10. 로지스틱스회귀분석

1. 로지스틱회귀분석의 이해

○ 로지스틱회귀모형

- 로지스틱 회귀분석은 출력변수가 범주형인 경우에 적용할 수 있는 방법임.
- 평의상 $y=0$ 또는 1의 값을 갖는다고 할 때, 단순선형회귀모형을 적용하는 경우 모형식의 결과가 범위 [0, 1]을 벗어날 수 있고, 오차항의 분포가 정규분포가 아니라는 문제점이 있음.
- 이에 대한 대안으로 연속이고 증가함수이며 [0, 1] 사이에서 값을 갖는 함수 $p(\boldsymbol{X})$에 대하여

$$P(Y=0|x)=p(\boldsymbol{X}\beta)$$

로 모형화 할 수 있음.

- 즉, $\boldsymbol{X}$가 주어졌을 때, Y의 조건부 평균이 아니라 조건부 확률을 적절한 연결함수(link function) p를 통해 모형화 하는 것이 로지스틱회귀모형의 아이디어임.
- 연결함수의 형태에 따라 로지스틱 모형, 검벨모형, 프로빗 모형 등이 있으나 계산상의 편리성으로 인하여 로지스틱회귀모형이 널리 사용됨.

- 로지스틱회귀모형은

$$P(Y=1|\boldsymbol{X})=\frac{\exp(\boldsymbol{X}\beta)}{1+\exp(\boldsymbol{X}\beta)}$$

이며, j번째 입력변수의 값이 1 증가할 때, 출력변수의 값이 0이 될 확률에 비해 1이 될 확률의 비의 증가율을 오즈비라고 하고, $\exp(\beta_j)$로 계산할 수 있음.

- 예를 들어, j번째 입력변수가 소득을 나타내고 출력변수가 구입여부를 나타낸다고 할 때, 회귀계수의 값이 3.72라고 한다면, 오즈비는 $\exp(3.72)=42$로 소득이 한 단위 증가하면 물품을 구매하지 않을 확률에 대한 구매할 확률이 42배 증가함을 알 수 있음.

- 로지스틱회귀모형에 대한 회귀계수를 추정하기 위해서는 최대우도추정법을 사용함.
- 하지만 선형회귀분석과 같이 정확한 회귀계수 추정량을 구할 수 없어 뉴튼 랩슨 방법과 같은 수치적 방법을 이용하여 회귀계수를 추정함.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

- 그리고 회귀계수의 유의성 ($H_0 : \beta_j = 0$) 검정은 우도비 검정통계량으로 계산함. 이 때 우도비 검정통계량은 근사적으로 자유도가 1인 카이제곱분포를 따르고, 그 값이 크면 회귀계수 값이 0이 아니라고 결론을 내립니다.

○로지스틱 모형을 이용한 분류

- 로지스틱회귀모형은 주어진 입력변수 $\boldsymbol{X}$에 대하여 출력변수 Y가 1이 될 확률 $P(Y=1|\boldsymbol{X})$를 추정하는데, 0과 1사이의 적당한 수 c를 절단값으로 선택하여 $P(Y=1|\boldsymbol{X})$가 c보다 크면 자료를 $Y=1$인 클래스로 분류하고, $P(Y=1|\boldsymbol{X})$가 c보다 작으면 자료를 $Y=0$인 클래스로 분류할 수 있음.

○로지스틱 모형의 예측력을 평가하는 지표

- Confusion Matrix
- 모형이 예측하는 값에는 True/False 두 가지가 있고, 각 예측값은 실제로 True이거나 False 일 수 있음. 이를 그림과 같이 모형의 예측값과 실제값을 각각 축으로 하는 분할표를 Confusion Matrix라고 함.

		실제값	
		True	False
예측값	True	TP (True Positive)	FP (False Positive)
	False	FN (False Negative)	TN (True Negative)

- 정확도(Precision)
- True로 예측한 분석대상 중에서 실제 값이 True인 비율을 말하며, 모형의 정확성을 나타내는 지표가 됨. 정확도가 높다는 것은 실제 False를 True로 잘못 예측하는 False Positive 오류가 작다는 뜻임.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

- 재현율(Recall)
- 실제 값이 True인 분석대상 중에서 True로 예측하여 모형이 적중한 비율을 말하며, 모형의 완전성을 나타내는 지표임. 재현율이 높다는 것은

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

실제 True를 False로 잘못 예측하는 False Negative 오류가 낮다는 뜻임.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F1 지표(F1-score)
- 정확도와 재현율이 균등하게 반영될 수 있도록 정확도와 재현율의 조화평균을 계산한 값으로, 모형의 예측력을 종합적으로 평가하는 지표임. 값이 높을수록 분류모형의 예측력이 좋다고 말할 수 있음.

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2. sklearn을 활용한 로지스틱회귀분석

- 로지스틱회귀분석을 실시하기 위해 먼저 sklearn 모듈을 활용한 로지스틱회귀분석 방법에 대해서 살펴보도록 하겠음.

- 앞서 살펴본 선형회귀분석과 마찬가지로 로지스틱회귀분석에서도 주어진 데이터를 100% 활용하여 선형회귀분석을 실시한 후 새롭게 데이터를 모형에 적용하였을 때 예측 성능이 떨어지는 경우가 많음.
- 이러한 overfitting 현상을 해소하기 위하여 교차검증 방법을 사용함.
- 로지스틱회귀분석을 수행하고자 할 때, 주어진 데이터를 Train 데이터와 Test 데이터로 분할은 sklearn.model_selection 모듈의 train_test_split() 함수를 이용하여 실시할 수 있음.

```
from sklearn.model_selection import train_test_split
train 객체명, test 객체명 = train_test_split(데이터셋, test_size=비율,
random_state=번호, stratify=출력변수 )
```

- train_test_split() 함수를 이용하는 방법은 선형회귀분석의 경우와 비슷하지만, 로지스틱회귀분석의 경우 또 다른 옵션인 stratify 옵션을 선택할 수 있음.
- stratify 옵션은 층화추출을 수행할 수 있게 해주는 옵션으로, stratify 옵션에 출력변수명을 입력하게 되면 출력변수에 따라 층화추출을 수행하게 됨.
- 로지스틱회귀분석의 경우 출력변수가 범주형 변수 형태이기 때문에

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

stratify 옵션을 이용하여 층화추출을 통한 Train 데이터와 Test 데이터로의 분리를 수행하는 것이 좋음.

