

5. 분류

데이터 과학자들은 자동으로 어떤 결정을 해야 하는 종류의 문제들을 자주 접한다. 이메일이 피싱 메일은 아닐까? 고객이 이탈할 가능성은 없을까? 웹 사용자가 광고를 클릭하려고 할까? 이러한 문제들을 **분류(classification)** 문제라고 한다.

분류는 가장 중요한 예측 유형이다. 데이터가 0인지 1인지(피싱인지/피싱이 아닌지, 클릭할지/클릭하지 않을지, 변경할지/변경하지 않을지)를 분류하거나, 여러 카테고리 중 어디에 속할지 예측하는 것을 목표로 한다.(예를 들면 지메일이 받은 이메일을 '소셜', '홍보', '프로모션' 등으로 필터링하는 것).

종종 단순한 (예/아니오 같은) 이진 분류가 아닌, 범주의 개수가 두 가지 이상인 분류를 해야 할 필요도 있다. 각 클래스에 속할 예측 확률이 알고 싶은 것이다.

모델에 단순히 이진 분류 결과를 할당하는 대신, 대부분의 알고리즘은 관심 클래스에 속할 확률 점수(경향 propensity)를 반환한다.

사실 R에서 로지스틱 회귀분석에 대한 기본 출력은 로그 오즈 척도를 따르며, 이것은 어떤 경향 점수로 변형된다. 그런 다음 이동 컷오프(절사)를 통해 경향 점수로부터 결정을 내릴 수 있다. 일반적인 접근 방식은 다음과 같다.

1. 어떤 레코드가 속할 거라고 생각되는 관심 클래스에 대한 컷오프 확률을 정한다.
2. 레코드가 관심 클래스에 속할 확률을 (모든 모델과 함께) 추정한다.
3. 그 확률이 컷오프 확률 이상이면 관심 클래스에 이 레코드를 할당한다.

컷오프가 높을수록 1로 예측되는 (관심 클래스에 속할) 레코드가 적어질 것이다. 컷오프가 낮을수록 더 많은 레코드가 1로 예측된다.

이장에서는 분류와 확률 추정을 위한 몇 가지 핵심기술을 다룬다. 분류와 수치 예측 모두에 사용할 수 있는 방법은 다음 장에서 추가적으로 설명한다.

범주 항목이 두 가지 이상이란?

대다수의 문제는 응답이 이진 형태이다. 그러나 일부 분류문제에서는 세 가지 이상의 결과가 있을 수 있다. 예를 들면, 가입한 지 1년 되는 고객들을 다음 세 가지로 분류할 수 있다. 계약을 해지하거나($Y = 2$), 월별 계약으로 변형하거나($Y = 1$), 새로운 장기 계약($Y = 0$)에 서명하거나, 목표는 $j = 0, 1, 2$ 에 대해 $Y = j$ 를 예측하는 것이다. 이 장에서 다루는 대부분의 분류 방법들을 세 가지 이상의 결과가 있는 문제에도 바로 사용하거나 살짝 변경해서 적용할 수 있다. 결과가 세 개 이상인 경우도, 조건부확률을 사용해서 여러 개의 이진 문제로 돌려서 생각해볼 수 있다.

예를 들면 앞의 재계약의 문제 경우에도, 두 가지 이진 예제 문제로 볼 수 있다.

- $Y = 0$ 인지 아니면 $Y > 0$ 인지 예측한다.
- $Y > 0$ 이라면 $Y = 1$ 인지 $Y = 2$ 인지를 예측한다.

이 경우 문제가 두 가지인 경우, 고객이 계약을 해지하는 경우와 재계약에 동의하는 경우(동의할 경우, 다시 어떤 유형의 재계약을 원하는지 고르면 된다)로 나누는 것이 좋다. 모델 피팅 관점에서도, 멀티 클래스 문제를 일련의 이진 문제로 변환하는 것이 종종 유리하다. 하나의 카테고리가 다른 카테고리 보다 훨씬 더 일반적인 경우 특히 그렇다.

5.1 나이브 베이즈

나이브 베이즈 알고리즘은 주어진 결과에 대해 예측변수 값을 관찰할 확률을 사용하여, 예측변수가 주어졌을 때, 결과 $Y = i$ 를 관찰할 확률을 추정한다.

용어정리

- 조건부확률 : 어떤 사건 ($Y = i$) 이 주어졌을 때, 해당 사건 ($X = i$)을 관찰할 확률 $P(X_i|Y_i)$
- 사후확률 : 예측 정보를 통합한 후, 결과의 확률(이와 달리, 사전확률에서는 예측변수에 대한 정보를 고려하지 않는다.)

베이지언 분류를 이해하기 위해, '나이브하지 않은' 베이지언 분류를 먼저 상상해보자. 각 레코드는 다음과 같이 분류할 수 있다.

1. 예측변수 프로파일이 동일한(즉, 예측변수의 값이 동일한) 모든 레코드들을 찾는다.
2. 해당 레코드들이 가장 많이 속한(즉, 가능성이 가장 많은) 클래스를 정한다.
3. 새 레코드에 해당 클래스를 지정한다.

이 방법은 모든 예측변수들이 동일하다면 같은 클래스에 할당될 가능성이 높기에, 표본에서 새로 들어온 레코드와 정확히 일치하는 데이터를 찾는 것에 무게를 두는 방식이다.

|

NOTE_ 표준 나이브 베이즈 알고리즘에서 예측변수는 범주형 변수(요인변수) 여야 한다. 연속형 변수를 다루는 두 가지 방법에 대해서는 5.1.3절에서 설명한다.

5.1.1 나이브하지 않은 베이지언 분류는 왜 현실성이 없을까?

예측변수의 개수가 일정 정도 커지게 되면, 분류해야 하는 데이터들은 대부분은 서로 완전 일치하는 경우가 거의 없다. 인구통계 변수를 기반으로 투표 결과를 예측하는 모델을 만든다고 하자. 꽤 크기가 큰 표본이라 하더라도, 지난 선거에서 투표한 미국 중서부 출신의 히스패닉계 남성이면서, 고소득자이고, 더 이전 선거에서는 투표 경험이 없고 등등에 대해 정확하게 일치하는 데이터는 없을 수 있다. 이게 단지 8가지 변수만을 고려했을 뿐이다. 대부분의 분류 문제에서 8개는 결코 많은 편이 아니다. 변수 하나만 추가한다고 해도, 동일한 5가지 변수 범주 중 하나에 속할 때, 레코드가 정확히 일치할 확률은 5배 감소한다.

CAUTION 그 이름과는 달리 나이브 베이즈는 베이지언 통계의 방법으로 간주되지 않는다. 나이브 베이즈는 상대적으로 통계 지식이 거의 필요 없는 데이터 중심의 경험적 방법이다. 베이즈 규칙과 비슷한 예측 계산이 들어가다 보니 이름을 그렇게 붙였을 뿐이다. 구체적으로는 결과가 주어졌을 때, 초반에 예측 변수의 확률을 계산하는 부분과 결과 확률을 최종적으로 계산하는 부분만 비슷하다.

5.1.2 나이브한 해법

앞서 알아본 나이브하지 않은 베이지언 방법과 달리, 나이브 베이즈 방법에서는 확률을 계산하기 위해 정확히 일치하는 레코드로만 제한할 필요가 없다. 대신 전체 데이터를 활용한다. 나이브 베이즈 방법에서 바뀌는 부분은 다음과 같다.

1. 이진 응답 $Y = i$ ($i = 0$ 또는 1)에 대해, 각 예측변수에 대한 조건부확률 $P(X_j|Y = i)$ 를 구한다.
이것은 $Y = i$ 가 주어질 때, 예측변수의 값이 나올 확률이다. 이 확률은 훈련 데이터에서 $Y = i$ 인 레코드들 중 X_j 값의 비율로 구할 수 있다.
2. 각 확률값을 곱한 다음, $Y = i$ 에 속한 레코드들의 비율을 곱한다.
3. 모든 클래스에 대해 1~2단계를 반복한다.
4. 2단계에서 모든 클래스에 대해 구한 확률 값을 모두 더한 값으로 클래스 i 의 확률을 나누면 결과 i 의 확률을 구할 수 있다.
5. 이 예측변수에 대해 가장 높은 확률을 갖는 클래스를 해당 레코드에 할당한다.


이 나이브 베이즈 알고리즘은 또한 예측변수 X_1, \dots, X_p 가 주어졌을 때의 출력 $Y = i$ 의 확률에 대한 방정식으로 표현될 수 있다.

$$P(X_1, X_2, \dots, X_p)$$

사실 $P(X_1, X_2, \dots, X_p)$ 값은 확률이 Y 에 독립이면서 0과 1 사이에 오도록 하기 위한 스케일링 계수이다.

$$P(X_1, X_2, \dots, X_p) = P(Y = 0)(P(X_1|Y = 0) P(X_2|Y = 0) \dots P(X_p|Y = 0)) + P(Y = 1)(P(X_1|Y = 1) P(X_2|Y = 1) \dots P(X_p|Y = 1))$$

왜 이 공식을 '나이브'라고 부르는 걸까? 결과가 주어졌을 때, 예측변수 벡터의 정확한 **조건부 확률**은 각 조건부확률 $P(X_j|Y = i)$ 의 곱으로 충분히 잘 할 수 있다는 단순한 가정을 기초로 하기 때문이다.

 image-20200417094620404

즉 $P(X_1, X_2, \dots, X_p|Y = i)$ 대신 $P(X_j|Y = i)$ 를 추정하면서, 우리는 X_j 가 $k \neq j$ 인 모든 X_k 가 서로 독립이라고 가정한 것이다.

