STAT 154: Homework 6 Solutions

Release date: **Sunday, April 7**

Due by: **11 PM, Sunday, April 21**

The honor code

(a) Please state the names of people who you worked with for this homework. You can also provide your comments about the homework here.

(b) Please type/write the following sentences yourself and sign at the end. We want to make it *extra* clear that nobody cheats even unintentionally.

I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at another students solutions and I have fairly credited all external sources in this write up.

Submission instructions

- It is a good idea to revisit your notes, slides and reading; and synthesize their main points BEFORE doing the homework.
- No .Rnw file is provided. You may use templates from previous homeworks if you want.
- **For the parts that ask you to implement/run some R code, your answer should look something like this (code followed by result):**

```
myfun<- function(){
show('this is a dummy function')
}
myfun()

## [1] "this is a dummy function"
```

Note that this is automatically generated if you use the R sweave environment.

- You need to submit the following:
  1. A pdf of your write-up to "HW6 write-up" that includes code for Problem 4.
  2. No *separate* code submission is required for this HW. You have to include the code in your submission for Problem 4 in the write-up itself.
- Ensure a proper submission to gradescope, otherwise it will not be graded. **This time we will not entertain any regrade requests for improper submission.**

Homework Overview

This homework covers kernel ridge regression and classification. The first problems attempts to make you comfortable with computational complexity related questions.

1 Computational complexity (10 pts)

Read about the big-O notation for the wiki page https://en.wikipedia.org/wiki/Big_O_notation and then answer the following using the big-O notation.

In the following questions, by computational complexity we refer to the number of addition/multiplication operations between two real numbers. To elaborate, we assume that two real numbers or multiplying them takes unit operations. Adding k real numbers has k computational complexity. Computing the multiplication of k pairs of numbers would have k computational complexity as well.

Now let $\mathbf{a}, \mathbf{v} \in \mathbb{R}^d$ and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$ and answer the following questions:

1. (2 pts) What is the computational complexity of computing $\mathbf{a} + \mathbf{v}$? What is the computational complexity of computing $\mathbf{a}^\top \mathbf{v}$?
   Ans. For adding two vectors, we see that

$$\mathbf{a} + \mathbf{v} = \begin{bmatrix} a_1 + v_1 \\ a_2 + v_2 \\ \vdots \\ a_d + v_d \end{bmatrix} \implies d \textbf{ addition operations} \implies O(d) \textbf{ computational complexity}.$$

   For the inner product, we see that

$$\mathbf{a}^\top \mathbf{v} = \sum_{i=1}^{d} a_i \cdot v_i \implies d \textbf{ multiplication operations and } (d-1) \textbf{ addition operations}$$
$$\implies 2d - 1 \textbf{ operations}$$
$$\implies O(d) \textbf{ computational complexity}.$$

2. (2 pts) What is the computational complexity of computing the matrix $\mathbf{A} + \mathbf{B}$? How much space does storing the matrix $\mathbf{A}$ require?
   Ans. The matrix addition is just adding all the entries if the two matrices:

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} A_{11} + B_{11} & \dots & A_{1d} + B_{1d} \\ A_{21} + B_{21} & \dots & A_{2d} + B_{2d} \\ \vdots & \ddots & \vdots \\ A_{n1} + B_{n1} & \dots & A_{nd} + B_{nd} \end{bmatrix} \implies nd \textbf{ addition operations}$$
$$\implies O(nd) \textbf{ computational complexity}.$$

   And to store a matrix of size $n \times d$, we need to store nd real numbers which implies the storage complexity for the matrix is $O(nd)$.

3. (4 pts) What is the computational complexity of computing the vector $\mathbf{A}\mathbf{v}$? How about computing the matrix $\mathbf{A}^\top \mathbf{B}$?

Ans. Let

$$\mathbf{A} = \begin{bmatrix} —\mathbf{a}_1^\top — \\ —\mathbf{a}_2^\top — \\ \vdots \\ —\mathbf{a}_n^\top — \end{bmatrix} \Longrightarrow \mathbf{Av} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{v} \\ \mathbf{a}_1^\top \mathbf{v} \\ \vdots \\ \mathbf{a}_n^\top \mathbf{v} \end{bmatrix} \Longrightarrow n \text{ inner products.}$$

Note that each inner product has $O(d)$ computational complexity since $\mathbf{a}_i, \mathbf{v} \in \mathbb{R}^d$ (from part 1 of this problem). Therefore the net complexity is $n \times O(d) = O(nd)$.
For matrix-matrix product, we see computing $\mathbf{A}^\top\mathbf{B}$ is equivalent to computing $\mathbf{A}^\top\mathbf{b}$ for all $\mathbf{b}$ which are the columns of the matrix $\mathbf{B}$. One computation takes $O(nd)$ time and there are d columns in the matrix B so the complexity is $O(nd^2)$.
Another way to get to the same answer is as follows:

$$\mathbf{B} = \begin{bmatrix} —\mathbf{b}_1^\top — \\ —\mathbf{b}_2^\top — \\ \vdots \\ —\mathbf{b}_n^\top — \end{bmatrix} \Longrightarrow \mathbf{A}^\top\mathbf{B} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} —\mathbf{b}_1^\top — \\ —\mathbf{b}_2^\top — \\ \vdots \\ —\mathbf{b}_n^\top — \end{bmatrix} = \sum_{i=1}^{n} \mathbf{a}_i\mathbf{b}_i^\top$$

Now each matrix $\mathbf{a}_i\mathbf{b}_i^\top$ is $d \times d$ and takes $O(d^2)$ to compute and we add n of such matrices implying $O(nd^2)$ complexity.

Remark: **More generally, the computational complexity of obtaining $\mathbf{AB}$ for $\mathbf{A} \in \mathbb{R}^{n\times d}, \mathbf{B} \in \mathbb{R}^{d\times p}$ is $O(ndp)$.**

4. (2 pts) What is the complexity of computing the vector $\mathbf{A}^\top\mathbf{Bv}$?
Hint: There are two ways to do it, one is naive and one is smart. You are encouraged to think and report both of them in your answer.
Ans. We can decompose the problem in two ways:

$$\underbrace{(\mathbf{A}^\top\mathbf{B})}_{\text{matrix-matrix product}}\mathbf{v} = \underbrace{\mathbf{Cv}}_{\text{matrix-vector product}} \Longrightarrow O(nd^2) + O(nd) \Longrightarrow O(nd^2) \textbf{ complexity}$$

$$\mathbf{A}^\top\underbrace{(\mathbf{Bv})}_{\text{matrix-vector product}} = \underbrace{\mathbf{A}^\top\widetilde{\mathbf{v}}}_{\text{matrix-vector product}} \Longrightarrow O(nd) + O(nd) \Longrightarrow O(nd) \textbf{ complexity}$$

Clearly, the second method is better. Think about it! While mathematically the two results are the same, computationally it does not make sense to compute the entire matrix $\mathbf{A}^\top\mathbf{B}$ since it is more expensive to do it (as we know from previous parts). Essentially in R, there will be a difference in amount of time these two codes would take (when n and d are large):

$$(t(A)\%*\%B)\%*\%v \quad \textbf{versus} \quad t(A)\%*\%(B\%*\%v)$$

In other words, where you put the brackets would change the run time of your algorithm, so be careful!

2 Kernel Methods (23 pts)

For the following problems, you can assume that inverting a $p \times p$ matrix takes order p^3 operations, i.e., the computational complexity of matrix inversion of size p is $O(p^3)$. Also for this problem, we use slightly different notation for dimensions in order to remain consistent with the note and the lecture.

