

Repeat the R codes in 4.7 Lab: Classification Methods.

Implementation of Chapter 4.7 Laboratory Exercises from 'An Introduction to Statistical Learning' (James et al., 2021), with detailed annotations.

Jason Huang

2025-11-27

目錄

4.7 Lab: Classification Methods	2
4.7.1 The Stock Market Data	2
4.7.2 Logistic Regression	5
Smarket 第一版作弊的表現 (glm 無測試集)	7
Smarket 第二版的表現 (glm 有測試集)	8
Smarket 第三版的表現 (只使用 Lag1 和 Lag2 的 GLM)	9
4.7.3 Linear Discriminant Analysis	10
Smarket 第四版的表現 (只使用 Lag1 和 Lag2 的 LDA)	12
4.7.4 Quadratic Discriminant Analysis	13
Smarket 第五版的表現 (只使用 Lag1 和 Lag2 的 QDA)	14
4.7.5 Naive Bayes	15
Smarket 第六版的表現 (只使用 Lag1 和 Lag2 的 Naive Bayes)	16
4.7.6 K-Nearest Neighbors	17
Smarket 第七版的表現 (只使用 Lag1 和 Lag2 的 KNN($k = 1$))	17
Smarket 第八版的表現 (只使用 Lag1 和 Lag2 的 KNN($k = 3$))	18
Caravan 第一版的表現 (KNN($k = 1$))	20
Caravan 第二版的表現 (KNN($k = 3$))	20
Caravan 第三版的表現 (KNN($k = 5$))	21
Caravan 第四版的表現 (GLM(glm.probs > .5))	21
Caravan 第五版的表現 (GLM(glm.probs > .25))	22
4.7.7 Poisson Regression	22

4.7 Lab: Classification Methods

4.7.1 The Stock Market Data

Smarket 資料集: 包含 2001 年初至 2005 年底期間標普 500 指數 1250 個交易日的收益率百分比。對於每個日期，我們記錄了前五個交易日 (Lag1 至 Lag5) 的收益率百分比，以及以下三項。

1. Volume 成交量 (前一天的交易股數，單位為十億股)
2. Today 當日收益率 (當日收益率百分比)
3. Direction 方向 (當日市場上漲或下跌)

```
library(ISLR2)
names(Smarket)

[1] "Year"      "Lag1"       "Lag2"       "Lag3"       "Lag4"       "Lag5"
[7] "Volume"    "Today"     "Direction"
```

維度: 看看這個資料集有幾列 (Rows) 和幾行 (Columns)。

```
dim(Smarket)
```

[1] 1250 9

資料分佈概況: 針對每一欄 (Column) 提供統計摘要。

```
summary(Smarket)
```

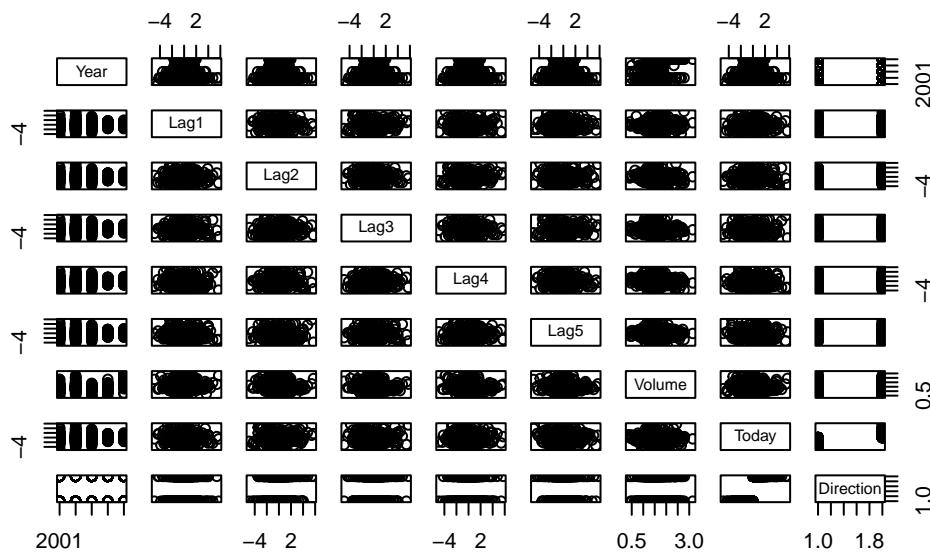
	Year	Lag1	Lag2	Lag3
Min.	:2001	Min. : -4.922000	Min. : -4.922000	Min. : -4.922000
1st Qu.	:2002	1st Qu. : -0.639500	1st Qu. : -0.639500	1st Qu. : -0.640000
Median	:2003	Median : 0.039000	Median : 0.039000	Median : 0.038500
Mean	:2003	Mean : 0.003834	Mean : 0.003919	Mean : 0.001716
3rd Qu.	:2004	3rd Qu. : 0.596750	3rd Qu. : 0.596750	3rd Qu. : 0.596750
Max.	:2005	Max. : 5.733000	Max. : 5.733000	Max. : 5.733000

	Lag4	Lag5	Volume	Today
Min.	: -4.922000	Min. : -4.92200	Min. : 0.3561	Min. : -4.922000
1st Qu.	: -0.640000	1st Qu. : -0.64000	1st Qu. : 1.2574	1st Qu. : -0.639500
Median	: 0.038500	Median : 0.03850	Median : 1.4229	Median : 0.038500
Mean	: 0.001636	Mean : 0.00561	Mean : 1.4783	Mean : 0.003138
3rd Qu.	: 0.596750	3rd Qu. : 0.59700	3rd Qu. : 1.6417	3rd Qu. : 0.596750
Max.	: 5.733000	Max. : 5.73300	Max. : 3.1525	Max. : 5.733000

	Direction
Down:	602
Up :	648

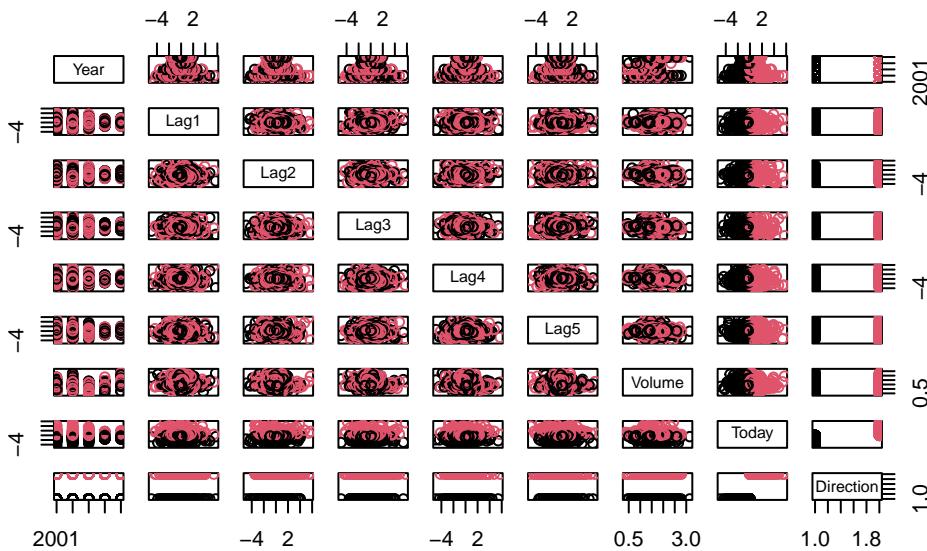
```
# 散佈圖矩陣：看看是否有趨勢存在。
```

```
pairs(Smarket)
```



```
# 把 Direction 為 Up 的點畫紅色，Down 的畫黑色。
```

```
pairs(Smarket, col = Smarket$Direction)
```



```
# 對「每一個變數（欄位）」跟「其他所有變數」兩兩互算相關性。
```

```
# 第 9 欄是文字 Direction，要把它拿掉才能算。
```