- 그럼 분석에 사용되는 Train 데이터를 활용하여 로지스틱회귀분석을 실시하는 방법을 알아보도록 하겠음.
- 분석에 사용되는 Train 데이터를 활용하여 로지스틱회귀분석 실시는 sklearn.linear_model 모듈의 LogisticRegression() 함수를 이용하여 실시할 수 있음.

```
from sklearn.linear_model import LogisticRegression
LogisticRegression(penalty='none').fit(독립변수, 종속변수)
```

- LogisticRegression().fit() 함수에 분석에 사용되는 Train 데이터의 입력변수와 출력변수를 입력함. 그리고 penalty 옵션은 로지스틱회귀분석을 수행함에 있어 회귀계수 추정시 패널티를 부여 여부를 지정하는 옵션임. 일반적인 통계프로그램에서 출력되는 로지스틱회귀분석의 결과는 패널티를 부여하지 않는 방식이므로 다른 통계프로그램과 같은 결과를 얻고자 할때는 penalty 옵션에 'none'을 입력함. 이번시간에는 패널티를 부여하지 않는 경우에 대해서만 살펴보도록 하겠음.
- 위 명령어를 사용하면 보통의 분석과는 다르게 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미로 LogisticRegression()이라는 메시지만 출력됨.
- 이는 R과 같은 다른 프로그램들과 마찬가지로 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.

- 따라서 LogisticRegression().fit() 함수를 실행시킨 결과를 객체에 할당하여 다음과 같은 함수를 활용하여 로지스틱회귀모형의 절편항, 기울기항, 정분류률 및 모형에 피팅된 출력변수의 예측값을 확인할 수 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

정분류률: 회귀모형 객체**.score**(입력변수, 출력변수)
회귀절편: 회귀모형 객체**.intercept_**
기울기: 회귀모형 객체**.coef_**
출력변수 예측값: 회귀모형 객체**.predict**(독립변수)
출력변수 확률값: 회귀모형 객체**.predict_proba**(독립변수)

- 로지스틱회귀모형의 분류정확도를 확인하기 위한 정분류률은 로지스틱 회귀모형의 결과를 할당한 객체를 활용하여 score() 함수를 사용하여 확인할 수 있음. 이 때 score() 함수에 정분류률 계산을 위한 입력변수와 출력변수를 차례로 입력하면 모형의 정분류률을 확인할 수 있음. 만약 Test 데이터의 입력변수와 출력변수를 입력하면 생성된 로지스틱 회귀모형에 Test 데이터를 적용하였을 때의 정분류률을 확인하여 모형을 평가할 수 있음.
- 다음으로 로지스틱회귀모형의 절편항과 기울기항을 확인하기 위해서는 로지스틱회귀모형의 결과를 할당한 객체를 활용하여 intercept_ 함수와 coef_ 함수를 사용하여 확인할 수 있음.
- 그리고 로지스틱회귀모형에 의해 예측된 출력변수의 값을 확인하기 위해서는 로지스틱회귀모형의 결과를 할당한 객체를 활용하여 predict() 함수를 사용하여 확인할 수 있음. 이 때 predict() 함수에 출력변수의 예측값을 계산하기 위한 입력변수를 입력하면 출력변수의 예측값을 확인할 수 있음. 만약 Test 데이터의 입력변수를 입력하면 생성된 로지스틱회귀모형에 Test 데이터를 적용하였을 때의 출력변수의 예측값을 확인할 수 있음.
- 또한, 로지스틱회귀모형에 의해 예측된 출력변수의 값이 1이 될 확률을 확인하기 위해서는 로지스틱회귀모형의 결과를 할당한 객체를 활용하여 predict_proba() 함수를 사용하여 확인할 수 있음.

- 그 외 모형 평가에 사용되는 분류표, ROC 곡선을 그리기 위해 필요한 거짓양성비율(False Positive Rate; fpr), 실제양성비율 (True Positive Rate; tpr), AUC값을 계산하기 위해서 sklearn.metrics 모듈의 confusion_matrix() 함수, accuracy_score() 함수, classification_report() 함수, roc_curve() 함수, auc() 함수를 이용하여 실시할 수 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

from **sklearn.metrics** import **confusion_matrix, accuracy_score, classification_report, roc_curve, auc**

분류표: **confusion_matrix**(출력변수 예측값, 출력변수)

정분류율: **accuracy_score**(출력변수, 출력변수 예측값)

분류 예측력 평가 지표: **classification_report**(출력변수, 출력변수 예측값)

fpr, tpr: **roc_curve**(출력변수, 출력변수 예측 확률)

AUC: **auc**(fpr, tpr)

- 로지스틱회귀모형을 평가하기 위해 실제 출력변수의 값과 예측값을 이원분할표의 형태로 나타내기 위해 confusion_matrix() 함수를 사용할 수 있음. 이 때, 이원분할표의 행 위치와 열 위치를 차례로 입력하여 분류표의 행과 열에 위치할 값을 지정할 수 있음.
- confusion_matrix() 함수에 의해 나타난 분류표 상의 전체 정분류율를 계산하기 위해 accuracy_score() 함수를 사용할 수 있음.이 때, 출력변수와 출력변수 예측값을 차례로 입력함.
- 로지스틱회귀모형을 평가하기 위한 지표인 presicion, recall, f1_score를 계산하기 위해 classification_report() 함수를 사용할 수 있음. 이 때, 출력변수와 출력변수 예측값을 차례로 입력함.
- 그리고 roc_curve() 함수를 통해 roc curve를 그리기 위한 fpr과 tpr을 계산할 수 있음. roc_curve() 함수에는 차례로 출력변수의 예측값과 출력변수가 1일 확률을 입력하는데요, 이 때 fpr, tpr 값이 차례로 출력되므로 roc_curve() 함수를 수행하였을 때 fpr과 tpr값이 저장될 객체를 지정하는 것이 좋습니다.
- 마지막으로 auc() 함수를 통해 roc curve의 아랫 면적인 AUC 값을 계산할 수 있음. auc() 함수에는 roc_curve() 함수를 통해 계산된 fpr과 tpr을 차례로 입력하여 auc 값을 계산할 수 있음.

3. statsmodels를 활용한 로지스틱회귀분석

- 앞서 살펴보았던 sklearn.linear_model 모듈을 이용하여 실시한 로지스틱회귀분석은 모형의 회귀계수 및 모형의 평가 지표등은 계산할 수 있지만, 입력변수들의 유의성 검정을 위한 p-value 등은 확인할 수 없는 단점이 존재함.
- 따라서 모형의 회귀계수 및 모형의 평가지표 뿐만 아니라 독립변수들의

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

유의성 검정을 위한 p-value를 확인하기 위해 또 다른 로지스틱회귀분석을 위한 statsmodels 모듈을 알아볼 필요가 있음.

- statsmodels 모듈에서 로지스틱회귀분석을 실시하기 위한 함수로는 OLS()와 ols() 함수가 존재함.

