

data('Auto')
myAuto <- Auto %>%
  mutate(mpg01 = (ifelse(mpg > median(mpg), 1, 0)))
```

(b)

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

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```r
pairs(myAuto)
```



From above, `cylinders`, `weight`, `displacement`, and `horsepower` might be useful to predict `mpg01`. To be specific, `mpg01` seems to have a strong inverse relationship with them.

(c)

```r
set.seed(50321222)
#split the 'myAuto' data randomly split by half as training data and testing data
train.num=sample(1:nrow(myAuto),.5*nrow(myAuto),replace=FALSE)
```

```r
Auto.train = myAuto[train.num,]
Auto.test = myAuto[-train.num,]
```

(d) LDA

1)

```r
#remain predictors related to mpg01 most
Auto.train <- Auto.train %>%
  dplyr::select(cylinders, weight, displacement, horsepower, mpg01)
Auto.test <- Auto.test %>%
  dplyr::select(cylinders, weight, displacement, horsepower, mpg01)
```

2)

```r
library(MASS)
```

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

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
lda.fit = lda(mpg01 ~ .,data = Auto.train)
lda.pred = predict(lda.fit, Auto.test)
#the test error of lda
mean(lda.pred$class != Auto.test$mpg01)
```

```
## [1] 0.1122449
```

(e) QDA

```r
qda.fit = qda(mpg01 ~ ., data = Auto.train)
qda.pred = predict(qda.fit, Auto.test)
#the test error of qda
mean(qda.pred$class != Auto.test$mpg01)
```

```
## [1] 0.1071429
```

(f) Logistic Regression

```r
glm.fit = glm(mpg01 ~ ., data = Auto.train, family = binomial)
glm.probs = predict(glm.fit, Auto.test, type = "response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
#the test error of logistic regression
mean(glm.pred != Auto.test$mpg01)
```

```
## [1] 0.1173469
```

(g) KNN

```r
library(class)
train.X = Auto.train[,-5]
test.X = Auto.test[,-5]
```

```r
# KNN(k=1)
knn.pred = knn(train.X, test.X, Auto.train[,5], k = 1)
#the test error of knn(k=1)
mean(knn.pred != Auto.test$mpg01)
```

```
## [1] 0.1326531
```

```r
# KNN(k=10)
knn.pred = knn(train.X, test.X, Auto.train[,5], k = 10)
#the test error of knn(k=10)
mean(knn.pred != Auto.test$mpg01)
```

```
## [1] 0.122449
```

```r
# KNN(k=100)
knn.pred = knn(train.X, test.X, Auto.train[,5], k = 100)
#the test error of knn(k=100)
mean(knn.pred != Auto.test$mpg01)
```

```
## [1] 0.1071429
```

According to the test errors, when K = 100, it has lowest test error value and thus has best performance among them.