나이브 베이즈 모델을 구하기 위해 사용할 만한, 몇 가지 R패키지들이 있다. 다음은 klaR 패키지를 사용하는 예제를 보여준다.

```
library(klaR)
naive_model <- NaiveBayes(outcome ~ purpose_ + home_ + emp_len_,
                           data = na.omit(loan_data))

> naive_model$table

$purpose_
      var
grouping  credit_card debt_consolidation home_improvement major_purchase
paid off  0.18759649          0.55215915          0.07150104          0.05359270
default   0.15151515          0.57571347          0.05981209          0.03727229
      var
grouping      medical      other small_business
paid off  0.01424728  0.09990737          0.02099599
default   0.01433549  0.11561025          0.04574126

$home_
      var
grouping  MORTGAGE      OWN      RENT
paid off  0.4894800  0.0808963  0.4296237
```

```

default  0.4313440 0.0832782 0.4853778

$emp_len_
      var
grouping  < 1 Year  > 1 Year
paid off 0.03105289 0.96894711
default  0.04728508 0.95271492

```

모델로부터 나온 결과는 조건부확률 $P(X_j|Y=i)$ 이다. 모델을 통해 다음과 같이 새로운 대출에 대한 결과를 예측할 수 있다.

```

> new_loan <- loan_data[147, c('purpose_', 'home_', 'emp_len_')]
> row.names(new_loan) <- NULL
> new_loan
      purpose_    home_  emp_len_
1 small_business MORTGAGE > 1 Year

```

이 경우, 모델은 연체를 예상한다.

```

> predict(naive_model, new_loan)
$class
[1] default
Levels: paid off default

$posterior
      paid off    default
[1,] 0.3463013 0.6536987

```

예측 결과에는 디폴트의 확률에 대한 posterior(사후) 추정도 함께 있다. 나이브 베이지언 분류기는 편향된 추정 결과를 예측하는 것으로 잘 알려져 있다. 하지만 $Y=1$ 인 확률에 따라 레코드들에 순위를 매기는 것이 목적이므로 확률의 비편향된 추정치를 굳이 구할 필요가 없다면, 나이브 베이즈도 나름 우수한 결과를 보인다.

5.1.3 수치형 예측변수

정의에 따라, 베이지언 분류기는 예측변수들이 범주형인 경우 (스팸 메일 분류에서 특정 단어, 야구, 문자열의 존재 여부 등)에 적합하다. 수치형 변수에 나이브 베이즈 방법을 적용하기 위해서는, 두 가지 접근법 중 하나를 따라야 한다.

- 수치형 예측변수를 비닝(binning)하여 범주형으로 변환한 뒤, 알고리즘을 적용한다.
- 조건부확률 $P(X_j|Y=i)$ 를 추정하기 위해 정규분포 같은 확률 모형을 사용한다.

CAUTION 훈련 데이터에 예측변수의 특정 카테고리에 해당하는 데이터가 없을 때에는, 다른 기법들처럼 이 변수를 무시하고 다른 변수들의 정보를 사용하는 대신, 나이브 베이즈 알고리즘은 데이터 결과에 대한 확률을 0으로 할당한다. 연속형 변수를 비닝할 때는, 이런 부분에 대해 주의를 기울여야 한다.

bin : 버리다.

주요개념

- 나이브 베이즈는 예측변수와 결과변수 모두 범주형(요인)변수 여야 한다.
- '각 출력 카테고리 안에서, 어떤 예측변수의 카테고리가 가장 가능성이 높은가?'를 답하고자 대답을 얻기위해 사용한다.
- 이 정보는 주어진 예측변수 값에 대해, 결과 카테고리의 확률을 추정하는 것으로 바뀐다.

5.3 판별분석

판별분석은 초창기의 통계 분류 방법이다.

용어 정리

- 공분산 : 하나의 변수가 다른 변수와 함께 변화하는 정도(유사한 크기와 방향)을 측정하는 지표
- 판별함수 : 예측변수에 적용했을 때, 클래스 구분을 최대화하는 함수
- 판별 가중치 : 판별함수를 적용하여 얻은 점수를 말하며, 어떤 클래스에 속할 확률을 추정하는 데 사용된다.

판별분석에는 여러 가지 기법이 있지만 그 가운데 가장 일반적으로 사용되는 것은 **선형판별분석(linear discriminant analysis (LDA))**이다. 피셔는 처음 제안했던 원래 방법은 실제 LDA와 약간 다르지만 동작하는 원리는 본질적으로 같다. 트리 모델이나 로지스틱 회귀와 같은 더 정교한 기법이 출현한 이후로는 LDA를 그렇게 많이 사용하지 않는다.

하지만 여전히 일부 응용 분야에서는 LDA를 사용하고 있으며, 주성분분석과 같이 아직도 많이 사용되는 다른 방법들과 연결된다. 또한 판별분석은 예측변수들의 중요성을 측정하거나 효과적으로 특징을 선택하는 방법으로도 사용된다.

CAUTION 선형판별분석과 약자가 같은 잠재 디리클레 할당(latent Dirichlet allocation)과 혼동하지 않도록 유의하자. 잠재 디리클레 할당은 텍스트와 자연어 처리에 사용되는 방법으로 선형판별분석과 아무런 관련이 없다.

5.2.1 공분산행렬

판별분석을 이해하려면, 두 개 이상의 변수 사이에 공분산이라는 개념을 먼저 도입해야 한다. 공분산이란 두 변수 x 와 z 사이의 관계를 의미하는 지표이다. \bar{x} 와 \bar{z} 는 각 변수의 평균을 나타낸다. x 와 z 사이의 공분산 $s_{x,z}$ 은 다음과 같다.

$$s_{x,z} = \sum_{i=1}^n (x_i - \bar{x})(z_i - \bar{z}) / (n - 1)$$

여기서 n 은 레코드의 개수를 의미한다. (n 대신 $n-1$ 을 사용했다는 점에 주목하자.)

상관계수 때와 마찬가지로 양수는 양의 관계를, 음수는 음의 관계를 나타낸다. 하지만 상관관계가 -1에서 -1 사이에서 정의됐다면, 공분산은 변수 x 와 z 에서 사용하는 척도와 동일한 척도에서 정의된다.

x 와 z 에 대한 공분산행렬 $\hat{\Sigma}$ 는 각 변수의 분산 s_x^2 과 s_z^2 을 대각원소로 놓고, 변수들 사이의 공분산을 비대각원소에 위치시킨 행렬이다.

$$\hat{\Sigma} = \begin{bmatrix} s_x^2 & s_{x,z} \\ s_{x,z} & s_z^2 \end{bmatrix}$$

Note_ 변수를 z 점수로 변환할 때 표준편차를 사용했던 것을 떠올려보자. 이를 확장하여 다변량분석에서 표준화 처리를 하기 위해 공분산행렬을 사용하는 것이다. 이를 마할라노비스 거리의 '다른 거리 지표'라고 부르며 LDA 함수와 관련이 있다.

5.2.2 피셔의 선형판별

간단한 설명을 위해, 두 개의 연속형 변수 (x,z)을 사용하여 이진 결과변수 y를 예측하려는 분류 문제가 있다고 하자. 기술적으로, 판별분석은 보통 예측변수가 정규분포를 따르는 연속적인 변수라는 가정이 있지만 실제로는 정규분포에서 벗어나거나 이진 예측변수에 대해서도 잘 동작한다.

피셔의 선형판별은 그룹 안의 편차와 다른 그룹 간의 편차를 구분한다. 구체적으로, 레코드를 두 그룹으로 나누는 방법을 찾기 위해, LDA는 '내부' 제곱합 $SS_{\text{내부}}$ (그룹 안의 변동을 측정)에 대한 $SS_{\text{사이}}$ (두 그룹 사이의 편차를 측정)의 **비율을 최대화하는 것을 목표로 한다**. 두 그룹은 $y=0$ 에 대해 (x_0, z_0) , $y=1$ 에 대해 (x_1, z_1) 으로 나뉘게 된다. 이 방법은 다음 제곱합 비율을 최대화하는 선형결합 $w_x x + w_z z$ 을 찾는다.

$$\frac{SS_{\text{사이}}}{SS_{\text{내부}}}$$

사이 제곱합의 각 값은 두 그룹 평균 사이의 거리 제곱을 말하며, 내부 제곱합은 공분산행렬에 의해 가중치가 적용된, 각 그룹 내의 평균을 주변으로 퍼져 있는 정도를 나타낸다. 직관적으로, 사이 제곱합을 최대화하고 내부 제곱합을 최소화하는 것이 두 그룹 사이를 가장 명확하게 나누는 방법이다.

5.2.3 간단한 예

R의 MASS패키지는 LDA 함수를 제공한다. 아래 예제는 두 예측변수 borrower_score와 payment_inc_ratio를 이용해 대출 데이터 표본에 이 함수를 적용하고 **선형판별자 가중치**를 구한다.

```
> loan_lda <- lda(outcome ~ borrower_score + payment_inc_ratio,
+                 data=loan3000)
> loan_lda$scaling
               LD1
borrower_score    7.17583880
payment_inc_ratio -0.09967559
```

Note_ 특징 선택에 판별분석 사용하기

LDA를 돌리기 전에 미리 예측변수들을 정규화했다면, 판별자 가중치는 변수의 중요도를 의미하게 된다. 따라서 특징 선택을 위해 계산상으로 효과적인 방법이다.

lda 함수를 이용해 다음과 같이 상환과 연체에 대한 확률을 계산할 수 있다.

```
> pred <- predict(loan_lda)
> head(pred$posterior)
  default paid off
1 0.5535437 0.4464563
2 0.5589534 0.4410466
3 0.2726962 0.7273038
4 0.5062538 0.4937462
5 0.6099525 0.3900475
6 0.4107406 0.5892594
```

예측에 대한 결과를 시각해서 볼 수 있다면 LDA가 잘 동작하는지 쉽게 알 수 있을 것이다. predict 함수의 출력값을 사용하여, 다음과 같이 체납에 대한 확률값을 그래프로 시각화할 수 있다.

```
pred <- predict(loan_lda)
lda_df <- cbind(loan3000, prob_default=pred$posterior[, 'default'])

x <- seq(from=.33, to=.73, length=100)
y <- seq(from=0, to=20, length=100)
newdata <- data.frame(borrower_score=x, payment_inc_ratio=y)
pred <- predict(loan_lda, newdata=newdata)
lda_df0 <- cbind(newdata, outcome=pred$class)

ggplot(data=lda_df, aes(x=borrower_score, y=payment_inc_ratio,
color=prob_default)) +
  geom_point(alpha=.6) +
  scale_color_gradient2(low='white', high='blue') +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0), lim=c(0, 20)) +
  geom_line(data=lda_df0, col='green', size=2, alpha=.8) +
  theme_bw()
```

판별 결과 얻은 가중치를 이용해, LDA는 실선을 이용해 예측변수 영역을 두 부분으로 나눈다.

