

Лабораторна 3: логістична регресія, лінійний та квадратичний дискримінантний аналіз та метод К-найближчих сусідів.

Опис даних

Використаємо набір даних Smarket. Цей набір даних складається з дохідності вираженої у відсотках для фондового індексу S&P 500 протягом 1 250 днів від початку 2001 року і до кінця 2005 року. Для кожної дати наявна дохідність для попередніх 5 днів, lag1, lag2,...,lag5. Змінна Volume містить дані про обсяг торгів попереднього дня у млн., Today – сьогоднішня дохідність, та змінна Direction, яка вказує чи зріс ринок чи впав.

```
> library(ISLR)
> names(Smarket)
[1] "Year"          "Lag1"          "Lag2"          "Lag3"          "Lag4"
[6] "Lag5"          "Volume"        "Today"         "Direction"
> dim(Smarket)
[1] 1250   9
> summary(Smarket)
      Year           Lag1           Lag2
Min. :2001   Min. :-4.92200   Min. :-4.92200
1st Qu.:2002  1st Qu.:-0.63950  1st Qu.:-0.63950
Median :2003   Median : 0.03900   Median : 0.03900
Mean   :2003   Mean   : 0.00383   Mean   : 0.00392
3rd Qu.:2004  3rd Qu.: 0.59675   3rd Qu.: 0.59675
Max.   :2005   Max.   : 5.73300   Max.   : 5.73300
      Lag3           Lag4           Lag5
Min. :-4.92200   Min. :-4.92200   Min. :-4.92200
1st Qu.:-0.64000  1st Qu.:-0.64000  1st Qu.:-0.64000
Median : 0.03850   Median : 0.03850   Median : 0.03850
Mean   : 0.00172   Mean   : 0.00164   Mean   : 0.00561
3rd Qu.: 0.59675  3rd Qu.: 0.59675  3rd Qu.: 0.59700
Max.   : 5.73300   Max.   : 5.73300   Max.   : 5.73300
      Volume          Today          Direction
Min.   :0.356   Min.   :-4.92200   Down:602
1st Qu.:1.257   1st Qu.:-0.63950   Up  :648
Median :1.423   Median : 0.03850
Mean   :1.478   Mean   : 0.00314
3rd Qu.:1.642   3rd Qu.: 0.59675
Max.   :3.152   Max.   : 5.73300
> pairs(Smarket)
```

Обчислимо кореляції між змінними

```

> cor(Smarket)
Error in cor(Smarket) : 'x' must be numeric
> cor(Smarket[,-9])
      Year    Lag1    Lag2    Lag3    Lag4    Lag5
Year  1.0000  0.02970  0.03060  0.03319  0.03569  0.02979
Lag1  0.0297  1.00000 -0.02629 -0.01080 -0.00299 -0.00567
Lag2  0.0306 -0.02629  1.00000 -0.02590 -0.01085 -0.00356
Lag3  0.0332 -0.01080 -0.02590  1.00000 -0.02405 -0.01881
Lag4  0.0357 -0.00299 -0.01085 -0.02405  1.00000 -0.02708
Lag5  0.0298 -0.00567 -0.00356 -0.01881 -0.02708  1.00000
Volume 0.5390  0.04091 -0.04338 -0.04182 -0.04841 -0.02200
Today  0.0301 -0.02616 -0.01025 -0.00245 -0.00690 -0.03486
      Volume   Today
Year   0.5390  0.03010
Lag1   0.0409 -0.02616
Lag2   -0.0434 -0.01025
Lag3   -0.0418 -0.00245
Lag4   -0.0484 -0.00690
Lag5   -0.0220 -0.03486
Volume 1.0000  0.01459
Today  0.0146  1.00000

```

Як і слід було очікувати, співвідношення між зміщеними значеннями дохідності та теперішнім значенням близькі до нуля. Іншими словами, немає значної кореляції між дохідностями а попередні дні і сьогоднішньою дохідністю. Єдиний висновок, що можна зараз зробити, що обсяги торгів зростають з часом.

```
> attach(Smarket)
```

```
> plot(Volume)
```

