

사전직무교육IT

데이터분석 - 분류

강사 이수현

목차



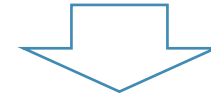
우리는 사람의 얼굴을 봤을 때 그 사람이 남성인지 여성인지 직관적으로 구분할 수 있습니다. 그런데 사람의 얼굴에서 어떠한 점을 기준으로 남성과 여성을 구분하는지는 명확하지 않습니다. 어떻게 보면 남성의 이목구비가 여성에 비해 큰 경향이 있는 것처럼 보이기도 합니다.

이번 활동에서는 머신러닝으로 성별을 분류하는 인공지능 모델을 만들어 봅니다.

문제 정의하기 성별 분류



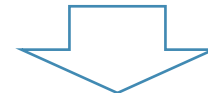
데이터 불러오기 이목구비 데이터 셋



데이터 탐색 및 시각화하기

- 범주형 수치형 변환
- 속성 간 상관관계 분석하기
- 데이터 나누기

훈련	테스트
----	-----



이 장에서는
다음의 순서로
살펴봅시다.

**이 장에서는
다음의 순서로
살펴봅시다.**



**모델 생성하기
로지스틱 회귀 모델**



모델 학습하기



모델 평가 및 예측하기

- 문제 해결에 필요한 정보
- 문제 해결 과정에서 필요한정보를 미리 살펴봅시다.
 1. 이 활동에 필요한 데이터 셋은 무엇이고, 이 데이터 셋은 어디에서 다운로드할 수 있나요?
 - 데이터 셋은 이목구비 데이터 셋으로, 캐글에서 다운로드할 수 있습니다.

2. 다운로드한 데이터 셋을 모델 학습에 사용하려면 어떻게 해야 할까요?

로지스틱 회귀 모델(분류 모델) 학습에 사용할 수 있도록 데이터 탐색 후 전처리 과정이 필요합니다.

결측치나 이상치가 없는지 확인하기, 범주형 데이터를 수치형 데이터로 변환하기.

성별 분류에 영향을 미치는 속성 선정하기.

훈련 데이터와 테스트 데이터 나누기 등을 합니다.

3. 모델 생성에서 사용할 모델을 찾아볼까요?

여기서는 성별 분류를 위해 로지스틱 회귀 모델을 사용합니다.

4. 모델 학습에서 우리가 해야 할 작업은 무엇일까요?

모델이 학습할 훈련 데이터와 훈련 데이터의 레이블 설정입니다.

5. 로지스틱 회귀 모델 성능을 나타내는 평가 지표는 무엇일까요?

테스트 데이터의 예측값과 실젯값을 비교하여 정확도를 산출합니다.

머신러닝에서 성별을 분류하는 문제를 해결하기 위해서는 속성별로 정리되어 있는 정형 데이터를 사용해야 한다.

이목구비 데이터 셋

이번 활동에서의 머신러닝은 딥러닝을 포함하지 않습니다.

- **데이터:** 사람의 얼굴 이미지 데이터(비정형 데이터)가 아니라 5,001명의 남녀 이목구비 크기를 정리한 정형 데이터
- **속성:** 머리카락 길이, 이마 너비, 이마 높이, 코 너비, 코 길이, 입술 두께, 인중 길이, 성별

이목구비 데이터 셋 속성

1	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender
2	1	11.8	6.1	1	0	1		1 Male
3	0	14	5.4	0	0	1		0 Female
4	0	11.8	6.3	1	1	1		1 Male
5	0	14.4	6.1	0	1	1		1 Male
6	1	13.5	5.9	0	0	0		0 Female
7	1	15.3	6.2	1	1	1		0 Male
8	0	13	5.2	0	0	0		0 Female
9	1	11.6	5.9	0	0	0		0 Female
10	1	12.9	7	1	1	1		1 Male
4994	1	11.6	5.9	0	0	0		1 Female
4995	1	12.9	7	1	1	1		1 Male
4996	1	11.6	5.9	0	0	0		1 Female
4997	1	12.9	7	1	1	1		1 Male
4998	1	13.6	5.1	0	0	0		0 Female
4999	1	11.9	5.4	0	0	0		0 Female
5000	1	12.9	5.7	0	0	0		0 Female
5001	1	13.2	6.2	0	0	0		0 Female
5002	1	15.4	5.4	1	1	1		1 Male

머리카락 길이(long_hair), 이마 너비(forehead_width_cm),
 이마 높이(forehead_height_cm), 코 너비(nose_wide),
 코 길이(nose_long), 입술 두께(lips_thin),
 인종 길이(distance_nose_to_lip_long), 성별(gender)

이목구비 데이터
 셋의 속성별
 설명은 244쪽에서!

정형 데이터와 비정형 데이터는 어떻게 다른가요?

정형 데이터는 csv 파일이나 스프레드시트 데이터처럼 행과 열로 표현할 수 있는 정해진 형식으로 저장되어 있어 특징을 쉽게 찾을 수 있다.

비정형 데이터는 전체 데이터의 80%를 차지하며, 이미지, 동영상, 텍스트, 음성 등으로 데이터 구조가 정해지지 않아 특징을 쉽게 찾을 수 없다.

이러한 비정형 데이터를 머신러닝으로 처리하기 위해서는 기술의 발전이 요구되는 실정이다.

데이터 셋 불러오기

- 이목구비 데이터 셋은 캐글(kaggle.com)에서 수집할 수 있다.
캐글은 기업이나 단체에서 해결하고자 하는 과제와 데이터를 등록하면 사람들이 이 문제를 해결하는 모델을 생성하고 경쟁하여 새로운 정보를 분석해 내는 대회 플랫폼이다.
- 공개되어 있는 데이터가 매우 많아서 인공지능 실습에 유용한 데이터를 쉽게 얻을 수 있고, 구글 계정이 있다면 쉽게 가입할 수 있다.

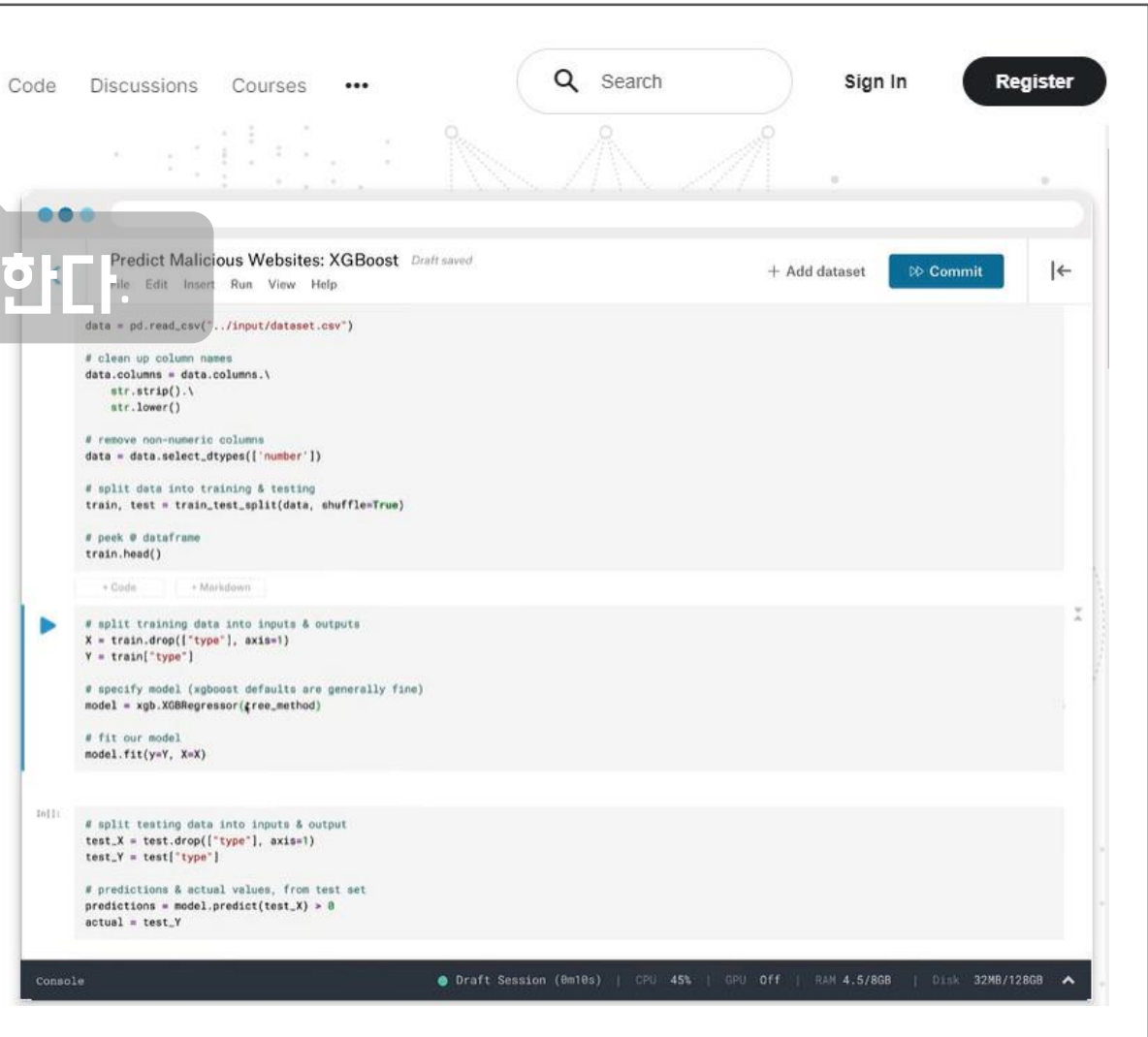
개글에 접속한 후 **Datasets**를 클릭한다.

