Name: Jingyi Yuan
UNI: jy2736
Class: STAT W4240

Homework 04

Question 1

(a).

```
########################################
# This code uses tm to preprocess the papers into a format useful for NB
preprocess.directory = function(dirname){

  # the directory must have all the relevant text files
  ds = DirSource(dirname)
  # Corpus will make a tm document corpus from this directory
  fp = Corpus( ds )
  # inspect to verify
  # inspect(fp[1])
  # another useful command
  # identical(fp[[1]], fp[["Federalist01.txt"]])
  # now let us iterate through and clean this up using tm functionality
    # make all words lower case
    fp = tm_map( fp , content_transformer(tolower));
    # remove all punctuation
    fp = tm_map( fp , removePunctuation);
    # remove stopwords like the, a, and so on.
    fp = tm_map( fp, removeWords, stopwords("english"));
    # remove stems like suffixes
    fp = tm_map( fp, stemDocument)
    # remove extra whitespace
    fp = tm_map( fp, stripWhitespace)
  # now write the corpus out to the files for our future use.
  # MAKE SURE THE _CLEAN DIRECTORY EXISTS
    dir.create( sprintf('%s_clean',dirname) )
    writeCorpus( fp , sprintf('%s_clean',dirname) )
}

########################################

##########################################

preprocess.directory("fp_hamilton_train")
preprocess.directory("fp_hamilton_test")
preprocess.directory("fp_madison_train")
preprocess.directory("fp_madison_test")

##########################################
```

(b).

```
########################################

hamilton.train = read.directory("fp_hamilton_train_clean")
hamilton.test = read.directory("fp_hamilton_test_clean")
madison.train = read.directory("fp_madison_train_clean")
madison.test = read.directory("fp_madison_test_clean")

########################################
```

(c).

```
all = list(hamilton.test, hamilton.train, madison.train, madison.test)
dictionary = make.sorted.dictionary.df(all)
```

(d).

```
dtm.hamilton.train = make.document.term.matrix(hamilton.train, dictionary)
dtm.hamilton.test = make.document.term.matrix(hamilton.test, dictionary)
dtm.madison.train = make.document.term.matrix(madison.train, dictionary)
dtm.madison.test = make.document.term.matrix(madison.test, dictionary)
```

(e).

```
D = nrow(dictionary)
mu = 1/D
logp.hamilton.train = make.log.pvec(dtm.hamilton.train, mu)
logp.hamilton.test = make.log.pvec(dtm.hamilton.test, mu)
logp.madison.train = make.log.pvec(dtm.madison.train, mu)
logp.madison.test = make.log.pvec(dtm.madison.test, mu)
```

Question 2

```
#problem2
h_train_num = nrow(dtm.hamilton.train)
m_train_num = nrow(dtm.madison.train)
train_num = h_train_num + m_train_num
log.prior.hamilton = log(h_train_num/train_num)
log.prior.madison = log(m_train_num/train_num)
result_h = vector()
result_m = vector()

naive.bayes = function(logp.hamilton.train, logp.madison.train, log.prior.hamilton, log.prior.madison, dtm.test){
  p_h = vector()
  p_m = vector()
  result= vector()
  for (i in 1:nrow(dtm.test)){
    p_h[i] = log.prior.hamilton + sum( logp.hamilton.train * dtm.test[i,] ) # y = 1
    p_m[i] = log.prior.madison + sum( logp.madison.train * dtm.test[i,] ) # y = 0
    if(p_h[i] > p_m[i]) {result[i] = 1}
    else {result[i] = 0}
  }
  return(result)
}

result_h = naive.bayes (logp.hamilton.train, logp.madison.train, log.prior.hamilton, log.prior.madison, dtm.hamilton.test)
result_m = naive.bayes (logp.hamilton.train, logp.madison.train, log.prior.hamilton, log.prior.madison, dtm.madison.test)
```

Question 3

```
> correct
[1] 0.6296296
> true_positives
[1] 1
> true_negatives
[1] 0.09090909
> false_positives
[1] 0.9090909
> false_negatives
[1] 0
```