○ Logit를 이용한 로지스틱회귀분석

- 그럼 먼저 Logit() 함수를 활용하여 로지스틱회귀분석을 실시하는 방법을 알아보도록 하겠음.

- Logit() 함수의 경우 앞서 살펴본 LogisticRegression() 함수와는 다르게 입력변수를 지정하면 자동으로 절편항을 입력하지 않음. 즉, 모형식에서 절편항이 제거된 상태로 분석이 이루어지기 때문에 절편항을 분석결과가 나타나게 하기 위해 선형회귀분석에서 실시하였던 것 처럼 statsmodels.api 모듈의 add_constant() 함수를 이용함.

- 그럼 절편항이 추가된 입력변수와 출력변수를 활용하여 로지스틱회귀분석을 실시하는 statsmodels.api 모듈의 Logit() 함수는 다음과 같이 이용할 수 있음.

```
from statsmodels.api import OLS
객체명 = OLS(출력변수, 입력변수).fit()
모형 결과: 객체명.summary()
회귀계수: 객체명.params
출력변수 확률값: 객체명.predict(입력변수)
```

- Logit().fit() 함수에 분석에 사용되는 Train 데이터의 출력변수와 입력변수를 입력함. 이 때, LogisticRegression() 함수와는 출력변수와 입력변수 입력 순서가 다르니 주의하기 바람.
- 위 명령어를 사용하면 LogisticRegression() 함수와 마찬가지로 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미의 메시지만 출력됨.
- 따라서 LogisticRegression() 함수와 마찬가지로 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수에 활용하여 결과를 확인할 수 있음.
- 로지스틱회귀모형의 각종 결과는 로지스틱회귀모형의 결과를 할당한 객체를 활용하여 summary() 함수를 사용하여 확인할 수 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

- 오즈비 계산을 위해 회귀계수값만 따로 출력하고 싶다면 Logit() 함수를 실행시킨 객체에 params 함수를 수행하여 확인할 수 있음.
- 그리고 LogisticRegression() 함수에서 사용되었던 predict() 함수가 ols() 에서는 출력변수의 예측값을 반환하는 것이 아니라 출력변수가 1이 될 확률값을 반환해 줌.
- 따라서 출력변수의 예측값을 확인하기 위해서는 predict() 함수를 통해 출력변수가 1이 될 확률값을 확인하고, 조건식을 통해 출력변수 예측값을 계산하여야 함.

○ logit을 이용한 회귀분석

- 앞서 설명한 LogisticRegression() 함수와 Logit() 함수는 모두 출력변수와 입력변수 데이터를 직접 입력하는 방식이었음.
- 출력변수와 입력변수를 직접 입력하는 방식이 아닌 변수이름을 활용하여 모형식을 입력하여 로지스틱회귀분석을 실시하는 statsmodels.formula.api 모듈의 logit() 함수는 다음과 같이 이용할 수 있음.

```
from statsmodels.formula.api import ols
객체명 = ols(모형식, data=데이터셋).fit()
모형 결과: 객체명.summary()
회귀계수: 객체명.params
출력변수 확률값: 객체명.predict(입력변수)
```

- logit().fit() 함수에 분석에 사용되는 모형식을 따옴표 안에 먼저 입력함. 이 때, 모형식 입력방법은 출력변수와 입력변수는 ~ 기호를 이용하여 구분함. 그리고 입력변수들은 + 기호를 이용하여 구분함. 즉, '출력변수 ~ 입력변수1 + 입력변수2 + ... ' 형태로 입력함.
- 그리고 data 옵션에 분석에 사용되는 데이터셋을 입력함. 이 때, 앞서 모형식에 사용되었던 변수명이 분석에 사용되는 데이터셋에 반드시 포함되어 있어야 함.
- logit() 함수도 앞선 명령어들과 마찬가지로 아무런 결과가 출력되지 않고, 분석이 실시되었다는 의미의 메시지만 출력됨.
- 따라서 분석이 수행된 결과를 다른 객체에 할당한 후 이를 다양한 함수

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

에 활용하여 결과를 확인할 수 있음.

- 로지스틱회귀모형의 각종 결과는 로지스틱회귀모형의 결과를 할당한 객체를 활용하여 summary() 함수를 사용하여 확인할 수 있음.
- 오즈비 계산을 위해 회귀계수값만 따로 출력하고 싶다면 logit() 함수를 실행시킨 객체에 params 함수를 수행하여 확인할 수 있음.
- 그리고 LogisticRegression() 함수에서 사용되었던 predict() 함수가 ols() 에서는 출력변수의 예측값을 반환하는 것이 아니라 출력변수가 1이 될 확률값을 반환해 줌.
- 따라서 출력변수의 예측값을 확인하기 위해서는 predict() 함수를 통해 출력변수가 1이 될 확률값을 확인하고, 조건식을 통해 출력변수 예측값을 계산하여야 함.

- seaborn 모듈의 titanic 데이터에서, alive를 예측하기 위해 age, fare, who를 사용한 로지스틱회귀분석을 실시하고자 함.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('titanic')

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

- info() 함수를 활용하여 자료를 확인한 결과 age에서 많은 결측값이 존재하여 dropna() 함수를 이용하여 결측값을 제거할 필요가 있음. 또한, 출력변수인 alive와 입력변수 중 who는 문자형 자료로 되어 있으므로 이를 확인할 필요가 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

```
In [4]: df = df.dropna(subset = ['alive', 'age', 'fare', 'who'], how='any', axis=0)

In [5]: df['who'].value_counts()
Out[5]:
man      413
woman    218
child     83
Name: who, dtype: int64

In [6]: df['alive'].value_counts()
Out[6]:
no     424
yes    290
Name: alive, dtype: int64
```

- 입력변수 중 who는 구성요소가 man, woman, child로 되어 있으므로 child를 기준으로 하는 더미변수로 변형시킬 필요가 있음. 또한, 출력변수 인 alive는 no와 yes로 구성되어 있어 이를 각각 0과 1로 변경할 필요가 있음.

```
In [7]: from sklearn.preprocessing import LabelEncoder

In [8]: df['alive_1']=LabelEncoder().fit_transform(df['alive'].values)
```

