

Python Advanced

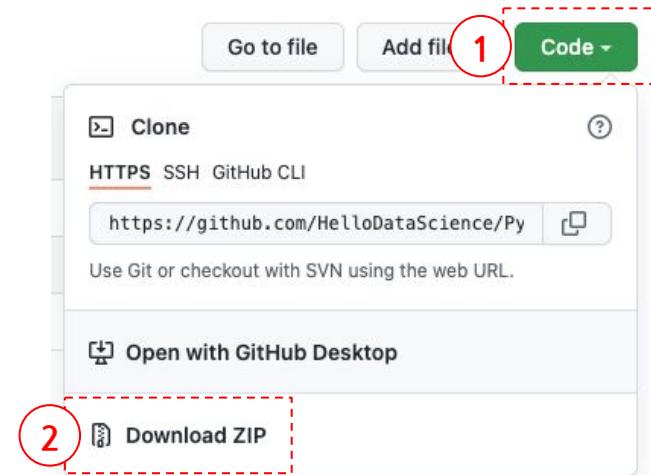
Statistical Analysis

강의 내용

- 탐색적 데이터 분석
- 기술통계 분석
- 확률분포의 이해
- 가설검정의 이해
- 데이터 분석 개요
- 선형 회귀분석
- 로지스틱 회귀분석

실습 데이터셋 내려받기

- 크롬에서 깃허브 저장소로 접속합니다. [주의] 크롬 아니면 에러 발생 가능!
 - URL: <https://github.com>HelloDataScience/PythonAdvanced>
- 초록색 Code와 Download ZIP을 차례대로 클릭하면 zip 파일을 다운로드 폴더에 저장합니다.
- zip 파일을 적당한 위치(예: 문서 폴더)에서 압축을 풀고 code와 data 폴더를 확인합니다.
 - code: Jupyter Notebook 파일을 포함하는 폴더입니다.
 - data: 실습 데이터 파일을 포함하는 폴더입니다.



탐색적 데이터 분석

탐색적 데이터 분석

- 데이터 분석 모델링에 앞서 탐색적 데이터 분석 Exploratory Data Analysis 을 실행함으로써 분석 데이터셋에 대한 이해도를 높일 수 있습니다.
- 탐색적 데이터 분석 과정에서 다양한 기술통계량을 계산하고 그래프로 데이터의 분포와 관계를 확인합니다.
- 결측값 NA 과 이상치 outlier 를 탐지하고 처리함으로써 데이터셋을 전처리하는 것 또한 탐색적 데이터 분석을 수행하는 목적 중 하나입니다.
 - 결측값은 입력된 값이 없는 상태입니다. 결측값을 대체하거나 삭제해야 합니다.
 - 이상치는 중심에서 멀리 떨어져 있는 값인데 전체 데이터의 관계를 왜곡할 수 있으므로 삭제하는 것이 좋습니다.

실습 데이터셋 소개

- 중고 자동차의 가격과 다양한 특성을 포함하는 텍스트 데이터입니다.
 - Price: 중고차 가격(달러)
 - Age: 중고차 나이(개월수)
 - KM: 주행거리(km)
 - FuelType: 연료의 종류(3종류)
 - HP: 마력
 - MetColor: 차량 색상(0/1)
 - Automatic: 미션의 종류(0/1)
 - CC: 엔진 배기량의 크기
 - Doors: 차량 문의 개수
 - Weight: 차량 무게(kg)

관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib # [참고] 라이브러리를 콤마로 나열할 수 있지만 가독성 측면에는 좋지 않습니다.
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3 # Jupyter Notebook에서 실수를 출력할 소수점 자리수를 3으로 설정합니다.
```

```
>>> pd.options.display.precision = 3 # pandas 옵션에서 실수를 출력할 소수점 자리수를 3으로 설정합니다.
```

작업 경로 확인 및 변경

- 현재 작업 경로를 확인합니다.

```
>>> os.getcwd()
```

- data 폴더로 작업 경로를 변경합니다.

```
>>> os.chdir(path = '../data')
```

- 현재 작업 경로에 있는 폴더명과 파일명을 출력합니다.

```
>>> os.listdir()
```

실습 데이터셋 준비

- 인터넷에 공유 중인 텍스트 데이터를 읽고 데이터프레임 df를 생성합니다.

```
>>> df = pd.read_csv(filepath_or_buffer = 'https://bit.ly/Used_Cars_Price')
```

- df의 정보를 확인합니다.

```
>>> df.info() # 행 개수, 열 개수, 열이름, 자료형, 열별 결측값 아닌 개수 및 자료형을 차례대로 확인합니다.
```

- df의 처음 5행을 출력합니다.

```
>>> df.head() # [참고] n 매개변수에 전달하는 인수의 기본값은 5입니다.
```

실습 데이터셋 전처리

- 범주형으로 변환할 열이름으로 리스트를 생성합니다.