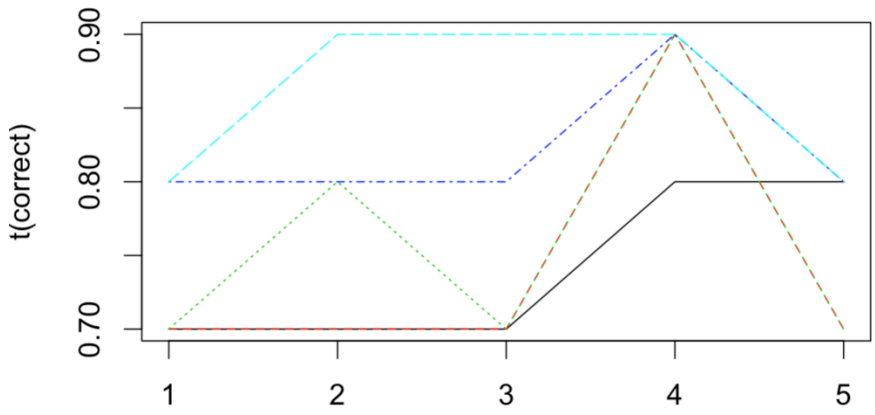
Question 4

(a).

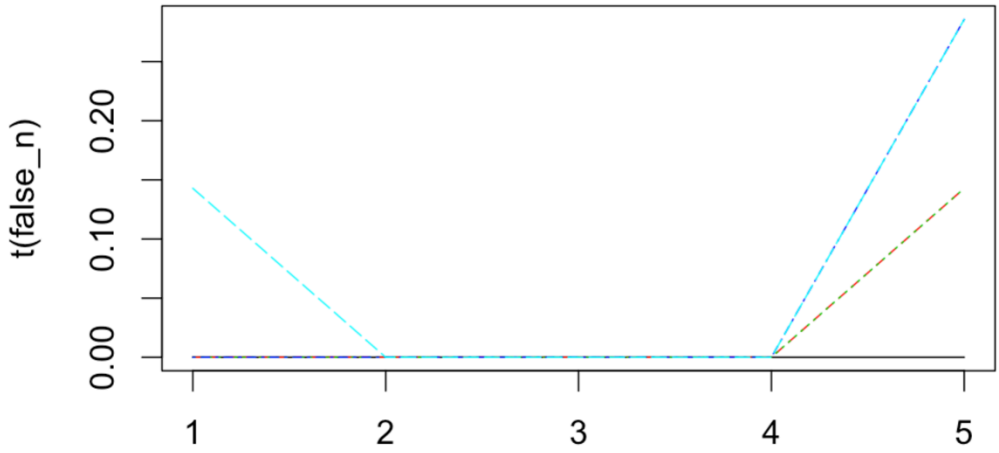
```
mu = c(0.1/D1 ,1/D1, 10/D1, 100/D1, 1000/D1)
```

The ith row means $\mu = mu[i]$.

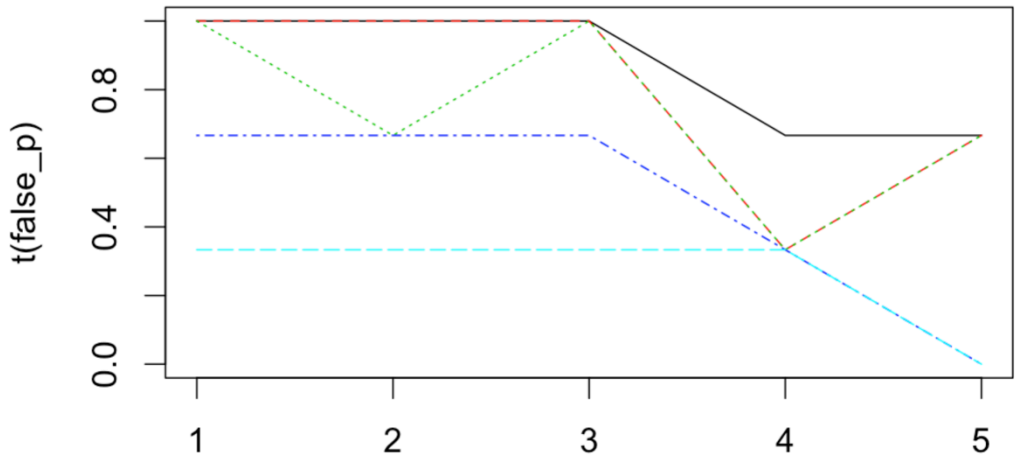
```
> correct
     [,1] [,2] [,3] [,4] [,5]
[1,]  0.7  0.7  0.7  0.8  0.8
[2,]  0.7  0.7  0.7  0.9  0.7
[3,]  0.7  0.8  0.7  0.9  0.7
[4,]  0.8  0.8  0.8  0.9  0.8
[5,]  0.8  0.9  0.9  0.9  0.8
```