직선에서 멀리 떨어진 예측결과일수록 신뢰도가 높다(즉, 확률이 0.5로부터 멀어진다).

Note_ 판별분석의 확장

먼저 예측변수가 많아질 경우다. 지금까지 예제에서는 예측변수가 두 개인 경우에 대해서만 이야기 했지만, LDA는 예측변수가 두 개보다 많아도 잘 동작한다. 단지 제한요소가 있다면 데이터 개수다. 변수 당 충분한 수의 레코드가 있어야 공분산을 계산할 수 있기 때문이다. 하지만 데이터 과학 응용 분야에서는 일반적으로 문제가 되지 않는다.

다음은 이차판별분석이다. 기본 판별분석의 다른 형태가 있는데, 그 가운데 가장 많이 알려진것이 이차 판별분석(quadratic discriminant analysis(QDA)) 이다. 이름과는 달리, QDA는 여전히 선형판별함수만 사용한다. LDA와 가장 큰 차이점은, LDA는 Y=0인 그룹과 Y=1인 그룹의 공분산행렬이 모두 동일해야 한다는 가정을 필요로 한다는 점이다. QDA에서는 이 두 그룹이 서로 다른 공분산을 갖을 수 있다. 실무적으로는, 대부분의 경우 이 차이가 그렇게 크지 않다.

주요개념

- 판별분석은 예측변수나 결과변수가 범주형이든 연속형이든 상관없이 잘 동작한다.
- 공분산행렬을 사용하여 한 클래스와 다른 클래스에 속한 데이터들을 구분하는 선형판별함수를 계산할 수 있다.
- 이 함수를 통해 각 레코드가 어떤 클래스에 속할 가중치 혹은 점수(각 클래스당 점수)를 구한다.

5.3 로지스틱 회귀

로지스틱 회귀는 결과가 이진형 변수라는 점만 빼면 다중선형회귀와 유사하다. 선형모형에 적합한 문제로 변환하기 위해 사용되는 다양한 변환 방법이 있다. 판별분석과 비슷하면서, 동시에 K 최근접 이웃이나 나이브 베이즈와는 다르게, 로지스틱 회귀는 데이터 위주의 접근 방식이라기보다, 구조화된 모델 접근 방식이라고 할 수 있다. 빠른 계산 속도와 새로운 데이터에 대한 빠른 점수 산정 덕분에 다양한 분야에서 널리 사용되고 있다.

용어정리

- 로짓 : (0~1이 아니라) $\pm\infty$ 범위에서 어떤 클래스에 속할 확률을 결정하는 함수 (유의어 : 로그 오즈)
- 오즈 : '실패'(0)에 대한 '성공'(1)의 비율
- 로그 오즈 : 변환 모델(선형)의 응답변수. 이 값을 통해 확률을 구한다.

어떻게 이진 결과변수로부터 선형으로 모델링이 가능한 결과변수를 구할 수 있을까? 그리고 그다음 다시 원래 이진 결과로 되돌릴 수 있을까?

5.3.1 로지스틱 반응 함수와 로짓

핵심 구성 요소는 로지스틱 반응 함수와 로짓이다. 여기서 우리는 확률 (0에서 1사이의 단위)을 선형 모델링에 적합한 더 확장된 단위로 매핑한다.

첫 번째 단계에서, 결과변수를 이진값으로 생각하기보다 라벨이 '1'이 될 확률 p 로 생각해보자.

당연히 p 를 다음과 같이 예측변수들의 선형함수로 모델링하고 싶은 유혹이 있을 것이다.

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q$$

그러나 이 모델을 피팅한다고 해도, 당연히 선형모델이다 보니 p 가 0과 1 사이로 딱 떨어지지 않을 수 있다. 더 이상 확률이라고 할 수 없다.

대신, 예측변수에 로지스틱 반응 혹은 역 로짓 함수라는 것을 적용해 p 를 모델링한다.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q)}}$$

이 변환을 통해 우리는 p 가 항상 0에서 1사이에 오도록 할 수 있다. 분모의 지수 부분을 구하려면 확률 대신 오즈비를 이용한다. 어딜 가나 내기 좋아하는 사람들에게 친숙한 오즈비 '성공(1)'과 '실패(0)'의 비율을 말한다. 확률의 관점에서, 오즈비는 사건이 발생할 확률을 사건이 발생하지 않을 확률로 나눈 비율이다. 예를 들면 어떤 말이 이길 확률이 0.5라면 '이기지 못할 확률'은 $(1-0.5) = 0.5$ 이고 오즈비는 1.0이다.

$$\text{오즈}(Y=1) = \frac{p}{1-p}$$

우리는 또한 역 오즈비 함수를 통해 확률값을 구할 수도 있다.

$$p = \frac{\text{오즈}}{1 + \text{오즈}}$$

오즈 수식을 로지스틱 반응 함수에 적용하면 다음과 같은 수식을 얻을 수 있다.

$$\text{오즈}(Y=1) = e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q)}$$

마지막으로 양변에 로그 함수를 취하면 우리는 예측변수에 대한 선형함수를 얻을 수 있다.

$$\log(\text{오즈}(Y=1)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q$$

로그 오즈함수, 또는 로짓 함수는 0과 1 사이의 확률 p 를 $-\infty$ 에서 $+\infty$ 까지의 값으로 매핑해준다.

다음그림을 참고하자. 이렇게 변환 과정이 모두 마무리되었다. 우리는 이제 어떤 확률을 예측할 수 있는 선형모델을 구했다. 이제 컷오프 (절사) 기준을 이용해 그 값보다 큰 확률값이 나오면 1로 분류하는 식의 과정을 통해 클래스 라벨을 구할 수 있다.


```

p <- seq(from=0.01, to=.99, by=.01)
df <- data.frame(p = p ,
                 logit = log(p/(1-p)),
                 odds = p/(1-p))

## Figure 5-2

png(filename=file.path(PSDS_PATH, 'figures', 'psds_0502.png'), width = 5,
height=4, units='in', res=300)
ggplot(data=df, aes(x=p, y=logit)) +
  geom_line() +
  labs(x = 'p', y='logit(p)') +
  theme_bw()
dev.off()

```

5.3.2 로지스틱 회귀와 GLM

앞서 유도한 로지스틱 회귀방정식에서 응답변수는 1의 이진 출력에 대한 로그 오즈 값이었다.

하지만 우리가 실제 관찰한 데이터는 로그 오즈 값이 아닌 이진 출력값이다. 따라서 이 방정식을 피팅하기 위해서는 특별한 확률 기법이 필요하다. 로지스틱 회귀는 선형회귀를 확장한 **일반화선형모형(GLM)**의 특별한 사례이다.

R에서 로지스틱 회귀를 구하려면 family인수를 binomial로 지정하고 glm함수를 사용해야 한다.

```

> ## Logistic regression
> logistic_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
+                       home_ + emp_len_ + borrower_score,
+                       data=loan_data, family='binomial')
> logistic_model

Call:  glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
emp_len_ + borrower_score, family = "binomial", data = loan_data)

Coefficients:
            (Intercept)            payment_inc_ratio
            1.63809                0.07974
purpose_debt_consolidation  purpose_home_improvement
            0.24937                0.40774
purpose_major_purchase      purpose_medical
            0.22963                0.51048
purpose_other               purpose_small_business
            0.62066                1.21526
            home_OWN          home_RENT
            0.04833                0.15732
emp_len_ > 1 Year            borrower_score
            -0.35673            -4.61264

Degrees of Freedom: 45341 Total (i.e. Null); 45330 Residual
Null Deviance: 62860
Residual Deviance: 57510 AIC: 57540
> summary(logistic_model)

Call:
glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
emp_len_ + borrower_score, family = "binomial", data = loan_data)

```

```

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.51951 -1.06908 -0.05853  1.07421  2.15528

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.638092   0.073708  22.224 < 2e-16 ***
payment_inc_ratio 0.079737   0.002487  32.058 < 2e-16 ***
purpose_debt_consolidation 0.249373   0.027615   9.030 < 2e-16 ***
purpose_home_improvement 0.407743   0.046615   8.747 < 2e-16 ***
purpose_major_purchase 0.229628   0.053683   4.277 1.89e-05 ***
purpose_medical 0.510479   0.086780   5.882 4.04e-09 ***
purpose_other 0.620663   0.039436  15.738 < 2e-16 ***
purpose_small_business 1.215261   0.063320  19.192 < 2e-16 ***
home_OWEN      0.048330   0.038036   1.271  0.204
home_RENT      0.157320   0.021203   7.420 1.17e-13 ***
emp_len_ > 1 Year -0.356731   0.052622  -6.779 1.21e-11 ***
borrower_score -4.612638   0.083558 -55.203 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 62857  on 45341  degrees of freedom
Residual deviance: 57515  on 45330  degrees of freedom
AIC: 57539

Number of Fisher Scoring iterations: 4

```

outcome이 응답변수이다. 대출을 모두 갚았으면 0, 연체 상태이면 1의 값을 갖는다. purpose_와 home_ 으로 시작하는 변수들은 각각 대출 목적과 주택 소유 상태를 나타내는 요인변수들이다.

회귀에서처럼, P 개의 수준을 갖는 요인변수는 $P - 1$ 개의 열로 표시할 수 있다. R에서는 기본적으로 기준 코딩을 사용하고, 기준 수준에 다른 수준들을 비교해서 사용한다. 이러한 요인변수들에 대한 기준 수준들은 각각 credit_card와 MORTGAGE이다. 변수 borrower_score은 차용인의 신용도(불량에서 우수까지)를 나타내는 0에서 1까지의 점수이다.