Логістична регресія

Побудуємо логістичну регресію для передбачення Direction на основі предикторів lag1, lag2,...,lag5 і Volume. Використаємо функція glm(), що призначена для оцінки узагальнених лінійних моделей, клас яких включає також логістичну регресію. Синтаксис функції glm() подібний до синтаксису функції lm(), за винятком того, що ми потрібно вказати, яка модель використовується, в нашому випадку, family = binomial.

```

> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
  data=Smarket,family=binomial)
> summary(glm.fit)

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

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
 -1.45   -1.20    1.07    1.15    1.33 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.12600   0.24074  -0.52    0.60    
Lag1        -0.07307   0.05017  -1.46    0.15    
Lag2        -0.04230   0.05009  -0.84    0.40    
Lag3         0.01109   0.04994   0.22    0.82    
Lag4         0.00936   0.04997   0.19    0.85    
Lag5         0.01031   0.04951   0.21    0.83    
Volume       0.13544   0.15836   0.86    0.39    
                                          
(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: 1742

```

Number of Fisher Scoring iterations: 3

Найменше p -значення відповідає Lag1. Негативний коефіцієнт біля цієї змінної вказує на те, що якщо вчора ринок мав позитивну дохідність, то сьогодні менше шансів зрости. Однак, оскільки p -значення дорівнює 0,15, то немає чітких доказів справжнього зв'язку між Lag1 та Direction. Ми використаємо функцію coef(), щоб отримати доступ лише до коефіцієнтів моделі. Ми також можемо використати функцію summary() для отримання інформації щодо побудованої моделі, наприклад, p -значення коефіцієнтів.

```

> coef(glm.fit)
(Intercept)          Lag1          Lag2          Lag3          Lag4
-0.12600      -0.07307     -0.04230      0.01109      0.00936
Lag5           Volume
 0.01031      0.13544

> summary(glm.fit)$coef
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.12600    0.2407   -0.523   0.601
Lag1         -0.07307   0.0502   -1.457   0.145
Lag2         -0.04230   0.0501   -0.845   0.398
Lag3          0.01109   0.0499   0.222   0.824
Lag4          0.00936   0.0500   0.187   0.851
Lag5          0.01031   0.0495   0.208   0.835
Volume        0.13544   0.1584   0.855   0.392

> summary(glm.fit)$coef[,4]
(Intercept)          Lag1          Lag2          Lag3          Lag4
 0.601         0.145         0.398         0.824         0.851
Lag5           Volume
 0.835         0.392

```

Функцію predict () можна використати для прогнозування ймовірності того, що ринок піде вгору, враховуючи значення предикторів. type="responce" вказує, що потрібно виводити ймовірності вигляду $P(Y = 1 | X)$. Якщо для функція predict () не задано набору даних, то обчислюються ймовірності для навчальних даних, які використовувались для оцінки моделі логістичної регресії. Ми вивели лише перші десять ймовірностей. Ми знаємо, що ці значення відповідають ймовірності зростання ринку, а не зниження, оскільки функція contrasts() вказує, що створено фіктивну змінну зі значенням 1 для Up.

```

> glm.probs=predict(glm.fit,type="response")
> glm.probs[1:10]
  1    2    3    4    5    6    7    8    9    10
0.507 0.481 0.481 0.515 0.511 0.507 0.493 0.509 0.518 0.489

> contrasts(Direction)
  Up
Down 0
Up   1

```

Для того, щоб зробити прогноз щодо того, чи буде ринок підніматися у певний день чи ні, ми повинні перетворити ці прогнозовані ймовірності на мітки класів, Up або Down. Наступні дві команди створюють вектор прогнозів на основі того, чи передбачена ймовірність росту ринку більша або менша 0,5.

```

> glm.pred=rep("Down",1250)
> glm.pred[glm.probs > .5] = "Up"