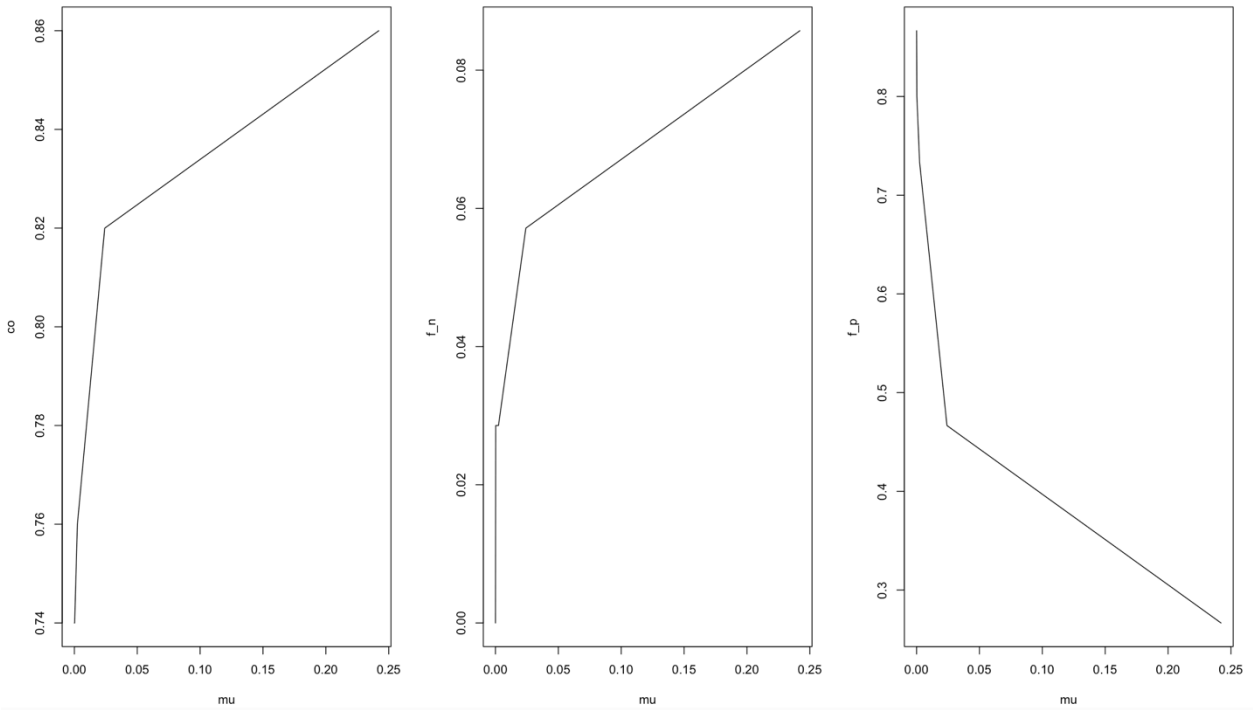
```
> false_n
          [,1] [,2] [,3] [,4]      [,5]
[1,] 0.0000000    0    0    0 0.0000000
[2,] 0.0000000    0    0    0 0.1428571
[3,] 0.0000000    0    0    0 0.1428571
[4,] 0.0000000    0    0    0 0.2857143
[5,] 0.1428571    0    0    0 0.2857143
```




```
> false_p
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0000000 1.0000000 1.0000000 0.6666667 0.6666667
[2,] 1.0000000 1.0000000 1.0000000 0.3333333 0.6666667
[3,] 1.0000000 0.6666667 1.0000000 0.3333333 0.6666667
[4,] 0.6666667 0.6666667 0.6666667 0.3333333 0.0000000
[5,] 0.3333333 0.3333333 0.3333333 0.3333333 0.0000000
```



We calculate the mean of each row and plot them against μ:



(b).