1. (2 pts) Let $\mathbf{X} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{y} \in \mathbb{R}^n$. Recall that the ridge estimate for the problem

$$\min_{\theta \in \mathbb{R}^\ell} \|\mathbf{X}\theta - \mathbf{y}\|_2^2 + \lambda \|\theta\|_2^2 \tag{1}$$

is given by $\widehat{\theta}_\lambda = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{X}^\top \mathbf{y}$. What is the computational complexity of computing this estimate?
Hint: You may use answers from previous question.
Ans. We use CC as a short form for computational complexity:

$$\begin{aligned} CC \text{ for computing the matrix } \mathbf{X}^\top \mathbf{X} &= O(n\ell^2) \\ CC \text{ for adding } \mathbf{X}^\top \mathbf{X} \text{ with } \lambda \mathbf{I}_\ell &= O(n\ell) \\ CC \text{ for inverting } \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell &= O(\ell^3) \\ CC \text{ for computing the vector } \mathbf{A}\mathbf{X}^\top \mathbf{y} &= O(n\ell) \quad \text{where } \mathbf{A} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell)^{-1}. \end{aligned}$$

Note that for the last part, we used the smart way from Q1.4; thus the net CC is $O(n\ell^2 + n\ell + \ell^3) = O(n\ell^2 + \ell^3)$, where note that we dropped $n\ell$ since $n\ell^2$ dominates it.
***Remark:* If $n \gg \ell$, then we can simply write it as $O(n\ell^2)$; and on the converse if $\ell \gg n$, the CC can be written as $O(\ell^3)$.**

2. (2 pts) Show that $\widehat{\theta}_\lambda = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y}$ is also a valid estimate for the ridge problem 1. What is the complexity of computing this estimate.
Ans. We take the first estimate (in previous part) as truth (since we know that expression for ridge estimate can be derived directly by setting the gradient of objective to zero.) Hence we need to show that

$$\mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{X}^\top \mathbf{y}$$

for which it suffices to show that

$$\begin{aligned} \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{X}^\top \\ \iff \qquad \mathbf{X}^\top &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n) \\ \iff \mathbf{X}^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_\ell) &= \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n) \\ \iff \mathbf{X}^\top \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{X}^\top &= \mathbf{X}^\top \mathbf{X}\mathbf{X}^\top + \lambda \mathbf{X}^\top, \end{aligned}$$

which is true. Hence we are done.

For CC, we see that

$$CC \text{ for computing the matrix } \mathbf{X}\mathbf{X}^\top = O(n^2\ell) \quad \text{(different than)} \mathbf{X}^\top\mathbf{X}$$
$$CC \text{ for adding } \mathbf{X}\mathbf{X}^\top \text{ with } \lambda\mathbf{I}_n = O(n)$$
$$CC \text{ for inverting } \mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_n = O(n^3)$$
$$CC \text{ for computing the vector } \mathbf{X}^\top\mathbf{A}\mathbf{y} = O(n^2 + n\ell) \quad \text{where } \mathbf{A} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_n)^{-1}.$$

and thus the net CC is $O(n^2\ell + n^2 + n^3 + n + n\ell) = O(n^2\ell + n^3)$ where we drop the less dominant terms.

3. (2 pts) Compare and contrast the computational complexities from the previous two parts.
Ans. The two estimates are usually called different names: ridge estimate for the first one $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_\ell)^{-1}\mathbf{X}^\top\mathbf{y}$ and the kernel estimate for the second estimate $\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_n)^{-1}\mathbf{y}$.
As discussed above ridge estimate has CC $O(n\ell^2+\ell^3)$ while the kernel estimate has CC $O(n^2\ell + n^3)$. Clearly the first method is preferable when $n \gg \ell$, i.e., the number of samples is much larger than the dimension of one sample point or equivalently the number of features. On the other hand, the kernel estimate is preferred when $\ell \gg n$, the number of features is much larger than the number of sample points.

4. (2 pts) Suppose we modify the problem (1) using a feature map $\phi : \mathbb{R}^\ell \to \mathbb{R}^d$ as follows:

$$\min_{\widetilde{\theta}\in\mathbb{R}^d} \left\| \mathbf{\Phi}\widetilde{\theta} - \mathbf{y} \right\|_2^2 + \lambda\|\widetilde{\theta}\|_2^2 \tag{2}$$

$$\text{where} \quad \mathbf{\Phi} = \begin{bmatrix} —\phi(\mathbf{x}_1)^\top— \\ \vdots \\ —\phi(\mathbf{x}_n)^\top— \end{bmatrix} \in \mathbb{R}^{n\times d}. \tag{3}$$

Can you provide a few reasons why we may want to do this? (Usually $d \geq \ell$ when we extend the $\mathbf{x}$ vectors in this fashion.)
Ans. Linear methods may not work with raw features x, and thus we may want to use higher order features for the data to fit a linear model well, e.g., polynomial features or cosine features etc. In simple words, linear methods are powerful when the features are rich enough, i.e., data which can not be described/classified using linear models in the raw feature space (i.e., just using x features) can be often be described/classified using rich enough features $\phi(\mathbf{x})$.

5. (8 pts) As discussed in class, often the choice of ϕ is (implicitly or explicitly) made such that

$$\phi(\mathbf{x})^\top\phi(\mathbf{z}) = k(x, z) \tag{4}$$

where $k : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}$ denotes the kernel function (that satisfies some nice properties). Can you compute the kernel functions for the following feature maps:

(a) $\phi(x) = [x_1 x_2, \frac{x_1^2}{\sqrt{2}}, \frac{x_2^2}{\sqrt{2}}]^\top$ where $\mathbf{x} \in \mathbb{R}^2$ with $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Ans. We have

$$\phi(\mathbf{x})^\top \phi(\mathbf{z}) = [x_1 x_2, \frac{x_1^2}{\sqrt{2}}, \frac{x_2^2}{\sqrt{2}}] \begin{bmatrix} z_1 z_2 \\ \frac{z_1^2}{\sqrt{2}} \\ \frac{z_2^2}{\sqrt{2}} \end{bmatrix} = x_1 x_2 z_1 z_2 + \frac{x_1^2 z_1^2}{2} + \frac{x_2^2 z_2^2}{2}$$

$$= \frac{(x_1 z_1 + x_2 z_2)^2}{2} = \frac{(\mathbf{x}^\top \mathbf{z})^2}{2}$$

If factor 2 is missing, loss of one point.

(b) $\phi(x) = [1, x, \frac{x^2}{\sqrt{2!}}, \frac{x^3}{\sqrt{3!}}, \ldots]$ where $x \in \mathbb{R}$.

Ans. We have

$$\phi(x)^\top \phi(z) = 1 + xz + \frac{x^2 z^2}{2!} + \frac{x^3 z^3}{3!} + \ldots = e^{xz}.$$

On the reverse side can you compute the feature map for the following kernel functions (it may not be possible in some cases):

(c) $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^2 + \mathbf{x}^\top \mathbf{z}$ for $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$.

Ans. We have

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= 1 + 2\mathbf{x}^\top \mathbf{z} + (\mathbf{x}^\top \mathbf{z})^2 + \mathbf{x}^\top \mathbf{z} \\ &= 1 + 3(\mathbf{x}^\top \mathbf{z}) + (\mathbf{x}^\top \mathbf{z})^2 \\ &= [1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2][1, \sqrt{3}z_1, \sqrt{3}z_2, \sqrt{2}z_1 z_2, z_1^2, z_2^2]^\top \end{aligned}$$

thus $\phi(\mathbf{x}) = [1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]^\top$. If the scalars are not right, loss of one point.

(d) $k(x, z) = e^{-(x-z)^2}$ for $x, z \in \mathbb{R}$.