이 변수는 K최근접 이웃 알고리즘을 사용하는 몇 가지 다른 변수로부터 만들어졌다.

5.3.3 일반화선형모형

일반화선형모형(GLM)은 회귀와 함께 두 번째로 가장 중요한 모델이다. GLM은 다음 두 가지 주요 구성 요소로 특징 지어진다.

- 확률분포 또는 분포군(로지스틱의 경우 이항분포)
- 응답을 예측변수에 매핑하는 연결함수(로지스틱 회귀의 경우 로짓)

분명히, 로지스틱 회귀는 GLM의 가장 널리 알려진 일반적인 형태이다. 데이터 과학자는 다른 유형의 GLM을 접할 것이다. 때로는 로짓 대신에 로그 연결 함수를 사용한다. 실제로 로그 연결을 사용하더라도, 매우 다른 결과가 발생할 가능성은 대부분의 응용 분야에서 거의 없다. 푸아송 분포는 일반적으로 카운트 데이터(예를 들면 사용자가 일정 시간 동안 웹 페이지를 방문한 횟수)를 모델링하는 데 사용된다. 다른 분포군으로는 **음이항분포**와 **감마 분포** 등이 있는데 이들은 경과 시간(예를 들면 고장 시간)을 모델링하는 데 자주 사용된다. 로지스틱 회귀와는 달리, 이 모델들을 GLM에 적용하는 것은 더 미묘한 차이를 발생시키므로 주의를 기울여야 한다.

이러한 방법의 유용성과 위험성을 모두 잘 알고 어느 정도 익숙하지 않다면, 피하는 것이 좋다.

5.3.4 로지스틱 회귀의 예측값

로지스틱 회귀에서 예측하는 값은 로그 오즈 $\hat{Y} = \log(\text{오즈}(Y = 1))$ 에 관한 값이다. 예측된 확률은 로지스틱 반응 함수에 의해 주어진다.

$$\hat{p} = \frac{1}{1+e^{-\hat{Y}}}$$

예를 들면 모델 `logistic_model`로부터 얻은 예측값을 살펴보자.

```
> pred <- predict(logistic_model)
> summary(pred)
      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
-2.704774 -0.518825 -0.008539  0.002564  0.505061  3.509606
```

이 값들을 다음과 같이 간단한 변환을 통해 확률값으로 바꿀 수 있다.

```
> prob <- 1/(1 + exp(-pred))
> summary(prob)
      Min. 1st Qu.  Median     Mean
 0.06269 0.37313 0.49787 0.50000
 3rd Qu.    Max.
 0.62365 0.97096
```

이 값들은 0에서 1 사이에 있을 뿐이지, 아직 이 예측 결과가 연체인지 아니면 빚을 갚는 것인지는 분명히 말해주지 않는다. K 최근접 이웃 분류기와 유사하게, 기본으로 1.5보다 큰 값을 사용하면 판별할 수 있다. 실무에서 희귀한 클래스에 속한 구성원을 확인하는 것이 목표라면 이 기준값을 더 낮게 할수록 좋은 경우가 종종 있다.

5.3.5 계수와 오즈비 해석하기

로지스틱 회귀의 장점 중 하나는 재계산 없이 새 데이터에 대해 빨리 결과를 계산할 수 있다는 점이다. 또 다른 하나는 모델을 해석하기가 다른 분류 방법들에 비해 상대적으로 쉽다는 점이다. 여기서 가장 중요한 개념은 바로 **오즈비**를 이해하는 것이다. 오즈비를 가장 쉽게 이해하는 방법은 이진 요인 변수 X 를 가지고 생각해 보는 것이다.

$$\text{오즈비} = \frac{\text{오즈}(Y=1|X=1)}{\text{오즈}(Y=1|X=0)}$$

위의 식은 $X = 1$ 일 때, $Y = 1$ 인 경우의 오즈와 $X = 0$ 일 때, $Y = 1$ 인 경우의 오즈를 비교한 것이라고 해석할 수 있다.

왜 굳이 확률 대신 오즈비를 사용해 이렇게 귀찮은 일을 하는 걸까? 로지스틱 회귀분석에서 계수 β_j 는 X_j 에 대한 오즈비의 로그 값이기에, 오즈비를 사용한다.

다음 예제가 이를 명확하게 이해하는 데 도움이 될 것이다. 5.3.2절에서 구한 모델에서, 변수 `purpose_small_business`에 대한 회귀계수는 1.21226이었다. 이것은 신용카드 빚을 갚기 위해 대출과 비교했을 때, 소규모 사업을 위한 대출은 $\exp(1.21226) = 3.4$ 만큼 대출을 갚는 것에 비해 빚을 갚지 않을 오즈비가 감소한다는 것을 의미한다.(오즈비가 감소하면, 안할 가능성이 높다는 것이다.)

분명히 소규모 사업을 창업하거나 확장하기 위한 목적의 대출은 다른 유형의 대출보다 훨씬 더 위험하다.

다음 그림은 오즈비가 1보다 클 경우, 오즈비와 로그 오즈비 사이의 관계를 보여준다. 계수가 로그 스케일이다 보니, 계수가 1만큼 증가할수록 결과적으로 오즈비는 $\exp(1) = 2.72$ 만큼 증가한다.

```
ggplot(data=df, aes(x=logit, y=odds)) +
  geom_line() +
  labs(x = 'log(odds ratio)', y='odds ratio') +
  ylim(1, 100) +
  xlim(0, 5) +
  theme_bw()
```

수치형 변수 X 에 대해서도 마찬가지로 비슷한 의미를 갖는다. X 에서 단위 크기 만큼 변화할 때 오즈비에서의 변화를 생각할 수 있다. 예를 들면, 소득에 대한 상환비율이 5에서 6만큼 증가했다고 하면, $\exp(0.08244) = 1.09$ 만큼 연체할 오즈비가 증가한다. 변수 borrow_score는 대출자의 신용도를 0(낮음)에서 1(높음)까지 변화한다. 현재 연체 중인 최악의 차용인에 대한 가장 우수한 차용인의 오즈비는 $\exp(-4.63890) = 0.01$ 정도로 훨씬 더 적다. 즉, 가장 신용이 불량한 차용인의 연체 위험도는 신용이 가장 좋은 차용자에 비해 100배 정도 더 안좋다!!

5.3.6 선형회귀와 로지스틱 회귀 : 유사점과 차이점

다중선형회귀와 로지스틱 회귀는 공통점이 많다. 두 가지 모두 예측변수와 응답변수를 선형 관계로 가정한다. 가장 좋은 모델을 탐색하고 찾는 과정도 아주 유사하다. 대부분의 선형모형에서, 예측변수에 스피라인 변환을 사용하는 방법은 로지스틱 회귀에서도 똑같이 적용할 수 있다.

하지만 로지스틱 회귀는 아래 두 가지 점에서 근본적인 차이가 있다.

- 모델을 피팅하는 방식(최소제곱을 사용할 수 없다)
- 모델에서 잔차의 특징과 분석

모델 피팅

선형회귀에서는 모델 피팅을 위해 최소제곱을 사용한다. RMSE와 R제곱 통계량을 사용하여 피팅의 성능을 평가한다. 로지스틱 회귀분석에서는 (선형회귀와는 달리) 닫힌 형태의 해가 없으므로 **최대우도추정(maximum likelihood estimation (MLE))**을 사용하여 모델을 피팅해야 한다. 최대우도추정이란, 우리가 보고 있는 데이터를 생성했을 가능성이 가장 큰 모델을 찾는 프로세스를 말한다. 로지스틱 회귀식에서 응답변수는 0이나 1이 아닌, 응답이 1인 로그 오즈비의 추정치이다. MLE는 예상 로그 오즈비가 관찰된 결과를 가장 잘 설명하는 모델을 찾는다. 알고리즘은 현재 파라미터에 기반하여 저무를 얻는 단계 (**피셔의 점수화**)와 적합성을 향상시키는 방향으로 파라미터를 업데이트하는 단계를 계속적으로 반복하는 준 뉴턴 최적화 메커니즘(quasi-Newton optimization)으로 동작한다.

최대우도추정

좀 더 자세히 다루기 위해, 통계 기호를 조금 사용하자. 일련의 데이터 (X_1, X_2, \dots, X_n) 과 파라미터 집합 θ 에 따른 확률모형 $P_\theta(X_1, X_2, \dots, X_n)$ 을 가지고 시작하자. MLE의 목표는 $P_\theta(X_1, X_2, \dots, X_n)$ 의 값을 최대화하는 파라미터의 집합 $\hat{\theta}$ 를 찾는 것이다. 즉, 이것은 다시 말해서 주어진 모델 P 에서 (X_1, X_2, \dots, X_n) 을 관측할 확률을 최대화하는 것이다. 피팅 과정에서 **편차**

라는 지표를 사용하여 모델을 평가한다.

$$\text{편차} = -2\log(P_\theta(X_1, X_2, \dots, X_n))$$

편차가 작을수록 모델 적합도가 높은 것을 의미한다.

다행히 SW에서 이러한 내용들을 처리해주기 때문에, 대부분의 사용자는 피팅 알고리즘의 세부 사항에 신경 쓸 필요가 없다. 대부분의 데이터 과학자들은 이것이 어떤 가정하에서 좋은 모델을 찾는 방법이라는 것을 이해하는 것 말고는 다른 피팅 방법에 대해 걱정할 필요는 없다.

CAUTION_ 요인변수 다루기

로지스틱 회귀에서 요인변수는 선형회귀에서처럼 인코딩하는 과정을 거쳐야 한다. 4.4절을 다시 참고하자. R이나 다른 SW에서, 이 과정을 자동으로 처리하고 보통은 기준 인코딩을 사용한다. 이 장에서 다루는 다른 모든 분류 방법들은 원-핫 인코딩 방법으로 사용한다.