Start with more than
a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code.

 REGISTER WITH GOOGLE

Register with Email



The screenshot shows the Kaggle website with the 'Datasets' tab highlighted in the navigation bar. Below the navigation bar, there's a search bar and 'Sign In' and 'Register' buttons. The main content area displays a Jupyter Notebook titled 'Predict Malicious Websites: XGBoost'. The notebook code includes data loading, cleaning, splitting, and model training using XGBoost. The console at the bottom shows the draft session status: 'Draft Session (0m10s) | CPU 45% | GPU Off | RAM 4.5/8GB | Disk 32MB/128GB'.

```
data = pd.read_csv("../input/dataset.csv")

# clean up column names
data.columns = data.columns.\
    str.strip().\
    str.lower()

# remove non-numeric columns
data = data.select_dtypes(['number'])

# split data into training & testing
train, test = train_test_split(data, shuffle=True)

# peek @ dataframe
train.head()

# split training data into inputs & outputs
X = train.drop(["type"], axis=1)
Y = train["type"]

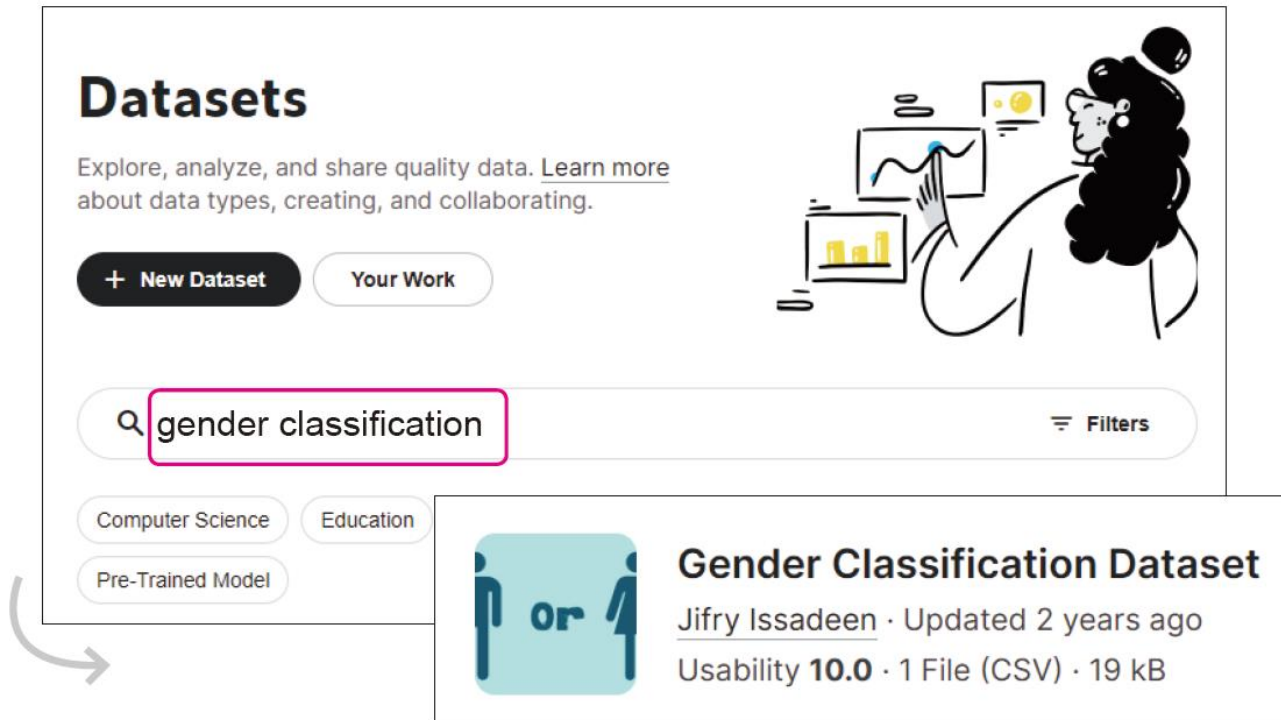
# specify model (xgboost defaults are generally fine)
model = xgb.XGBRegressor(tree_method)

# fit our model
model.fit(y=Y, X=X)

# split testing data into inputs & output
test_X = test.drop(["type"], axis=1)
test_Y = test["type"]

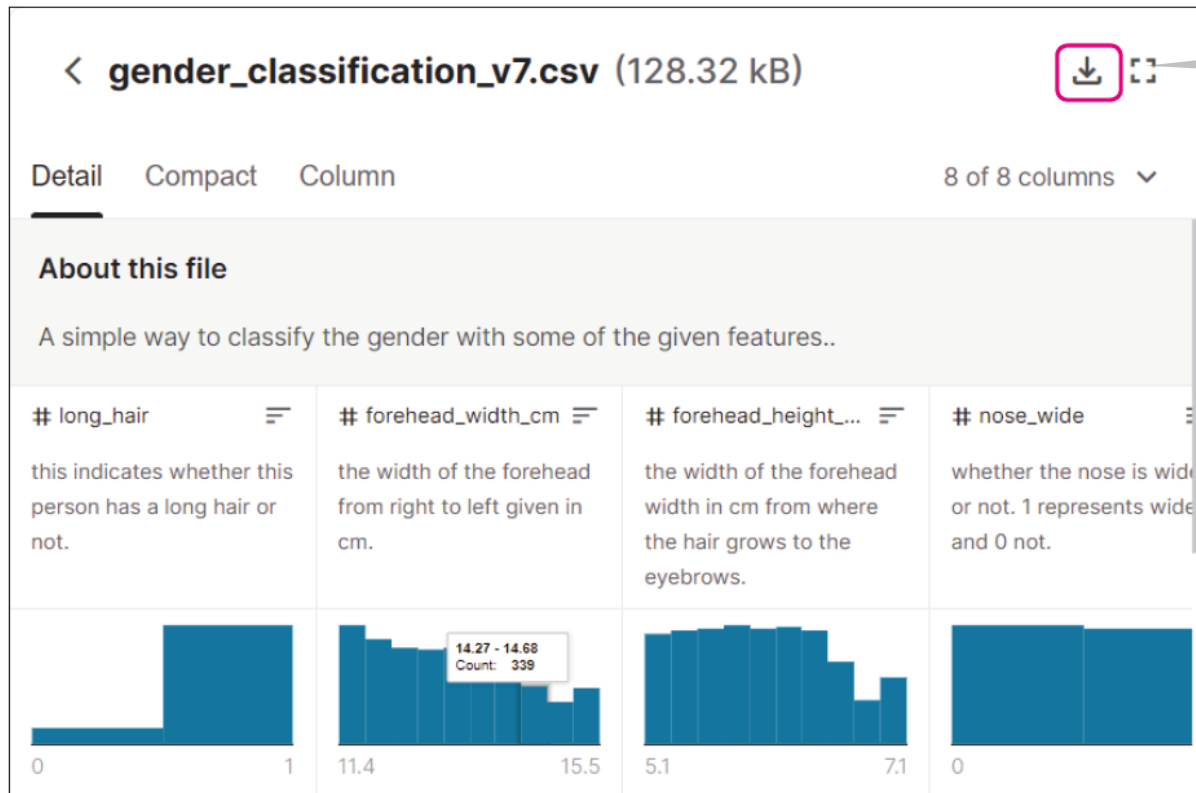
# predictions & actual values, from test set
predictions = model.predict(test_X) > 0
actual = test_Y
```

데이터 셋 다운로드하기



검색창에서 'gender classification'이라고 검색하고, 검색 결과 중 Jifry Issadeen이 등록한 'Gender Classification Dataset'을 클릭합니다.

데이터 셋 다운로드하기



다운로드 버튼

데이터 셋 업로드하기

1. 구글 드라이브와 코랩 연동하기



```
1 from google.colab import drive  
2 drive.mount('/content/drive')
```

2. 구글 드라이브 파일에 접근 허용하기

'Google Drive에 연결' 버튼 클릭

오른쪽 코드는 구글 코랩 라이브러리에서 드라이브와 관련된 부분만 가져온 후 드라이브를 연동하겠다는 명령입니다.

노트북에서 Google Drive 파일에 액세스하도록 허용하시겠습니까?

이 노트북에서 Google Drive 파일에 대한 액세스를 요청합니다. Google Drive에 대한 액세스 권한을 부여하면 노트북에서 실행되는 코드가 Google Drive의 파일을 수정할 수 있게 됩니다. 이 액세스를 허용하기 전에 노트북 코드를 검토하시기 바랍니다.