Ans. We have

$$k(x, z) = e^{-(x-z)^2} = e^{-x^2} e^{2xz} e^{-z^2}$$

Note that e^{-x^2} and e^{-z^2} are already separate, so we only need to write e^{-2xz} as a product of functions of just x and just z which we already have from the previous part.

$$e^{2xz} = e^{\sqrt{2x} \cdot \sqrt{2z}} = [1, \sqrt{2}x, \frac{(\sqrt{2}x)^2}{\sqrt{2!}}, \frac{(\sqrt{2}x)^3}{\sqrt{3!}}, \ldots][1, \sqrt{2}z, \frac{(\sqrt{2}z)^2}{\sqrt{2!}}, \frac{(\sqrt{2}z)^3}{\sqrt{3!}}, \ldots]^\top.$$

Hence we have the feature as

$$\phi(x) = e^{-x^2}[1, \sqrt{2}x, \frac{(\sqrt{2}x)^2}{\sqrt{2!}}, \frac{(\sqrt{2}x)^3}{\sqrt{3!}}, \ldots]^\top$$

If people have written, we tried hard and couldn't find anything: 0.5 points.
Remark: **For Gaussian kernel, it is possible to find features in one dimensions, but if we use the kernel in higher dimensions, in general the closed form expression for ϕ is hard to find.**

6. (2 pts) What are the two different ways of computing the solution to the problem (2)?
Ans. The problem (2) is same as linear regression except that we have new features now: in place of $\mathbf{X}$ we use $\mathbf{\Phi}$ and hence the two estimates are given by

$$\widehat{\widetilde{\theta}} = (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathbf{I}_d)^{-1} \mathbf{\Phi}^\top \mathbf{y} \quad \text{and} \quad \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y}.$$

7. (5 pts) We now compare the complexity of the two estimates for a polynomial kernel. Compare and contrast the computational complexity of the two estimates when $\mathbf{x} \in \mathbb{R}^\ell$ and the feature map ϕ is chosen such that the corresponding kernel function (4) is given by

$$k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^p.$$

Discuss when is an estimate better than the other (on computational grounds for computing prediction given a new data point).
Ans. We will use the fact that $\phi(\mathbf{x})^\top \phi(\mathbf{z}) = k(\mathbf{x}, \mathbf{z})$. Let c denote the cost of computing $k(\mathbf{x}, \mathbf{z})$ once. In the polynomial kernel case, we have

$$\begin{aligned} \textbf{computing } (1 + \mathbf{x}^\top \mathbf{z})^p &\implies \textbf{one inner product in } \mathbb{R}^\ell \textbf{ and raising to power } p \\ &\implies c = O(\ell) + O(\log p) \textbf{ computational complexity}. \end{aligned}$$

Given $\mathbf{x}_{\text{new}}$, we have

$$\widehat{y}_{\text{new}} = \phi(\mathbf{x}_{\text{new}})^\top \widehat{\widetilde{\theta}}.$$

Since we have polynomial kernel, the size of $\phi(x_{\text{new}}) \in \mathbb{R}^d$ is $d = \binom{\ell+p}{\ell} \sim p^\ell$. To compute the ridge-like estimate:

CC for computing $(\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathbf{I}_d)^{-1} \mathbf{\Phi}^\top \mathbf{y}$ is $O(nd^2 + d^3)$ where $d = p^\ell$

and then for computing $\phi(\mathbf{x}_{\text{new}})^\top \widehat{\widetilde{\theta}}$ the CC is $O(d)$

So if we have n_{train} and n_{test} training and test points, the net CC is

$$O(n_{\text{train}} d^2 + d^3 + n_{\text{test}} d) \quad \text{where } d = p^\ell.$$

Kernel-estimate:

$$\begin{aligned} \widehat{y}_{\text{new}} &= \phi(\mathbf{x}_{\text{new}})^\top \widehat{\widetilde{\theta}} \\ &= \phi(\mathbf{x}_{\text{new}})^\top \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \\ &= [k(\mathbf{x}_{\text{new}}, \mathbf{x}_1), \ldots, k(\mathbf{x}_{\text{new}}, \mathbf{x}_{n_{\text{train}}})](\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{y} \end{aligned}$$

Computing $(\mathbf{K}+\lambda \mathbf{I}_n)^{-1}\mathbf{y}$ where $n = n_{\text{train}}$ takes

$$O(\underbrace{n_{\text{train}}^2 c}_{\text{computing K}} + \underbrace{n_{\text{train}}^3}_{\text{inversion}} + \underbrace{n_{\text{train}}^2}_{\text{matrix-vector product}}) \quad \text{where } c = O(\ell + \log p)$$

and then computing the new $[k(\mathbf{x}_{\text{new}}, \mathbf{x}_1), \ldots, k(\mathbf{x}_{\text{new}}, \mathbf{x}_{n_{\text{train}}})]$ takes $O(n_{\text{train}} c)$ time for each new data point and then the inner product takes $O(n_{\text{train}})$ time. Thus the net complexity for computing predictions for n_{test} data points is

$$\begin{aligned} &O(n_{\text{train}}^2 c + n_{\text{train}}^3 + n_{\text{train}}^2 + n_{\text{test}}(n_{\text{train}}(1+c))) \\ &= O(n_{\text{train}}^2 c + n_{\text{train}}^3 + n_{\text{test}} n_{\text{train}} c) \quad \text{where } c = \ell + \log p. \end{aligned}$$

***Remark:* If $n_{\text{test}} = 1$ and $n_{\text{train}} = n$, the previous examples can be simplified to**

$$O(nd^2 + d^3) \quad \textbf{vs} \quad O(n^2 \log d + n^3)$$

where we have used that $c = \ell + \log p = O(\ell \log p) = O(\log d)$. Thus if d is large, Kernel estimate is much easier to compute.
Deduct one point if only these simplified expressions and discussions are provided.

3 LDA and linear regression (10 pts)

ESLII book https://web.stanford.edu/~hastie/Papers/ESLII.pdf: Question 4.2 (all 5 parts); Page 135-136

(a)

Ans. We follow the steps from Page 5 of the notes n18.pdf: For LDA, for $j = 1, 2$, we have

$$\begin{aligned} \mathbb{P}(Y = j | X = x) &\propto \mathbb{P}(Y = j) p(X = x | Y = j) \\ &\propto \frac{N_j}{N} \cdot \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det(\widehat{\mathbf{\Sigma}})}} \exp\left(-\frac{(x - \widehat{\mu}_j)^\top \widehat{\mathbf{\Sigma}}^{-1} (x - \widehat{\mu_j})}{2} \right) \end{aligned}$$

where

$$\widehat{\mu}_j = \frac{1}{N_j} \sum_{Y_i = j} x_i$$

is the sample mean of j-th class. And

$$\widehat{\mathbf{\Sigma}} = \frac{1}{N-2} \sum_{i=1}^{N} (x_i - \widehat{\mu}_{Y_i})(x_i - \widehat{\mu}_{Y_i})^\top$$

is the estimate for the common-covariance across all classes.
Remark: Note that the choice of the normalization with $N-2$ or N does not

really matter when N is large. N is chosen for MLE estimator while $N-2$ is chosen to make the variance estimator unbiased. Recall the sample variance definition, the normalization there is often $N-1$ in place of N. A similar thing is also being done here.

A point x is classified to class 2 if

$$\begin{aligned}
&\mathbb{P}(Y=2|X=x) > \mathbb{P}(Y=1|X=x) \\
\iff \quad & \frac{N_2}{N}\frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\det(\widehat{\mathbf{\Sigma}})}}e^{\left(-\frac{(x-\widehat{\mu}_1)^\top \widehat{\mathbf{\Sigma}}^{-1}(x-\widehat{\mu_1})}{2}\right)} > \frac{N_1}{N}\frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\det(\widehat{\mathbf{\Sigma}})}}e^{\left(-\frac{(x-\widehat{\mu}_2)^\top \widehat{\mathbf{\Sigma}}^{-1}(x-\widehat{\mu_2})}{2}\right)} \\
\iff \quad & -\frac{(x-\widehat{\mu}_2)^\top \widehat{\mathbf{\Sigma}}^{-1}(x-\widehat{\mu_2})}{2} > -\frac{(x-\widehat{\mu}_1)^\top \widehat{\mathbf{\Sigma}}^{-1}(x-\widehat{\mu_1})}{2} + \log\frac{N_1}{N} - \log\frac{N_2}{N} \\
\iff \quad & x^\top \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2-\widehat{\mu}_1) > \frac{1}{2}\widehat{\mu}_2^\top \widehat{\mathbf{\Sigma}}^{-1}\widehat{\mu}_2 - \frac{1}{2}\widehat{\mu}_1^\top \widehat{\mathbf{\Sigma}}^{-1}\widehat{\mu}_1 + \log\frac{N_1}{N} - \log\frac{N_2}{N}
\end{aligned}$$

as claimed.