```

Перша команда створює вектор з 1250 елементів Down. Другий рядок перетворює всі елементи на Up, для яких передбачена ймовірність зростання ринку перевищує 0,5. Враховуючи ці прогнози, функція table() може бути використана для створення матриці помилок, щоб визначити, скільки спостережень були класифіковані правильно або неправильно.

```

> table(glm.pred, Direction)
    Direction
glm.pred Down Up
  Down    145 141
  Up      457 507
> (507+145) / 1250
[1] 0.5216
> mean(glm.pred==Direction)
[1] 0.5216

```

Діагональні елементи матриці помилок вказують на правильні прогнози, тоді як позадіагоналі – неправильні. Отже, наша модель правильно передбачила, що 507 днів, що ринок зросте і 145 – знизиться, загалом $507 + 145 = 652$ правильні прогнози. Функцію `mean()` можна використати для обчислення частки днів, коли прогноз був правильним. У цьому випадку логістична регресія правильно спрогнозувала рух ринку у 52,2% випадків. На перший погляд здається, що модель логістичної регресії працює трохи краще, ніж випадкове вгадування. Однак цей результат вводить в оману оскільки ми тренували і тестували модель на тому ж наборі з 1250 спостережень. Іншими словами, $100 - 52,2 = 47,8\%$ – це коефіцієнт помилок у навчанні. Ми бачили раніше, що частота помилок у навчанні часто надмірно оптимістична – це має тенденцію занижувати рівень помилок тесту. З метою кращої оцінки точності моделі логістичної регресії, ми можемо оцінити модель використовуючи частину даних, а потім перевірити, наскільки добре вона передбачає на решті даних. Це дасть більш реалістичний рівень помилок у тому сенсі, що на практиці нас буде цікавити ефективність нашої моделі не на даних на яких ми оцінювали нашу модель, а на майбутніх даних, які на момент побудови моделі є невідомими. Для реалізації цієї стратегії ми спочатку створимо відповідний вектор зі спостережень з 2001 по 2004 рр. Потім ми використаємо цей вектор, щоб створити набір даних за 2005 рік.

```

> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252   9
> Direction.2005=Direction[!train]

```

Об'єкт `train` – це вектор із 1 250 елементів, що відповідає спостереженням у нашему наборі даних. Елементи вектора, які відповідають спостереженням, що мали місце до 2005 р., встановлено як `TRUE`, тоді як для тих, що відповідають спостереженням у 2005 р. встановлено як `FALSE`. Об'єкт `train` по суті є логічний вектор, оскільки його елементи мають значення `TRUE` і `FALSE`. Булеві вектори можна використовуватися для отримання підмножини рядків або стовпців матриці. Наприклад, команда `Smarket[train,]` вибере підматрицю набору даних фондового ринку, що відповідають лише датам до 2005 р., оскільки це ті, для яких елементи `train` є `TRUE`. Символ `!` може використовуватися для заперечення всіх елементів булевого вектора. Тому `Smarket[!train,]` це підматриця даних фондового ринку, що містить лише спостереження з датами 2005 рік. Тепер ми оцінимо модель логістичної регресії, використовуючи лише підмножину спостережень, які відповідають датам до 2005 року, використовуючи аргумент

subset. Потім ми дослідимо прогнозованій ймовірності зростання фондового ринку для кожного дня в нашому тестовому наборі, тобто для днів 2005 року.

```
> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,  
    data=Smarket ,family=binomial ,subset=train)  
> glm.probs=predict(glm.fit ,Smarket .2005 ,type="response ")
```

Ми навчили та протестували нашу модель на двох абсолютно різних наборах даних: навчання проводилося з використанням лише дат до 2005 року, і тестування проводилось з використанням лише дат 2005 року. Ми обчислили прогнози на 2005 рік та порівняли їх із реальними рухами ринку протягом цього періоду часу.

```
> glm.pred=rep ("Down " ,252)  
> glm.pred[glm.probs > .5] ="Up "  
> table(glm.pred ,Direction .2005)  
    Direction .2005  
glm.pred Down Up  
    Down    77 97  
    Up      34 44  
> mean(glm.pred==Direction .2005)  
[1] 0.48  
> mean(glm.pred!=Direction .2005)  
[1] 0.52
```

Позначення != означає, «не дорівнює», і тому в останній команді обчислюється коефіцієнт помилок тестового набору. Результати досить невтішні: помилка тесту становить 52%, що гірше, ніж випадкове вгадування! Звичайно, цей результат є передбачуваним оскільки, як правило, не можна точно передбачити майбутнє зростання/спадання ринку на основі даних за попередні дні.