- 먼저 출력변수의 값을 0과 1로 변형시키기 위해 sklearn.preprocessing 모듈의 LabelEncoder() 함수를 사용하였음. LabelEncoder() 함수에 fit_transform() 함수를 추가로 적용하여 values 함수를 통한 alive 변수의 값을 0과 1로 변형하여 데이터프레임의 마지막에 새롭게 alive_1이라는 변수명으로 저장하였음.

```
In [9]: import pandas as pd

In [10]: df1=pd.get_dummies(df['who'],drop_first=True)

In [11]: print(df1)
     man  woman
0      1      0
1      0      1
2      0      1
3      0      1
4      1      0
..   ...    ...
885    0      1
886    1      0
887    0      1
889    1      0
890    1      0

[714 rows x 2 columns]
```

```
In [12]: df = pd.concat([df, df1],axis=1)

In [13]: print(df)
     survived  pclass     sex   age  sibsp  ...  alive  alone alive_1 man woman
0           0       3    male  22.0      1  ...     no  False       0   1     0
1           1       1  female  38.0      1  ...    yes  False       1   0     1
2           1       3  female  26.0      0  ...    yes   True       1   0     1
3           1       1  female  35.0      1  ...    yes  False       1   0     1
4           0       3    male  35.0      0  ...     no   True       0   1     0
..        ...     ...     ...   ...    ...  ...    ...    ...     ...  ..   ...
885         0       3  female  39.0      0  ...     no  False       0   0     1
886         0       2    male  27.0      0  ...     no   True       0   1     0
887         1       1  female  19.0      0  ...    yes   True       1   0     1
889         1       1    male  26.0      0  ...    yes   True       1   1     0
890         0       3    male  32.0      0  ...     no   True       0   1     0

[714 rows x 18 columns]
```

- 그리고 입력변수 중 하나인 who 변수를 더미변수로 만들기 위해 pandas 모듈의 get_dummies() 함수를 사용하였음. 그 결과 새롭게 man과 woman 변수가 생성되었고 이를 pandas 모듈의 concat() 함수를 이용하여 데이터프레임에 더미변수를 추가하였음.

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

터를 구분하기 위해 train_test_split() 함수를 사용하였음.

```
In [14]: from sklearn.model_selection import train_test_split

In [15]: train, test = train_test_split(df, test_size=0.3, random_state=1, stratify=df['alive_1'])

In [16]: y_train=train['alive_1']
    ...: X_train=train[['age','fare', 'man', 'woman']]
    ...: y_test=test['alive_1']
    ...: X_test=test[['age','fare', 'man', 'woman']]
```

- tran_test_split() 함수의 stratify 옵션에 출력변수인 alive_1을 입력하여 출력변수에 따른 층화추출 방법으로 Train 데이터와 Test 데이터를 추출하였음. 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 독립변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 종속변수를 선택하여 각각 할당하였음.

- Train 데이터의 독립변수 자료인 X_train과 Train 데이터의 종속변수 자료인 y_train을 각각 LogisticRegression().fit() 함수에 대입하여 로지스틱 회귀분석을 실시하였음.

```
In [17]: from sklearn.linear_model import LogisticRegression

In [18]: lr = LogisticRegression(penalty='none').fit(X_train, y_train)
```

- LogisticRegression().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 lr이라는 객체에 할당하였고, 이를 활용하여 회귀계수, 오즈비, 정분류율를 각각 계산하였음.

```
In [19]: print('회귀상수: ', lr.intercept_)
회귀상수:  [0.21196612]

In [20]: print('회귀계수: ', lr.coef_)
회귀계수:  [[-0.01092474  0.01027057 -1.76505059  0.97353757]]

In [21]: from numpy import exp

In [22]: print('오즈비: ', exp(lr.coef_))
오즈비:  [[0.98913472 1.0103235  0.17117813 2.64729291]]

In [23]: lr.score(X_train, y_train)
Out[23]: 0.7835671342685371
```

- score() 함수를 이용하여 정분류율를 계산한 결과 0.783567로 나타났고, intercept_ 함수를 이용하여 절편을 계산한 결과 0.21196으로 나타났음. 그리고 coef_ 함수를 이용하여 독립변수인 age, fare, man, woman들의 기울기를 계산한 결과 각각 –0.0109, 0.01027, -1.76505, 0.97.537로 나타났음.
- 이러한 회귀계수 값을 numpy 모듈의 exp() 함수를 이용하여 age, fare,

통계청 통계교육원 본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

man, woman들의 오즈비를 계산한 결과 0.98913, 1.01032, 0.171178, 2.64729로 나타났음.

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 로지스틱회귀모형을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [24]: y_train_pred = lr.predict(X_train)

In [25]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [26]: confusion_matrix(y_train_pred, y_train)
Out[26]:
array([[235,  47],
       [ 61, 156]], dtype=int64)

In [27]: accuracy_score(y_train_pred, y_train)
Out[27]: 0.7835671342685371

In [28]: lr_report = classification_report(y_train, y_train_pred)

In [29]: print(lr_report)
              precision    recall  f1-score   support

           0       0.83      0.79      0.81       296
           1       0.72      0.77      0.74       203

    accuracy                           0.78       499
   macro avg       0.78      0.78      0.78       499
weighted avg       0.79      0.78      0.78       499
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 alive가 no와 yes를 각각 235개, 156개 데이터에 대해서 제대로 분류하여 정분류율는 0.783567로 나타났음. 그리고 alive가 no인 경우 precision, recall, f1-score는 각각 0.83, 0.79, 0.81로 나타났고, alive가 yes인 경우 precision, recall, f1-score는 각각 0.72, 0.77, 0.74로 나타났음.

- 그리고 Train 데이터의 sklearn.metrics 모듈의 roc_curve() 함수를 이용하여 fpr과 tpr을 구한 뒤 이를 이용하여 auc와 ROC curve를 그리고자 함.

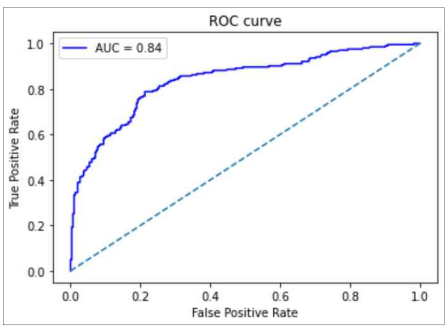
```
In [37]: from sklearn.metrics import roc_curve, auc

In [38]: y_train_prob = lr.predict_proba(X_train)[:,1]

In [39]: fpr, tpr, _ = roc_curve(y_train, y_train_prob)

In [40]: roc_auc = auc(fpr, tpr)

In [41]: import matplotlib.pyplot as plt
    ...: plt.plot(fpr, tpr, 'b', label='AUC = %.2f' %roc_auc)
    ...: plt.plot([0,1], [0,1], '--')
    ...: plt.ylabel('True Positive Rate')
    ...: plt.xlabel('False Positive Rate')
    ...: plt.title('ROC curve')
    ...: plt.legend(loc='best')
Out[41]: <matplotlib.legend.Legend at 0x22bafc3f208>
```



본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

- 그 결과 ROC curve의 면적인 AUC는 0.84의 값을 가지는 것을 확인할 수 있고, matplotlib.pyplot의 plot() 함수를 이용하여 x축에 fpr, y축에 tpr를 지정하여 그림을 그려 ROC curve를 확인할 수 있음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 로지스틱회귀모형을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [30]: lr.score(X_test, y_test)
Out[30]: 0.772093023255814