(b)

Ans. For OLS solution, we have

$$\begin{aligned}
\nabla_\beta \sum_{i=1}^N (y_i-\beta_0-\beta^\top x_i)^2 = 0 &\Longrightarrow \sum_{i=1}^N (y_i-\beta_0-\beta^\top x_i)(-x_i) = 0 \\
&\Longrightarrow \left(\sum_{i=1}^N x_i x_i^\top\right)\beta + \beta_0 \sum_{i=1}^N x_i = \sum_{i=1}^N y_i x_i \qquad (5)
\end{aligned}$$

$$\begin{aligned}
\textbf{and } \frac{\partial}{\partial\beta_0}\sum_{i=1}^N (y_i-\beta_0-\beta^\top x_i)^2 = 0 &\Longrightarrow \sum_{i=1}^N (y_i-\beta_0-\beta^\top x_i)(-1) = 0 \\
&\Longrightarrow \beta_0 = \frac{1}{N}\sum_{i=1}^N y_i - (\frac{1}{N}\sum_{i=1}^N x_i)^\top\beta \qquad (6)
\end{aligned}$$

Now we note the following things:

$$\begin{aligned}
\sum_{i=1}^N y_i &= \sum_{i\in\textbf{class 1}} y_i - \sum_{i\in\textbf{class 2}} y_i = N_1\left(\frac{-N}{N_1}\right) + N_2\left(\frac{N}{N_2}\right) = 0 \\
\sum_{i=1}^N x_i &= \sum_{i\in\textbf{class 1}} x_i + \sum_{i\in\textbf{class 2}} x_i = N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2 \\
\sum_{i=1}^N x_i y_i &= \sum_{i\in\textbf{class 1}} y_i x_i + \sum_{i\in\textbf{class 2}} y_i x_i = \left(\frac{-N}{N_1}\right)N_1\widehat{\mu}_1 + \left(\frac{N}{N_2}\right)N_2\widehat{\mu}_2 = N(\widehat{\mu}_2-\widehat{\mu}_1).
\end{aligned}$$

Hence, the equations (5) and (6) simplify to

$$\beta_0 = \frac{1}{N}\sum_{i=1}^{N} y_i - (\frac{1}{N}\sum_{i=1}^{N} x_i)^\top \beta = 0 - \left(\frac{N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2}{N}\right)^\top \beta$$

$$\implies \quad \beta_0 = -\left(\frac{N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2}{N}\right)^\top \beta \quad \textbf{and} \tag{7}$$

$$\left(\sum_{i=1}^{N} x_i x_i^\top\right)\beta + \left(\sum_{i=1}^{N} x_i\right)\beta_0 = \sum_{i=1}^{N} y_i x_i$$

$$\implies \quad \left(\sum_{i=1}^{N} x_i x_i^\top\right)\beta + (N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2)\beta_0 = N(\widehat{\mu}_1 - \widehat{\mu}_2)$$

$$\implies \quad \left(\sum_{i=1}^{N} x_i x_i^\top\right)\beta - (N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2)\frac{1}{N}(N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2)^\top\beta = N(\widehat{\mu}_1 - \widehat{\mu}_2)$$

$$\implies \quad \left(\sum_{i=1}^{N} x_i x_i^\top - \frac{1}{N}(N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2)(N_1\widehat{\mu}_1 + N_2\widehat{\mu}_2)^\top\right)\beta = N(\widehat{\mu}_1 - \widehat{\mu}_2). \tag{8}$$

Note that the question asks us to show that (Equation 4.56)

$$\left[(N-2)\widehat{\mathbf{\Sigma}} + \frac{N_1 N_2}{N}(\widehat{\mu}_2 - \widehat{\mu}_1)(\widehat{\mu}_2 - \widehat{\mu}_1)^\top\right]\beta = N(\widehat{\mu}_1 - \widehat{\mu}_2) \tag{9}$$

where we have also used the expression for the matrix $\mathbf{\Sigma}_B = (\widehat{\mu}_2 - \widehat{\mu}_1)(\widehat{\mu}_2 - \widehat{\mu}_1)^\top$. Thus it is left to show that the LHS of the equation (9) is equal to the LHS of the equation (8). Simplifying the LHS of equation (9), we find that

$$\left[(N-2)\widehat{\mathbf{\Sigma}} + \frac{N_1 N_2}{N}(\widehat{\mu}_2 - \widehat{\mu}_1)(\widehat{\mu}_2 - \widehat{\mu}_1)^\top\right] = \sum_{i\in\textbf{class 1}} (x_i - \widehat{\mu}_1)(x_i - \widehat{\mu}_1)^\top + \sum_{i\in\textbf{class 2}} (x_i - \widehat{\mu}_2)(x_i - \widehat{\mu}_2)^\top + \frac{N_1 N_2}{N}(\widehat{\mu}_1\widehat{\mu}_1^\top + \widehat{\mu}_2\widehat{\mu}_2^\top - \widehat{\mu}_1\widehat{\mu}_2^\top - \widehat{\mu}_2\widehat{\mu}_1^\top) \tag{10}$$

Note that

$$\begin{aligned}
\sum_{i\in\textbf{class 1}} (x_i - \widehat{\mu}_1)(x_i - \widehat{\mu}_1)^\top &= \sum_{i\in\textbf{class 1}} (x_i x_i^\top + \widehat{\mu}_1\widehat{\mu}_1^\top - x_i\widehat{\mu}_1^\top - \widehat{\mu}_1 x_i^\top) \\
&= \left(\sum_{i\in\textbf{class 1}} x_i x_i^\top\right) + N_1\widehat{\mu}_1\widehat{\mu}_1^\top - \left(\sum_{i\in\textbf{class 1}} x_i\right)\widehat{\mu}_1^\top - \widehat{\mu}_1\left(\sum_{i\in\textbf{class 1}} x_i\right)^\top) \\
&= \left(\sum_{i\in\textbf{class 1}} x_i x_i^\top\right) + N_1\widehat{\mu}_1\widehat{\mu}_1^\top - N_1\widehat{\mu}_1\widehat{\mu}_1^\top - \widehat{\mu}_1 N_1\widehat{\mu}_1^\top \\
&= \left(\sum_{i\in\textbf{class 1}} x_i x_i^\top\right) - N_1\widehat{\mu}_1\widehat{\mu}_1^\top.
\end{aligned}$$

Similarly, we can compute that

$$\sum_{i \in \textbf{class 2}} (x_i - \widehat{\mu}_1)(x_i - \widehat{\mu}_1)^\top = \left(\sum_{i \in \textbf{class 2}} x_i x_i^\top \right) - N_2 \widehat{\mu}_2 \widehat{\mu}_2^\top .$$