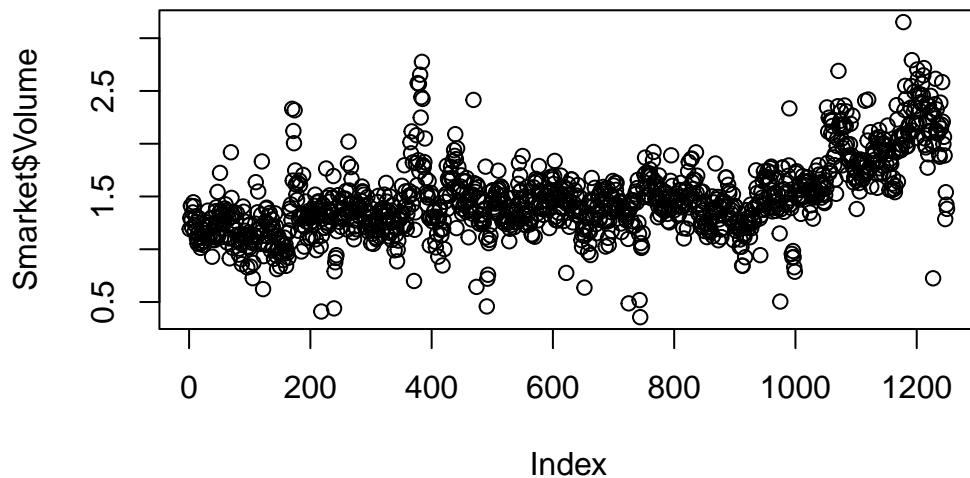
```
cor(Smarket[,-9])
```

	Year	Lag1	Lag2	Lag3	Lag4
Year	1.00000000	0.029699649	0.030596422	0.033194581	0.035688718
Lag1	0.02969965	1.00000000	-0.026294328	-0.010803402	-0.002985911
Lag2	0.03059642	-0.026294328	1.00000000	-0.025896670	-0.010853533
Lag3	0.03319458	-0.010803402	-0.025896670	1.00000000	-0.024051036
Lag4	0.03568872	-0.002985911	-0.010853533	-0.024051036	1.00000000
Lag5	0.02978799	-0.005674606	-0.003557949	-0.018808338	-0.027083641
Volume	0.53900647	0.040909908	-0.043383215	-0.041823686	-0.048414246
Today	0.03009523	-0.026155045	-0.010250033	-0.002447647	-0.006899527
	Lag5	Volume	Today		
Year	0.029787995	0.53900647	0.030095229		
Lag1	-0.005674606	0.04090991	-0.026155045		
Lag2	-0.003557949	-0.04338321	-0.010250033		
Lag3	-0.018808338	-0.04182369	-0.002447647		
Lag4	-0.027083641	-0.04841425	-0.006899527		
Lag5	1.000000000	-0.02200231	-0.034860083		
Volume	-0.022002315	1.00000000	0.014591823		
Today	-0.034860083	0.01459182	1.000000000		

從相關性可看出滯後變數與當日收益率之間的相關性接近零，似乎沒有相關性。

唯一顯著的相關性存在於 Year 年份和 Volume 成交量之間，相關係數 0.53900647。

```
# 繪製隨著時間過去 (x 軸往右)，成交量 (y 軸) 呈現逐漸上升的趨勢。
plot(Smarket$Volume)
```



圖表重點顯示出整體趨勢往上跑、趨勢大、變異高。

4.7.2 Logistic Regression

```
# 擬合一個邏輯迴歸模型，以使用 Lag1 到 Lag5 和 Volume 來預測 Direction。
# glm = generalized linear model
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial
)
# 看看模型訓練得好不好
summary(glm.fits)
```

Call:

```
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
  Volume, family = binomial, data = Smarket)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.126000	0.240736	-0.523	0.601
Lag1	-0.073074	0.050167	-1.457	0.145
Lag2	-0.042301	0.050086	-0.845	0.398
Lag3	0.011085	0.049939	0.222	0.824
Lag4	0.009359	0.049974	0.187	0.851
Lag5	0.010313	0.049511	0.208	0.835
Volume	0.135441	0.158360	0.855	0.392

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1731.2 on 1249 degrees of freedom
 Residual deviance: 1727.6 on 1243 degrees of freedom
 AIC: 1741.6

Number of Fisher Scoring iterations: 3

1. Estimate (估計值): 告訴你變數跟結果 (股市上漲) 的關係是正向還是負向。Pr(>|z|) (P-value · P 值): 希望這個數字小於 0.05 · 表示這個變數是真的有用。
 2. Std. Error (Standard Error · 標準誤差): 估計係數 (Estimate) 可能偏離真實值的平均誤差範圍 · 數字越小越精準也越好。
 3. z value (z 統計量): 這個係數離 0 有多遠 ? 0 代表這變數對結果完全沒有影響 · 數字越大代表真的有作用 · 通常要 > 1.96 。
- 這裡最小的 p 值與 Lag1 相關。這個預測變數的負係數 (-0.073074) 表明 · 如果市場昨天上漲 · 那麼今天上漲的可能性較小。然而 · p 值高達 0.15 · 仍然相對較大 · 因此沒有明確的證據表明 Lag1 與趨勢之間存在真正的關聯。

```
# 利用剛剛訓練好的模型 · 計算出每一天股市上漲的機率有多高。  

# 使用 glm.fits 模型來對整個資料集進行「預測」。  

# type = "response" · 不看 Log-Odds ( 對數勝算比 ) · 直接看機率 (Probability)。  

glm.probs <- predict(glm.fits, type = "response")  

# glm.probs 會儲存 1250 個數字 · 每個數字都在 0 到 1 之間 · 代表模型預測當天會 Up 上漲的機率。  

# 顯示 glm.probs 陣列中的前 10 個計算結果。  

glm.probs[1:10]
```

1	2	3	4	5	6	7	8
0.5070841	0.4814679	0.4811388	0.5152224	0.5107812	0.5069565	0.4926509	0.5092292
9	10						
0.5176135	0.4888378						

```
# 查看 R 語言如何將類別變數 (Direction)「轉譯」成數字，以供迴歸模型 (glm) 計算。
contrasts(Smarket$Direction)
```

```
Up
Down 0
Up 1

# 為了預測市場在特定日期是上漲還是下跌，必須將這些預測機率轉換為類別標籤，即上漲或下跌。
# 建立一個包含 1250 個 Down 元素的向量。
glm.pred <- rep("Down", 1250)
# 將所有預測市場上漲機率超過 0.5 的元素轉換為 Up。
glm.pred[glm.probs > .5] = "Up"
```

Smarket 第一版作弊的表現 (glm 無測試集)

```
# 使用 table() 函數產生混淆矩陣，以確定有多少觀測值被正確分類或錯誤分類。
# 混淆矩陣的對角線元素表示正確的預測，而非對角線元素表示錯誤的預測。
table(glm.pred, Smarket$Direction)
```

```
glm.pred Down Up
Down 145 141
Up 457 507
```

```
# 我們的模型正確預測了市場上漲 507 天，下跌 145 天，總共有  $507 + 145 = 652$  次正確預測。
# 使用 mean() 函數計算預測正確的天數佔總天數的比例。
mean(glm.pred == Smarket$Direction)
```

[1] 0.5216

正確預測市場走勢的機率為 52.16%。邏輯迴歸模型似乎比隨機猜測略好 (大於 50%)?

這個結果具有誤導性，因為我們使用同一組包含 1250 個觀測值的資料集對模型進行了訓練和測試。過於樂觀!! 低估測試誤差率!!

為了更好地評估邏輯迴歸模型在此情境下的準確性，我們可以使用部分資料擬合模型，然後檢驗它對剩餘資料的預測效果。

```
# 訓練集 (2001 年至 2004 年)
train <- (Smarket$Year < 2005)
# 測試集 (2005 年之後)
Smarket.2005 <- Smarket[!train, ]
# 看看測試集的維度
```

```

dim(Smarket.2005)

[1] 252   9

# 把測試集的方向額外存著
Direction.2005 <- Smarket$Direction[!train]
# 使用 subset 參數，僅使用日期在 2005 年之前的觀測值子集來擬合邏輯迴歸模型。
glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial, subset = train
)
# 儲存對測試集預測的機率！
glm.probs <- predict(glm.fits, Smarket.2005,
                      type = "response")
# 建立一個包含 252 個 Down 元素的向量。
glm.pred <- rep("Down", 252)
# 將所有測試集預測市場上漲機率超過 0.5 的元素轉換為 Up。
glm.pred[glm.probs > .5] <- "Up"

```

Smarket 第二版的表現 (glm 有測試集)