아니요

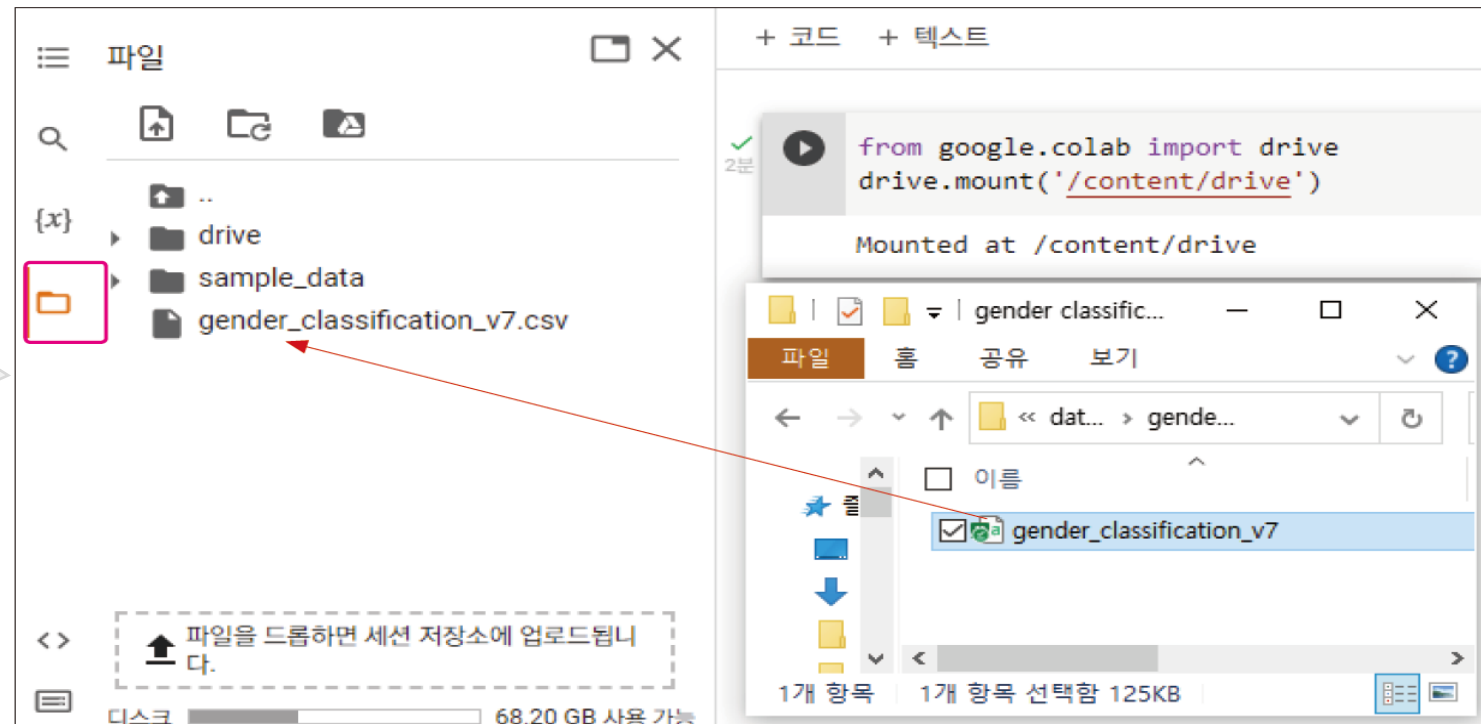
Google Drive에 연결

데이터 셋 업로드하기 (mount방법)



4. 파일 아이콘 클릭 후, 다운로드한 데이터 셋을 파일 창에 끌어다 놓으면, 'gender_classification_v7.csv' 파일 업로드 확인하기

이제 이 파일의 경로는
'/content/gender_
classification_
n_v7.csv'입니다.



* 데이터 셋 업로드하기 (탐색기로 직접 올려주는 방법)



4. 파일 아이콘 클릭 후, 다운로드한 데이터 셋을 파일 창에 끌어다 놓으면, 'gender_classification_v7.csv' 파일 업로드 확인하기

파일



sample_data

이제 이 파일의 경로는
'/content/gender_
classification_v7.csv'
입니다.



+ 코드 + 텍스트

✓
25
초



```
1 from google.colab import files  
2 filename=list(files.upload().keys())[0]
```

파일 선택

gender_classification_v7.csv

gender_classification_v7.csv

d: 2022. 1. 4. - 100% done

파일 읽어 들이기

1. 이목구비 데이터 셋 코랩으로 불러오기

데이터프레임 객체 = 판다스 객체.read_csv('파일명.csv')

판다스 라이브러리의 read_csv() 함수를 사용하여 csv 형식의 파일을 코랩으로 읽어 들임.

- 파일의 형식은 csv(comma-separated values)이므로 판다스 라이브러리에서 제공하는 파일 읽기 명령(read_csv) 사용
- 판다스 라이브러리를 제일 먼저 불러옴.

파일 읽어 들이기

2 이목구비 데이터 셋을 df라는 이름으로 사용하기

```
1 import pandas as pd
2 df = pd.read_csv('gender_classification_v7.csv')
3 df
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender
0	1	11.8	6.1	1	0	1	1	Male
1	0	14.0	5.4	0	0	1	0	Female
2	0	11.8	6.3	1	1	1	1	Male
3	0	14.4	6.1	0	1	1	1	Male
4	1	13.5	5.9	0	0	0	0	Female

작업 내용 미리보기

- 데이터를 불러온 후 데이터 셋에 어떤 속성과 값들이 포함되어 있는지 살펴보고, 성별을 분류하는 데 필요한 속성을 선정
- 이 속성을 활용하여 인공지능 모델을 학습
 - 데이터 살펴보기
 - 데이터 시각화하기
 - 상관관계 분석하기
 - 훈련 데이터와 테스트 데이터 나누기

데이터 탐색 및 전처리

데이터 살펴보기

- 속성 탐색
- 결측치 또는 이상치 탐색

데이터 시각화하기

- 데이터 속성값의 분포 시각화
- 성별 분류에 영향을 미치는 속성 탐색

상관관계 분석하기

- 원-핫 인코딩 수행
- 상관관계 분석을 통해 성별 분류에 영향을 미치는 속성 선정

데이터 나누기

- 훈련 데이터와 테스트 데이터 분리

상관관계 분석은 수치형 데이터로 처리, 범주형 데이터(문자열 값)를 수치형 데이터로 변환하는 원-핫 인코딩 수행

- 데이터
살펴보기

- 몇 개의 데이터와 속성이 있는지, 어떤 속성이 있는지 살펴본다.
- 판다스 라이브러리의 `info()` 메소드를 사용한다.

데이터프레임 객체.info

`info()` 메소드를 통해 데이터 개수, 속성 개수, 속성명, 결측치,
속성의 데이터 타입 등 확인

데이터 기초 정보 확인하기

1 df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5001 entries, 0 to 5000
Data columns (total 8 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   long_hair                            5001 non-null   int64
1   forehead_width_cm                   5001 non-null   float64
2   forehead_height_cm                  5001 non-null   float64
3   nose_wide                           5001 non-null   int64
4   nose_long                           5001 non-null   int64
5   lips_thin                           5001 non-null   int64
6   distance_nose_to_lip_long           5001 non-null   int64
7   gender                              5001 non-null   object
dtypes: float64(2), int64(5), object(1)
memory usage: 312.7+ KB
```

속성명	설명
long_hair	머리카락 길이(0: 짧은 머리, 1: 긴 머리)
forehead_width_cm	이마 너비(cm)
forehead_height_cm	이마 높이(cm)
nose_wide	코 너비(0: 좁은 코, 1: 넓은 코)
nose_long	코 길이(0: 짧은 코, 1: 긴 코)
lips_thin	입술 두께(0: 얇은 입술, 1: 두꺼운 입술)
distance_nose_to_lip_long	인중 길이(0: 짧은 인중, 1: 긴 인중)
gender	성별(male: 남성, female: 여성)

데이터 기초 정보 확인하기

이 데이터 셋은 총 5,001개의 데이터와 8개의 속성으로 구성되어 있고, 각 속성이 5,001개의 값을 가진 것으로 보아 결측치가 없음(non-null)을 확인할 수 있습니다.

속성별 데이터 유형은 머리카락 길이(long_hair), 코 너비(nose_wide), 코 길이(nose_long), 입술 두께(lips_thin), 인종 길이(distance_nose_to_lip_long)는 정수형(int64),

이마 너비(forehead_width_cm)와 이마 높이(forehead_height_cm)는 실수형(float64),

성별(gender)은 범주형(object)인 것을 알 수 있습니다.

데이터 기초 정보 확인하기

실제로 데이터가 어떻게 생겼는지 확인해 봅시다.
전체 데이터를 불러오지 않고 **head()** 메소드를 사용하여
0~4행까지의 데이터만 살펴봅니다.