5.3.7 모델 평가하기

다른 분류 방법들과 마찬가지로, 모델이 새로운 데이터를 얼마나 정확하게 분류하는가를 기준으로 로지스틱 회귀를 평가한다. 선형회귀와 같이, 표준 통계 도구들을 사용해 모델을 평가하고 향상시킬 수 있다. 예측된 계수들과 함께, R은 계수들의 표준오차(SE), z점수, p값을 출력한다.

```
logistic_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
                        home_ + emp_len_ + borrower_score,
                        data=loan_data, family='binomial')

logistic_model

> summary(logistic_model)

Call:
glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
    emp_len_ + borrower_score, family = "binomial", data = loan_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.51951  -1.06908  -0.05853   1.07421   2.15528

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.638092   0.073708  22.224 < 2e-16 ***
payment_inc_ratio  0.079737   0.002487  32.058 < 2e-16 ***
purpose_debt_consolidation 0.249373   0.027615   9.030 < 2e-16 ***
purpose_home_improvement  0.407743   0.046615   8.747 < 2e-16 ***
purpose_major_purchase    0.229628   0.053683   4.277 1.89e-05 ***
purpose_medical          0.510479   0.086780   5.882 4.04e-09 ***
purpose_other            0.620663   0.039436  15.738 < 2e-16 ***
purpose_small_business    1.215261   0.063320  19.192 < 2e-16 ***
home_OWEN                0.048330   0.038036   1.271   0.204
home_RENT                0.157320   0.021203   7.420 1.17e-13 ***
emp_len_ > 1 Year        -0.356731   0.052622  -6.779 1.21e-11 ***
borrower_score          -4.612638   0.083558 -55.203 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 62857  on 45341  degrees of freedom
Residual deviance: 57515  on 45330  degrees of freedom
AIC: 57539

Number of Fisher Scoring iterations: 4
```

p값을 해석할 때, 회귀에서 언급했던 주의사항도 같이 따라온다. 통계적인 유의성을 측정하는 지표로 보기보다는 변수의 중요성을 나타내는 상대적인 지표로 봐야한다. 이진 응답변수가 있는 로지스틱 회귀모형은 RMSE나 R 제곱이 있을 수 없다. 대신 분류 문제에서 가장 일반적으로 사용되는 측정 지표들을 사용할 수 있다.

선형회귀에 적용되었던 많은 개념이 로지스틱 회귀(그리고 다른 GLM들)에도 똑같이 이어진다.

예를 들면 여기에서도 단계적 회귀, 상호작용 항 도입, 스플라인 항 포함 등을 모두 사용할 수 있다. 로지스틱 회귀에 적용되던 교란변수나 변수 상관과 관련한 문제들도 동일하게 고려해야 한다. mgcv패키지를 이용해 일반화가법모형을 이용할 수도 있다.

```
> logistic_gam <- gam(outcome ~ s(payment_inc_ratio) + purpose_ +  
+                       home_ + emp_len_ + s(borrower_score),  
+                       data=loan_data, family='binomial')  
> logistic_gam
```

Family: binomial

Link function: logit

Formula:

outcome ~ s(payment_inc_ratio) + purpose_ + home_ + emp_len_ +
s(borrower_score)

Estimated degrees of freedom:

7.66 4.17 total = 21.83

UBRE score: 0.2681506

로지스틱 회귀가 선형회귀와 다른 부분은 바로 잔차에 관한 내용이다. 회귀에서 처럼 편잔차를 계산하는 것은 다음과 같이 수월하다.

```
library(mgcv)  
logistic_gam <- gam(outcome ~ s(payment_inc_ratio) + purpose_ +  
+                   home_ + emp_len_ + s(borrower_score),  
+                   data=loan_data, family='binomial')  
logistic_gam  
  
terms <- predict(logistic_gam, type='terms')  
partial_resid <- resid(logistic_gam) + terms  
df <- data.frame(payment_inc_ratio = loan_data[, 'payment_inc_ratio'],  
+               terms = terms[, 's(payment_inc_ratio)'],  
+               partial_resid = partial_resid[, 's(payment_inc_ratio)'])  
  
## Code for Figure 5-4  
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0504.png'),  
+    width = 5, height=4, units='in', res=300)  
  
ggplot(df, aes(x=payment_inc_ratio, y=partial_resid, solid = FALSE)) +  
  geom_point(shape=46, alpha=.4) +  
  geom_line(aes(x=payment_inc_ratio, y=terms),  
+           color='red', alpha=.5, size=1.5) +  
  labs(y='Partial Residual') +  
  xlim(0, 25) +  
  theme_bw()
```

그림에서 점들이 뭉쳐 있는 구름 같은 모양이 두 군데 있고, 추정 결과로 얻은 회귀선이 그 사이를 지나가고 있다. 위쪽 구름은 1의 응답(연체)을 의미하고, 아래쪽 구름은 0의 응답(대출 상환)을 의미한다.

결과변수가 이진형이기 때문에, 로지스틱회귀에서 얻은 잔차는 보통 이러한 형태를 띄게 된다. 로지스틱 회귀에서 편잔차는 회귀에서보다 덜 중요하긴 하지만, 비선형성을 검증하고 영향력이 큰 레코드들을 확인하는 데 여전히 유용하다.

CAUTION_ summary 함수에서 어떤 출력은 사실상 무시할 수 있다. 분포도 파라미터는 다른 유형의 GLM에는 적용되는 반면, 로지스틱 회귀에는 적용되지 않는다. 잔차 편차나 점수화 반복 횟수는 최대우도 피팅 방법과 관련이 있다.

주요개념

- 로지스틱 회귀는 출력이 이진변수라는 점만 빼면, 선형회귀와 매우 비슷하다.
- 선형모형과 비슷한 형태의 모델을 만들기 위해, 응답변수로 오즈비의 로그값을 사용하는 등의 몇 가지 변환이 필요하다.
- 반복 과정을 통해 선형모형을 피팅하고 나면, 로그 오즈비는 다시 확률값으로 변환된다.
- 로지스틱 회귀는 계산속도가 빠르고 새로운 데이터에 대해서도 빠르게 결과를 구할 수 있다는 장점 때문에 많이 사용한다.

5.4 분류 모델 평가하기

예측 모델링에서, 수많은 모델을 시도해보고 각각에 홀드아웃 표본(시험표본(test sample) 혹은 타당성검사 표본(validation sample) 이라고도 부른다)을 적용하고 성능을 평가하는 것은 아주 일반적이다. 가장 정확한 예측을 얻기 위해 기본적으로 필요한 일들이다.

용어 정리

- 정확도 : 정확히 분류된 비율
- 혼동행렬 : 분류에서 예측된 결과와 실제 결과에 대한 레코드의 개수를 표시한 테이블(이진형인 경우 2x2)
- 민감도(sensitivity) : 정확히 분류된 1의 비율(유의어 : 재현율 recall)
- 특이도(specificity) : 정확히 분류된 0의 비율
- 정밀도(precision) : 정확히 1이라고 예측된 1의 비율
- ROC곡선(ROC Curve) : 민감도와 특이도를 표시한 그림
- 리프트(lift) : 모델이 다른 확률 컷오프에 대해 (비교적 드문) 1을 얼마나 더 효과적으로 구분하는지 나타내는 측정 지표

분류 성능을 측정하는 가장 간단한 방법은 정확히 예측한 것들의 비율이 얼마인지 보는 것이다.

대부분의 분류 알고리즘에서 각 데이터에 대해 1이 될 확률값을 추정하여 할당한다.

(역자 주석: 모든 방법이 비편향 확률 예측을 하는 것은 아니다. 대부분 비편향 확률 추정을 통해 얻은 순위 정보를 알려준다. 이런 경우에도 컷오프 방법은 같은 방식으로 동작한다.)

가장 기본적인 컷오프 기준값은 0.5 즉 50%이다. 확률이 0.5보다 크면 분류 결과는 1, 그렇지 않으면 0이 된다. 또 다른 방법은, 실제 데이터에서 1이 차지하는 비율을 컷오프로 사용하는 방법이 있다.

정확도는 아래 수식과 같다.

$$\text{정확도} = \frac{\sum \text{참 양성} + \sum \text{참 음성}}{\text{표본 크기}}$$

5.4.1 혼동행렬

혼동행렬은 분류 결과를 나타내는 가장 대표적인 행렬이다. 혼동행렬은 응답 유형별로 정확한 예측과 잘못된 예측의 수를 한 번에 보여주는 표다. R에서는 여러 가지 패키지를 사용하여 혼동행렬을 구할 수 있다. 물론 이진 경우에는 간단히 손으로 계산할 수도 있다.

혼동행렬을 설명하기 위해 균형 잡힌 데이터, 즉 동일한 수의 대출 연산/상환 데이터를 이용해 학습한 모델 *logistic_gam* 을 생각해보자. 일반적인 관례에 따라 $Y = 1$ 은 관심이 있는 사건(연체)에 해당하고 $Y = 0$ 은 그 반대인 통상적 사건(상환)으로 두겠다. 다음은 전체 훈련 데이터 (불균형)에 적용한 *logistic_gam* 모델의 혼동행렬을 계산한다.

```
pred <- predict(logistic_gam, newdata=loan_data)
pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(loan_data$outcome=='default')
true_pos <- (true_y==1) & (pred_y==1)
true_neg <- (true_y==0) & (pred_y==0)
false_pos <- (true_y==0) & (pred_y==1)
false_neg <- (true_y==1) & (pred_y==0)
conf_mat <- matrix(c(sum(true_pos), sum(false_pos),
                      sum(false_neg), sum(true_neg)), 2, 2)
colnames(conf_mat) <- c('Yhat = 1', 'Yhat = 0')
rownames(conf_mat) <- c('Y = 1', 'Y = 0')
conf_mat

> conf_mat
      Yhat = 1 Yhat = 0
Y = 1    14293     8378
Y = 0     8051    14620
```

결과에서 열은 예측값이고 행은 실제 결과를 의미한다. 행렬의 대각원소들은 정확히 예측한 데이터의 수를 의미하며 비대각원소들은 부정확한 예측의 수를 의미한다. 예를 들면 14,293 건의 연체는 정확히 연체라는 예측결과를 보였다. 하지만 8,501건의 연체는 대출을 상환한다고 잘못된 예측을 했다.

 image-20200417155010861

위 그림은 이진 응답변수 Y 에 대한 혼동행렬과 또 다른 지표들을 보여준다. 대출 데이터의 예제에서 보았듯, 실제 응답변수는 행을 따라, 그리고 예측 응답변수는 열을 따라 표시된다(행과 열의 의미를 바꾸어 사용하는 경우도 종종 있다). 다각 방향의 칸(왼쪽 상단, 오른쪽 하단)은 예측변수 \hat{Y} 가 정확한 값을 예측하는 경우의 수를 의미한다. 여기서 눈에 띄지 않지만 중요한 지표 중 하나는 **거짓 양성비율(정밀도와 대응되는 개념)**이다. 결과가 1인 데이터의 수가 희박할 때, 모든 예측 응답변수에 대해 거짓 양성 값의 비율이 높아져, 예측 결과는 1이지만 실제로는 0일 가능성이 높은 상황이 된다. 이 문제는 광범위하게 적용되는 의료 검진 검사(예를 들면 유방 조영술)를 어렵게 하는 요인이다. 상대적으로 발생하는 비율이 드물기 때문에, 검사 결과가 양성으로 나왔다고 해서 그것이 바로 유방암을 의미하지는 않는다. 이러한 점들이 대중에 혼동을 가져다줄 수 있다.