```

# 使用預測的跟真實的值，繪製混淆矩陣。
table(glm.pred, Direction.2005)

```

		Direction.2005
glm.pred	Down	Up
Down	77	97
Up	34	44

```

# 計算預測正確的天數佔總天數的比例
mean(glm.pred == Direction.2005)

```

```
[1] 0.4801587
```

```
# 測試錯誤率高達 52%，比隨機猜測還要糟糕！
```

或許移除那些對預測方向沒有幫助的變量，我們可以更有效的模型！

使用與反應變數無關的預測變數往往會導致測試錯誤率上升（因為這類預測變數會增加方差，而偏差不會相應減少），因此移除這類預測變數反而可能帶來改善。

```
# 只使用 Lag1 和 Lag2 重新擬合邏輯迴歸模型。
# 因為在原始邏輯迴歸模型中，這兩個變數具有相對較高的預測能力。
glm.fits <- glm(Direction ~ Lag1 + Lag2, data = Smarket,
                     family = binomial, subset = train)
# 儲存對測試集預測的機率！第三版！
glm.probs <- predict(glm.fits, Smarket.2005,
                      type = "response")
# 建立一個包含 252 個 Down 元素的向量。第三版！
glm.pred <- rep("Down", 252)
# 將所有測試集預測市場上漲機率超過 0.5 的元素轉換為 Up。第三版！
glm.pred[glm.probs > .5] <- "Up"
```

Smarket 第三版的表現 (只使用 Lag1 和 Lag2 的 GLM)

```
# 使用預測的跟真實的值，繪製混淆矩陣。第三版！
table(glm.pred, Direction.2005)
```

		Direction.2005	
		glm.pred	Down
Down	Down	35	35
	Up	76	106

```
# 計算預測正確的天數佔總天數的比例。第三版！
mean(glm.pred == Direction.2005)
```

[1] 0.5595238

```
# 預測市場上漲的表現好像不錯？
106 / (106 + 76)
```

[1] 0.5824176

混淆矩陣顯示，在邏輯迴歸預測市場上漲的日子裡，準確率達到了 58%。這提示了一種可能的交易策略：在模型預測市場上漲的日子買入，在預測市場下跌的日子避免交易。我們想要預測 Lag1 和 Lag2 分別等於 1.2 和 1.1 的那一天的方向，以及它們分別等於 1.5 和 -0.8 的那一天的方向。

```
# 我們想要預測 Lag1 和 Lag2 分別等於 1.2 和 1.1 的那一天的方向。
# 以及它們分別等於 1.5 和 -0.8 的那一天的方向。
predict(glm.fits,
        newdata =
          data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)),
```

```

    type = "response"
)

```

1	2
0.4791462	0.4960939

4.7.3 Linear Discriminant Analysis

對 Smarket 資料進行 LDA 分析。`lda()` 函數的語法與 `lm()` 和 `glm()` 函數完全相同，只是缺少 `family` 選項。

LDA 代表 Linear Discriminant Analysis (線性判別分析)，第二個分類模型。LDA 的目標是讓不同的類別分得越開越好。找到一個最佳的線性組合 (一條線或一個平面)，讓不同類別的資料點在投影到這條線上後，彼此間距離最遠，而每個類別內部的點又最緊密。

```

library(MASS)
# 訓練 LDA 模型
# 使用 Lag1 ( 昨天的回報率 ) 和 Lag2 ( 前天的回報率 ) 來預測 Direction ( 今天漲跌 )。
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = Smarket,
                  subset = train)
# 看看輸出結果
lda.fit

```

Call:

```
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
```

Prior probabilities of groups:

Down	Up
0.491984	0.508016

Group means:

	Lag1	Lag2
Down	0.04279022	0.03389409
Up	-0.03954635	-0.03132544

Coefficients of linear discriminants:

	LD1
Lag1	-0.6420190
Lag2	-0.5135293

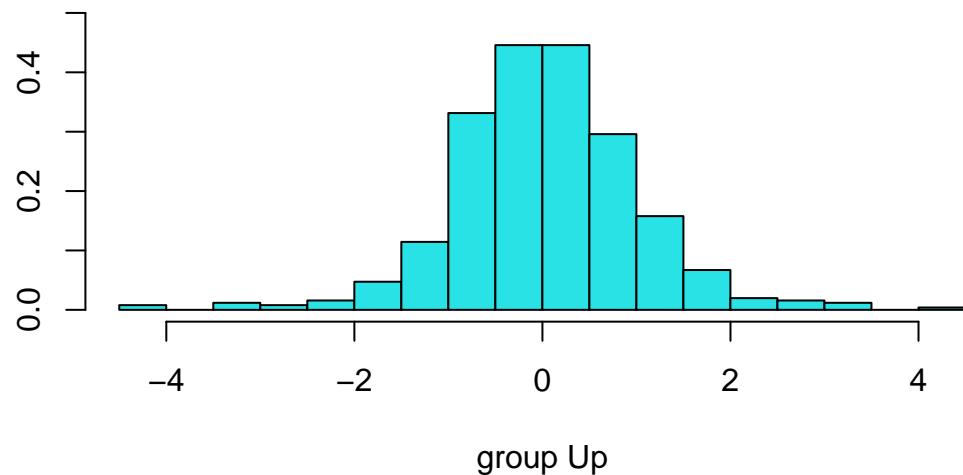
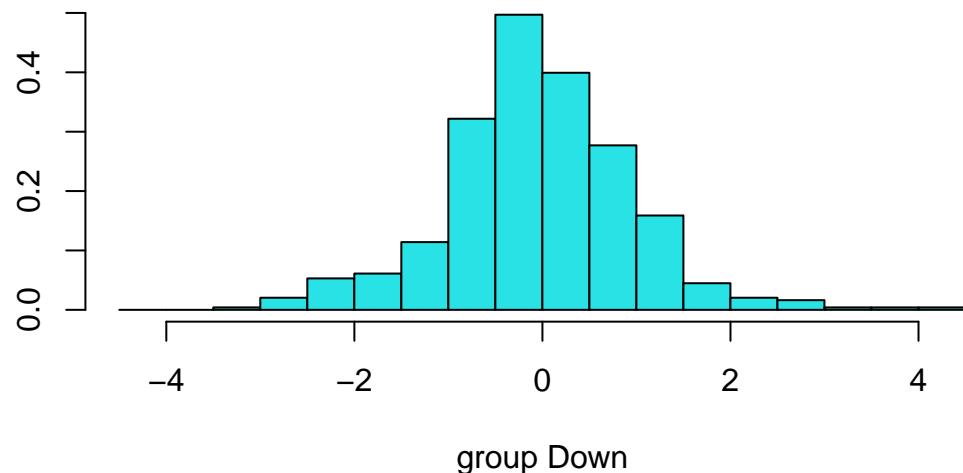
1. Prior probabilities of groups (群組先驗機率) 告訴你訓練集裡，Down 和 Up 各佔多少比例。

2. Coefficients of linear discriminants (線性判別係數) 是 LDA 找到的那條最佳分界線的係數。

3. 這條線是 $LD1 = -0.642 \times Lag1 - 0.514 \times Lag2$ ，看 $LD1$ 的值是大於還是小於 0，來決定是 Up 還是 Down。若該值較小，則 LDA 分類器預測市場下跌；較大則 LDA 分類器預測市場上漲。

```
# 畫個圖看看！
```

```
plot(lda.fit)
```



```
# X 軸是線性判別分數 (Linear Discriminant Score, 或稱 LD Score)。
# Y 軸是這些 LD 分數的分佈密度。
# 這張圖是在看「資料被投影到最優判別線後，分得開還是分不開？」
# 目標是讓這兩個直方圖的平均值（中心點）分得越開越好。
# 圖表表現不太好，不過還是看看預測表現如何？
lda.pred <- predict(lda.fit, Smarket[!train,])
```

Smarket 第四版的表現 (只使用 Lag1 和 Lag2 的 LDA)