데이터프레임 객체.head(확인할 데이터 개수)

head() 메소드를 통해 데이터 상단의 일부 데이터 확인

head()의 괄호 안에 숫자를 입력하지 않으면
기본값인 5로 설정되어 0~4행의 데이터가 출력

데이터 기초 정보 확인하기

이목구비 데이터 셋의 속성들이 어떻게 테이블로 정리되어 있는지 출력하기

```
1 df.head( )
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender
0	1	11.8	6.1	1	0	1	1	Male
1	0	14.0	5.4	0	0	1	0	Female
2	0	11.8	6.3	1	1	1	1	Male
3	0	14.4	6.1	0	1	1	1	Male
4	1	13.5	5.9	0	0	0	0	Female

데이터 통계치 살펴보기

describe() 메소드를 사용하여 데이터들의 통계량(개수, 평균, 표준편차, 최솟값, 최댓값, 4분위수)을 파악하기

데이터프레임 객체.describe()

describe() 메소드를 통해 데이터 개수, 평균, 표준편차,
최솟값, 최댓값, 4분위수 파악

수치값을 갖는 속성에 대해서만 통계치를 출력하며 결측치는 제외됩니다.
성별(gender) 속성은 문자열이므로 통계치가 출력되지 않습니다.

데이터 기초 정보 확인하기

8개 속성 중 성별을 제외한 수치의 값을 갖는 7개 속성의 통계량 출력하기

```
1 df.describe( )
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long
count	5001.000000	5001.000000	5001.000000	5001.000000	5001.000000	5001.000000	5001.000000
mean	0.869626	13.181484	5.946311	0.493901	0.507898	0.493101	0.498900
std	0.336748	1.107128	0.541268	0.500013	0.499988	0.500002	0.500049
min	0.000000	11.400000	5.100000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	12.200000	5.500000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	13.100000	5.900000	0.000000	1.000000	0.000000	0.000000
75%	1.000000	14.000000	6.400000	1.000000	1.000000	1.000000	1.000000
max	1.000000	15.500000	7.100000	1.000000	1.000000	1.000000	1.000000

데이터 통계치 살펴보기

- 머리카락 길이(long_hair), 코 너비(nose_wide), 코 길이(nose_long), 입술 두께(lips_thin), 인종 길이(distance_nose_to_lip_long) 속성의 값은 0.0~1.0의 범위에 있는 것
- 평균(mean)을 살펴보면 머리카락 길이(long_hair) 속성의 값은 약 0.87로 대체로 머리카락 길이가 긴 사람들의 데이터가 많다는 것
- 이마 너비(forehead_width_cm)와 이마 높이(forehead_height_cm)를 제외한 나머지 속성의 평균이 약 0.5인 것을 고려할 때, 전반적으로 반반씩 섞여 있을 가능성(예: 코 너비가 넓은 사람과 좁은 사람이 대략 반반일 가능성)을 유추

데이터 통계치 살펴보기

- 단위가 cm이고 실수형 데이터 유형인 이마 너비 (forehead_width_cm) 와 이마 높이 (forehead_height_cm)의 통계치도 확인 가능
- 이마 너비 (forehead_width_cm)의 경우에는 최소 11.4cm~최대 15.5cm의 범위에 있고 평균은 13.18cm이며, 이마 높이 (forehead_height_cm)의 경우에는 최소 5.1cm~최대 7.1cm의 범위에 있고 평균은 5.95cm임을 알 수 있음.

데이터 통계치 살펴보기

이목구비 데이터 셋의 속성을 데이터프레임으로 살펴본 결과, 실제 데이터가 어떻게 구성되었는지 기본 정보와 통계량을 통해 쉽게 확인할 수 있었으나 각 속성값의 분포를 비교하기에는 어렵다.

데이터 시각화하기

- 데이터를 그래프로 그려 시각화하면 데이터 값의 분포를 파악하거나 평균을 쉽게 비교
- 이목구비 데이터를 시각화하여 성별을 분류하는 데 영향을 미칠만한 속성에는 무엇이 있는지 살펴보기

성별 분포 확인하기

- 데이터의 비율이 한쪽으로 치우쳐 있으면 치우친 성으로 인식할 가능성이 크기 때문에 남녀 데이터의 비율이 비슷하게
- 각 속성의 고윳값의 개수를 출력하기 위해 **value_counts()** 메소드 사용

데이터프레임 객체['속성명'].value_counts()

속성명에 포함된 각 고윳값의 개수 출력

성별 분포 확인하기

성별 분포를 확인하기 위하여 맷플롯립 라이브러리를 불러
와 성별(gender) 속성값을 비교하는 그래프 그리기

```
1 import matplotlib.pyplot as plt  
2 df['gender'].value_counts( )
```

Female 2501

Male 2500

Name: gender, dtype: int64

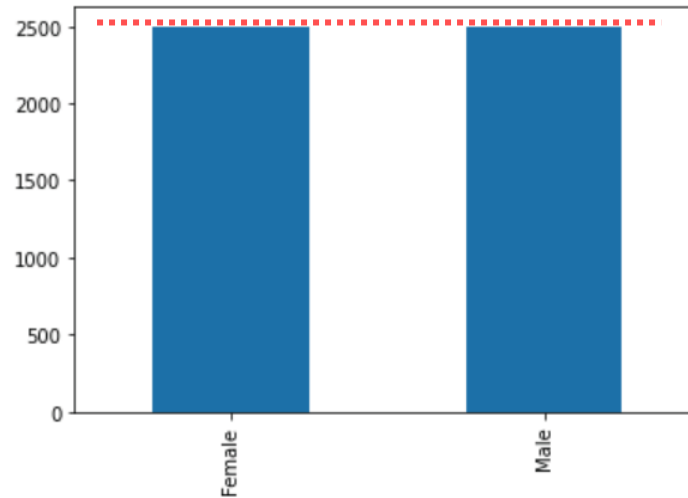
위 결과에서 여성은 2,501명, 남성은 2,500명으로
한눈에 봐도 **성별 균형이 맞는 것** 확인 가능

성별 분포 확인하기

- 막대그래프를 그려 살펴보기
- `plot.bar()` 명령 사용

```
1 df['gender'].value_counts().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f7daa1a5b90>



성별 균형이 맞는지
막대그래프를 통해
쉽게 확인 가능

성별간 머리카락 길이 확인하기

- 머리카락 길이가 영향을 미칠 가능성이 있는지 성별과 머리카락 길이의 평균을 비교하는 그래프 그리기
- 머리카락 길이(long_hair) 속성을 남성과 여성으로 분류하기 위해 **groupby()** 메소드 사용

데이터프레임 객체[속성명1].groupby(데이터프레임 객체[속성명2])

속성명2의 고유태를 기준으로 속성명1을 그룹화

예) `df['long_hair'].groupby(df['gender'])`는

gender값인 male, female 2개의 그룹으로 long_hair를 분류함.

성별간 머리카락 길이 확인하기

- `groupby()`를 사용하여 머리카락 길이(`long_hair`)를 'male'과 'female' 그룹으로 분류
- 각각의 평균(`mean()`)을 소수점 아래 둘째 자리까지 반올림(`round(2)`)한 결과값을 `long_hair_count`으로 설정

```
1 long_hair_count = df['long_hair'].groupby(df['gender']).mean( ).round(2)
2 long_hair_count
```

```
gender
Female    0.87
Male      0.87
Name: long_hair, dtype: float64
```

성별간 머리카락 길이 확인하기

실행 결과를 통해 여성과 남성의 머리카락 평균 길이가 0.87로 동일한 것을 알 수 있습니다.
따라서 머리카락 길이로는 남성과 여성을 분류하기 어려울 수 있음을 예상해 볼 수 있습니다.

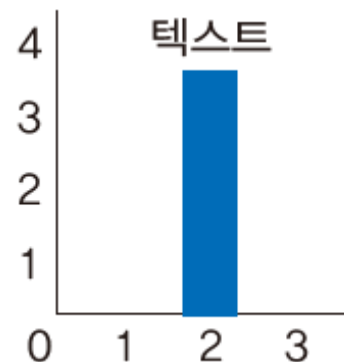
성별간
머리카락 길이
확인하기

- `plot.bar()` 명령을 사용하여 막대그래프를 그린 후, `text()` 메소드로 각 막대의 윗부분에 출력
- 막대그래프의 각 막대에 값을 출력하기 위해 `plt.text()` 메소드 사용

`plt.text(x축 위치, y축 위치, 출력할 내용)`

그래프의 (x, y) 위치에 '출력할 내용' 출력

예) `plt.text(2, 3, '텍스트')`는 오른쪽 그래프처럼 출력



성별간
머리카락 길이
확인하기

다른 속성들도 남녀 간의 평균 차이가 있는지
동일한 방식으로 확인하기



```
1 long_hair_count.plot.bar( ) # 막대그래프 그리기
2 plt.text(0, long_hair_count[0] + 0.02, long_hair_count[0],
3 va = 'center', ha = 'center') # Female에 대한 평균 출력
4 plt.text(1, long_hair_count[1] + 0.02, long_hair_count[1],
5 va = 'center', ha = 'center') # Male에 대한 평균 출력
6 plt.show( )
```

성별간 머리카락 길이 확인하기