Нагадаємо, що наша модель логістичної регресії мала надзвичайно великі розначення пов'язані з усіма предикторами. Оскільки, використання предикторів, які не мають впливають на залежну змінну, як правило, викликає погрішення тестової помилки, то можливо, видаливши деякі змінні ми отримаємо більш ефективну модель. Ми побудуємо логістичну регресію з використанням предикторів Lag1 і Lag2.

```
> glm.fit=glm(Direction~Lag1+Lag2 ,data=Smarket ,family=binomial ,  
    subset=train)  
> glm.probs=predict(glm.fit ,Smarket .2005 ,type="response ")  
> glm.pred=rep ("Down " ,252)  
> glm.pred[glm.probs > .5] ="Up "  
> table(glm.pred ,Direction .2005)  
    Direction .2005  
glm.pred Down Up  
    Down    35 35  
    Up      76 106  
> mean(glm.pred==Direction .2005)  
[1] 0.56  
> 106/(106+76)  
[1] 0.582
```

Отримані результати виглядають краще: 56% щоденних рухів були правильно передбачені. Матриця помилок вказує, що у дні коли логістична регресія

передбачає падіння ринку, точність становить 50%. Однак у дні, коли прогнозується зростання ринку, точність становить 58%. Припустимо, що ми хочемо передбачити дохідність, пов'язану з конкретним значенням Lag1 та Lag2. Зокрема, ми хочемо передбачити напрямок руху ринку на день, коли Lag1 і Lag2 дорівнюють 1,2 і 1,1, відповідно, і в день, коли вони дорівнюють 1,5 та -0,8. Ми робимо це за допомогою функції predict () .

```
> predict(glm.fit,newdata=data.frame(Lag1=c(1.2,1.5),
  Lag2=c(1.1,-0.8)),type="response")
   1         2
0.4791    0.4961
```

Лінійний дискрімінантний аналіз

Використаємо лінійний дискрімінантний аналіз для даних Smarket. Ми зробимо за допомогою функції lda(), яка є частиною бібліотеки MASS. Синтаксис функції lda() ідентичний синтаксису lm() та синтаксису glm() за винятком відсутності опції family. Ми оцінюємо модель на основі лише спостереження до 2005 року.

```
> library(MASS)
> lda.fit=lda(Direction~Lag1+Lag2,data=Smarket,subset=train)
> lda.fit
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
```

Prior probabilities of groups:

| | |
|-------|-------|
| Down | Up |
| 0.492 | 0.508 |

Group means:

| | Lag1 | Lag2 |
|------|---------|---------|
| Down | 0.0428 | 0.0339 |
| Up | -0.0395 | -0.0313 |

Coefficients of linear discriminants:

| | LD1 |
|-----------------|--------|
| Lag1 | -0.642 |
| Lag2 | -0.514 |
| > plot(lda.fit) | |

Отримали, що $\hat{\pi}_1 = 0.492$, а $\hat{\pi}_1 = 0.508$, тобто 49,2% навчальних спостережень відповідають дням, протягом яких ринок спадав. Ми також отримали групові середні; це середні показники кожного предиктора в кожному класі і вони використовуються в цьому методі як оцінки μ_k . Можемо зробити висновок, що існує тенденція, що протягом попередніх 2 днів дохідності є негативним у дні, коли ринок зростає, і тенденція для попередніх 2 днів дохідності є позитивними, коли ринок спадає. На основі оцінених коефіцієнти можемо побудувати правило для прийняття рішень на основі лінійного дискрімінантного аналізу. Іншими словами, якщо $-0,642 \times \text{Lag1} - 0,514 \times \text{Lag2}$ велике, тоді класифікатор буде прогнозувати зростання ринку, і якщо невеликий, то – падіння ринку. Функція plot () будує графіки лінійних дискримінантів, отриманих обчисленням $-0,642 \times \text{Lag1} - 0,514 \times \text{Lag2}$ для кожного з навчальних спостережень.