Thus the expression on RHS of equation (10) can be simplified as

$$\begin{aligned}
&\sum_{i \in \textbf{class 1}} (x_i - \widehat{\mu}_1)(x_i - \widehat{\mu}_1)^\top + \sum_{i \in \textbf{class 2}} (x_i - \widehat{\mu}_2)(x_i - \widehat{\mu}_2)^\top + \frac{N_1 N_2}{N} (\widehat{\mu}_1 \widehat{\mu}_1^\top + \widehat{\mu}_2 \widehat{\mu}_2^\top - \widehat{\mu}_1 \widehat{\mu}_2^\top - \widehat{\mu}_2 \widehat{\mu}_1^\top) \\
&= \left(\sum_{i \in \textbf{class 1}} x_i x_i^\top \right) + \left(\sum_{i \in \textbf{class 2}} x_i x_i^\top \right) - N_1 \widehat{\mu}_1 \widehat{\mu}_1^\top - N_2 \widehat{\mu}_2 \widehat{\mu}_2^\top + \frac{N_1 N_2}{N} (\widehat{\mu}_1 \widehat{\mu}_1^\top + \widehat{\mu}_2 \widehat{\mu}_2^\top - \widehat{\mu}_1 \widehat{\mu}_2^\top - \widehat{\mu}_2 \widehat{\mu}_1^\top) \\
&= \sum_{i=1}^{N} x_i x_i^\top - \frac{N_1 N_2}{N} (\widehat{\mu}_1 \widehat{\mu}_2^\top + \widehat{\mu}_2 \widehat{\mu}_1^\top) + \left(\frac{N_1 N_2}{N} - N_1 \right) \widehat{\mu}_1 \widehat{\mu}_1^\top + \left(\frac{N_1 N_2}{N} - N_2 \right) \widehat{\mu}_2 \widehat{\mu}_2^\top \\
&\overset{(i)}{=} \sum_{i=1}^{N} x_i x_i^\top - \frac{N_1 N_2}{N} (\widehat{\mu}_1 \widehat{\mu}_2^\top + \widehat{\mu}_2 \widehat{\mu}_1^\top) - \left(\frac{N_1^2}{N} \right) \widehat{\mu}_1 \widehat{\mu}_1^\top - \left(\frac{N_2^2}{N} \right) \widehat{\mu}_2 \widehat{\mu}_2^\top \\
&\overset{(i)}{=} \sum_{i=1}^{N} x_i x_i^\top - \frac{1}{N} (N_1 \widehat{\mu}_1 + N_2 \widehat{\mu}_2)(N_1 \widehat{\mu}_1 + N_2 \widehat{\mu}_2)^\top,
\end{aligned}$$

which is the same as the LHS of the equation (8) and the proof is complete. Here in step (i) we have used the fact that $N_1 + N_2 = N \implies N_1 N_2 - N N_1 = -N_1^2$ and similarly that $N_1 N_2 - N N_2 = -N_2^2$. The proof is now done.

(c)

Ans. From equation 4.56 or (9), we have that

$$\begin{aligned}
& \left[(N-2)\widehat{\mathbf{\Sigma}} + \frac{N_1 N_2}{N} (\widehat{\mu}_2 - \widehat{\mu}_1)(\widehat{\mu}_2 - \widehat{\mu}_1)^\top \right] \beta = N(\widehat{\mu}_1 - \widehat{\mu}_2) \\
\implies \quad & (N-2)\widehat{\mathbf{\Sigma}}\beta + \frac{N_1 N_2}{N} (\widehat{\mu}_2 - \widehat{\mu}_1)(\widehat{\mu}_2 - \widehat{\mu}_1)^\top \beta = N(\widehat{\mu}_1 - \widehat{\mu}_2) \\
\implies \quad & (N-2)\widehat{\mathbf{\Sigma}}\beta = \underbrace{\left[N - \left(\frac{N_1 N_2 \cdot (\widehat{\mu}_2 - \widehat{\mu}_1)^\top \beta}{N} \right) \right]}_{\textbf{scalar}} (\widehat{\mu}_1 - \widehat{\mu}_2) \\
\implies \quad & \beta \propto \widehat{\mathbf{\Sigma}}^{-1} (\widehat{\mu}_2 - \widehat{\mu}_1).
\end{aligned}$$

Doing this much work gets full points.
In simple words, although the RHS depends on β, it just means that we need to find a β in the direction of $\widehat{\mathbf{\Sigma}}(\widehat{\mu}_2 - \widehat{\mu}_1)$ such that it satisfies that equation. To find exact solution, we assume $\beta = c\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1)$ and plug it into the

equation and solve for c:

$$(N-2)c(\widehat{\mu}_2-\widehat{\mu}_1)=\left[N-\left(\frac{N_1N_2\cdot(\widehat{\mu}_2-\widehat{\mu}_1)^\top\widehat{\mathbf{\Sigma}}^{-1}c(\widehat{\mu}_2-\widehat{\mu}_1)}{N}\right)\right](\widehat{\mu}_1-\widehat{\mu}_2)$$

$$\implies \quad c_\star=\frac{N}{N-2+\frac{N_1N_2}{N}\cdot(\widehat{\mu}_2-\widehat{\mu}_1)^\top\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2-\widehat{\mu}_1)}$$

$$\implies \quad \widehat{\beta}_{\mathbf{OLS}}=c_\star\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2-\widehat{\mu}_1).$$

Remark: **Students are not expected to compute the exact solution.**

(d)

Ans. Let ω,δ be two new distinct codings for the labels of class 1 and 2 respectively and let y_i' denote the new codings. Then we are supposed to show that the solution to the problem

$$\min_{\beta,\beta_0}\sum_{i=1}^{N}(c_1y_i+c_2-\beta_0-x_i^\top\beta)^2$$

satisfies

$$\beta\propto\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2-\widehat{\mu}_1).$$

First, we note that we can uniquely solve for c_1 and c_2 such that

$$\omega=c_1(-N/N_1)+c_2\quad\textbf{and}\quad\delta=c_1(N/N_2)+c_2$$

In other words, given any coding ω,δ we can find c_1 and c_2 uniquely such that

$$y_i'=c_1y_i+c_2.$$

Thus the least squares problem becomes:

$$\min_{\beta,\beta_0}\sum_{i=1}^{N}(y_i'-\beta_0-x_i^\top\beta)^2=\min_{\beta,\beta_0}\sum_{i=1}^{N}(c_1y_i+c_2-\beta_0-x_i^\top\beta)^2$$

Doing a change of variables: $\beta_0\to c_2+c_1\widetilde{\beta}_0$ and $\beta\to c_1\widetilde{\beta}$, we find that the problem simplifies to

$$\begin{aligned}\min_{\beta,\beta_0}\sum_{i=1}^{N}(c_1y_i+c_2-\beta_0-x_i^\top\beta)^2&=\min_{\widetilde{\beta},\widetilde{\beta_0}}(c_1y_i+c_2-c_2-c_1\widetilde{\beta}_0-x_i^\top c_1\widetilde{\beta})^2\\&=c_1^2\min_{\widetilde{\beta},\widetilde{\beta_0}}(y_i-\widetilde{\beta}_0-x_i^\top\widetilde{\beta})^2\end{aligned}$$

and whoa, we recover the original problem where y_i takes values as $-N/N_1$ and N/N_2 implying that optimal $\widetilde{\beta}$ points in the LDA vector direction, i.e.,

$\widetilde{\beta} \propto \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1)$ **Since β is simply a scalar multiple of $\widetilde{\beta}$, the solution for the original OLS problem with the new codings is also pointing in the same direction.**
Remark: **Any other valid argument based on re-simplifying the equations is also correct. BUT simply saying oh its obvious or some trivial argument like that will not get any points.**

(e)

Ans. Using $\widehat{\beta} = c\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1)$ and $\widehat{\beta}_0 = (\frac{N_1}{N}\widehat{\mu}_1 + \frac{N_2}{N}\widehat{\mu}_2)^\top \widehat{\beta}$ (from equation (7)), we find that

$$\widehat{f}(x) = \widehat{\beta}_0 + x^\top \widehat{\beta} = ((\frac{N_1}{N}\widehat{\mu}_1 + \frac{N_2}{N}\widehat{\mu}_2)^\top + x)^\top c\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1)$$

Thus the decision rule (for class 2) is

$$\widehat{f}(x) > 0 \iff x^\top \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1) + (\frac{N_1}{N}\widehat{\mu}_1 + \frac{N_2}{N}\widehat{\mu}_2)^\top \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1) > 0$$

Comparing with LDA rule (equation 4.11 in the ESL book Page 109):

$$x^\top \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1) > \frac{1}{2}\widehat{\mu}_2^\top \widehat{\mathbf{\Sigma}}^{-1}\widehat{\mu}_2 - \frac{1}{2}\widehat{\mu}_1^\top \widehat{\mathbf{\Sigma}}^{-1}\widehat{\mu}_1 + \log\frac{N_1}{N} - \log\frac{N_2}{N}$$

we see that the two rules are equivalent only if $N_1 = N_2 = N/2$.

4 Applied problem for classification (7 pts)

ISL Book: 4.11 (all 7 parts) **Show code in the write-up pdf for all parts.**

(a) We look at some density plots, boxplots and pairwise scatter plots of some important features. From the plots, we se that most of the plotted features are useful for classifying; although cylinders and displacement (highly correlated among themselves) seem to have a different distribution under the two models.