```
1 long_hair_count.plot.bar() # 막대그래프 그리기
2 plt.text(0, long_hair_count[0] + 0.02, long_hair_count[0],
3         va = 'center', ha = 'center') # Female에 대한 평균 출력
4 plt.text(1, long_hair_count[1] + 0.02, long_hair_count[1],
5         va = 'center', ha = 'center') # Male에 대한 평균 출력
6 plt.show()
```

[다른속성들도
남녀간의평균치이 있는지
동일한방식으로확인하기]

x축 값: 막대그래프 순서
0은 제일 왼쪽 막대

y축 값: 여성 머리카락
길이의 평균값

y축 값에서
0.02 위의 값

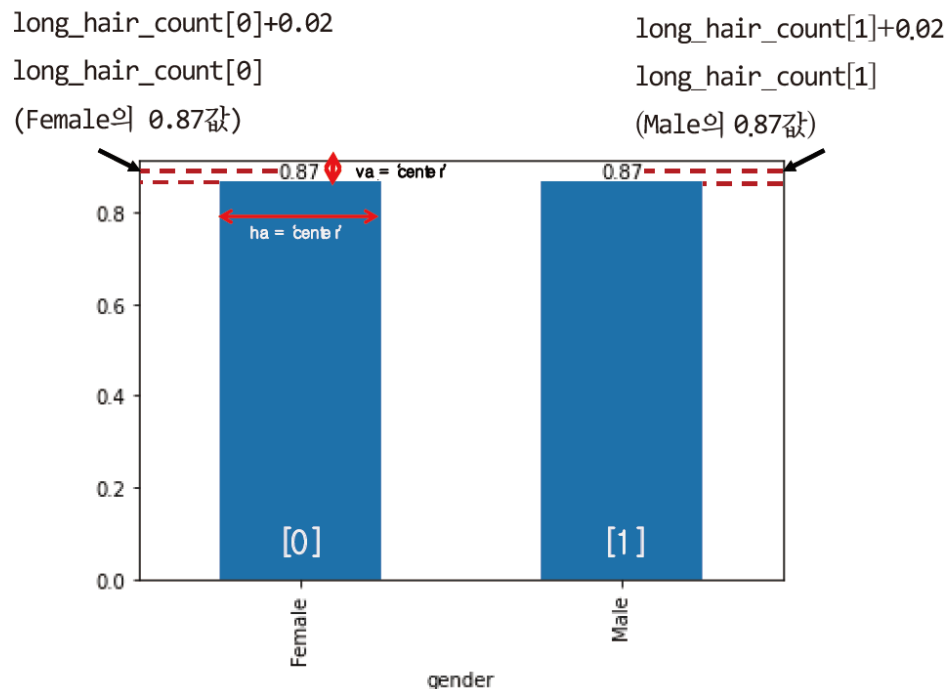
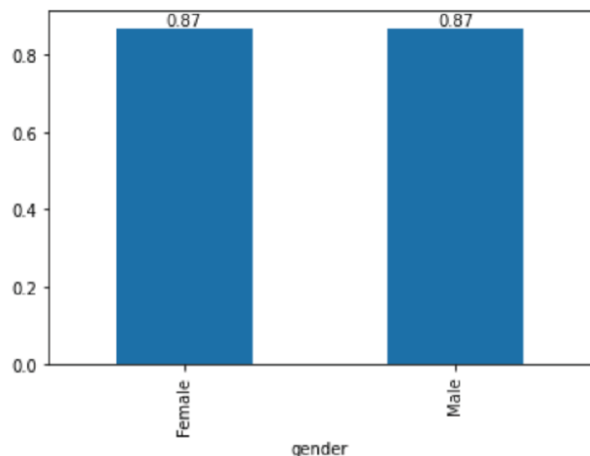
여성 머리카락
길이 평균값

y축 방향으로 가운데
(막대 윗부분만 해당)

x축 방향으로
막대 가운데

```
plt.text(0, long_hair_count[0] + 0.02, long_hair_count[0], va = 'center', ha = 'center')
plt.text(1, long_hair_count[1] + 0.02, long_hair_count[1], va = 'center', ha = 'center')
```

데이터 탐색 및 전처리하기



x축 값: 막대그래프 순서
0은 제일 왼쪽 막대

y축 값: 여성 머리카락
길이의 평균값

y축 값에서
0.02 위의 값

여성 머리카락
길이 평균값

y축 방향으로 가운데
(막대 윗부분만 해당)

x축 방향으로
막대 가운데

```
plt.text(0, long_hair_count[0] + 0.02, long_hair_count[0], va = 'center', ha = 'center')  
plt.text(1, long_hair_count[1] + 0.02, long_hair_count[1], va = 'center', ha = 'center')
```

성별간 머리카락 길이 확인하기

gender	long_hair
Female	0.87
Male	0.87

← long_hair_count[0]

← long_hair_count[1]

- long_hair_count에는 오른쪽 표의 내용이 저장
- 성별의 각 값은 long_hair_count[0]과 long_hair_count[1]로 사용 가능

상관관계 분석하기

성별(gender) 속성과 관련이 깊은 속성이라면 남녀를 예측하는 데 영향을 미치는 속성

남녀를 분류하는 데 어느 속성이 성별과 관련이 깊은지 확인하기 위하여 상관관계(correlation) 분석 실시

상관관계 되짚어보기

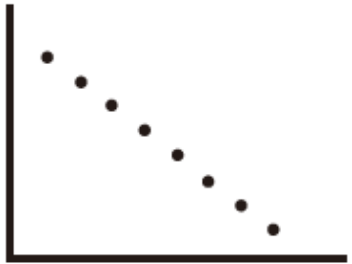
상관관계 분석

두 속성 간에 관련성을 직선으로 표현하고 이 직선에 얼마나 밀집되어 있는가(관련성이 높은가)를 $-1.0 \sim 1.0$ 사이의 실숫값 r 로 설명하는 분석 방법

- 양수는 양의 상관관계, 음수는 음의 상관관계를 의미
- 절대값 1(11)에 가까울수록 두 속성 간의 관련성이 높음.
- 0에 가까울수록 두 속성 간의 관련성이 낮음.

상관관계 되짚어보기

r 값의 범위에 따른 상관관계의 정도를 그래프로 표현



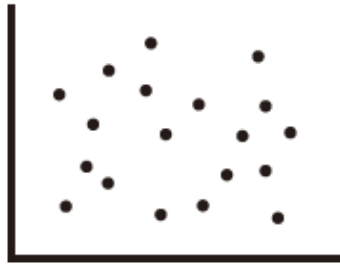
$$r = -1$$

음의 상관관계가
강하다.



$$-1 < r < 0$$

음의 상관관계가
있기는 하다.



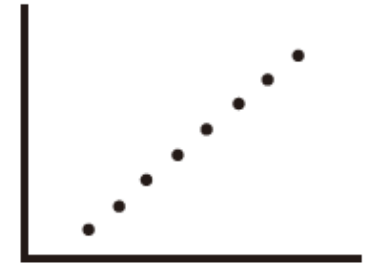
$$r = 0$$

상관관계가
없다.



$$0 < r < 1$$

양의 상관관계가
있기는 하다.



$$r = 1$$

양의 상관관계가
강하다.

상관관계 되짚어보기



$r = -1$



$-1 < r < 0$



$r = 0$



$0 < r < 1$



$r = 1$

상관관계 r 의 절댓값	상관관계의 정도
0.0~0.2	상관관계가 거의 없다.
0.2~0.4	상관관계가 낮다.
0.4~0.6	상관관계가 있다.
0.6~0.8	상관관계가 높다.
0.8~1.0	상관관계가 매우 높다.

상관관계 되짚어보기

상관관계를 분석하기 위해서는 속성값들이 모두 **수치형 데이터**

'Female', 'Male'과 같은 문자열 값을 갖는 성별(gender) 속성은 **범주형 데이터**이므로 상관관계 분석을 위해 수치형으로 변환

나머지 속성들은 이미 정수형(int), 실수형(float)인 수치형 데이터로 변환하지 않음.



범주형 데이터를 수치형 데이터로 변환하는 작업은
원-핫 인코딩 활용

원-핫 인코딩 방법

원-핫 인코딩

범주의 개수만큼 속성을 만들고 범주마다
0이나 1을 입력하는 방법

1개의 속성만 1로 표기하고 나머지 속성에는 0을 표기한다고 해서
'원-핫(one-hot)'이라는 이름이 붙음.

원-핫 인코딩 방법

법주의 고유값을 속성명으로 하는 새로운 속성 생성
해당하는 법주에 1, 아닌 법주에 0 표기

PET		PET_dog	PET_cat	PET_rabbit
dog		1	0	0
cat	→	0	1	0
rabbit		0	0	1
dog		1	0	0

원-핫 인코딩 방법

- 표현 형식이 간단한 판다스 라이브러리의 **get_dummies()** 사용
- 특정 속성을 인코딩하기 위해 데이터프레임에서 columns를 사용하여 인코딩할 속성명 제시

```
판다스 객체.get_dummies(데이터프레임 객체, columns = ['인코딩할 속성명'], drop_first = True)  
# columns를 명시하지 않으면 데이터프레임의 모든 범주형 데이터를 원-핫 인코딩 처리함.  
# 특정 속성만 원-핫 인코딩하려면 columns에 속성명 제시(생략 가능)  
# drop_first = True를 추가하면 원-핫 인코딩을 수행하여 생성된 새로운 속성 중 첫 번째 속성  
삭제(생략 가능)
```

원-핫 인코딩 적용

이목구비 데이터 셋(df)에서 성별(gender) 속성에 대하여
원-핫 인코딩을 수행하고
그 결과를 df_onehot이라는 이름으로 사용



```
1 df_onehot = pd.get_dummies(df, columns = ['gender'])  
2 df_onehot
```

원-핫 인코딩 적용



	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender_Female	gender_Male
0	1	11.8	6.1	1	0	1	1	0	1
1	0	14.0	5.4	0	0	1	0	1	0
2	0	11.8	6.3	1	1	1	1	0	1
3	0	14.4	6.1	0	1	1	1	0	1
4	1	13.5	5.9	0	0	0	0	1	0
...
4996	1	13.6	5.1	0	0	0	0	1	0
4997	1	11.9	5.4	0	0	0	0	1	0
4998	1	12.9	5.7	0	0	0	0	1	0
4999	1	13.2	6.2	0	0	0	0	1	0
5000	1	15.4	5.4	1	1	1	1	0	1

5001 rows × 9 columns

원-핫 인코딩 적용

- **성별(gender) 속성이 원-핫 인코딩을 통해 범주 (Female, Male)의 개수만큼 2개 속성(gender_Female, gender_Male)이 새롭게 생성되었고, 해당하는 범주에만 1이 체크된 것 확인**
- **이처럼 `get_dummies()`를 통해 원-핫 인코딩된 후에는 기존의 범주형 데이터 속성(gender)은 삭제**

원-핫 인코딩 적용

- gender_Female의 값이 1일 때, gender_Male의 값은 항상 0이고, gender_Female의 값이 0일 때, gender_Male의 값은 항상 1
- 2개의 속성 중 1개의 속성만 사용하면 되므로, **drop_first = True**를 추가하여 새로운 속성 중 첫 번째 속성의 gender_Female 속성 삭제