5.4.2 희귀 클래스 문제

많은 문제에서, 분류해야 할 클래스 간에 불균형이 존재하는 경우가 대부분이다. 즉 한 클래스가 다른 클래스에 비해 경우의 수가 훨씬 많은 경우가 존재한다. 합법적 보험 청구 대 사기성 보험 청구, 또는 웹사이트의 단순 방문객 대 실 구매자의 경우를 예로 들 수 있을 것이다. 보통은 데이터 수가 상대적으로 작은 (희귀한) 클래스 (예를 들면 사기성 보험 청구)가 관심의 대상이 되기 때문에 통상적으로 1로 지정하고, 반대로 수가 많은 경우를 0으로 지정한다. 일반적인 경우, 1이 더 중요한 사건을 의미한다. 예를 들면 사기성 보험 청구를 정확히 잡아내는 것은 몇 천 달러의 돈을 아끼는 결과를 가져다준다. 반대로 사기성이 아닌 보험 청구를 정확하게 파악하는 것은, 사기성으로 의심되는 보험 청구를 일일이 손으로 확인하는 데 드는 비용과 노력만을 절약해줄 뿐이다.

클래스를 쉽게 분리하기 어려운 경우에는, 가장 **정확도**가 높은 분류 모델은 모든 것을 무조건 0으로 분류하는 모델일 수도 있다. 예를 들어, 인터넷 쇼핑물의 방문객 중 0.1%만이 실제 구매를 한다면, '모든 방문객이 구매를 하지 않을 것이다'라고 예측하는 모델의 정확도는 99.9%가 될 것이다. 그러나 이 모델은 결국 있으나 마나다. 대신, 비구매자를 잘못 구분해서 전반적인 정확도가 비록 떨어지더라도, 실제 구매자를 잘 골라내는 모델이 있다면 그 모델을 선호할 것이다.

5.4.3 정밀도, 재현율, 특이도

정확도 외에도 모델의 성능을 표현하기 위해 사용되는 다른 여러 지표들이 있다. 이들 중 몇 가지는 통계에서 오랜 역사를 가지고 있다. 특히 진단 검진의 기대 성능을 많이 다루는 생물통계학 분야에서 역사가 깊다. 먼저 **정밀도**란, 예측된 양성 결과의 정확도를 의미한다.

$$\text{정밀도} = \frac{\text{참 양성}}{\sum \text{참 양성} + \sum \text{거짓 양성}}$$

재현율(recall)은 **민감도**라고 부르기도 하는데, 양성 결과를 예측하는 모델의 능력을 평가한다.

즉 양성 데이터에 대해 정확히 1이라고 예측하는 결과의 비율을 의미한다. **민감도**란 생물통계학과 의료 진단학에서 주로 사용하던 용어이고, **재현율**이란 말은 머신러닝 분야에서 좀 더 많이 사용된다. 재현율의 정의는 다음과 같다.

$$\text{재현율} = \frac{\text{참 양성}}{\sum \text{참 양성} + \sum \text{거짓 음성}}$$

마지막 하나는 **특이도**로서, 이는 음성 결과를 정확히 예측하는 능력을 측정한다.

$$\text{특이도} = \frac{\text{참 음성}}{\sum \text{참 음성} + \sum \text{거짓 양성}}$$

이들을 코드로 표현하면 다음과 같다.

```
# precision
conf_mat[1,1]/sum(conf_mat[,1])
# recall
conf_mat[1,1]/sum(conf_mat[1,])
# specificity
conf_mat[2,2]/sum(conf_mat[2,])
```

5.4.4 ROC곡선

앞에서 다룬 내용에서 눈치챌겠지만 재현율과 특이도 사이에는 트레이드오프 관계(시소 관계)가 있다. 1을 잘 잡아낸다는 것은 그만큼 0을 1로 잘못 예측할 가능성도 그만큼 0을 1로 잘못 예측할 가능성도 높아지는 것을 의미한다. 이상적인 분류기란, 0을 1이라고 잘못 분류하지도 않으면서 동시에 1을 정말 잘 분류하는 것을 의미한다.

이러한 트레이드오프 관계를 표현하기 위한 지표가 바로 '수신자 조작 특성(receiver operation characteristic)' 곡선, 보통은 줄여서 **ROC곡선**이다. ROC곡선은 **x축의 특이도에 대한 y축의 재현율(민감도)**을 표시한다.

역자 각주: ROC 곡선은 제2차 세계대전 중 레이더 신호를 정확히 분류해 적기 출현을 방위 부대에 미리 알리기 위한 목적으로 수신국의 성능을 올리는데 처음 사용되었다.

ROC곡선은 레코드를 분류할 때 사용하는 컷오프 값을 바꿀 때 재현율과 특이도 사이의 트레이드오프 관계를 잘 보여준다. y축에 민감도(재현율)를 표시하면서, x축에는 다음과 같은 두 가지 형태로 표시할 수 있다.

- x축 왼쪽에 1부터 오른쪽에 0까지 특이도를 표시한다.
- y축 아래에 0부터 위쪽에 1까지 재현율을 표시한다.

어느 방법을 사용하든지 곡선의 모양은 동일하다. ROC곡선을 계산하는 과정은 다음과 같다.

1.1로 예측한 확률에 따라 가장 1이 되기 쉬운 것부터 1이 되기 어려운 순으로 레코드를 정렬한다.

2. 정렬된 순서대로 점증적으로 특이도와 재현율을 계산한다.

R에서 ROC곡선을 얻는 방법은 매우 간단하다. 아래 코드는 대출 데이터에 대한 ROC를 계산한다.

```
idx <- order(-pred)
recall <- cumsum(true_y[idx]==1)/sum(true_y==1)
specificity <- (sum(true_y==0) - cumsum(true_y[idx]==0))/sum(true_y==0)
roc_df <- data.frame(recall = recall, specificity = specificity)
ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100)/100), aes(x=x, y=1-x),
            linetype='dotted', color='red') +
  theme_bw()
```

위 그림은 이 코드의 결과를 보여준다. 점선은 랜덤으로 예측했을 때의 결과를 의미한다. 극단적으로 효과적인 분류기(또는 의료 분야에서 극단적으로 효과적으로 진단 검사) ROC 곡선이 왼쪽 상단에 가까운 형태를 보일 것이다. 즉 0을 1로 잘못 예측하는 경우 없이, 1을 정확히 예측할 것이다.

이 모델에서 적어도 50% 정도의 특이도를 원한다면 재현율은 약 75% 정도가 될 것이다.

Note_ 정밀도-재현율 곡선

ROC 곡선과 함께, 정밀도-재현율(PR) 곡선을 사용하기도 한다. ROC 곡선과 마찬가지로 방법으로 PR 곡선을 구할 수 있다. 확률이 낮은 경우에서 높은 경우로 데이터를 정렬한 후에, 차례대로 정밀도와 재현율을 계산한다. PR곡선은 클래스 간 데이터 불균형이 심할 때 특히 유용하다.

5.4.5 AUC

ROC 곡선 자체로는 아주 훌륭한 시각화도구이지만, 분류가 성능을 나타내는 어떤 하나의 값을 주지는 않는다. 하지만 ROC 곡선을 이용해 곡선 아래 면적(AUC(area underneath curve))이라는 지표를 구할 수 있다. AUC는 간단히 말해 ROC 곡선의 아래쪽 면적을 의미한다. AUC값이 높을수록 더 좋은 분류기라고 할 수 있다. AUC가 1이라는 것은 0을 1로 잘못 예측하는 경우 없이, 1을 정확히 분류하는 완벽한 분류기를 의미한다.

최악의 분류기는 ROC 곡선이 가운데를 지나가는 직선인 경우, 즉 AUC가 0.5다. ruddndlek.

다음 그림은 대출 모델에 대한 ROC곡선의 아래쪽 면적을 보여준다.

```
ggplot(roc_df, aes(specificity)) +
  geom_ribbon(aes(ymin=0, ymax=recall), fill='blue', alpha=.3) +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  labs(y='recall') +
  theme_bw()
```

AUC값은 수치적분을 통해 구할 수 있다.

```
> ## AUC calculation
> sum(roc_df$recall[-1] * diff(1-roc_df$specificity))
[1] 0.6926232
> head(roc_df)
      recall specificity
1 4.410921e-05 1.0000000
2 8.821843e-05 1.0000000
3 8.821843e-05 0.9999559
4 1.323276e-04 0.9999559
5 1.764369e-04 0.9999559
6 2.205461e-04 0.9999559
```

모델의 AUC 값은 약 0.69 이다.