(Note that density plot for variables that take only discrete values are not a good idea in general – although we provide it here for your visualization.)

```
library(ISLR)
library(ggplot2)
data("Auto")

# I created two variables, one factor and one no-factor.
# factor is useful for EDA with ggplot, and no-factor is useful
# for doing classification. You don't have to do the same.
# This is simply to illustrate the usefulness of the two ways
# to convert the feature.

mpgfactor <- factor(ifelse(Auto$mpg<=median(Auto$mpg), 0, 1))
```

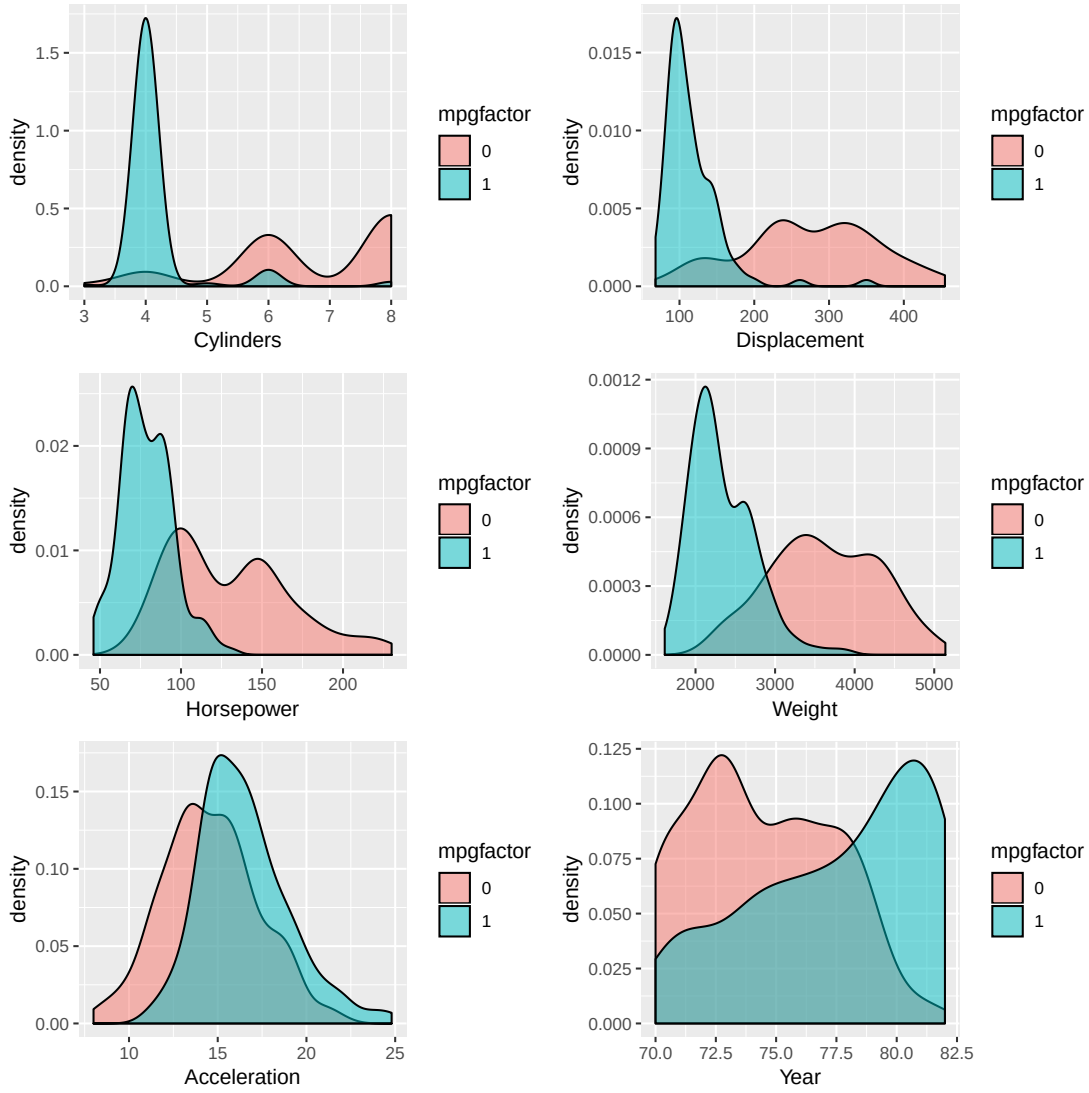
```
mpg01 <- ifelse(Auto$mpg<=median(Auto$mpg), 0, 1)
Auto <- data.frame(Auto, mpgfactor, mpg01)

colnames(Auto)
```

```
##  [1] "mpg"          "cylinders"    "displacement" "horsepower"
##  [5] "weight"       "acceleration" "year"         "origin"
##  [9] "name"         "mpgfactor"    "mpg01"
```

```
library(gridExtra)

# correlation with labels
cor(Auto$mpg01, Auto[2:7])
```

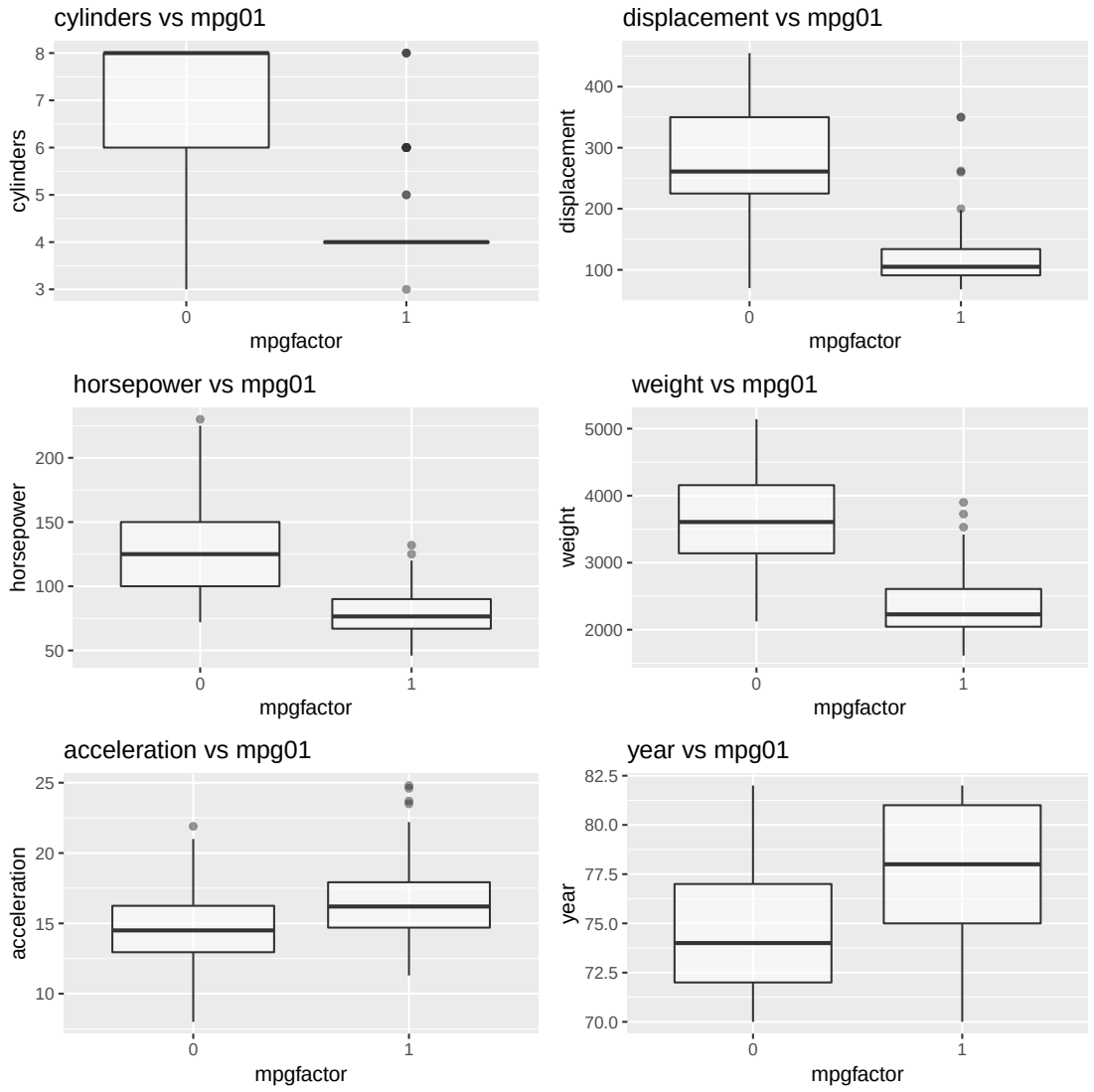
```
##       cylinders displacement horsepower     weight acceleration      year
## [1,] -0.7591939   -0.7534766 -0.6670526 -0.7577566    0.3468215 0.4299042
```

```
# density plots
p1 <- ggplot(Auto) + geom_density(aes(x=cylinders, fill=mpgfactor),
                                  alpha=0.5)  + labs(x="Cylinders")
p2 <- ggplot(Auto) + geom_density(aes(x=displacement, fill=mpgfactor),
                                  alpha=0.5)  + labs(x="Displacement")
p3 <- ggplot(Auto) + geom_density(aes(x=horsepower, fill=mpgfactor),
                                  alpha=0.5)  + labs(x="Horsepower")
p4 <- ggplot(Auto) + geom_density(aes(x=weight, fill=mpgfactor),
                                  alpha=0.5)  + labs(x="Weight")
p5 <- ggplot(Auto) + geom_density(aes(x=acceleration, fill=mpgfactor),
                                  alpha=0.5)  + labs(x="Acceleration")
p6<- ggplot(Auto) + geom_density(aes(x=year, fill=mpgfactor),
                                 alpha=0.5)  + labs(x="Year")
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=2)
```