In [31]: y_test_pred = lr.predict(X_test)

In [32]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [33]: confusion_matrix(y_test_pred, y_test)
Out[33]:
array([[102,  23],
       [ 26,  64]], dtype=int64)

In [34]: accuracy_score(y_test_pred, y_test)
Out[34]: 0.772093023255814

In [35]: lr_report1 = classification_report(y_test, y_test_pred)

In [36]: print(lr_report1)
              precision    recall  f1-score   support

           0       0.82      0.80      0.81       128
           1       0.71      0.74      0.72        87

    accuracy                           0.77       215
   macro avg       0.76      0.77      0.76       215
weighted avg       0.77      0.77      0.77       215
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 alive가 no와 yes를 각각 102개, 64개 데이터에 대해서 제대로 분류하여 정분류율는 0.77209로 나타났음. 그리고 alive가 no인 경우 precision, recall, f1-score는 각각 0.82, 0.80, 0.81로 나타났고, alive가 yes인 경우 precision, recall, f1-score는 각각 0.71, 0.74, 0.72로 나타났음.
- 이는 Train 데이터를 활용하여 나타난 결과와 큰 차이가 존재하지 않는다는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 Logit() 함수에 대입하여 로지스틱회귀분석을 실시하고자 함.
- Logit() 함수는 앞서 설명하였듯이 절편항을 기본적으로 제공하지 않기 때문에 상수항을 추가한 입력변수 데이터를 생성해야 함. 따라서 이를 위해 statsmodels.api 모듈의 add_constant() 함수를 사용하여 입력변수

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

데이터에 절편항을 추가하였음.

```
In [58]: from statsmodels.api import add_constant

In [59]: X_train = add_constant(X_train)

In [60]: X_test = add_constant(X_test)

In [61]: print(X_train)
     const   age      fare  man  woman
741    1.0  36.0   78.8500    1      0
302    1.0  19.0    0.0000    1      0
514    1.0  24.0    7.4958    1      0
113    1.0  20.0    9.8250    0      1
632    1.0  32.0   30.5000    1      0
..     ...   ...       ...  ...    ...
435    1.0  14.0  120.0000    0      0
480    1.0   9.0   46.9000    0      0
115    1.0  21.0    7.9250    1      0
796    1.0  49.0   25.9292    0      1
236    1.0  44.0   26.0000    1      0

[499 rows x 5 columns]
```

- 그 결과 입력변수 데이터 0번째 열에 const라는 1로만 구성된 변수가 추가된 것을 확인할 수 있음.

- 절편항이 추가된 입력변수 데이터와 출력변수 데이터를 활용하여 statsmodels.api 모듈의 Logit().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 lr_result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 로지스틱회귀분석 결과를 확인하였음.

```
In [62]: from statsmodels.api import Logit

In [63]: lr_result = Logit(y_train, X_train).fit()
Optimization terminated successfully.
         Current function value: 0.483677
         Iterations 6

In [64]: lr_result.summary()
Out[64]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                alive_1   No. Observations:                  499
Model:                          Logit   Df Residuals:                      494
Method:                           MLE   Df Model:                            4
Date:                Mon, 30 Aug 2021   Pseudo R-squ.:                 0.2842
Time:                        14:09:43   Log-Likelihood:                -241.35
converged:                       True   LL-Null:                       -337.16
Covariance Type:            nonrobust   LLR p-value:                 2.381e-40
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2120      0.284      0.745      0.456      -0.346       0.769
age           -0.0109      0.010     -1.080      0.280      -0.031       0.009
fare           0.0103      0.003      3.479      0.001       0.004       0.016
man           -1.7651      0.418     -4.218      0.000      -2.585      -0.945
woman          0.9735      0.427      2.281      0.023       0.137       1.810
==============================================================================
"""
```

- 그 결과 LogisticRegression() 함수의 결과와 같은 절편항, 기울기항 등등을 확인할 수 있음.
- 그리고 LogisticRegression() 함수에서 확인할 수 없었던 계수들의 유의

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

성을 확인할 수 있음. age, fare, man, woman 변수들의 p-value는 각각 0.280, 0.001, 0.000, 0.023으로 나타나 fare, man, woman이 출력변수에 유의한 영향을 끼친다는 것을 확인할 수 있음.

```
In [65]: from numpy import exp

In [66]: exp(lr_result.params)
Out[66]:
const    1.236109
age      0.989135
fare     1.010323
man      0.171178
woman    2.647293
dtype: float64
```

- 그리고 로지스틱회귀분석을 실행시킨 객체에 params 함수를 대입하여 로지스틱회귀모형의 회귀계수를 확인할 수 있고, 이를 이용하여 오즈비를 계산하기 위해 numpy 모듈의 exp() 함수를 적용하면 age, fare, man, woman의 오즈비는 각각 0.989135, 1.010323, 0.171178, 2.647293인 것을 확인할 수 있음.

- 또한, sklearn 모듈에서 사용되었던 predict() 함수는 Logit() 함수에서는 출력변수가 1이 될 확률을 계산하는 함수임. 따라서 확률값에 절단값을 지정하여 출력변수의 예측값을 추가로 계산해주어야 함. 절단값이 0.5인 경우는 around() 함수를 이용하여 계산할 수 있고, 0.5 이외의 경우는 numpy 모듈의 where() 함수를 통해 계산할 수 있음.

```
In [67]: y_train_prob = lr_result.predict(X_train)

In [68]: print(y_train_prob)
741    0.242956
302    0.146708
514    0.149530
113    0.744201
632    0.169465
         ...
435    0.784399
480    0.644588
115    0.154320
796    0.714328
236    0.145948
Length: 499, dtype: float64
```

```
In [69]: from numpy import around

In [70]: y_train_pred = around(y_train_prob)

In [71]: print(y_train_pred)
741    0.0
302    0.0
514    0.0
113    1.0
632    0.0
      ...
435    1.0
480    1.0
115    0.0
796    1.0
236    0.0
Length: 499, dtype: float64
```

- 위 예제는 절단값을 0.5로 지정하여 around() 함수를 사용한 결과임. 확률값이 0.5이하인 경우는 0, 0.5이상인 경우는 1로 예측한 것을 확인할 수 있음.