```
1 df_onehot = pd.get_dummies(df, columns = ['gender'], drop_first = True)
2 df_onehot
```


원-핫 인코딩 적용

gender_Female 속성이 삭제되고 gender_Male 속성만 남은 것 확인



	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender_Male
0	1	11.8	6.1	1	0	1	1	1
1	0	14.0	5.4	0	0	1	0	0
2	0	11.8	6.3	1	1	1	1	1
3	0	14.4	6.1	0	1	1	1	1
4	1	13.5	5.9	0	0	0	0	0
...
4996	1	13.6	5.1	0	0	0	0	0
4997	1	11.9	5.4	0	0	0	0	0
4998	1	12.9	5.7	0	0	0	0	0
4999	1	13.2	6.2	0	0	0	0	0
5000	1	15.4	5.4	1	1	1	1	1

5001 rows × 8 columns

상관관계 분석

- 데이터프레임의 속성 간에 관련 정도를 나타내는 상관관계를 분석하기 위해서는 **corr()** 메소드를 사용

데이터프레임 객체.corr()

상관관계(correlation) 분석

상관관계 분석



```
1 corr = df_onehot.corr( ).round(2)
2 corr
```



	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender_Male
long_hair	1.00	-0.01	-0.01	0.01	0.01	0.01	-0.03	-0.01
forehead_width_cm	-0.01	1.00	0.26	0.57	0.56	0.56	0.25	0.33
forehead_height_cm	-0.01	0.26	1.00	0.21	0.19	0.21	0.22	0.28
nose_wide	0.01	0.57	0.21	1.00	0.57	0.56	0.57	0.76
nose_long	0.01	0.26	0.19	0.57	1.00	0.56	0.56	0.74
lips_thin	0.01	0.26	0.21	0.56	0.56	1.00	0.57	0.74
distance_nose_to_lip_long	-0.03	0.25	0.22	0.57	0.56	0.57	1.00	0.75
gender_Male	-0.01	0.33	0.28	0.76	0.74	0.74	0.75	1.00

성별(gender) 속성을 원-핫 인코딩한 결과인 df_onehot 데이터프레임에 대하여 상관관계 분석 수행, 상관관계 값(상관계수)을 소수점 아래 둘째 자리까지 반올림한 결과를 corr로 설정

상관관계 분석

- gender_Male 속성의 상관관계 값이 **실수형 수치값으로 나타난 것 확인**
- 속성별로 상관관계를 파악하려면 일일이 **수치를 비교해야 하는 불편함 발생**

히트맵으로
표현

히트맵

색(heat, 열)으로 배열(map, 지도)을 나타내는
그래프

히트맵을 사용하면 배열에 두 속성 간의 관계가
색으로 표현되어 한눈에 파악하기 쉬움.

히트맵으로
표현

heatmap() 메소드 내에 **annot = True** 속성을 설정하면
히트맵 각 셀에 해당하는 값 표기하기

시본 객체.heatmap(데이터프레임명, annot = True)

annot = True는 히트맵의 각 셀에 해당하는 값을 셀에 표기

히트맵으로
표현

히트맵은 시본(seaborn) 라이브러리를 사용하여 쉽게
구현 가능



```
1 import seaborn as sns  
2 sns.heatmap(corr, annot = True) # 히트맵에 값(수치값) 포함하여 그리기
```

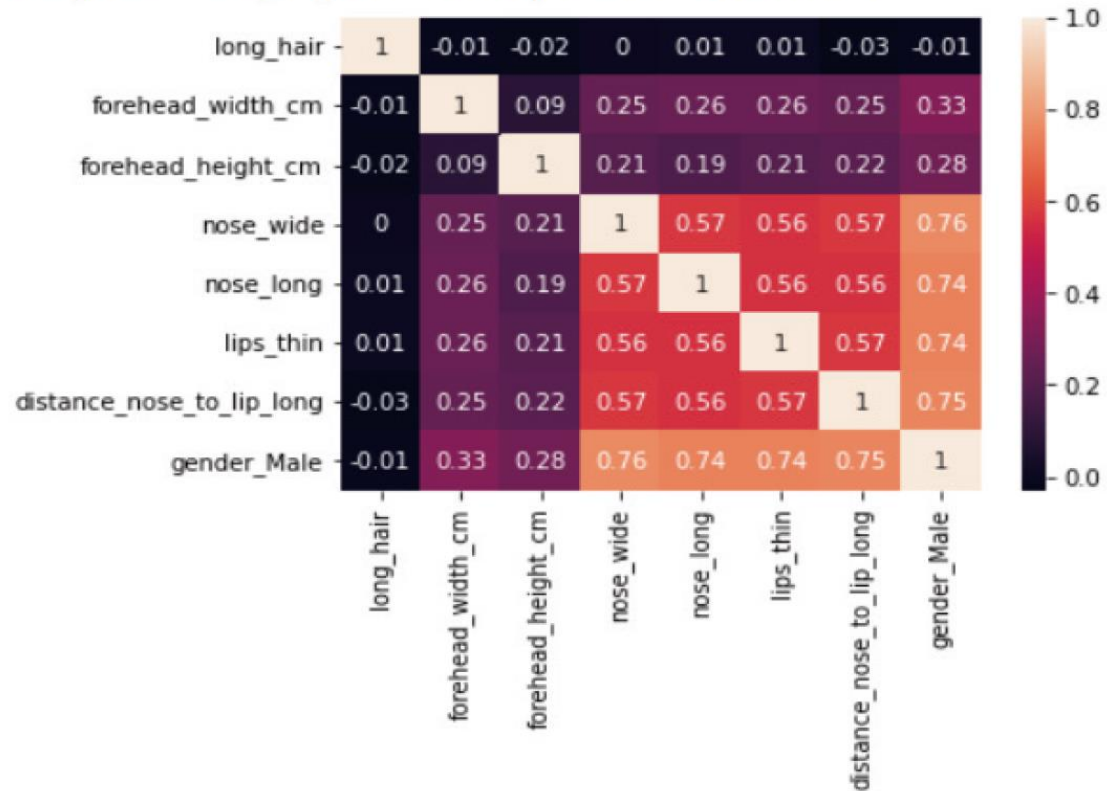
히트맵으로 표현



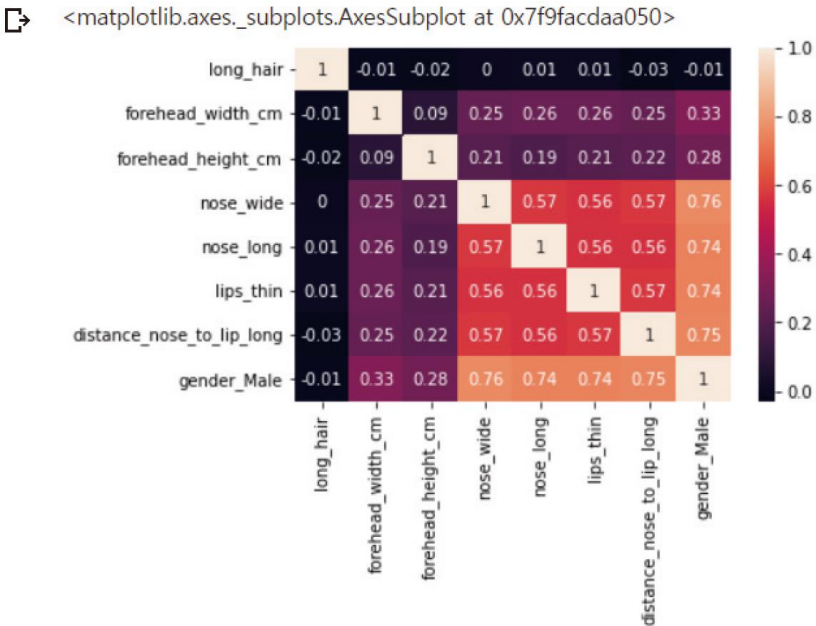
```
1 import seaborn as sns  
2 sns.heatmap(corr, annot = True) # 히트맵에 값(수치값) 포함하여 그리기
```



<matplotlib.axes._subplots.AxesSubplot at 0x7f9facdaa050>



히트맵으로 표현



- 우측의 색깔 막대(color bar)에서 **밝은 색상일수록 속성 간의 관련성이 깊은 것을 확인 가능**
- 머리카락 길이(long_hair)는 상관관계가 거의 없고,
- 이마 너비(forehead_width_cm), 이마 높이(forehead_height_cm)는 상관관계가 낮으며,
- 코 너비(nose_wide), 코 길이(nose_long), 입술 두께(lips_thin), 인중 길이(distance_nose_to_lip_long)는 상관관계가 높음.

히트맵으로 표현

높은 상관관계를 갖는 코너비(nose_wide),
코길이(nose_long), 입술 두께(lips_thin),
인중 길이(distance_nose_to_lip_long)를 선택



단, 처리 시간이나 비용을 고려하기보다 정확도를 조금이라도 높이기 위해 낮은 상관관계를 갖는
이마 너비(forehead_width_cm)와
이마 높이(forehead_height_cm) 추가

**독립 변수와
종속 변수 구분**

- **독립 변수:** 영향을 미치는 속성
- **종속 변수:** 독립 변수의 영향을 받는 속성

독립 변수	종속 변수
이마 너비(forehead_width_cm), 이마 높이 (forehead_height_cm), 코 너비(nose_wide), 코 길이 (nose_long), 입술 두께(lips_thin), 인중 길이(distance_nose_to_lip_long)	성별(gender_Male)

독립 변수 객체 = 데이터프레임 객체[[속성명1, 속성명2, ...]]

종속 변수 객체 = 데이터프레임 객체[속성명]

독립 변수와 종속 변수 구분