```
# 繪製混淆矩陣
table(lda.pred$class, Smarket$Direction[!train])
```

	Down	Up
Down	35	35
Up	76	106

預測市場的表現如何？55.95%。0..0 好喔。

$(35+106)/252$

[1] 0.5595238

```
# 使用訓練好的 lda.fit 模型，對 2005 年的測試資料進行預測
lda.pred <- predict(lda.fit, Smarket.2005)
# 查看 lda.pred 這個預測結果物件裡面包含了哪些資訊
names(lda.pred)
```

[1] "class" "posterior" "x"

```
# `class` 包含 LDA 對市場趨勢的預測。
# `posterior` 是一個矩陣，其第 k 列包含對應觀測值屬於第 k 類的後驗機率。
# `x` 包含前面所描述的線性判別圖。
# 從預測結果中，取出模型判定的最終類別 (Class)。
lda.class <- lda.pred$class
# 製作混淆矩陣
table(lda.class, Direction.2005)
```

	Direction.2005	
	Down	Up
Down	35	35
Up	76	106

```
# 計算模型預測的整體準確率
mean(lda.class == Direction.2005)

[1] 0.5595238

# 計算模型預測 Down (下跌) 機率大於或等於 50% 的總天數。
sum(lda.pred$posterior[, 1] >= .5)

[1] 70

# 計算模型預測 Down (下跌) 機率小於 50% 的總天數 (即 Up 的天數)。
sum(lda.pred$posterior[, 1] < .5)

[1] 182

# 顯示前 20 天，模型預測 Down (下跌) 的機率是多少。
lda.pred$posterior[1:20, 1]

  999     1000     1001     1002     1003     1004     1005     1006
0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016 0.4872861
  1007     1008     1009     1010     1011     1012     1013     1014
0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761 0.4744593 0.4799583
  1015     1016     1017     1018
0.4935775 0.5030894 0.4978806 0.4886331

# 顯示前 20 天，模型最終判斷是 Down 還是 Up。
lda.class[1:20]

[1] Up   Down Up   Up   Up
[16] Up   Up   Down Up   Up
Levels: Down Up

# 計算模型預測 Down (下跌) 機率大於 90% 的天數，用來看模型對自己的預測有多「自信」。
sum(lda.pred$posterior[, 1] > .9)

[1] 0
```

2005 年沒有一天達到這個閾值！2005 年全年下降的最大後驗機率僅為 52.02%。

4.7.4 Quadratic Discriminant Analysis

QDA 代表 Quadratic Discriminant Analysis (二次判別分析)。

假設每個類別擁有獨立的變異結構，分界線是曲線或非線性 (Quadratic Boundary)。

QDA 更靈活，理論上可以達到更好的分離效果。

```
# 訓練 QDA 模型，用 Lag1 和 Lag2 預測 Direction，只使用訓練集。
qda.fit <- qda(Smarket$Direction ~ Lag1 + Lag2, data = Smarket,
                 subset = train)
# 輸出模型訓練結果
qda.fit
```

Call:

```
qda(Smarket$Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
```

Prior probabilities of groups:

	Down	Up
	0.491984	0.508016

Group means:

	Lag1	Lag2
Down	0.04279022	0.03389409
Up	-0.03954635	-0.03132544

```
# 使用訓練好的 qda.fit 模型，對 2005 年的測試集進行預測，取出最終的分類結果。
```

```
qda.class <- predict(qda.fit, Smarket.2005)$class
```

Smarket 第五版的表現 (只使用 Lag1 和 Lag2 的 QDA)

```
# 製作混淆矩陣，比較 QDA 的預測結果 和 2005 年實際的漲跌，計算預測的對錯次數。
table(qda.class, Direction.2005)
```

Direction.2005		
qda.class	Down	Up
Down	30	20
Up	81	121

```
# 計算模型在 2005 年測試集上的準確率
mean(qda.class == Direction.2005)
```

[1] 0.5992063

有趣的是 QDA 預測的準確率也接近 60%，對於眾所周知難以精確建模的股票市場數據而言，這樣的準確率相當令人印象深刻，顯示所假設的二次型是合理的。

4.7.5 Naive Bayes

Naive Bayes 是一種基於貝氏定理 (Bayes' Theorem) 的分類算法。核心思想是利用已知的特徵 (如 Lag1、Lag2) · 來計算某個類別 (如 Up 或 Down) 發生的條件機率 · 試圖計算給定 Lag1 和 Lag2 · 今天上漲的機率。

這算法被稱為「樸素」 · 是因為它做了一個非常大膽 (通常不成立) 的假設 · 假設所有的預測變數 (Lag1, Lag2...) 彼此之間是獨立的 (Independent) · 不受其他變數影響。

```
library(e1071)
# 訓練 Naive Bayes 模型 · 用 Lag1 和 Lag2 預測 Direction · 只使用訓練集。
nb.fit <- naiveBayes(Smarket$Direction ~ Lag1 + Lag2, data = Smarket,
                      subset = train)
# 輸出模型訓練結果
nb.fit
```

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

Y

Down	Up
0.491984	0.508016

Conditional probabilities:

Lag1

Y	[,1]	[,2]
Down	0.04279022	1.227446
Up	-0.03954635	1.231668

Lag2

Y	[,1]	[,2]
Down	0.03389409	1.239191
Up	-0.03132544	1.220765

從剛剛輸出結果包含每個類別中每個變數的估計平均值和標準差 · 我們能輕鬆驗證這一點!

計算訓練集中 · 在「下跌日」(Direction == "Down") 時 · Lag1 的實際平均值和標準差。

由以下證實了 Naive Bayes 假設這些數據是服從常態分佈 · 並用這些 mean, sd 來計算機率。

```
mean(Smarket$Lag1[train] [Smarket$Direction[train] == "Down"])

[1] 0.04279022

sd(Smarket$Lag1[train] [Smarket$Direction[train] == "Down"])

[1] 1.227446

# 使用訓練好的模型，對 2005 年測試集進行預測，輸出最終判定的類別 (Up 或 Down)。
nb.class <- predict(nb.fit, Smarket.2005)
```

Smarket 第六版的表現 (只使用 Lag1 和 Lag2 的 Naive Bayes)

```
# 製作混淆矩陣，比較 Naive Bayes 的預測結果 和 2005 年實際的漲跌，計算預測的對錯次數。
table(nb.class, Direction.2005)
```

		Direction.2005
nb.class	Down	Up
Down	28	20
Up	83	121

```
# 計算模型在 2005 年測試集上的準確率
mean(nb.class == Direction.2005)
```

[1] 0.5912698

```
# Naive Bayes 演算法在此資料集的表現非常出色，預測準確率超過 59%。
# 雖然略遜於二次判別分析 (QDA)，但遠勝於線性判別分析 (LDA)。
```

```
# 使用模型計算 2005 年每一天，市場是 Down 或 Up 的原始機率 (Raw Probability)。
nb.preds <- predict(nb.fit, Smarket.2005, type = "raw")
# 顯示前 5 筆資料預測出的 Down 和 Up 機率。
nb.preds[1:5, ]
```

	Down	Up
[1,]	0.4873164	0.5126836
[2,]	0.4762492	0.5237508
[3,]	0.4653377	0.5346623
[4,]	0.4748652	0.5251348
[5,]	0.4901890	0.5098110

4.7.6 K-Nearest Neighbors

執行 KNN 演算法。此函數需要四個輸入：

1. 包含與訓練資料相關的預測變數的矩陣，此標記為 train.X。
2. 包含與待預測資料相關的預測變數的矩陣，此標記為 test.X。
3. 包含訓練觀測值的類別標籤的向量，此標記為 train.Direction。
4. K 值，即分類器所使用的最近鄰數量。

使用 cbind() 函數（列綁定）將 Lag1 和 Lag2 變數綁定到兩個矩陣中，一個用於訓練集，另一個用於測試集。

```
library(class)
# 訓練集
train.X <- cbind(Smarket$Lag1, Smarket$Lag2)[train, ]
# 測試集
test.X <- cbind(Smarket$Lag1, Smarket$Lag2)[!train, ]
# 方向集
train.Direction <- Smarket$Direction[train]
# 設定固定種子
set.seed(1)
```

Smarket 第七版的表現 (只使用 Lag1 和 Lag2 的 KNN($k = 1$))