The best value for μ is $\mu = 1000\frac{1}{|D|}$. As shown in the three matrices above, when $\mu = 1000\frac{1}{|D|}$, the correct classification rate is the highest (the light blue line). Although the proportion of false negative is increased slightly, the proportion of false positive decreases from 1 to 0.33, and as mentioned, the total correct classification rate increases. We know that:

$$\hat{p}(X_{ij} = k \mid Y_i = \tilde{y}) = \frac{\mu + \sum_{i=1}^{m} \sum_{j=1}^{n_i} \mathbf{1}_{\{X_{ij}=k, Y_j=\tilde{y}\}}}{\mu|D| + \sum_{i=1}^{m} n_i \mathbf{1}_{\{Y_i=\tilde{y}\}}}.$$

When μ is small, the Bayes classifier has a tendency to classify the data as hamilton because we have more data in the hamilton training set. When μ becomes large, the correct increases since the boundary is changed. To some point, the correct becomes the largest and decreases lately. When μ is extremely large, the bayes classifier will classify all testing set as madison.

(c).

```
> correct
[1] 0.6296296 0.6296296 0.7037037 0.8518519 0.9259259
> false_n
[1] 0.0000 0.0000 0.0000 0.0000 0.0625
> false_p
[1] 0.90909091 0.90909091 0.72727273 0.36363636 0.09090909
```

The answer from (b) is still the best value. As mentioned in (b), with the increasing of μ, although the proportion of false negative is increased slightly, the proportion of false positive decreases from 0.91 to 0.09, and the total correct classification rate increases. Considering the meaning of μ mentioned in (b), $\mu = 1000\frac{1}{|D|}$ is still the best value.

(d).

```
> percentage_error_c
[1] 0.1752941 0.1752941 0.0800000 0.0373913 0.0712000
```

The size of the training set may account for the differences. We can increase the size of the training set. We can do this by increasing K or adding more data to the training dataset (for both hamilton and madison dataset), thus the differences between the cross-validation rate estimates and the rates on the training sets could be minimized.

Question 5

We have:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

Since we cannot assume $\sigma_1^2 = \ldots = \sigma_K^2$ for QDA, we have:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)}{\sum_{l=1}^{K}\pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2)}$$

so:

$$\log(p_k(x)) \propto \log[\pi_k \frac{1}{\sqrt{2\pi}\sigma_k}\exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)]$$

$$= \log[\pi_k \frac{1}{\sqrt{2\pi}\sigma_k}] - \frac{1}{2\sigma_k^2}(x-\mu_k)^2$$

Obviously we can see that these is a x^2, thus the Bayes' classifier is not linear.

Question 6

(a).

For the training set, since QDA is more flexible, it may fit the data better than LDA even if the Bayes decision boundary is linear. But since we do not know the exact data, we cannot tell exactly which one performs better and can only expect QDA to be better.

For the testing set, we may expect LDA performs better than QDA since the boundary is linear and QDA may overfit.

(b).

Since the boundary is non-linear, we expect QDA to perform better on both the training and the testing set. QDA is more flexible and can fit the training set well. Also, since the boundary is non-linear, it will perform better than LDA on the testing set.

(c).

In general, as the sample size n increases, we expect the test prediction accuracy of QDA relative to LDA to improve, because QDA is more flexible and takes the variances into consideration. As the sample size increases, QDA can fit the dataset better and precisely since the variances do not affect the result as much as in LDA.

(d).

False.

If the Bayes decision boundary for a given problem is linear, QDA may overfit on the testing dataset thus may achieve an inferior test error rate.