Функція predict () повертає список із трьома елементами. Перший елемент, class, містить прогнози щодо руху ринку. Другий елемент, posterior – це матриця, k-й стовпець якої містить апостеріорну ймовірність того, що відповідне спостереження належить k-му класу. Нарешті, x містить лінійні дискримінанти.

```
> lda.pred=predict(lda.fit, Smarket.2005)
> names(lda.pred)
[1] "class"      "posterior"   "x"
```

Результати тестування подібні до результатів отриманих для логістичної регресії

```
> lda.class=lda.pred$class
> table(lda.class,Direction.2005)
    Direction.2005
lda.pred Down Up
    Down     35 35
    Up       76 106
> mean(lda.class==Direction.2005)
[1] 0.56
```

Застосування порогу 50% до апостеріорних ймовірностей дозволяє нам відтворити передбачення, що містяться в lda.pred\$class.

```
> sum(lda.pred$posterior[,1]>=.5)
[1] 70
> sum(lda.pred$posterior[,1]<.5)
[1] 182
```

Зверніть увагу на те, що апостеріорна ймовірність, що виводиться моделлю відповідає ймовірність спадання ринку:

```
> lda.pred$posterior[1:20,1]
> lda.class[1:20]
```

Якщо ми б хотіли використати поріг для апостеріорної ймовірності, відмінний від 50%, наприклад, припустимо що ми хочемо передбачити зменшення ринку лише в тому випадку, якщо ми сильно впевнені, що ринок справді зменшиться в той день – тобто, якщо апостеріорна ймовірність становить не менше 90%.

```
> sum(lda.pred$posterior[,1]>.9)
[1] 0
```

Для жодного дня 2005 року не досягнуто цього порогу! Насправді найбільша апостеріорна ймовірність спаду за весь 2005 рік становила 52,02%.

Квадратичний дискримінантний аналіз

Побудуємо модель квадратичного дискримінантного аналізу до даних Smarket. Робимо це за допомогою функції qda (), яка також є частиною бібліотеки MASS. Синтаксис цієї функції ідентичний синтаксису lda().

```

> qda.fit=qda(Direction~Lag1+Lag2 ,data=Smarket ,subset=train)
> qda.fit
Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
  Down      Up 
0.492 0.508 

Group means:
          Lag1    Lag2
Down  0.0428  0.0339
Up   -0.0395 -0.0313

```

Вихідні результати містять групові середні. Але вони не містить коефіцієнтів лінійних дискримінантів, оскільки даний класифікатор використовує квадратичну, а не лінійна функція предикторів. Функція predict() працює точно так само, як і для лінійного аналізу.

```

> qda.class=predict(qda.fit,Smarket.2005)$class
> table(qda.class,Direction.2005)
           Direction.2005
qda.class Down  Up
      Down 30  20
      Up   81 121
> mean(qda.class==Direction.2005)
[1] 0.599

```

Цікаво, що отримані прогнози є точними майже в 60% випадків. Цей рівень точності є досить вражаючим для даних фондового ринку. Це говорить про те, що прийнята квадратна форма краще описує справжнє співвідношення ніж лінійні форми, розглянуті вище.

Метод K-найближчих сусідів

Використаємо метод K-найближчих сусідів з допомогою функції knn(), яка є частиною бібліотеки class. Ця функція працює інакше, ніж інші функції для оцінки моделей, які ми використовували раніше. Замість двоступеневого підходу, при якому ми спочатку оцінюємо модель, а потім використовуємо її для побудови передбачення, knn() формує передбачення за допомогою однієї команди. Функція вимагає чотирьох входів.

1. Матриця, що містить предиктори, пов'язані з навчальними даними, позначені train.X.
2. Матриця, що містить предиктори, пов'язані з даними, для яких ми хочемо робити прогнози, позначені test.X.
3. Вектор, що містить мітки класів для навчальних спостережень, Direction.
4. Значення для K, кількість найближчих сусідів, які слід використовувати при класифікації.