```
# KNN 訓練 ( $k = 1$ )
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
# 混淆矩陣，比較 KNN( $k = 1$ ) 的預測結果和 2005 年實際的漲跌，計算預測的對錯次數。
table(knn.pred, Direction.2005)
```

		Direction.2005	
		Down	Up
Down	43	58	
Up	68	83	

```
# 計算 KNN( $k = 1$ ) 模型在 2005 年測試集上的準確率
(83 + 43) / 252
```

[1] 0.5

Smarket 第八版的表現 (只使用 Lag1 和 Lag2 的 KNN($k = 3$))

```
# KNN 訓練 ( $k = 3$ )
knn.pred <- knn(train.X, test.X, train.Direction, k = 3)
# 混淆矩陣 · 比較 KNN( $k = 3$ ) 的預測結果 和 2005 年實際的漲跌 · 計算預測的對錯次數。
table(knn.pred, Direction.2005)
```

```
Direction.2005
knn.pred Down Up
Down    48 54
Up      63 87

# 計算 KNN( $k = 3$ ) 模型在 2005 年測試集上的準確率
mean(knn.pred == Direction.2005)
```

[1] 0.5357143

很明顯可知結果略有改善 · 但進一步增加 K 值並沒有帶來任何改進。對於此資料集 · QDA 似乎是我們目前考察過的方法中效果最好的。

KNN 在 Smarket 資料集的表現並不理想 · 但它通常也能取得令人印象深刻的結果。我們試試把 KNN 方法應用於 Caravan 資料集。

此資料集包含 85 個預測變量 · 用於衡量 5822 位個體的統計特徵。反應變數是購買情況 · 它表示特定個體是否購買了房車保險。在該資料集中 · 只有 6% 的人購買了房車保險。

```
# 維度: 看看這個資料集有幾列 ( Rows ) 和幾行 ( Columns )。
dim(Caravan)
```

[1] 5822 86

```
# 鎖定 Caravan 這資料集 · 之後能少打一點。
attach(Caravan)
# # 資料分佈概況: 針對這一欄 Purchase 提供統計摘要。
summary(Purchase)
```

```
No   Yes
5474  348
```

```
# 確認一下購買的比例是否 6% 左右
348 / 5822
```

[1] 0.05977327

由於 KNN 分類器透過識別與給定測試觀測值最接近的觀測值來預測其類別 · 因此變數的尺度至關重

要。解決這個問題的一個好方法是將資料標準化 scale，使所有標準化變數的平均值為零，標準差為一。

```
# 在標準化資料時，我們排除第 86 列，因為那是 qualitative Purchase variable。
```

```
standardized.X <- scale(Caravan[, -86])
```

```
# 原本的標準差差很多
```

```
var(Caravan[, 1])
```

```
[1] 165.0378
```

```
var(Caravan[, 2])
```

```
[1] 0.1647078
```

```
# 現在標準化後的 x 的每一列的標準差均為 1，平均值為 0。
```

```
var(standardized.X[, 1])
```

```
[1] 1
```

```
var(standardized.X[, 2])
```

```
[1] 1
```

```
# 分割測試集（包含前 1000 個觀測值）
```

```
test <- 1:1000
```

```
# 標準化的訓練集（包含剩餘的觀測值）
```

```
train.X <- standardized.X[-test, ]
```

```
# 標準化的測試集
```

```
test.X <- standardized.X[test, ]
```

```
# 把訓練集的 Y 弄出來
```

```
train.Y <- Purchase[-test]
```

```
# 把測試集的 Y 弄出來
```

```
test.Y <- Purchase[test]
```

```
# 設定固定種子
```

```
set.seed(1)
```

```
# KNN 訓練 (k = 1)
```

```
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
```

```
# 在 1000 個測試觀測值上，KNN 錯誤率略低於 12%。
```

```
mean(test.Y != knn.pred)
```

```
[1] 0.118
```

```
# 然而因為只有 6% 的客戶購買了保險，我們可以透過始終預測「否」來將錯誤率降低到 6%。
```

```
mean(test.Y != "No")
```

```
[1] 0.059
```

如果公司嘗試向隨機選擇的客戶銷售保險，那麼成功率只有 6%，考慮到相關成本，這可能太低了，因此公司更希望只向那些可能購買保險的客戶銷售。

所以整體錯誤率並非公司關注的重點。相反地公司關注的是被正確預測會購買保險的客戶比例。

Caravan 第一版的表現 (KNN(k = 1))

```
# 製作混淆矩陣，比較 KNN(k = 1) 的預測結果 和實際的 Y 值，計算預測的對錯次數。
```

```
table(knn.pred, test.Y)
```

	test.Y	
knn.pred	No	Yes
No	873	50
Yes	68	9

```
# 計算 KNN(k = 1) 模型的準確率
```

```
9 / (68 + 9)
```

```
[1] 0.1168831
```

結果表明對於被預測會購買保險的客戶，K=1 的 KNN 演算法比隨機猜測的效果要好得多。在 77 位這樣的客戶中，有 9 位 (11.7%) 實際購買了保險。這個比例是隨機猜測的兩倍。

Caravan 第二版的表現 (KNN(k = 3))

```
# 試試看 KNN 訓練 (k = 3)
```

```
knn.pred <- knn(train.X, test.X, train.Y, k = 3)
```

```
# 混淆矩陣，比較 KNN(k = 3) 的預測結果 和實際的 Y 值，計算預測的對錯次數。
```

```
table(knn.pred, test.Y)
```

	test.Y	
knn.pred	No	Yes
No	920	54
Yes	21	5

```
### Caravan 第二版的表現 (KNN(k = 3)) =====
```

```
# 計算 KNN(k = 3) 的成功率 (預測會保險且真的有保險的)
```

```
5 / 26
```

```
[1] 0.1923077
```

當 K=3 時，成功率提升至 19%

Caravan 第三版的表現 (KNN(k = 5))

```
# 試試看 KNN 訓練 (k = 5)
knn.pred <- knn(train.X, test.X, train.Y, k = 5)
# 混淆矩陣，比較 KNN(k = 5) 的預測結果 和實際的 Y 值，計算預測的對錯次數。
table(knn.pred, test.Y)
```

	test.Y	
knn.pred	No	Yes
No	930	55
Yes	11	4

```
# 計算 KNN(k = 5) 的成功率 (預測會保險且真的有保險的)
4 / 15
```

[1] 0.2666667

當 K=3 時，成功率提升至 19%；當 K=5 時，成功率則高達 26.7%。這比隨機猜測的成功率高出四倍以上。看來 KNN 演算法確實能在一個複雜的資料集中找到一些真正的規律！

雖然使用 K=5 的 KNN 演算法預測的成本效益高，但預測結果只有 15 位客戶會購買保險，實際上保險公司可能希望投入資源說服更多潛在客戶購買保險。

作為對比，我們也可以將邏輯迴歸模型 (logistic regression model) 擬合到資料中。

Caravan 第四版的表現 (GLM(glm.probs > .5))

```
# Purchase ~ . 代表資料集裡除了因變數 (Purchase) 以外，所有其他的變數。
# 在 R 語言的索引中，負號代表「排除 (Exclude)」。
# subset = -test 代表請排除掉 test 向量裡包含的所有行編號。
glm.fits <- glm(Purchase ~ ., data = Caravan,
                  family = binomial, subset = -test)
# 使用訓練好的模型，對測試集進行預測，輸出最終判定的類別。
glm.probs <- predict(glm.fits, Caravan[test, ],
                      type = "response")
# 1000 個先預設是"No"(不保險的比較多)
glm.pred <- rep("No", 1000)
# 使用 0.5 作為分類器的預測機率閾值，超過的標記改成"Yes"
glm.pred[glm.probs > .5] <- "Yes"
```

```
# 製作混淆矩陣，比較 GLM 的預測結果 和實際的 Y 值。
```

```
table(glm.pred, test.Y)
```

```
test.Y
glm.pred No Yes
No 934 59
Yes 7 0
```

```
# 只有 7 個測試樣本被預測為會購買保險。更糟的是我們對這 7 個樣本的預測都是錯的！
```

Caravan 第五版的表現 (GLM(glm.probs > .25))