```
1 x = df_onehot[['forehead_width_cm', 'forehead_height_cm',  
2               'nose_wide', 'nose_long', 'lips_thin',  
3               'distance_nose_to_lip_long']] # 독립 변수  
4 y = df_onehot['gender_Male'] # 종속 변수  
5 display(x)  
6 display(y)
```



	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	0	1
0	11.8	6.1	1	0	1	1	1	0
1	14.0	5.4	0	0	1	0	2	1
2	11.8	6.3	1	1	1	1	3	1
3	14.4	6.1	0	1	1	1	4	0
4	13.5	5.9	0	0	0	0
...	4996	0
4996	13.6	5.1	0	0	0	0	4997	0
4997	11.9	5.4	0	0	0	0	4998	0
4998	12.9	5.7	0	0	0	0	4999	0
4999	13.2	6.2	0	0	0	0	5000	1
5000	15.4	5.4	1	1	1	1	Name: gender_Male, Length: 5001, dtype: uint8	

5001 rows x 6 columns

`df_onehot`**[속성명 리스트]**

원-핫 인코딩을 수행한 데이터프레임
`df_onehot`의 속성이라는 것을 나타내는 대괄호

안쪽의 대괄호는 여러 개의 속성들로
구성된 리스트를 나타내는 대괄호

#대괄호를 한 번 만 사용하지 않도록 유의

- **훈련데이터
테스트 데이터
나누기**
- 인공지능 모델을 학습시키기 위해 이목구비 데이터 셋을 **훈련 데이터와 테스트 데이터로 구분**
- 일반적으로 훈련 데이터와 테스트 데이터는 **7 : 3 비율로 나누기**

훈련 데이터 테스트 데이터 나누기

- **사이킷런(sklearn) 라이브러리**의 모듈을 사용하여 전체 데이터를 훈련 데이터와 테스트 데이터로 쉽게 나누기
- 사이킷런의 `model_selection` 모듈에서 **`train_test_split`**을 불러오는 방법 사용

```
1 from sklearn.model_selection import train_test_split
```

훈련 데이터 테스트 데이터 나누기

- 훈련 데이터와 테스트 데이터를 나누기 위해 **train_test_split()** 사용
- train_test_split()은 실행될 때마다 훈련 데이터와 테스트 데이터를 새롭게 구성

학습에 사용할 독립 변수 객체, 테스트에 사용할 독립 변수 객체, 학습에 사용할 종속 변수 객체

테스트에 사용할 종속 변수 객체 = **train_test_split** (독립 변수, 종속 변수, **test_size** = 테스트 데이터 비율)

독립 변수: 영향을 미치는 속성, 종속 변수: 영향을 받는 속성

테스트 데이터 비율: 실수 형태로 나타내며, 예를 들어 0.3을 입력한다면

전체 데이터의 30%를 테스트 데이터로 배정하겠다는 의미, test_size를 생략하면 기본 값 0.25로 설정

훈련 데이터 테스트 데이터 나누기

독립 변수

종속 변수



```
1 X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.3)
```

훈련 데이터(70%)		테스트 데이터(30%)	
독립 변수 (X_train)	종속 변수 (Y_train)	독립 변수 (X_test)	종속 변수 (Y_test)

테스트 데이터의 비율을 0.3(30%)으로 설정하였으므로 훈련 데이터는 70%

사이킷런(scikit-learn)

- Python 라이브러리 중에서 머신러닝할 때 가장 많이 사용
- 1~2줄의 코드만으로도 머신러닝을 실행시킬 수 있을 만큼 사용이 편리
- 사이킷런을 사용하면 훈련 데이터와 테스트 데이터를 나누거나, 모델을 선택하여 학습시키는 등 다양한 작업 가능

- 분류를 위한 머신러닝 모델에는 다양한 종류가 있으며 각 모델마다 데이터를 학습하는 방식과 분류 정확도가 조금씩 다름.
- 성별을 분류하기 위하여 **로지스틱 회귀(Logistic Regression)** 모델 사용

1. 사이킷런을 이용하여 다음과 같이 한 줄로 로지스틱 회귀 모델을 불러오기

```
1 from sklearn.linear_model import LogisticRegression
```

2. LogisticRegression() 함수를 통해 로지스틱 회귀 모델 생성

생성한 모델을 LR_model이라는 이름으로 사용

```
1 LR_model = LogisticRegression( ) # Logistic Regression 모델 생성
```

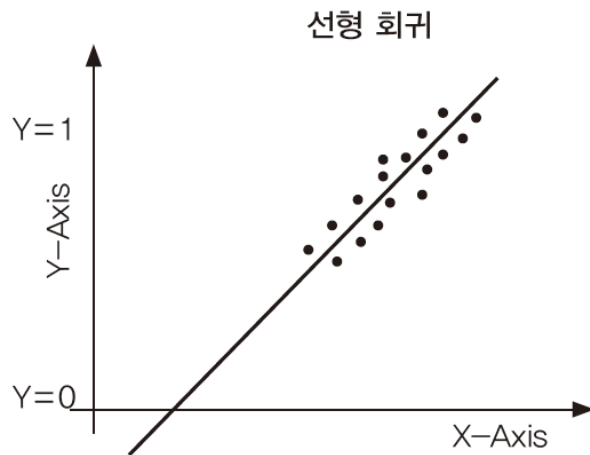
현재 생성된 모델 LR_model은 사람에 비유하면 아직 학습하지 않은 뇌구조에 해당

로지스틱 회귀(Logistic Regression) 모델(분류 모델)

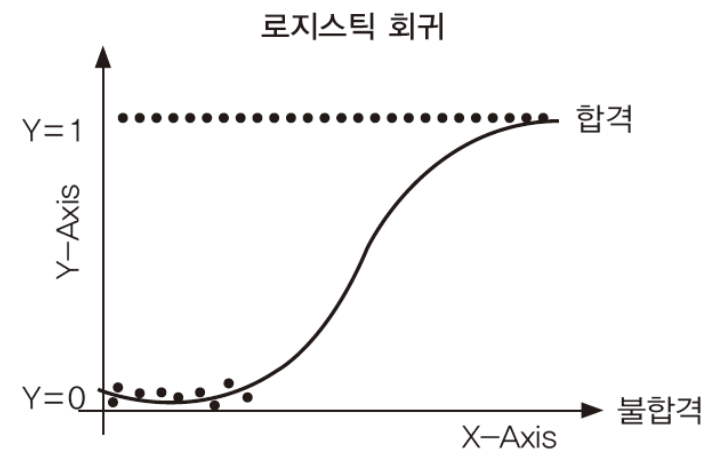
- 결괏값을 예측하는 회귀식에 로짓(logit, 로그 함수)을 이용하여 결괏값을 $0 \sim 1$ 사이의 범위로 출력하여 분류하는 머신러닝 모델
- 합격·불합격, 남·녀, 생존·사망 등과 같은 범주형 데이터에 대한 범주 예측 가능

보충

선형 회귀 vs. 로지스틱 회귀



종속 변수가 학습 시간에 따른 성적과 같이 **수치형 데이터일 때 사용**



종속 변수가 선형 회귀선에 로그 함수를 적용하여 0과 1로 분류할 수 있게 되어 있으며 **범주형 데이터일 때 사용**

머신러닝에서 모델 학습을 할 때 모델이 학습할 훈련 데이터(독립 변수)와
훈련 데이터의 레이블(종속 변수) 설정하기

모델 객체.fit(훈련 데이터, 훈련 데이터의 레이블)

훈련 데이터와 훈련 데이터의 레이블은 값만 학습하기 위해 '.values'와 함께 제시

- X_train을 훈련 데이터로, Y_train을 훈련 데이터의 레이블로 설정
- **fit()** 함수를 사용하면 쉽게 학습 가능



```
1 LR_model.fit(X_train.values, Y_train.values)
```


학습에서 속성명 사용하지 않기

- 모델 학습에 값만 사용하는 경우에는 훈련 데이터 X_train과 훈련 데이터의 레이블 Y_train에 '.values'라는 속성이 함께 제시
- '.values'를 제시하지 않으면, 이후 예측 과정에서 다음과 같은 경고 문구가 출력되므로, 테스트 데이터에도 적용

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names "X does not have valid feature names, but"
```

학습에서 속성명 사용하지 않기

이러한 문구가 출력되는 이유는 학습에 사용한 데이터는 속성명과 값으로 구성된 데이터프레임이고, 이후 모델을 평가(predict())한 결과로 출력되는 형태는 속성명(feature name)이 없는 배열(array)이기 때문입니다.

따라서 '.values'를 사용하면 모델이 속성명을 제외한 값만 학습하기 때문에 이러한 경고 문구가 나타나지 않습니다.