Ми використаємо функцію cbind(), для з'єднання Lag1 та Lag2 у дві матриці, одну для навчального набору, а іншу для тестового набору даних.

```

> library(class)
> train.X=cbind(Lag1,Lag2)[train,]
> test.X=cbind(Lag1,Lag2)[!train,]
> train.Direction=Direction[train]

```

Тепер функцію `knn()` можна використовувати для прогнозування руху ринку в 2005 р. `set.seed()` встановлюється для випадку, якщо кілька спостережень розглядається як найближчі сусіди, то R буде випадковим чином вибирати потрібну кількість.

```

> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Direction,k=1)
> table(knn.pred,Direction.2005)
    Direction.2005
knn.pred Down Up
  Down     43 58
  Up      68 83
> (83+43)/252
[1] 0.5

```

Результати з використанням $K=1$ не дуже хороши, оскільки лише 50% значень правильно передбачені. Повторимо аналіз, використовуючи $K = 3$.

```

> knn.pred=knn(train.X,test.X,train.Direction,k=3)
> table(knn.pred,Direction.2005)
    Direction.2005
knn.pred Down Up
  Down     48 54
  Up      63 87
> mean(knn.pred==Direction.2005)
[1] 0.536

```

Результати дещо покращилися. Але подальше збільшення K не призводить до покращення результатів. Отже, для цих даних квадратичний дискримінантний аналіз забезпечує найкращі результати серед методів, які ми розглядали до цього часу.

Метод K-найближчих сусідів для даних Caravan

Цей набір даних включає 85 предикторів, які вимірюють демографічні характеристики для 5822 особи. Залежна змінна: `Purchase`, яка вказує, чи купила дана особа страховий поліс чи ні. У цьому наборі даних придбали страховий поліс лише 6% людей.

```

> dim(Caravan)
[1] 5822   86
> attach(Caravan)
> summary(Purchase)
  No  Yes 
5474 348 
> 348/5822
[1] 0.0598

```

Оскільки даний класифікатор передбачає клас даного тестового спостереження за визначенням найближчих до нього спостережень, масштабування змінних має значення. Будь-які змінні великого масштабу матимуть набагато більший вплив

на відстань між спостереженнями, а отже, і на сам класифікатор, ніж змінні, які є в малому масштабі. Наприклад, уявіть набір даних, що містить дві змінні, зарплату та вік (вимірюється в доларах і роках відповідно). Що стосується даного методу, то різниця в зарплаті у 1000 доларів величезна порівняно з різницею у віці 50 років. Отже, заробітна плата буде визначати результати класифікації методом К-найближчих сусідів, а вік ефекту майже не матиме. Це суперечить нашій інтуїції, оскільки насправді різниця в зарплаті \$ 1 000 є досить малою порівняно з різницею у віці 50 років. Крім того, важливість масштабу для нашого класифікатора призводить до ще одного питання: якщо ми вимірювали зарплату в японських ієнах, або якщо вимірювали вік у хвилинах, тоді ми отримали б зовсім інші результати класифікації ніж ті, які ми отримаємо, якщо ці дві змінні вимірюються в доларах та роках. Хороший спосіб вирішити цю проблему - стандартизувати дані так, щоб усім змінним відповідало середнє значення нуль та стандартне відхилення одиниця. Тоді всі змінні будуть у порівнянному масштабі. Можемо зробити це за допомогою функції `Scale()`. При стандартизації даних ми виключаємо стовпець 86, оскільки це є якісна змінна `Purchase`.

```
> standardized.X=scale(Caravan [,-86])
> var(Caravan [,1])
[1] 165
> var(Caravan [,2])
[1] 0.165
> var(standardized.X[,1])
[1] 1
> var(standardized.X[,2])
[1] 1
```

Тепер кожен стовпець `standardized.X` має стандартне відхилення один та середнє значення нуль. Тепер розділимо спостереження на тестовий набір, що містить перші 1000 спостереження та навчальний набір, що містить решту спостережень. Ми підбираємо модель на навчальних даних, використовуючи $K = 1$, і оцінюємо її точність на тестових даних.

```
> test=1:1000
> train.X=standardized.X[-test,]
> test.X=standardized.X[test,]
> train.Y=Purchase [-test]
> test.Y=Purchase [test]
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Y,k=1)
> mean(test.Y!=knn.pred)
[1] 0.118
> mean(test.Y!="No")
[1] 0.059
```