```
# 重來一次，1000 個先預設是"No"(不保險的比較多)
```

```
glm.pred <- rep("No", 1000)
```

```
# 改成使用 0.25 作為分類器的預測機率閾值，超過的標記改成"Yes"
```

```
glm.pred[glm.probs > .25] <- "Yes"
```

```
### Caravan 第五版的表現 (GLM(glm.probs > .25) )=====
```

```
# 製作混淆矩陣，比較 GLM 的預測結果 和實際的 Y 值。
```

```
table(glm.pred, test.Y)
```

```
test.Y
glm.pred No Yes
No 919 48
Yes 22 11
```

```
# 計算 GLM(glm.probs > .25) 的成功率（預測會保險且真的有保險的）
```

```
11 / (22 + 11)
```

```
[1] 0.3333333
```

改為在預測購買機率超過 0.25 時就進行預測，結果好得多!! 預測 33 人會購買保險，並且其中約 33% 的人的預測是正確的!! 這比隨機猜測好五倍以上！

4.7.7 Poisson Regression

試試看對 Bikeshare 資料集擬合了一個 Poisson Regression model，該資料集衡量了華盛頓特區每小時的自行車租賃數量 (騎乘者數量)。

```
# 鎖定 Bikeshare 這資料集，之後能少打一點。
```

```
attach(Bikeshare)
```

```
# 維度：看看這個資料集有幾列 ( Rows ) 和幾行 ( Columns )。
dim(Bikeshare)

[1] 8645 15

# 看看有那些變數
names(Bikeshare)

[1] "season"      "mnth"        "day"         "hr"          "holiday"
[6] "weekday"     "workingday"   "weathersit"  "temp"        "atemp"
[11] "hum"         "windspeed"    "casual"      "registered"  "bikers"

# 做個 least squares linear regression model 試試
mod.lm <- lm(
  bikers ~ mnth + hr + workingday + temp + weathersit,
  data = Bikeshare
)
# 看看模型訓練概況，提供統計摘要。
summary(mod.lm)
```

Call:

```
lm(formula = bikers ~ mnth + hr + workingday + temp + weathersit,
  data = Bikeshare)
```

Residuals:

Min	1Q	Median	3Q	Max
-299.00	-45.70	-6.23	41.08	425.29

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-68.632	5.307	-12.932	< 2e-16 ***
mnthFeb	6.845	4.287	1.597	0.110398
mnthMarch	16.551	4.301	3.848	0.000120 ***
mnthApril	41.425	4.972	8.331	< 2e-16 ***
mnthMay	72.557	5.641	12.862	< 2e-16 ***
mnthJune	67.819	6.544	10.364	< 2e-16 ***
mnthJuly	45.324	7.081	6.401	1.63e-10 ***
mnthAug	53.243	6.640	8.019	1.21e-15 ***
mnthSept	66.678	5.925	11.254	< 2e-16 ***
mnthOct	75.834	4.950	15.319	< 2e-16 ***

mnthNov	60.310	4.610	13.083	< 2e-16	***
mnthDec	46.458	4.271	10.878	< 2e-16	***
hr1	-14.579	5.699	-2.558	0.010536	*
hr2	-21.579	5.733	-3.764	0.000168	***
hr3	-31.141	5.778	-5.389	7.26e-08	***
hr4	-36.908	5.802	-6.361	2.11e-10	***
hr5	-24.135	5.737	-4.207	2.61e-05	***
hr6	20.600	5.704	3.612	0.000306	***
hr7	120.093	5.693	21.095	< 2e-16	***
hr8	223.662	5.690	39.310	< 2e-16	***
hr9	120.582	5.693	21.182	< 2e-16	***
hr10	83.801	5.705	14.689	< 2e-16	***
hr11	105.423	5.722	18.424	< 2e-16	***
hr12	137.284	5.740	23.916	< 2e-16	***
hr13	136.036	5.760	23.617	< 2e-16	***
hr14	126.636	5.776	21.923	< 2e-16	***
hr15	132.087	5.780	22.852	< 2e-16	***
hr16	178.521	5.772	30.927	< 2e-16	***
hr17	296.267	5.749	51.537	< 2e-16	***
hr18	269.441	5.736	46.976	< 2e-16	***
hr19	186.256	5.714	32.596	< 2e-16	***
hr20	125.549	5.704	22.012	< 2e-16	***
hr21	87.554	5.693	15.378	< 2e-16	***
hr22	59.123	5.689	10.392	< 2e-16	***
hr23	26.838	5.688	4.719	2.41e-06	***
workingday	1.270	1.784	0.711	0.476810	
temp	157.209	10.261	15.321	< 2e-16	***
weathersitcloudy/misty	-12.890	1.964	-6.562	5.60e-11	***
weathersitlight rain/snow	-66.494	2.965	-22.425	< 2e-16	***
weathersitheavy rain/snow	-109.745	76.667	-1.431	0.152341	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 76.5 on 8605 degrees of freedom

Multiple R-squared: 0.6745, Adjusted R-squared: 0.6731

F-statistic: 457.3 on 39 and 8605 DF, p-value: < 2.2e-16

hr(0) 和 mnth(Jan) 的第一級被視為基準值，因此未提供它們的係數估計值，它們的係數估計值為零，所有其他級別均相對於這些基準值進行測量。

2月的係數為 6.845，表示在其他變數不變的情況下，2月的乘客數量平均比1月多約7人。3月的乘

客數量則比 1 月多約 16.5 人。

```
# 查看 R 語言如何將類別變數 ( hr & mnth )「轉譯」成數字，以供後續模型計算。
contrasts(Bikeshare$hr) = contr.sum(24)
contrasts(Bikeshare$mnth) = contr.sum(12)
# 訓練一個 linear model 試試
mod.lm2 <- lm(
  bikers ~ mnth + hr + workingday + temp + weathersit,
  data = Bikeshare
)
# 看看模型訓練的狀況
summary(mod.lm2)
```

Call:

```
lm(formula = bikers ~ mnth + hr + workingday + temp + weathersit,
  data = Bikeshare)
```

Residuals:

Min	1Q	Median	3Q	Max
-299.00	-45.70	-6.23	41.08	425.29

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	73.5974	5.1322	14.340	< 2e-16 ***
mnth1	-46.0871	4.0855	-11.281	< 2e-16 ***
mnth2	-39.2419	3.5391	-11.088	< 2e-16 ***
mnth3	-29.5357	3.1552	-9.361	< 2e-16 ***
mnth4	-4.6622	2.7406	-1.701	0.08895 .
mnth5	26.4700	2.8508	9.285	< 2e-16 ***
mnth6	21.7317	3.4651	6.272	3.75e-10 ***
mnth7	-0.7626	3.9084	-0.195	0.84530
mnth8	7.1560	3.5347	2.024	0.04295 *
mnth9	20.5912	3.0456	6.761	1.46e-11 ***
mnth10	29.7472	2.6995	11.019	< 2e-16 ***
mnth11	14.2229	2.8604	4.972	6.74e-07 ***
hr1	-96.1420	3.9554	-24.307	< 2e-16 ***
hr2	-110.7213	3.9662	-27.916	< 2e-16 ***
hr3	-117.7212	4.0165	-29.310	< 2e-16 ***
hr4	-127.2828	4.0808	-31.191	< 2e-16 ***

```

hr5           -133.0495   4.1168 -32.319 < 2e-16 ***
hr6           -120.2775   4.0370 -29.794 < 2e-16 ***
hr7           -75.5424   3.9916 -18.925 < 2e-16 ***
hr8           23.9511    3.9686   6.035 1.65e-09 ***
hr9           127.5199   3.9500  32.284 < 2e-16 ***
hr10          24.4399    3.9360   6.209 5.57e-10 ***
hr11          -12.3407   3.9361  -3.135  0.00172 **
hr12          9.2814     3.9447   2.353  0.01865 *
hr13          41.1417    3.9571  10.397 < 2e-16 ***
hr14          39.8939    3.9750  10.036 < 2e-16 ***
hr15          30.4940    3.9910   7.641 2.39e-14 ***
hr16          35.9445    3.9949   8.998 < 2e-16 ***
hr17          82.3786    3.9883  20.655 < 2e-16 ***
hr18          200.1249   3.9638  50.488 < 2e-16 ***
hr19          173.2989   3.9561  43.806 < 2e-16 ***
hr20          90.1138    3.9400  22.872 < 2e-16 ***
hr21          29.4071    3.9362  7.471 8.74e-14 ***
hr22          -8.5883    3.9332  -2.184  0.02902 *
hr23          -37.0194   3.9344  -9.409 < 2e-16 ***
workingday     1.2696    1.7845   0.711  0.47681
temp          157.2094   10.2612  15.321 < 2e-16 ***
weathersitcloudy/misty -12.8903   1.9643  -6.562 5.60e-11 ***
weathersitlight rain/snow -66.4944   2.9652 -22.425 < 2e-16 ***
weathersitheavy rain/snow -109.7446  76.6674  -1.431  0.15234
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