- 다음으로 Train 데이터인 train 데이터를 logit() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- train 데이터를 독립변수 데이터셋과 종속변수 데이터셋으로 구분하지 않고 모형식을 활용하여 statsmodels.formula.api 모듈의 logit().fit() 함

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [87]: from statsmodels.formula.api import logit

In [88]: lr_result = logit(formula='alive_1 ~ age + fare + man + woman',data=train).fit()
Optimization terminated successfully.
         Current function value: 0.483677
         Iterations 6

In [89]: lr_result.summary()
Out[89]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                alive_1   No. Observations:                  499
Model:                          Logit   Df Residuals:                      494
Method:                           MLE   Df Model:                            4
Date:                Mon, 30 Aug 2021   Pseudo R-squ.:                 0.2842
Time:                        15:36:36   Log-Likelihood:                -241.35
converged:                       True   LL-Null:                       -337.16
Covariance Type:            nonrobust   LLR p-value:                2.381e-40
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.2120      0.284      0.745      0.456      -0.346       0.769
age           -0.0109      0.010     -1.080      0.280      -0.031       0.009
fare           0.0103      0.003      3.479      0.001       0.004       0.016
man           -1.7651      0.418     -4.218      0.000      -2.585      -0.945
woman          0.9735      0.427      2.281      0.023       0.137       1.810
==============================================================================
"""
```

- 그 결과 앞서 실시한 Logit() 함수와 같은 형태의 결과를 얻을 수 있음.

4. 로지스틱회귀분석의 실습

○ seaborn 모듈의 penguins 데이터에서, 성별(sex)을 예측하기 위해 bill_length_mm, bill_depth_mm, species를 사용한 로지스틱회귀분석을 실시하고자 함.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('penguins')

In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

- info() 함수를 활용하여 자료를 확인한 결과 sex, bill_length_mm, bill_depth_mm에서 결측값이 존재하여 dropna() 함수를 이용하여 결측값을 제거할 필요가 있음. 또한, 출력변수인 sex와 입력변수 중 species는 문자형 자료로 되어 있으므로 이를 확인할 필요가 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

```
In [4]: df = df.dropna(subset = ['sex', 'species', 'bill_length_mm', 'bill_depth_mm'], how='any', axis=0)

In [5]: df['sex'].value_counts()
Out[5]:
Male      168
Female    165
Name: sex, dtype: int64

In [6]: df['species'].value_counts()
Out[6]:
Adelie       146
Gentoo       119
Chinstrap     68
Name: species, dtype: int64
```

- 입력변수 중 species는 구성요소가 Adelie, Gentoo, Chinstrap으로 되어 있으므로 Adelie를 기준으로 하는 더미변수로 변형시킬 필요가 있음. 또한, 출력변수 인 sex는 Female과 male로 구성되어 있어 이를 각각 0과 1로 변경할 필요가 있음.

```
In [7]: from sklearn.preprocessing import LabelEncoder

In [8]: df['sex_1']=LabelEncoder().fit_transform(df['sex'].values)
```

- 먼저 출력변수의 값을 0과 1로 변형시키기 위해 sklearn.preprocessing 모듈의 LabelEncoder() 함수를 사용하였음. LabelEncoder() 함수에 fit_transform() 함수를 추가로 적용하여 values 함수를 통한 sex 변수의 값을 0과 1로 변형하여 데이터프레임의 마지막에 새롭게 sex_1이라는 변수명으로 저장하였음.

```
In [9]: import pandas as pd

In [10]: df1=pd.get_dummies(df['species'],drop_first=True)

In [11]: print(df1)
     Chinstrap  Gentoo
0            0       0
1            0       0
2            0       0
4            0       0
5            0       0
..         ...     ...
338          0       1
340          0       1
341          0       1
342          0       1
343          0       1

[333 rows x 2 columns]
```

```
In [12]: df = pd.concat([df, df1],axis=1)

In [13]: print(df)
    species     island  bill_length_mm  ...  sex_1  Chinstrap  Gentoo
0    Adelie  Torgersen            39.1  ...      1          0       0
1    Adelie  Torgersen            39.5  ...      0          0       0
2    Adelie  Torgersen            40.3  ...      0          0       0
4    Adelie  Torgersen            36.7  ...      0          0       0
5    Adelie  Torgersen            39.3  ...      1          0       0
..      ...        ...             ...  ...    ...        ...     ...
338  Gentoo     Biscoe            47.2  ...      0          0       1
340  Gentoo     Biscoe            46.8  ...      0          0       1
341  Gentoo     Biscoe            50.4  ...      1          0       1
342  Gentoo     Biscoe            45.2  ...      0          0       1
343  Gentoo     Biscoe            49.9  ...      1          0       1

[333 rows x 10 columns]
```

- 그리고 입력변수 중 하나인 species 변수를 더미변수로 만들기 위해 pandas 모듈의 get_dummies() 함수를 사용하였음. 그 결과 새롭게 Chinstrap과 Gentoo 변수가 생성되었고 이를 pandas 모듈의 concat() 함수를 이용하여 데이터프레임에 더미변수를 추가하였음.

- 다음으로 모형을 생성하기 위한 Train 데이터와 검증을 위한 Test 데이터를 구분하기 위해 train_test_split() 함수를 사용하였음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

```
In [14]: from sklearn.model_selection import train_test_split

In [15]: train, test = train_test_split(df, test_size=0.3, random_state=1, stratify=df['sex_1'])

In [16]: y_train=train['sex_1']
    ...: X_train=train[['bill_length_mm', 'bill_depth_mm', 'Chinstrap', 'Gentoo']]
    ...: y_test=test['sex_1']
    ...: X_test=test[['bill_length_mm', 'bill_depth_mm', 'Chinstrap', 'Gentoo']]
```

- tran_test_split() 함수의 stratify 옵션에 출력변수인 sex_1을 입력하여 출력변수에 따른 층화추출 방법으로 Train 데이터와 Test 데이터를 추출하였음. 이 때 Train 데이터는 train 객체에 할당하고, Test 데이터는 test 객체에 할당하였음. 그런 후 X_train과 X_test 객체에 train 객체와 test 객체의 독립변수를 선택하여 각각 할당하였고, y_train과 y_test 객체에 train 객체와 test 객체의 종속변수를 선택하여 각각 할당하였음.

- Train 데이터의 독립변수 자료인 X_train과 Train 데이터의 종속변수 자료인 y_train을 각각 LogisticRegression().fit() 함수에 대입하여 로지스틱 회귀분석을 실시하였음.

```
In [17]: from sklearn.linear_model import LogisticRegression

In [18]: lr = LogisticRegression(penalty='none').fit(X_train, y_train)
```

- LogisticRegression().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 lr이라는 객체에 할당하였고, 이를 활용하여 회귀계수, 오즈비, 정분류율을 각각 계산하였음.