Вектор `test` є числовим зі значеннями від 1 до 1 000. Введення `standardized.X[test,]` дає підматрицю даних, що містить спостереження індекси яких знаходяться в межах від 1 до 1 000, тоді як `standardized.X [-test,]` - дані з індексами які не попадають в межі від 1 до 1 000. Частота помилок побудованої моделі на 1000 тестових спостережень трохи менше 12%. На перший погляд це може здатися досить хорошим результатом. Однак оскільки купували страхівку лише 6%

клієнтів, ми могли б знизити рівень помилок до 6%, завжди передбачаючи "Ні" незалежно від значень предикторів!

Припустимо, що спроби продати страховку мають певні нетривіальні витрати. Наприклад, можливо, продавець повинен відвідати кожного потенційного клієнта. Якщо компанія намагається продати страховку випадковому клієнту, тоді рівень успіху складе лише 6%, що може бути занадто низьким, враховуючи витрати. Натомість компанія хотіла б намагатися продати страховку лише тим клієнтам, які, ймовірно, її придобають. Отже, загальний коефіцієнт помилок не представляє інтересу. Натомість представляє інтерес частка осіб, для яких правильно передбачено, що вони придобають страховку. Виявляється, метод K-найближчих сусідів з $K = 1$ працює набагато краще, ніж випадкове вгадування серед клієнтів, для яких передбачається придбання страховки. Серед 77 таких клієнти, 9 або 11,7%, насправді купляють страховку. Це вдвічі більше ніж результат, який можна було б отримати при випадковому вгадуванні.

```
> table(knn.pred, test.Y)
      test.Y
knn.pred  No Yes
  No    873   50
  Yes    68    9
> 9/(68+9)
[1] 0.117
```

При використанні $K = 3$, рівень успіху зростає до 19%, а при $K = 5$ - 26,7%. Це в чотири рази перевищує показник, який є результатом випадкового вгадування. Отже, даний підхід знаходить деякі реальні закономірності у складному наборі даних!

```
> knn.pred=knn(train.X,test.X,train.Y,k=3)
> table(knn.pred,test.Y)
      test.Y
knn.pred  No Yes
  No    920   54
  Yes    21    5
> 5/26
[1] 0.192
> knn.pred=knn(train.X,test.X,train.Y,k=5)
> table(knn.pred,test.Y)
      test.Y
knn.pred  No Yes
  No    930   55
  Yes    11    4
> 4/15
[1] 0.267
```

Як порівняння, ми також можемо пристосувати модель логістичної регресії до даних. Якщо ми використовуємо 0,5 як прогнозовану межу ймовірності для класифікатора, тоді ми маємо проблему: передбачається лише сім придбань страховки для тестових даних. Причому, модель помилляється із ними усіма! Однак ми не зобов'язані використовувати відсікання 0,5. Якщо ми замість цього передбачимо покупку коли прогнозована ймовірність покупки перевищує 0,25, ми отримуємо набагато кращі результати: ми прогнозуємо, що 33 особи

придбають страховку, і ми правильно прогнозуємо для 33% цих людей. Це в п'ять разів краще ніж випадкове вгадування!

```
> glm.fit=glm(Purchase~,data=Caravan,family=binomial,
   subset==test)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> glm.probs=predict(glm.fit,Caravan[test,],type="response")
> glm.pred=rep("No",1000)
> glm.pred[glm.probs>.5]="Yes"
> table(glm.pred,test.Y)
    test.Y
glm.pred  No Yes
      No  934  59
      Yes   7   0
> glm.pred=rep("No",1000)
> glm.pred[glm.probs>.25]="Yes"
> table(glm.pred,test.Y)
    test.Y
glm.pred  No Yes
      No  919  48
      Yes  22  11
> 11/(22+11)
[1] 0.333
```