Residual standard error: 76.5 on 8605 degrees of freedom
Multiple R-squared:  0.6745,    Adjusted R-squared:  0.6731
F-statistic: 457.3 on 39 and 8605 DF,  p-value: < 2.2e-16

```

在 mod.lm2 中，除了 hr 和 mnth 的最後一個等級外，其他所有等級都會報告係數估計值。重要的是，在 mod.lm2 中，mnth 最後一個等級的係數估計值不為零：它等於所有其他等級係數估計值總和的負值。

其中 1 月份的係數為 -46.087，表示在其他變數不變的情況下，1 月份的乘客數量通常比年平均值少 46 人。

```

# 從這能看出編碼方式的選擇其實並不重要，兩者預測的差距非常非常小。
sum((predict(mod.lm) - predict(mod.lm2))^2)

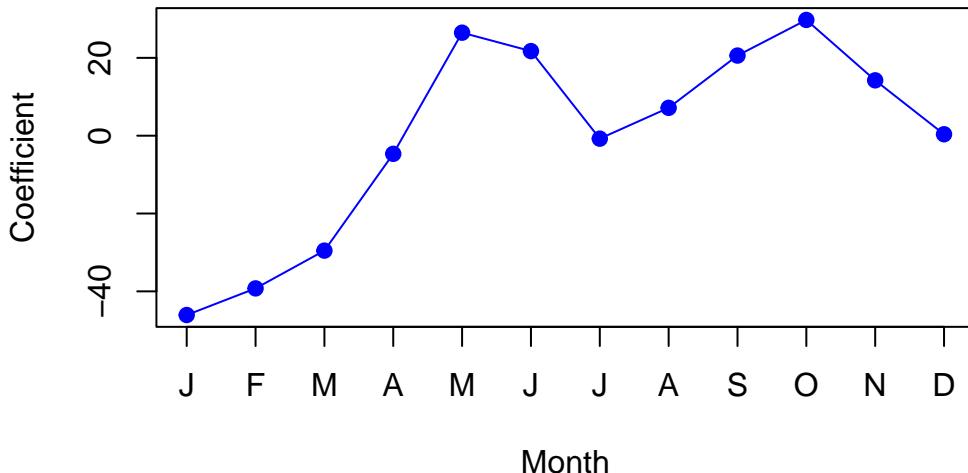
```

```
[1] 1.586608e-18
```

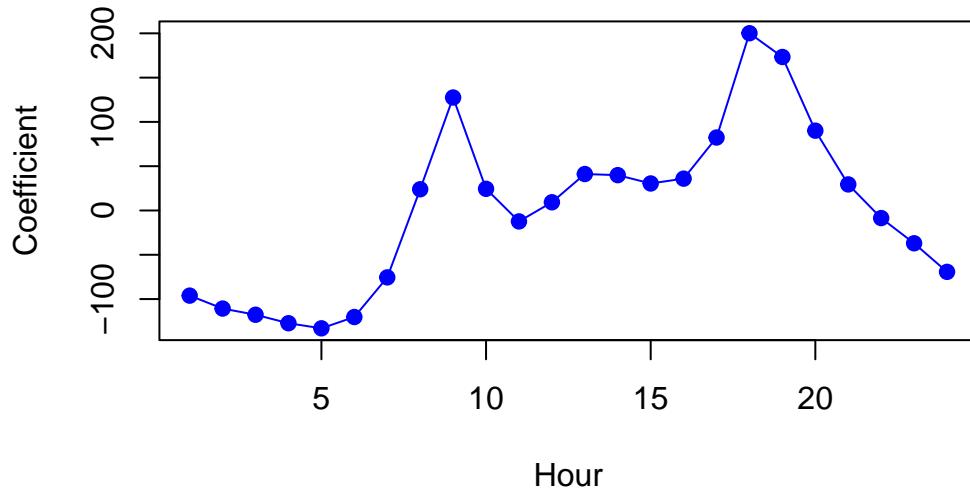
```
# 從此處可看出平方差之和為零
all.equal(predict(mod.lm), predict(mod.lm2))
```

[1] TRUE

```
# 提取 mod.lm2 中 12 個月份的影響係數 (1 月至 11 月的係數可獲得)。
# 第 12 個月 (Dec) 的係數是前面 11 個月係數總和的負值。
coef.months <- c(coef(mod.lm2)[2:12],
                    -sum(coef(mod.lm2)[2:12]))
# 畫出 12 個月份的係數值。Y 軸表示該月相比於年平均的差異。
plot(coef.months, xlab = "Month", ylab = "Coefficient",
      xaxt = "n", col = "blue", pch = 19, type = "o")
# 標記月份在 x 軸上
axis(side = 1, at = 1:12, labels = c("J", "F", "M", "A",
                                         "M", "J", "J", "A", "S", "O", "N", "D"))
```



```
# 計算小時係數。第 24 個小時的係數一樣也是前面 11 個月係數總和的負值。
coef.hours <- c(coef(mod.lm2)[13:35],
                  -sum(coef(mod.lm2)[13:35]))
# 畫出 24 個小時的係數值。Y 軸表示該小時相比於日平均的差異。
plot(coef.hours, xlab = "Hour", ylab = "Coefficient",
      col = "blue", pch = 19, type = "o")
```



```
# 訓練一個 Poisson Regression model，預測自行車租賃數量 (bikers)。
# 計數資料 (人數) 通常服從 Poisson 分佈，所以這個模型理論上很合適。
mod.pois <- glm(
  bikers ~ mnth + hr + workingday + temp + weathersit,
  data = Bikeshare, family = poisson
)
# 輸出 Poisson 模型的詳細結果
summary(mod.pois)
```

Call:

```
glm(formula = bikers ~ mnth + hr + workingday + temp + weathersit,
     family = poisson, data = Bikeshare)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	4.118245	0.006021	683.964	< 2e-16 ***
mnth1	-0.670170	0.005907	-113.445	< 2e-16 ***
mnth2	-0.444124	0.004860	-91.379	< 2e-16 ***
mnth3	-0.293733	0.004144	-70.886	< 2e-16 ***
mnth4	0.021523	0.003125	6.888	5.66e-12 ***
mnth5	0.240471	0.002916	82.462	< 2e-16 ***
mnth6	0.223235	0.003554	62.818	< 2e-16 ***
mnth7	0.103617	0.004125	25.121	< 2e-16 ***

mnth8	0.151171	0.003662	41.281	< 2e-16	***
mnth9	0.233493	0.003102	75.281	< 2e-16	***
mnth10	0.267573	0.002785	96.091	< 2e-16	***
mnth11	0.150264	0.003180	47.248	< 2e-16	***
hr1	-0.754386	0.007879	-95.744	< 2e-16	***
hr2	-1.225979	0.009953	-123.173	< 2e-16	***
hr3	-1.563147	0.011869	-131.702	< 2e-16	***
hr4	-2.198304	0.016424	-133.846	< 2e-16	***
hr5	-2.830484	0.022538	-125.586	< 2e-16	***
hr6	-1.814657	0.013464	-134.775	< 2e-16	***
hr7	-0.429888	0.006896	-62.341	< 2e-16	***
hr8	0.575181	0.004406	130.544	< 2e-16	***
hr9	1.076927	0.003563	302.220	< 2e-16	***
hr10	0.581769	0.004286	135.727	< 2e-16	***
hr11	0.336852	0.004720	71.372	< 2e-16	***
hr12	0.494121	0.004392	112.494	< 2e-16	***
hr13	0.679642	0.004069	167.040	< 2e-16	***
hr14	0.673565	0.004089	164.722	< 2e-16	***
hr15	0.624910	0.004178	149.570	< 2e-16	***
hr16	0.653763	0.004132	158.205	< 2e-16	***
hr17	0.874301	0.003784	231.040	< 2e-16	***
hr18	1.294635	0.003254	397.848	< 2e-16	***
hr19	1.212281	0.003321	365.084	< 2e-16	***
hr20	0.914022	0.003700	247.065	< 2e-16	***
hr21	0.616201	0.004191	147.045	< 2e-16	***
hr22	0.364181	0.004659	78.173	< 2e-16	***
hr23	0.117493	0.005225	22.488	< 2e-16	***
workingday	0.014665	0.001955	7.502	6.27e-14	***
temp	0.785292	0.011475	68.434	< 2e-16	***
weathersitcloudy/misty	-0.075231	0.002179	-34.528	< 2e-16	***
weathersitlight rain/snow	-0.575800	0.004058	-141.905	< 2e-16	***
weathersitheavy rain/snow	-0.926287	0.166782	-5.554	2.79e-08	***