```
In [19]: print('회귀상수: ', lr.intercept_)
회귀상수:  [-64.61307371]

In [20]: print('회귀계수: ', lr.coef_)
회귀계수:  [[ 0.66381094  2.10963611 -7.04917564  1.68654281]]

In [21]: from numpy import exp

In [22]: print('오즈비: ', exp(lr.coef_))
오즈비:  [[1.94217978e+00 8.24524035e+00 8.68124309e-04 5.40077688e+00]]

In [23]: lr.score(X_train, y_train)
Out[23]: 0.8927038626609443
```

- score() 함수를 이용하여 정분류율를 계산한 결과 0.8927로 나타났고, intercept_ 함수를 이용하여 절편을 계산한 결과 –64.6130으로 나타났음. 그리고 coef_ 함수를 이용하여 독립변수인 bill_length_mm, bill_depth_mm, Cinstrap, Gentoo 들의 기울기를 계산한 결과 각각 0.6638, 2.1096, -7.0491, 1.6865로 나타났음.
- 이러한 회귀계수 값을 numpy 모듈의 exp() 함수를 이용하여 bill_length_mm, bill_depth_mm, Cinstrap, Gentoo들의 오즈비를 계산한

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

결과 1.9421, 8.2452, 0.0008, 5.4007로 나타났음.

- predict() 함수에 Train 데이터의 입력변수 데이터인 X_train을 대입하여 Train 데이터의 출력변수 예측값을 y_train_pred 객체에 할당하여 로지스틱회귀모형을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [24]: y_train_pred = lr.predict(X_train)

In [25]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [26]: confusion_matrix(y_train_pred, y_train)
Out[26]:
array([[102,  12],
       [ 13, 106]], dtype=int64)

In [27]: accuracy_score(y_train_pred, y_train)
Out[27]: 0.8927038626609443

In [28]: lr_report = classification_report(y_train, y_train_pred)

In [29]: print(lr_report)
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       115
           1       0.89      0.90      0.89       118

    accuracy                           0.89       233
   macro avg       0.89      0.89      0.89       233
weighted avg       0.89      0.89      0.89       233
```

- 분류표, 정분류율, precision, recall, f1 score 등을 계산하기 위하여 sklearn.metrics 모듈의 confusion_matrix, accuracy_score, classification_report() 함수를 이용하였음. 그 결과 모형 생성에 사용된 Train 데이터에서는 sex가 Female과 Male을 각각 102개, 106개 데이터에 대해서 제대로 분류하여 정분류율는 0.8927로 나타났음. 그리고 sex가 Female인 경우 precision, recall, f1-score는 각각 0.89, 0.89, 0.89로 나타났고, sex가 Male인 경우 precision, recall, f1-score는 각각 0.89, 0.90, 0.89로 나타났음.

- 그리고 Train 데이터의 sklearn.metrics 모듈의 roc_curve() 함수를 이용하여 fpr과 tpr을 구한 뒤 이를 이용하여 auc와 ROC curve를 그리고자 함.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

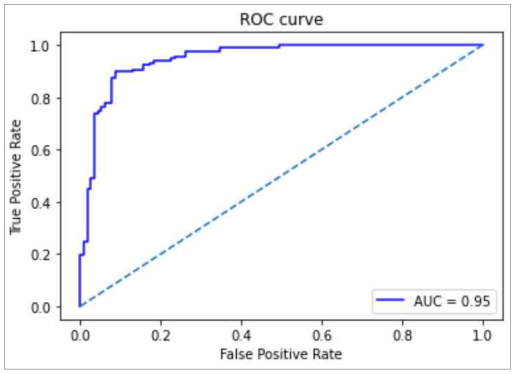
```
In [37]: from sklearn.metrics import roc_curve, auc

In [38]: y_train_prob = lr.predict_proba(X_train)[:,1]

In [39]: fpr, tpr, _ = roc_curve(y_train, y_train_prob)

In [40]: roc_auc = auc(fpr, tpr)

In [41]: import matplotlib.pyplot as plt
    ...: plt.plot(fpr, tpr, 'b', label='AUC = %.2f' %roc_auc)
    ...: plt.plot([0,1], [0,1], '--')
    ...: plt.ylabel('True Positive Rate')
    ...: plt.xlabel('False Positive Rate')
    ...: plt.title('ROC curve')
    ...: plt.legend(loc='best')
Out[41]: <matplotlib.legend.Legend at 0x259abbf96c8>
```



- 그 결과 ROC curve의 면적인 AUC는 0.95의 값을 가지는 것을 확인할 수 있고, matplotlib.pyplot의 plot() 함수를 이용하여 x축에 fpr, y축에 tpr를 지정하여 그림을 그려 ROC curve를 확인할 수 있음.

- 다음으로 predict() 함수에 Test 데이터의 독립변수 데이터인 X_test를 대입하여 Test 데이터의 종속변수 예측값을 y_test_pred 객체에 할당하여 로지스틱회귀모형을 평가하는 지표인 정분류율, precision, recall, f1 score 등을 계산하였음.

```
In [30]: lr.score(X_test, y_test)
Out[30]: 0.86

In [31]: y_test_pred = lr.predict(X_test)

In [32]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

In [33]: confusion_matrix(y_test_pred, y_test)
Out[33]:
array([[45,  9],
       [ 5, 41]], dtype=int64)

In [34]: accuracy_score(y_test_pred, y_test)
Out[34]: 0.86

In [35]: lr_report1 = classification_report(y_test, y_test_pred)

In [36]: print(lr_report1)
              precision    recall  f1-score   support

           0       0.83      0.90      0.87        50
           1       0.89      0.82      0.85        50

    accuracy                           0.86       100
   macro avg       0.86      0.86      0.86       100
weighted avg       0.86      0.86      0.86       100
```

- 그 결과 모형 생성에 사용된 Test 데이터에서는 sex가 Famale과 Male을 각각 45개, 41개 데이터에 대해서 제대로 분류하여 정분류율은

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

0.86으로 나타났음. 그리고 sex가 Female인 경우 precision, recall, f1-score는 각각 0.83, 0.90, 0.87로 나타났고, sex가 Male인 경우 precision, recall, f1-score는 각각 0.89, 0.82, 0.85로 나타났음.

- 이는 Train 데이터를 활용하여 나타난 결과와 큰 차이가 존재하지 않는다는 것을 확인할 수 있음.