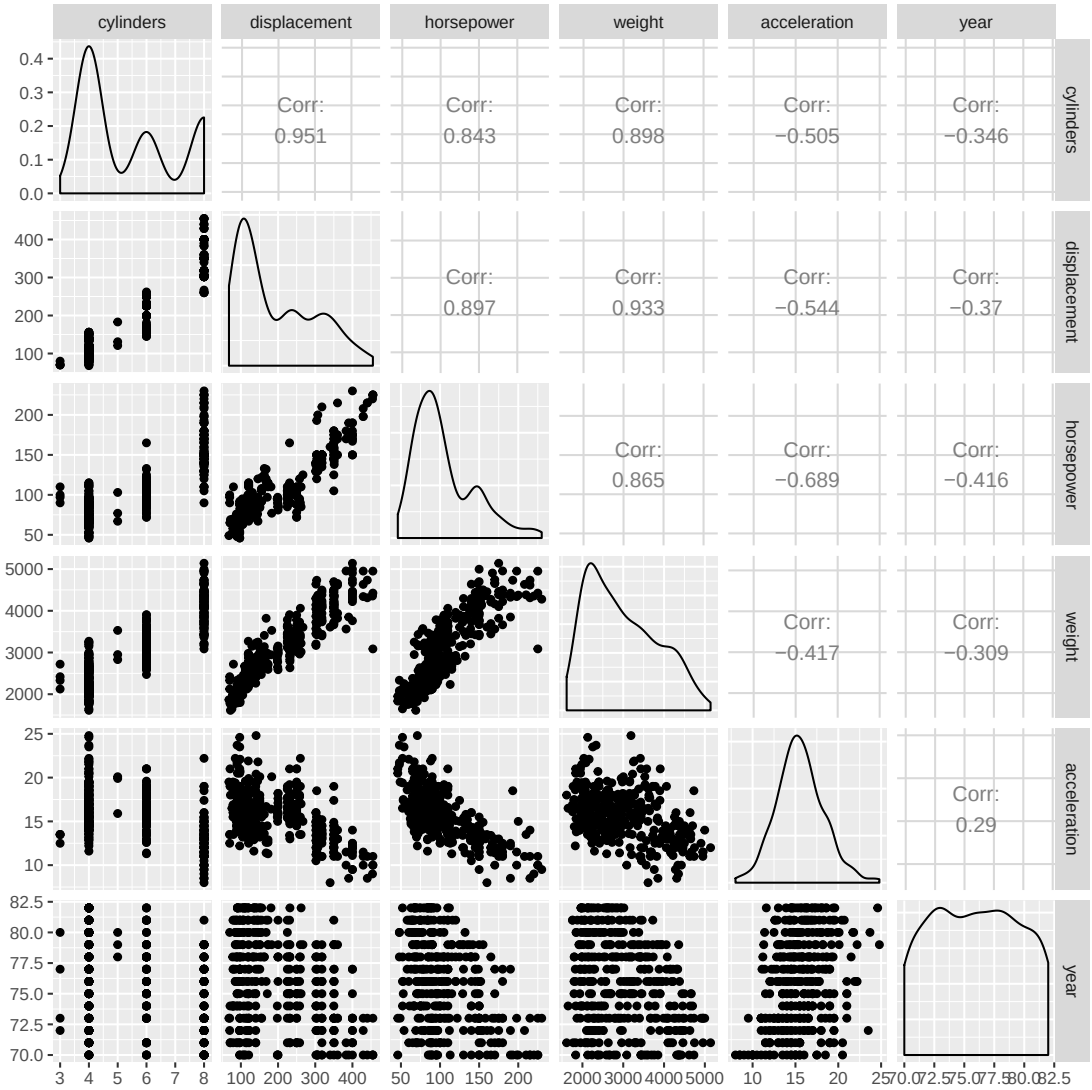



```
# box plots
p1 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=cylinders),
                                  alpha=0.5)  + labs(title = "cylinders vs mpg01")
p2 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=displacement),
                                  alpha=0.5)  + labs(title = "displacement vs mpg01")
p3 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=horsepower),
                                  alpha=0.5)  + labs(title = "horsepower vs mpg01")
p4 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=weight),
                                  alpha=0.5)  + labs(title = "weight vs mpg01")
p5 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=acceleration),
                                  alpha=0.5)  + labs(title = "acceleration vs mpg01")
p6 <- ggplot(Auto) + geom_boxplot(aes(x=mpgfactor, y=year),
                                  alpha=0.5)  + labs(title = "year vs mpg01")
```