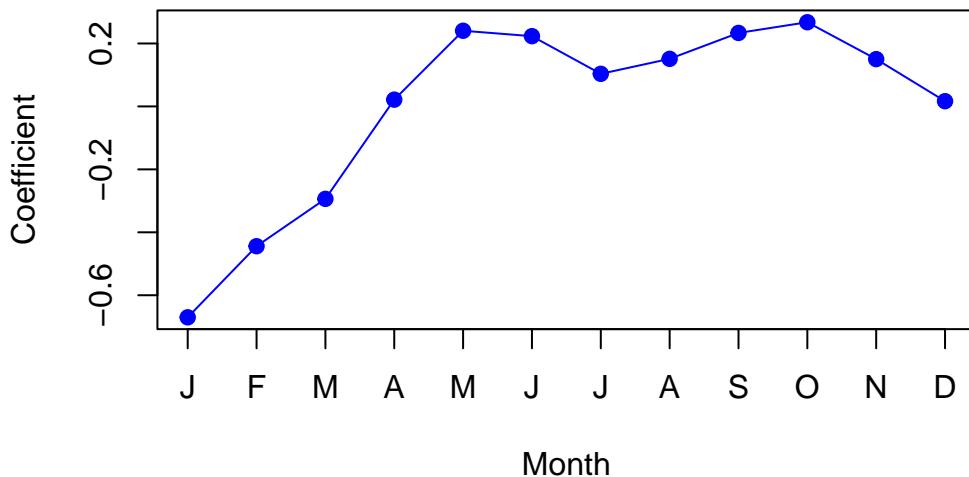
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

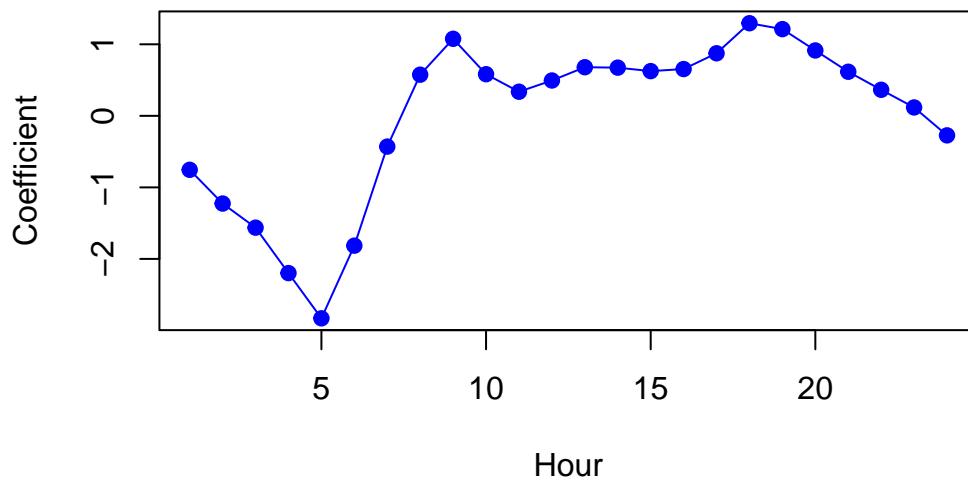
Null deviance: 1052921 on 8644 degrees of freedom
 Residual deviance: 228041 on 8605 degrees of freedom
 AIC: 281159

Number of Fisher Scoring iterations: 5

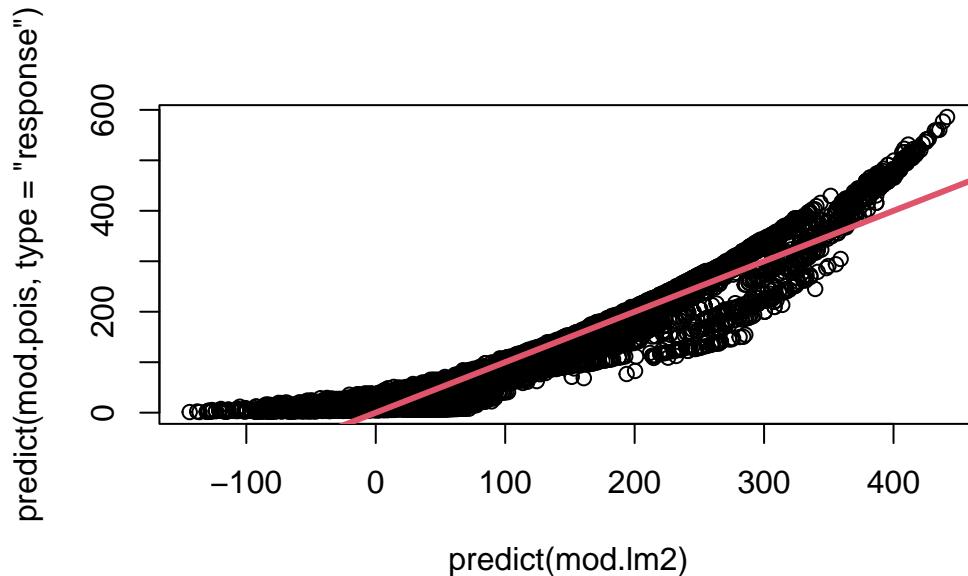
```
# 提取 mod.pois 的月份係數。
coef.mnth <- c(coef(mod.pois)[2:12],
                 -sum(coef(mod.pois)[2:12]))
# 畫出 Poisson 模型的月份係數
plot(coef.mnth, xlab = "Month", ylab = "Coefficient",
      xaxt = "n", col = "blue", pch = 19, type = "o")
# 標記 X 軸為月份
axis(side = 1, at = 1:12, labels = c("J", "F", "M", "A", "M", "J",
                                         "J", "A", "S", "O", "N", "D"))
```



```
# 提取 mod.pois 的小時係數
coef.hours <- c(coef(mod.pois)[13:35],
                  -sum(coef(mod.pois)[13:35]))
# 畫出 Poisson 模型的小時係數
plot(coef.hours, xlab = "Hour", ylab = "Coefficient",
      col = "blue", pch = 19, type = "o")
```



```
# 比較預測值：線性模型 (mod.lm2) 的預測值與 Poisson 模型 (mod.pois) 的預測值。
# 使用參數 type = "response" 指定 R 輸出 exp( ^0 + ^1X1+...+^pXp)
plot(predict(mod.lm2), predict(mod.pois, type = "response"))
# 畫一條斜率為 1 的對角參考線。
# 如果所有點都落在這條線上，代表兩個模型的預測結果完全相同。
abline(0, 1, col = 2, lwd = 3)
```



Poisson Regression model 的預測結果與線性模型的預測結果相關。然而圖表的形狀不是直線，而是

一條明顯向上彎曲的曲線，揭示了兩個模型的核心差異。

1. 線性模型失敗了，在前面出現了負值 (-100)，預測了負的自行車數量。
2. Poisson 模型預測的極端值更高 (曲線彎曲)，在 X 軸超過 200 之後，黑點明顯地衝到紅線的上方，而且差距越來越大。因此對於極低或極高的客流量，Poisson Regression model 的預測結果往往大於線性模型的預測結果。