CAUTION_ 거짓 양성 비율에 대한 혼동

거짓 양성/비율은 종종 특이도나 민감도와 혼동되어 설명된다.(심지어 출판물과 소프트웨어에서도!!) 때로 거짓 양성 비율은 검사 결과가 양성으로 잘못 나온 음성 데이터의 비율로 정의되기도 한다. 예를 들어 네트워크 침입 탐지 등에서 실제 음성인 신호가 양성으로 탐지된 신호의 비율을 나타내는 데 사용된다.

5.4.6 리프트

분류기 성능을 평가하는 지표로 AUC를 사용하면 단순히 정확도만을 사용하는 것보다는 나은 결과를 얻을 수 있다. 전체적인 정확도도 높이면서 실무에서 중요한 1을 더 정확히 분류해야 하는 트레이드오프를 얼마나 잘 처리하는지 평가할 수 있기 때문이다. 하지만 케이스 문제에서는 모든 레코드를 0으로 분류하지 않도록 하려면 모델의 확률 컷오프를 0.5미만으로 낮춰야 하는 문제가 있다. 이러한 경우 0.4, 0.3 또는 그 이하의 확률도 레코드를 1로 분류하기에 충분할 수 있다. 즉 1의 중요성을 너무 크게 반영하여 1을 과대평가하는 결과를 낳을 수 있다.

컷오프를 변경하면 1을 포착할 가능성이 높아질 수 있다.(0을 1로 잘못 분류하는 경우가 발생하더라도) 하지만 그렇다면 최적의 컷오프란 무엇일까?

리프트 개념을 사용하면 이 질문에 대한 직접적인 답변을 잠시 보류할 수 있다. 대신, 각각 1로 예측될 확률이 있는 레코드들을 정렬한다고 하자. 예를 들면 상위 10% 레코드를 1로 분류하는 알고리즘이, 눈감고 아무거나 선택하는 경우와 비교할 때 얼마나 나은가? 무작위로 선택했을 때, 0.1%의 정확도를

얻었다. 상위 10%에서 0.3%의 결과를 얻었다면, 이 알고리즘은 상위 10%에서 3의 **리프트**(다른 표현으로 **이득**)를 갖는다고 할 수 있다. 이 값은 매 십분위수 마다 혹은 데이터 범위에서 연속적인 값을 따라 얻을 수 있다.

리프트 차트를 계산하려면 먼저 y축에 재현율을 그리고 x축에 총 레코드 수를 나타내는 **누적이득 차트**를 작성해야 한다.

리프트 곡선은 랜덤 선택을 의미하는 대각선에 대한 누적이득의 비율을 말한다.

십분위 이득 차트는 전자 상거래가 등장하기 이전부터 사용된, 예측 모델링에서 매우 오래된 기술 중 하나이다. 특히 광고 메일 전문가들 사이에서 인기가 있었다. 광고 메일을 무차별적으로 발송한다면 이는 매우 값비싼 광고 방법이다. 대신 광고주들은 가장 돈이 될 만한 잠재적인 고객을 선별하기 위한 예측 모델(초기에는 매우 단순한 모델)을 사용했다.

CAUTION_ 업리프트

가끔 리프트와 동일한 의미로 업리프트 라는 표현을 사용한다. 하지만 이 용어는 좀 더 제한적인 상황에서 사용된다. A/B 검정을 수행하고, 처리 A나 B 가운데 하나를 예측변수로 사용하는 예측 모델에서 사용된다.

업리프트는 처리 A와 처리 B 사이의 개별적인 한 케이스에 대해 예측된 결과의 향상을 의미한다.

처음에는 예측변수를 A로 놓고 개별 데이터를 점수화하고 그리고 다시 예측 변수를 B로 바꿔서 한 다음 점수를 보고 결정한다. 영업 담당자나 정치 선거 컨설턴트가 고객이나 유권자를 대상으로 한 두 메시지 중에 어느 것이 더 효과적인지를 결정하는 데 등에 이 방법을 사용한다.

리프트 곡선은 레코드를 1로 분류하기 위한 확률 컷오프 값에 따른 결과의 변화를 한눈에 볼 수 있게 해 준다. 적합한 컷오프 값을 결정하기 위한 중간 단계로 활용할 수 있다. 예를 들면 국세청은 세무감사에 사용할 수 있는 일정량의 자원만 보유하고 있기에, 가장 가능성 있는 세무 사기꾼을 잡기 위해 이 자원들을 사용하기 원한다. 자원 제약을 염두에 두고 당국은 감사를 진행할지 말지 결정하기 위한 기준을 추정하기 위해 리프트 차트를 사용한다.

주요개념

- 정확도(예측한 분류 결과가 몇 퍼센트 정확한지)는 모델을 평가하는 가장 기본적인 단계이다.
- 다른 평가 지표들(재현율, 특이도, 정밀도)은 좀 더 세부적인 성능 특징을 나타낸다.(예를 들면 재현율 모델이 1을 얼마나 정확히 분류하는지를 나타낸다).
- AUC(ROC 곡선 아래 면적)는 모델의 1과 0을 구분하는 능력을 보여주기 위해 가장 보편적으로 사용되는 지표이다.
- 이와 비슷하게 리프트는 모델이 1을 얼마나 효과적으로 분류해내는지를 측정한다. 가장 1로 분류될 가능성이 높은 거부터 매 십분위마다 이를 계산한다.

5.5 불균형 데이터 다루기

앞 절에서는 분류 모델을 평가할 때 단순 정확도 외에 사용할 다른 성능 지표들에 알아보았다. 이들이 데이터가 매우 드문 불균형 데이터 (온라인 구매, 보험 청구 사기 등)에 적합하다는 사실도 알아보았다.

이번 절에서는 불균형 데이터에서 예측 모델링 성능을 향상시킬 몇 가지 방법에 대해 알아본다.

용어정리

- 과소표본 (undersample): 분류모델에서 개수가 많은 클래스 데이터 중 일부 소수만을 사용하는 것(유의어 : 다운 샘플)
- 과잉표본 (oversample): 분류 모델에서 희귀 클래스 데이터를 중복해서, 필요하면 부트스트랩 해서 사용하는 것(유의어 : 업샘플)
- 상향 가중치(up weight) 혹은 하향 가중치 (down weight) : 모델에서 희귀(혹은 다수) 클래스에 높은 (혹은 낮은) 가중치를 주는 것
- 데이터 생성 (data generation) : 부트스트랩과 비슷하게 다시 샘플링한 레코드를 빼고 원래 원본과 살짝 다르게 데이터르 생성하는 것
- z-점수(z-score) : 표준화 결과
- K : 최근접 이웃 알고리즘에서 이웃들의 개수

5.5.1 과소표본추출

앞서 다룬 대출 데이터와 같이 데이터 개수가 충분하다면, 다수의 데이터에 해당하는 클래스에서 과소표본추출(다운샘플링)을 해서 모델링할 때 0과 1의 데이터 개수 간의 균형을 맞출 수 있다.

과소표본추출의 기본 아이디어는 다수의 클래스에 속한 데이터들 중에 중복된 레코드들이 많을 것이라는 사실에서 출발한다. 작지만 더 균형잡힌 데이터는 모델 성능에 좋은 영향을 주게 되고, 데이터를 준비하는 과정이나 모델을 검증하는 과정이 좀 더 수월하다.

어느 정도의 데이터를 충분하다고 할 수 있을까? 이는 응용 분야에 따라 달라진다. 하지만 일반적으로 소수 클래스의 데이터가 수만 개 정도 있다면 충분하다고 할 수 있다. 물론 1과 0을 분리하기가 쉽다면, 더 적은 데이터로 충분할 수도 있다.

5.3절에서 분석한 대출 데이터는 균형 잡힌 학습 데이터를 통해 얻은 것이다. 데이터의 절반은 대출을 모두 갚은 경우이고, 나머지 절반은 갚지 않은 경우이다. 예측값도 비슷하게 나왔다. 절반은 확률이 0.5보다 낮았고, 나머지 절반은 0.5보다 컸다. 하지만 전체 대출 데이터에서는 18% 정도만이 연체 상태였다.

```
## Code for undersampling
mean(full_train_set$outcome=='default')
[1] 0.1889455
```

모델을 학습하는 데 전체 데이터를 사용한다면 어떻게 될까?

```
> full_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
+                   home_ + emp_len_ + dti + revol_bal + revol_util,
+                   data=full_train_set, family='binomial')
> pred <- predict(full_model)
> mean(pred > 0)
[1] 0.003942094
```

대출의 약 0.39% 정도만이 연체 상태일 것이라고 예측하므로 기대되는 값보다 1/12 작은 수준이다. 모든 데이터를 동일하게 학습에 사용하다 보니, 대출을 갚는다는 예측이 대출을 갚지 않는다는 예측을 압도하는 결과를 보이는 것이다. 얼핏 생각해봐도, 빚을 갚지 않는 사람보다는 빚을 갚는 사람의 정보가 훨씬 많기에 빚을 갚지 않고 있는 데이터에 대해서도 이와 유사한 빚을 갚는 사람의 정보를 찾을 가능성이 높아진다. 균형잡힌 데이터를 사용했을 때는 거의 50% 정도 빚을 갚지 않는다고 예측했었다.

5.5.2 과잉표본추출과 상향/하향 가중치

과소표본 방식의 약점으로 지적받는 부분은 일부 데이터가 버려지기 때문에 모든 정보를 활용하지 못한다는 점이다. 상대적으로 작은 데이터 집합에서, 희귀 클래스 경우의 레코드가 몇백 혹은 몇천 개라면, 다수 클래스에 대한 과소표본추출은 정말 유용한 정보까지 버리게 되는 결과를 초래할 수 있다. 이런 경우, 다수 클래스를 과소표본추출하는 대신, 복원추출방식(부트스트래핑)으로 희귀 클래스의 데이터를 희귀 클래스의 데이터를 **과잉표본추출(업 샘플링)** 해야한다.