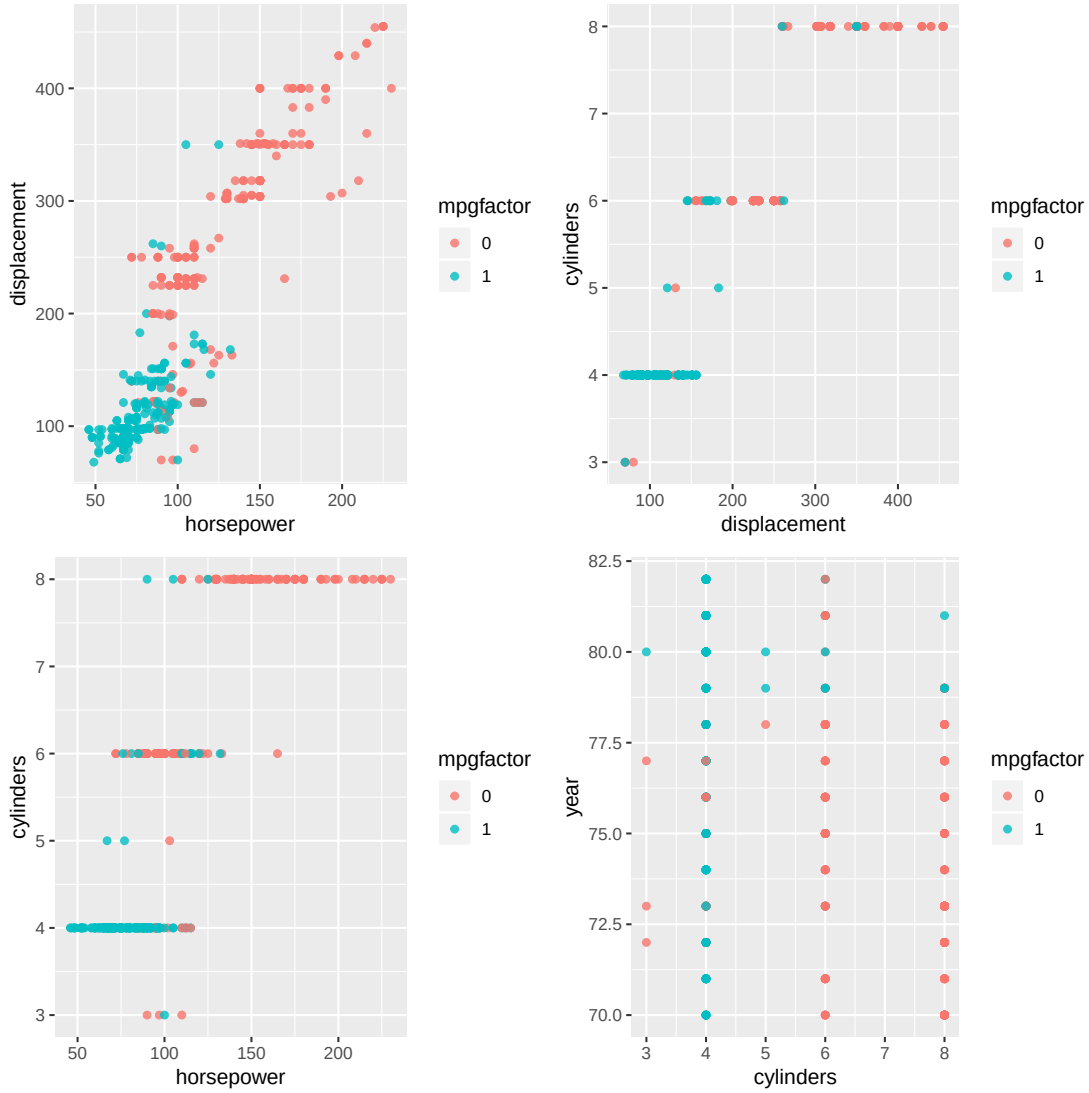
```
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=2)
```




```
# Correlations
library("GGally")
ggpairs(Auto[, 2:7])
```




```r
# some pairwise scatter plots
p1 <- ggplot(Auto) +
  geom_point(aes(x=horsepower, y=displacement, color=mpgfactor), alpha=0.8)

p2 <- ggplot(Auto) +
  geom_point(aes(x=displacement, y=cylinders, color=mpgfactor), alpha=0.8)

p3 <- ggplot(Auto) +
  geom_point(aes(x=horsepower, y=cylinders, color=mpgfactor), alpha=0.8)

p4 <- ggplot(Auto) +
  geom_point(aes(x=cylinders, y=year, color=mpgfactor), alpha=0.8)
grid.arrange(p1, p2, p3, p4, ncol=2)
```



(b) Splitting the data

```
n <- nrow(Auto)
set.seed(100)
train_idx <- sample(1:n, size=0.7*n)
Auto_train <- Auto[train_idx,]
Auto_test <- Auto[-train_idx,]
```

(c) LDA (we use only top four features based on correlation and other EDA from previous parts).

```r
# LDA
library(MASS)
lda_auto <- lda(mpg01 ~ cylinders+displacement+horsepower+weight,
                data=Auto, subset = train_idx)
lda_auto
```

```
## Call:
## lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto,
##     subset = train_idx)
##
## Prior probabilities of groups:
##         0         1
## 0.5109489 0.4890511
##
## Group means:
##   cylinders displacement horsepower   weight
## 0  6.678571     272.3071  130.62143 3623.614
## 1  4.156716     115.1269   77.29104 2340.119
##
## Coefficients of linear discriminants:
##                         LD1
## cylinders    -4.919825e-01
## displacement -7.876694e-05
## horsepower    1.394448e-03
## weight       -9.071885e-04
```

```r
lda_pred <- predict(lda_auto, Auto)
cat("LDA Train error:",mean(lda_pred$class[train_idx] != Auto$mpg01[train_idx]),
    "LDA Test error:", mean(lda_pred$class[-train_idx] != Auto$mpg01[-train_idx]))
```

```
## LDA Train error: 0.1094891 LDA Test error: 0.09322034
```

(d) QDA

```r
# QDA
qda_auto <- qda(mpg01 ~ cylinders+displacement+horsepower+weight,
                data=Auto_train)

qda_pred <- predict(qda_auto, Auto)
cat("QDA Train error:", mean(qda_pred$class[train_idx] != Auto$mpg01[train_idx]),
    "QDA Test error:", mean(qda_pred$class[-train_idx] != Auto$mpg01[-train_idx]))
```

```
## QDA Train error: 0.1058394 QDA Test error: 0.07627119
```

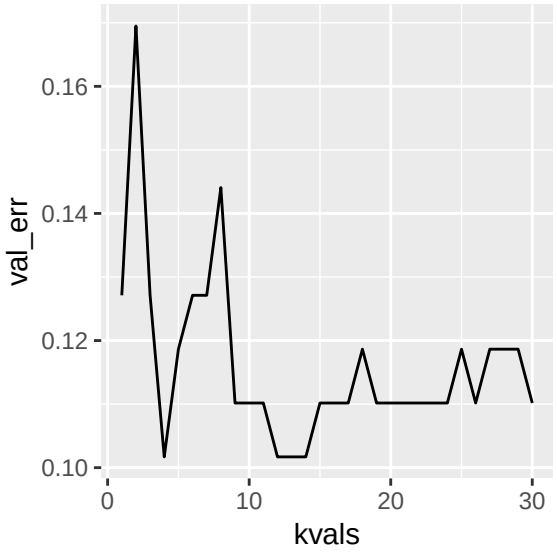
(e) Logistic Regression

```
# Logistic Regression
log_auto <- glm(mpg01~cylinders+displacement+horsepower+weight,
                data=Auto_train, family = 'binomial')
summary(log_auto)

##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight, family = "binomial", data = Auto_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2794  -0.2552  -0.0024   0.3721   3.4005
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  11.8333557  2.1083828   5.613 1.99e-08 ***
## cylinders    -0.1385465  0.4378371  -0.316  0.75167
## displacement -0.0104388  0.0104052  -1.003  0.31575
## horsepower   -0.0532929  0.0167749  -3.177  0.00149 **
## weight       -0.0015868  0.0008575  -1.851  0.06424 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.71  on 273  degrees of freedom
## Residual deviance: 149.61  on 269  degrees of freedom
## AIC: 159.61
##
## Number of Fisher Scoring iterations: 7

logistic_prob <- predict(log_auto, Auto, type="response")
logistic_pred <- 0*1:n
logistic_pred[logistic_prob>0.5] = 1
cat("Logistic Train error:", mean(logistic_pred[train_idx] != Auto$mpg01[train_idx]),
    "Logistic Test error:", mean(logistic_pred[-train_idx] != Auto$mpg01[-train_idx]))

## Logistic Train error: 0.1021898 Logistic Test error: 0.08474576
```

(f) K-NN: We try different values of k and plot the errors. We see that for our settings (a seed should be set) it is 4. The answer can vary for students; a good range of k (>= 5

values) should be tried.

```
library(class)
set.seed(1000)
features <- c("cylinders", "displacement", "horsepower", "weight")
kvals <- 1:30
val_err <- 0*1:30
for (k in kvals){
  knn <- knn(Auto_train[, features], Auto_test[, features],
             Auto_train[, "mpg01"], k=k)
  val_err[k] <- mean(knn!=Auto_test$mpg01)
}
ggplot() + geom_line(aes(x=kvals, y=val_err))
```




```
which.min(val_err)

## [1] 4
```