학습에서 속성명 사용하지 않기

훈련 데이터 입력							
	속성명 1	속성명 2	속성명	속성명 5	속성명 6	속성명 7	
0							
1							
2			값				
⋮							
4999							
5000							

속성명과 값으로 구성된 데이터프레임

학습

모델 평가 결과

값만 학습

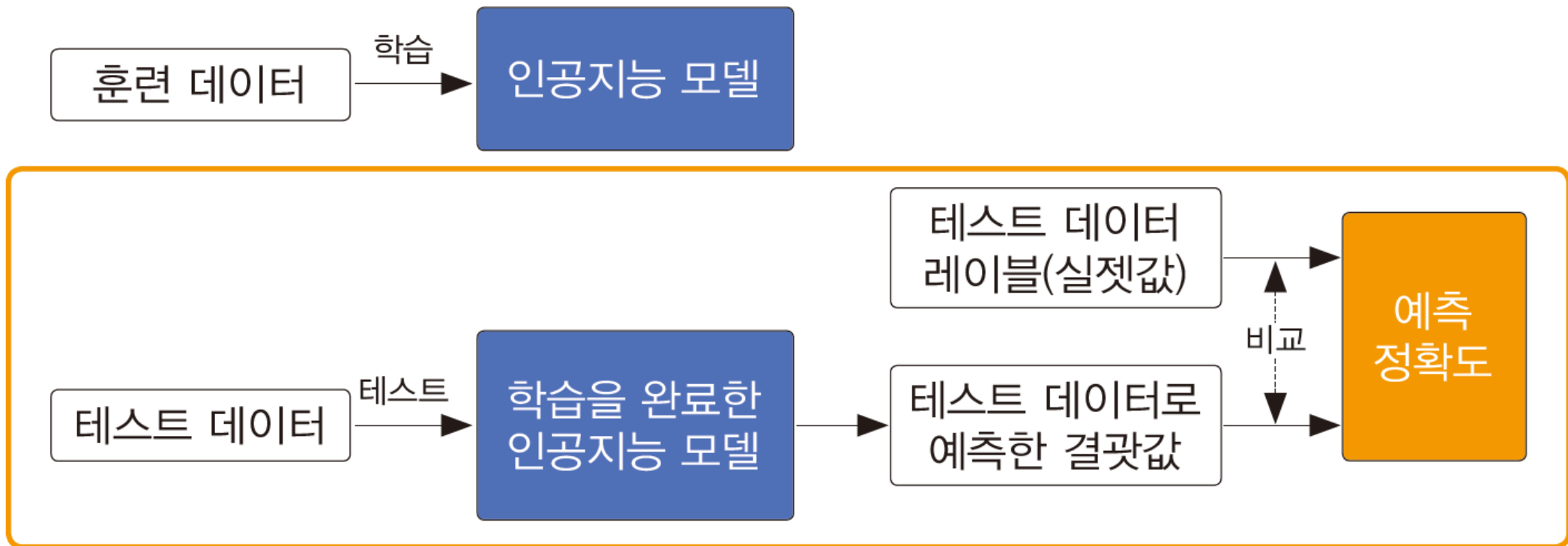


```
array([1, 1, 1,..., 0, 1, 1], dtype = uint8)
```



속성명이 없는 배열

모델을 생성하여 학습시키고 난 후 학습이 제대로 되었는지 평가



- 모델 평가
- 모델이 학습을 완료하고 나면 학습이 잘되었는지 평가하기

예측

테스트 데이터를 이용하여 학습이
잘되었는지 평가

형식: `predict(테스트 데이터)`

정확도 산출

테스트 데이터의 레이블(실제값)과
예측한 결과값을 비교하여 정확도
산출

형식: `accuracy_score(테스트 데이터
의 레이블, 테스트 데이터로 예측한
결과값)`

테스트 데이터의
예측값

학습을 완료한 모델(LR_model)이 학습을 잘했는지 평가하기
위해 **predict()** 메소드를 사용하여 테스트 데이터(X_test)로
성별을 예측하기

모델 객체.predict(테스트 데이터)

테스트 데이터의 예측값

학습을 완료한 모델이 예측한 결과를 `predict_results`라는 이름으로 설정하기



```
1 predict_results = LR_model.predict(X_test.values)
2 predict_results
```



```
array([1, 1, 1, ..., 0, 1, 1], dtype = uint8)
```

'gender_Female'(0: 남성, 1: 여성) 속성을 삭제하고 'gender_Male'(0: 여성, 1: 남성) 속성을 종속 변수로 설정한 것을 기억하도록 합니다.

테스트 데이터의 예측값

데이터 나누기를 랜덤으로 실시하기 때문에 예측 결과가 다르게 출력될 수 있어요.

LR_model 모델이 테스트 데이터(X_test) 중 **첫 번째 데이터를 남성(Male)으로 예측한 것을 알 수** 있습니다.

정확도 산출

- 실젯값의 테스트 데이터 레이블(`Y_test`)과 모델이 테스트 데이터를 예측한 값을 비교하여 정확도 산출
- 정확도를 산출하기 위해 `accuracy_score()` 메소드를 사용하여 모델이 예측한 값과 실젯값이 얼마나 일치하는지 평가하여 0.0~1.0 사이의 실숫값으로 출력

`accuracy_score(테스트 데이터의 레이블, 테스트 데이터 예측값)`

테스트 데이터의 레이블과 테스트 데이터의 예측값을 비교하여 모델의 정확도 산출

정확도 산출

사이킷런의 **metrics** 모듈을 사용하여 정확도 산출하기



```
1 from sklearn.metrics import accuracy_score  
2 accuracy_score(Y_test, predict_results)
```



```
0.9713524317121919
```

실행 결과는 매번 달라질 수 있어요.

0.9713...으로, 이는 모델이 약 97%의 정확도를 보인다고 해석할 수 있습니다.

- 성별 예측
- 새로운 데이터를 입력하여 **predict_proba()** 메소드를 사용하면 예측한 결과가 남성이 아닐 확률과 남성일 확률을 실숫값으로 출력하기

모델 객체.predict_proba(새로운 데이터)

새로운 데이터에 대한 예측 결과가 레이블에 속할 확률 출력

성별 예측

- 넘파이 라이브러리를 불러와 새로운 데이터를 배열로 제시
- 독립 변수 6개의 속성값을 각각 16.3, 6.0, 0, 0, 0, 0으로 생성하여 `choi`라는 이름으로 설정하기
- 새로 입력한 데이터를 모델이 어떻게 분류했는지 확인하기

```
1 import numpy as np # 넘파이 라이브러리 호출
2 choi = np.array([[16.3, 6.0, 0, 0, 0, 0]]) # 독립 변수에 속하는 속성들의 새로운 값 입력
3 LR_model.predict_proba(choi)
```

```
array([[0.99012047, 0.00987953]])
```

성별 예측

choi가 남성이 아닐 확률(여성일 확률)은 약 0.99,
남성일 확률은 약 0.01인 것을 확인할 수 있습니다.
이는 로지스틱 회귀 모델이 choi를 여성으로
분류했다는 것을 의미합니다.

문제 정의하기

문제 상황 이해하기

해결할 문제: 이목구비 크기로 성별 분류

데이터 불러오기

캐글(kaggle)에서 이목구비 데이터 셋 불러오기

데이터 탐색 및 전처리하기

데이터 속성 살펴보기(데이터 유형, 결측치, 속성명 등)

데이터 시각화하기(성별 분류에 영향을 미치는 속성 탐색)

범주형 속성값을 수치형으로 변환하기(원-핫 인코딩)

속성 간 상관관계 분석하기(속성 간의 관계를 히트맵으로 시각화)

훈련 데이터와 테스트 데이터 나누기



모델 생성하기

로지스틱 회귀 모델 생성하기



모델 학습하기

모델 학습하기(훈련 데이터 사용)



모델 평가 및 예측하기

성능 평가하기(테스트 데이터 사용)
성별 예측하기(새로운 데이터 사용)

1. 원-핫 인코딩

범주형 데이터를 수치형으로 변환하는 작업으로 범주의 고윳값 개수만큼 속성을 만들고 범주마다 0 또는 1의 값을 입력하는 방법입니다. 이때 해당되는 범주에는 1이, 해당되지 않는 범주에는 0이 표기됩니다. 원-핫 인코딩을 수행하고 나면 결과 예측에 영향을 미치는 속성을 찾기 위한 상관관계 분석 과정을 수행합니다.

2. 독립 변수와 종속 변수

결과 예측에 영향을 미치는 속성은 독립 변수, 독립 변수의 영향을 받는 속성은 종속 변수입니다.

3. 로지스틱 회귀 모델

결괏값을 예측하는 회귀식에 로짓(logit, 로그 함수)을 이용하여 결괏값을 0~1 사이의 범위로 출력하여 분류하는 머신러닝 모델입니다.

4. 분류 모델 평가 지표의 정확도

훈련 데이터로 학습시킨 인공지능 모델이 테스트 데이터(실제값과 예측값 비교)를 입력 받았을 때 얼마나 정확하게 분류하는지를 나타내는 값입니다.

5. 사이킷런 라이브러리

훈련 데이터와 테스트 데이터를 나누거나 모델을 선택하여 학습시키는 데 사용하는 라이브러리입니다.

'머신러닝 문제 해결'의 '성별을 분류하다' 활동에서 분류 모델인 로지스틱 회귀 모델을 사용하여 남녀의 성별을 분류해 보았다.

우선 우리는 문제 해결을 위해 이목구비 데이터 셋(gender_classification_v7.csv)의 속성과 속성 간 상관관계를 살펴보면서 성별 분류에 영향을 미치는 속성(독립 변수)과 영향을 받는 속성(종속 변수)이 무엇인지 확인하였다.

이 과정에서 상관관계 분석을 위해 문자열 값을 갖는 범주형 데이터를 원-핫 인코딩하여 수치형 데이터로 변환하고, 히트맵으로 시각화하여 종속 변수와 상관관계가 있는 속성이 무엇인지 쉽게 파악하였다.