데이터에 가중치를 적용하는 방식으로 이와 비슷한 효과를 얻을 수 있다. 많은 분류 알고리즘에서 상향/하향 가중치를 데이터에 적용하기 위해 weight 라는 인수를 지원한다. 예를 들면 glm 함수에서 weight 라는 인수를 사용해서 대출 데이터에 가중치 벡터를 적용해보자.

```
## Code for oversampling/up weighting
wt <- ifelse(full_train_set$outcome=='default', 1/mean(full_train_set$outcome ==
'default'), 1)
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
                  home_ + emp_len_ + dti + revol_bal + revol_util,
                  data=full_train_set, weight=wt, family='quasibinomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.5767208
```

연체에 대한 가중치를 1/p로 두었다. 여기서 p는 연체의 확률값이다. 그리고 대출 상환에 대한 가중치는 1로 두었다. 연체와 상환의 가중치 합은 거의 동일하다. 이렇게 하면 예측값의 평균은 과소추정 때 구한 0.39%가 아닌 43%로 쫓나 뛰었다.

가중치를 적용하는 방식이 희귀 클래스를 업샘플링하거나 다수 클래스를 다운 샘플링하는 방법을 대체할 수 있다.

NOTE_ 손실 함수

많은 분류 혹은 회귀 알고리즘은, 어떤 기준 혹은 손실 함수(loss function)를 최적화한다고 볼 수 있다. 예를 들면 로지스틱 회귀는 편차를 최소화하려고 한다. 어떤 자료에서는 회귀 클래스 때문에 생길 수 있는 문제를 피하기 위해 손실 함수를 수정하는 방법을 제안하기도 한다. 실제로 이 방법을 적용하는 것은 어렵다. 분류 알고리즘의 손실 함수를 직접적으로 수정하는 것은 복잡하고 어렵다. 반면에 가중치를 사용하는 방법은 가중치가 높은 데이터를 선호하고 가중치가 낮은 데이터의 줄여주는 식으로 손실 함수를 변경하는 쉬운 방법이다.

5.5.3 데이터 생성

부트스트랩을 통한 업샘플링 방식의 변형으로 기존에 존재하는 데이터를 살짝 바꿔 새로운 레코드를 만드는 **데이터 생성**방법이 있다. 이 방법에는 데이터의 개수가 제한적이기 때문에 알고리즘을 통해 분류 '규칙'을 생기에는 정보가 충분하지 않다는 직관이 바탕에 깔려 있다. 비슷하지만 기존의 데이터와는 다른 데이터를 생성해서 좀 더 로버스트한 분류 규칙을 담을 수 있는 기회를 주고자 하는 것이다. 이는 통계에서 부스팅이나 배깅 같은 앙상블 모델에 담겨 있는 개념과 매우 비슷하다.

합성 소수 과잉표본 기법(Synthetic Minority Oversampling Technique)의 약자인 SMOTE 알고리즘은 발표와 동시에 주목을 받았다. SMOTE 알고리즘은 업샘플링된 레코드와 비슷한 레코드를 찾고, 원래 레코드와 이웃 레코드의 랜덤 가중평균으로 새로운 합성 레코드를 만든다. 여기에 대해 각각의 예측변수에 대해 개별적 가중치를 생성한다. 새로 합성된 업샘플 레코드의 개수는 데이터의 균형을 맞추기 위해 필요한 업샘플링 비율에 따라 달라진다.

R에서 SMOTE를 구현한 몇 가지 패키지들이 있다. 이 가운데 불균형 데이터를 처리할 수 있는 가장 종합적인 패키지는 unbalanced이다. 'Racing' 알고리즘을 포함하여 다양한 기법들을 제공한다.

그러나 SMOTE 알고리즘은 무척 간단하기에 knn 패키지를 사용하여 R로 직접 구현할 수도 있다.

5.5.4 비용 기반 분류

실무적으로, 분류 규칙을 결정할 때 정확도나 AUC만으로는 충분하지 않을 수 있다. 종종 추정 비용은 거짓 양성 대 거짓 음성으로 결정될 수 있고, 최상의 컷 오프를 결정하려면 이러한 비용들을 종합적으로 고려할 필요가 있다. 예를 들면 신규 대출에서 연체로 인해 예상되는 비용이 C라고 하고 대출 상환을 통해 얻을 수 있는 수익을 R이라고 하자. 이때 신규 대출의 기대 수익은 다음과 같다.

$$\text{기대 수익} = P(Y = 0) \times R + P(Y = 1) \times C$$

여기서 대출 결과를 단순히 연체나 상환, 둘 중 하나로 결정하는 대신에, 대출을 통해 얻을 수 있는 기대 수익이 있는지 없는지를 결정하는 것이 더 말이 된다. 대출을 갚지 않을 확률을 예측하는 것은 중간 단계이고, 결국은 사업의 목적인 기대 수익을 결정하기 위해 대출의 전체적인 가치를 얻어내야 한다. 예를 들면 가치가 적은 대출보다는 연체 확률이 더 높더라도 가치가 더 큰 대출을 선호하는 편이 나올 수도 있는 것이다.

5.5.5 예측 결과 분석

AUC와 같은 단일 성능 지표로는 어떤 상황에서 모델의 적합성을 여러 가지 측면으로 보기 어려울 수 있다. 다음 그림은 대출 데이터의 두 가지 예측변수 borrow_score와 payment_inc_ratio를 사용해서 구한 4개의 서로 다른 모델에 대한 결정 규칙을 보여준다. 선형판별분석(LDA), 로지스틱 회귀분석, GAM을 이용한 로지스틱 회귀, 트리 모델 4가지를 사용했다. 선들의 왼쪽 상단 영역은 연체 예측에 해당한다. LDA와 로지스틱 선형회귀는 거의 비슷한 결과를 보인다. 트리 모델이 가장 이상한 결과를 보인다. 사실 대출 신청자의 신용점수가 올라가면 예측이 대출 상환에서 연체 쪽으로 움직이는 상황이 발생하긴 한다! 최

종적으로 GAM을 이용한 로지스틱 회귀모형이 트리 모델과 다른 선형모형들을 서로 타협하는 결과를 보여준다.

```
# Code for Figure 5-8: comparison of methods
loan_tree <- rpart(outcome ~ borrower_score + payment_inc_ratio,
                  data=loan3000,
                  control = rpart.control(cp=.005))

lda_pred <- lda_df0[, c('borrower_score', 'payment_inc_ratio')]
lda_pred$method = 'LDA'

tree_pred <- data.frame(borrower_score = c(0.375, 0.375, 0.525, 0.525, 0.625,
0.625),
                      payment_inc_ratio = c(0, 9.732, 9.732, 8.772, 8.772,
20),
                      method = rep('Tree', 6))

glm0 <- glm(outcome ~ (payment_inc_ratio) + (borrower_score),
            data=loan3000, family='binomial')
y <- seq(from=0, to=20, length=100)
x <- (-glm0$coefficients[1] - glm0$coefficients[2]*y)/glm0$coefficients[3]
glm0_pred <- data.frame(borrower_score=x, payment_inc_ratio=y,
method='Logistic')

gam1 <- gam(outcome ~ s(payment_inc_ratio) + s(borrower_score),
            data=loan3000, family='binomial')
# newdata = gam0_pred

gam_fun <- function(x){
  rss <- sum(predict(gam1, newdata=data.frame(borrower_score=x,
payment_inc_ratio=y))^2)
}
est_x <- nlminb(newdata$borrower_score, gam_fun )
gam1_pred <- data.frame(borrower_score=est_x$par, payment_inc_ratio=y,
method="GAM")

loan_fits <- rbind(lda_pred,
                  tree_pred,
                  glm0_pred,
                  gam1_pred)

## Code for Figure 5-8
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0508.png'), width = 6,
height=4, units='in', res=300)
ggplot(data=loan_fits, aes(x=borrower_score, y=payment_inc_ratio, color=method,
linetype=method)) +
  geom_line(size=1.5) +
  theme(legend.key.width = unit(2,"cm")) +
  guides(linetype = guide_legend(override.aes = list(size = 1)))
dev.off()
```

차원이 높아지고 GAM 이나 트리모델을 사용하는 경우, 예측 규칙을 시각화하기가 쉽지 않다.

다만 어떤 경우에도 예측값에 대한 탐색 분석은 할 가치가 있는 일이다.

주요 개념

- 데이터의 심각한 불균형(즉, 관심 있는 결과의 데이터가 희박할 때)은 분류 알고리즘에서 문제가 될 수 있다.
- 한 방법은 다수의 데이터를 다운샘플링하거나 희귀한 데이터를 업샘플링해서 학습 데이터의 균형을 맞추는 것이다.
- 갖고 있는 1의 데이터를 모두 사용해도 그 개수가 너무 적을 때는, 희귀한 데이터에 대해 부트스트랩 방식을 사용하거나 기존의 데이터와 유사한 합성 데이터를 만들기 위해 SMOTE를 사용한다.
- 데이터의 불균형이 존재할 경우 보통은 어느 한쪽(1의 클래스)을 정확히 분류하는 것에 더 높은 점수를 주게 되어 있고, 이러한 가치 비율이 평가지표에 반영되어야 한다.

5.6 마치며

분류란 어떤 레코드가 두 가지(또는 소수의) 범주 중 어디에 속할지를 예측하는 프로세스로, 예측 분석을 위한 기본적인 도구이다. 대출을 갚지 못할 것인가(예, 아니오)? 사전 지불 방식을 사용할 것인가? 웹 방문자가 링크를 클릭할 것인가? 소비자가 상품을 구매할 것인가? 해당 보험 청구가 사기인가? 보통, 이러한 분류 문제에서 한 클래스가 주요 관심 사항(예를 들면 사기성 보험 청구)이며, 이진 분류에서 이 클래스를 1로 지정하고 다른 클래스는 0으로 지정한다. 이 프로세스의 핵심은 관심 있는 클래스에 속할 확률인 **경향 점수**를 추정하는 것이다. 일반적으로 겪게 되는 문제는, 관심 있는 클래스가 상대적으로 드문 상황이다. 예 단순 정확도 외에 다양한 모델 평가 지표에 대한 논의를 이번 장 후반에 다뤘다. 이런 상황에서는 모든 레코드에 대해 무조건 0이라고 예측하는 것이 높은 정확도를 얻게 되므로, 이러한 지표들이 매우 중요하다는 사실을 반드시 기억하자,