- 다음으로 Train 데이터의 입력변수 자료인 X_train과 Train 데이터의 출력변수 자료인 y_train을 각각 Logit() 함수에 대입하여 로지스틱회귀분석을 실시하고자 함.
- Logit() 함수는 앞서 설명하였듯이 절편항을 기본적으로 제공하지 않기 때문에 상수항을 추가한 입력변수 데이터를 생성해야 함. 따라서 이를 위해 statsmodels.api 모듈의 add_constant() 함수를 사용하여 입력변수 데이터에 절편항을 추가하였음.

```
In [57]: from statsmodels.api import add_constant

In [58]: X_train = add_constant(X_train)

In [59]: X_test = add_constant(X_test)

In [60]: print(X_train)
     const  bill_length_mm  bill_depth_mm  Chinstrap  Gentoo
28     1.0           37.9           18.6          0       0
234    1.0           45.8           14.6          0       1
48     1.0           36.0           17.9          0       0
199    1.0           49.0           19.6          1       0
169    1.0           58.0           17.8          1       0
..     ...            ...            ...        ...     ...
277    1.0           45.5           15.0          0       1
258    1.0           44.0           13.6          0       1
342    1.0           45.2           14.8          0       1
299    1.0           45.2           16.4          0       1
96     1.0           38.1           18.6          0       0

[233 rows x 5 columns]
```

- 그 결과 입력변수 데이터 0번째 열에 const라는 1로만 구성된 변수가 추가된 것을 확인할 수 있음.

- 절편항이 추가된 입력변수 데이터와 출력변수 데이터를 활용하여 statsmodels.api 모듈의 Logit().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 lr_result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 로지스틱회귀분석 결과를 확인하였음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

```
In [61]: from statsmodels.api import Logit

In [62]: lr_result = Logit(y_train, X_train).fit()
Optimization terminated successfully.
         Current function value: 0.287318
         Iterations 8

In [63]: lr_result.summary()
Out[63]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                  sex_1   No. Observations:                  233
Model:                          Logit   Df Residuals:                      228
Method:                           MLE   Df Model:                            4
Date:                Mon, 30 Aug 2021   Pseudo R-squ.:                  0.5854
Time:                        15:40:24   Log-Likelihood:                -66.945
converged:                       True   LL-Null:                       -161.48
Covariance Type:            nonrobust   LLR p-value:                 8.365e-40
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -64.6130      8.583     -7.528      0.000     -81.436     -47.790
bill_length_mm     0.6638      0.115      5.748      0.000       0.437       0.890
bill_depth_mm      2.1096      0.339      6.228      0.000       1.446       2.774
Chinstrap         -7.0493      1.365     -5.165      0.000      -9.724      -4.374
Gentoo             1.6866      1.419      1.188      0.235      -1.095       4.468
==============================================================================
"""
```

- 그 결과 LogisticRegression() 함수의 결과와 같은 절편항, 기울기항 등 등을 확인할 수 있음.
- 그리고 LogisticRegression() 함수에서 확인할 수 없었던 계수들의 유의성을 확인할 수 있음. bill_length_mm, bill_depth_mm, Cinstrap, Gentoo 변수들의 p-value는 각각 0.0000, 0.0000, 0.0000, 0.235으로 나타나 bill_length_mm, bill_depth_mm, Cinstrap이 출력변수에 유의한 영향을 끼친다는 것을 확인할 수 있음.

```
In [64]: from numpy import exp

In [65]: exp(lr_result.params)
Out[65]:
const             8.688037e-29
bill_length_mm    1.942196e+00
bill_depth_mm     8.245048e+00
Chinstrap         8.680313e-04
Gentoo            5.400829e+00
dtype: float64
```

- 그리고 로지스틱회귀분석을 실행시킨 객체에 params 함수를 대입하여 로지스틱회귀모형의 회귀계수를 확인할 수 있고, 이를 이용하여 오즈비를 계산하기 위해 numpy 모듈의 exp() 함수를 적용하면 bill_length_mm, bill_depth_mm, Cinstrap, Gentoo의 오즈비는 각각 1.9421, 8.2450, 0.0008, 5.4008인 것을 확인할 수 있음.

- 또한, sklearn 모듈에서 사용되었던 predict() 함수는 Logit() 함수에서는 출력변수가 1이 될 확률을 계산하는 함수임. 따라서 확률값에 절단값

통계청 통계교육원 본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.

을 지정하여 출력변수의 예측값을 추가로 계산해주어야 함. 절단값이 0.5인 경우는 around() 함수를 이용하여 계산할 수 있고, 0.5 이외의 경우는 numpy 모듈의 where() 함수를 통해 계산할 수 있음.

```
In [66]: y_train_prob = lr_result.predict(X_train)

In [67]: print(y_train_prob)
28     0.446340
234    0.151460
48     0.049573
199    0.901433
169    0.987755
         ...
277    0.253790
258    0.006511
342    0.154522
299    0.842350
96     0.479335
Length: 233, dtype: float64
```

```
In [68]: from numpy import around

In [69]: y_train_pred = around(y_train_prob)

In [70]: print(y_train_pred)
28     0.0
234    0.0
48     0.0
199    1.0
169    1.0
      ...
277    0.0
258    0.0
342    0.0
299    1.0
96     0.0
Length: 233, dtype: float64
```

- 위 예제는 절단값을 0.5로 지정하여 around() 함수를 사용한 결과임. 확률값이 0.5이하인 경우는 0, 0.5이상인 경우는 1로 예측한 것을 확인할 수 있음.

- 다음으로 Train 데이터인 train 데이터를 logit() 함수에 대입하여 선형회귀분석을 실시하고자 함.
- train 데이터를 독립변수 데이터셋과 종속변수 데이터셋으로 구분하지 않고 모형식을 활용하여 statsmodels.formula.api 모듈의 logit().fit() 함수를 실행시킨 결과는 출력되지 않으므로 이를 result라는 객체에 할당하였고, 이를 summary() 함수를 적용하여 선형회귀분석 결과를 확인하였음.

```
In [86]: from statsmodels.formula.api import logit

In [87]: lr_result = logit(formula='sex_1 ~ bill_length_mm + bill_depth_mm + Chinstrap + Gentoo',data=train).fit()
Optimization terminated successfully.
         Current function value: 0.287318
         Iterations 8

In [88]: lr_result.summary()
Out[88]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                  sex_1   No. Observations:                  233
Model:                          Logit   Df Residuals:                      228
Method:                           MLE   Df Model:                            4
Date:                Mon, 30 Aug 2021   Pseudo R-squ.:                 0.5854
Time:                        15:42:08   Log-Likelihood:                -66.945
converged:                       True   LL-Null:                       -161.48
Covariance Type:            nonrobust   LLR p-value:                 8.365e-40
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -64.6130      8.583     -7.528      0.000     -81.436     -47.790
bill_length_mm     0.6638      0.115      5.748      0.000       0.437       0.890
bill_depth_mm      2.1096      0.339      6.228      0.000       1.446       2.774
Chinstrap     -7.0493      1.365     -5.165      0.000      -9.724      -4.374
Gentoo         1.6866      1.419      1.188      0.235      -1.095       4.468
==============================================================================
"""
```

- 그 결과 앞서 실시한 Logit() 함수와 같은 형태의 결과를 얻을 수 있음.

본 문서는 저작권자의 동의 없이 무단 복제 및 배포할 수 없습니다.