Problem 7

(a).

```
> mpg01
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 [48] 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 [95] 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1
[142] 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0
[189] 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
[236] 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0
[283] 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
[330] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
[377] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
```

Change the first column of Auto from mpg to mpg01 and form a data.frame data_auto:

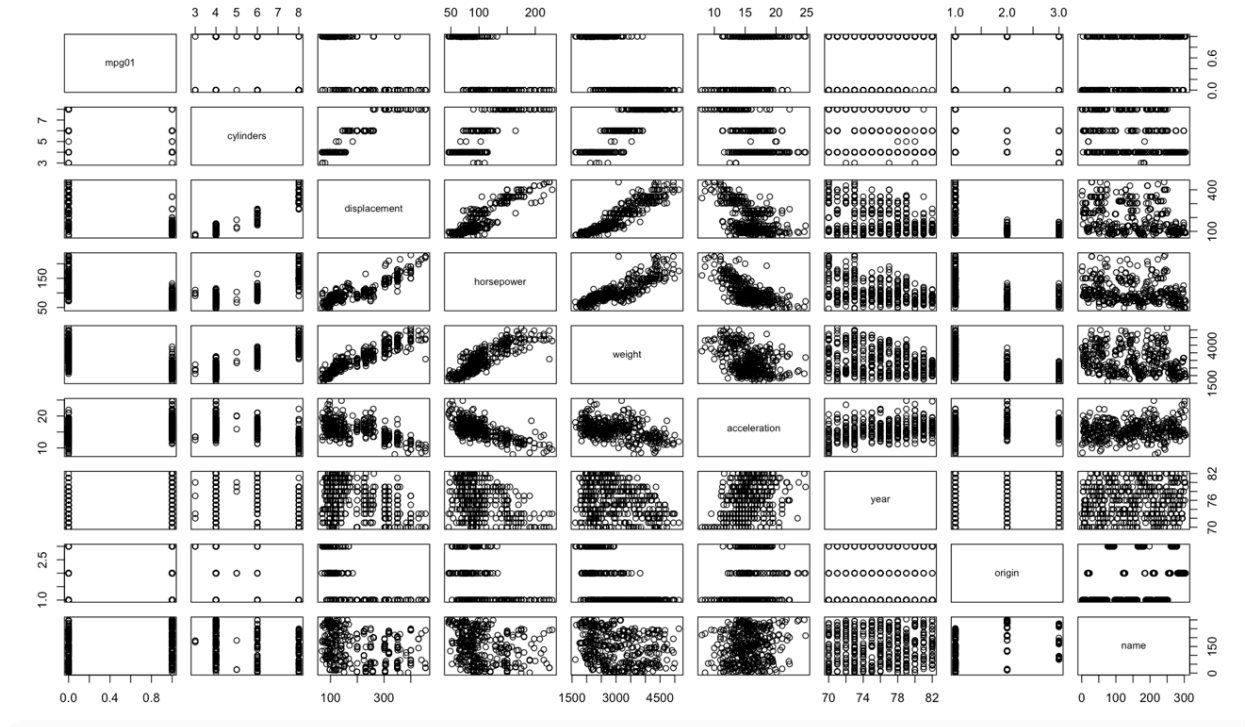
data_auto	392 obs. of 9 variables

Part of data_auto is shown below:

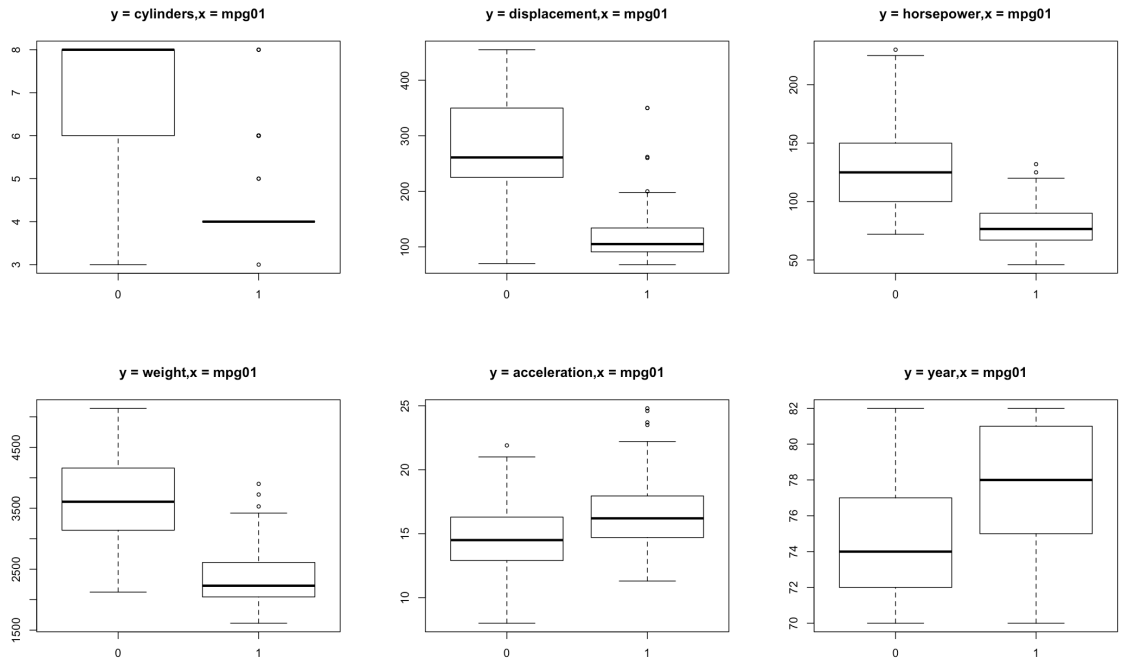
	mpg01	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
1	0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle m
2	0	8	350.0	165	3693	11.5	70	1	buick skylark 320
3	0	8	318.0	150	3436	11.0	70	1	plymouth satellite
4	0	8	304.0	150	3433	12.0	70	1	amc rebel sst
5	0	8	302.0	140	3449	10.5	70	1	ford torino
6	0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
7	0	8	454.0	220	4354	9.0	70	1	chevrolet impala
8	0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
9	0	8	455.0	225	4425	10.0	70	1	pontiac catalina
10	0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
11	0	8	383.0	170	3563	10.0	70	1	dodge challenger se
12	0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
13	0	8	400.0	150	3761	9.5	70	1	chevrolet monte carl

(b).

First use the function paris() to see the relationship between each variables:



Then we use boxplot():



From the picture we can see that acceleration, year, origin and name have little influence on the value of mpg01, while cylinders, displacement, horsepower and weight and seem to be useful in predicting mpg01.

(c).

We split the data by using the variable "year":

```
#(c)
train_num = (year%%2 == 0)  # if the year is even
test_num = !train_num
train = data_auto[train_num, ]
test = data_auto[test_num, ]
mpg01_test = mpg01[test_num]
```

Then we have our training set and test set:

test	182 obs. of 9 variables
train	210 obs. of 9 variables

(d).

Perform LDA using the four variables mentioned above (cylinders, displacement, horsepower and weight) to predict mpg01:

```
> table(pred.test, mpg01_test)
         mpg01_test
pred.test  0  1
        0 86  9
        1 14 73
```

```
> test_error_lda
[1] 0.1263736
```

(e).

Perform QDA using the four variables mentioned above (cylinders, displacement, horsepower and weight) to predict mpg01:

```
> table(pred.test_qda, mpg01_test)
             mpg01_test
pred.test_qda  0  1
            0 89 13
            1 11 69
```

```
> test_error_qda
[1] 0.1318681
```

(f).

Perform logistic regression using the four variables mentioned above (cylinders, displacement, horsepower and weight) to predict mpg01:

```
> LR_a$coefficients
  (Intercept)    cylinders displacement   horsepower       weight
17.658730372 -1.028031664  0.002461740 -0.050610857 -0.002922352
```

```
> table(pred.test_LR, mpg01_test)
             mpg01_test
pred.test_LR  0  1
            0 89 11
            1 11 71
```

```
> test_error_LR
[1] 0.1208791
```

(g).

Perform kNN using the four variables mentioned above (cylinders, displacement, horsepower and weight) to predict mpg01:

We use five values of K:

```
k = c(1,5,10,50,100)
```

```
> test_error_kNN
[1] 0.1538462 0.1483516 0.1593407 0.1483516 0.1428571
```

When K = 100, the test error is smallest. So in these values of K, K = 100 seems to perform the best on this data.