```
>>> cols = ['MetColor', 'Automatic']
```

- 지정한 변수를 문자형으로 일괄 변환합니다.

```
>>> df[cols] = df[cols].astype(str)
```

- df의 열별 자료형을 확인합니다.

```
>>> df.dtypes # MetColor와 Automatic의 자료형이 object로 바뀌었습니다.
```

실습 데이터셋 전처리(계속)

- 실수 및 정수형 변수의 기술통계량을 확인합니다.

```
>>> df.describe() # 실수 및 정수형 변수의 결측값 아닌 개수, 평균, 표준편차, 최솟값, 사분위수 및 최댓값을 확인합니다.
```

- df를 KM로 오름차순 정렬하고 처음 5행을 출력합니다.

```
>>> df.sort_values(by = ['KM']).head()
```

- df에서 KM가 1보다 큰 행을 선택하고 행이름을 초기화합니다.

```
>>> df = df[df['KM'].gt(1)].reset_index(drop = True)
```

실습 데이터셋 전처리(계속)

- 범주형 변수의 기술통계량을 확인합니다.

```
>>> df.describe(include = object) # 문자형 변수의 결측값 아닌 개수, 중복 제거한 원소 개수, 최빈값  
    및 최빈값의 빈도수를 확인합니다.
```

- 범주형 변수의 범주별 상대도수를 출력합니다.

```
>>> df['FuelType'].value_counts(normalize = True)
```

```
>>> df['MetColor'].value_counts(normalize = True)
```

```
>>> df['Automatic'].value_counts(normalize = True)
```

시각화 설정: 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import seaborn as sns # 고급 시각화 함수를 포함하는 라이브러리입니다.
```

```
>>> import matplotlib.pyplot as plt # 그래프 크기, 제목, 축이름 등을 지정할 때 사용합니다.
```

```
>>> import matplotlib.font_manager as fm # 한글폰트를 지정할 때 사용합니다.
```

- 테스트용 그래프를 그립니다.

```
>>> sns.histplot(data = df, x = 'Price')
```

```
>>> plt.title(label = '중고차 가격 분포'); # [참고] 시각화 코드 마지막에 추가한 세미콜론(;)은  
plt.show() 함수와 같은 기능을 실행합니다.
```

위 코드를 실행하면 한글을 네모로 출력하므로 한글폰트를 설정해야 합니다.

[참고] 한글폰트 외 그래프 크기 및 해상도 등 다양한 그래픽 파라미터를 설정할 수 있습니다.

시각화 설정: 한글폰트명 탐색

- 현재 사용 중인 컴퓨터에 설치한 전체 폰트 파일명을 리스트로 반환합니다.

```
>>> fontList = fm.findSystemFonts(fontext = 'ttf'); fontList
```

- 특정 문자열을 갖는 파일명으로 리스트를 생성합니다.

```
>>> fontPath = [font for font in fontList if 'Gamja' in font]; fontPath
```

- 반복문으로 폰트명을 출력합니다.

```
>>> for font in fontPath:
```

```
    print(fm.FontProperties(fname = font).get_name())
```

반복문을 실행한 결과에서 마음에 드는 폰트명을 선택하고 plt.rc() 함수에 지정합니다.
[참고] rc는 runtime configuration를 의미하며, pyplot을 실행하는 환경을 의미합니다.

시각화 설정: 그래픽 파라미터 설정

- 그래프 크기와 해상도를 설정합니다.

```
>>> plt.rc(group = 'figure', figsize = (4, 4), dpi = 150)
```

- 한글폰트와 글자 크기를 설정합니다.

```
>>> plt.rc(group = 'font', family = 'Gamja Flower', size = 10)
```

- 축에 유니코드 마이너스를 출력하지 않도록 설정합니다.

```
>>> plt.rc(group = 'axes', unicode_minus = False) # [참고] 왼쪽 코드를 설정하지 않으면  
음수 앞에 '‑'를 출력합니다.
```

- 범례에 채우기 색과 테두리 색을 추가합니다. # fc는 facecolor(채우기), ec는 edgecolor(테두리)
관련 매개변수이고 '0'은 검정, '1'은 흰색입니다.

```
>>> plt.rc(group = 'legend', frameon = True, fc = '1', ec = '0')
```

[참고] 그래픽 파라미터 설정 관련 모듈 생성

- 시각화 라이브러리 호출, 스타일시트, 한글폰트 및 그래픽 파라미터를 설정하는 코드를 모듈(py 파일)로 저장하면 필요할 때마다 호출할 수 있으므로 편리합니다.
- Anaconda 메인에서 Text File을 열고, 모듈(py 파일)로 저장할 시각화 설정 관련 코드를 붙여넣습니다.
- 상단 메뉴에서 File → Save Text As를 클릭하고 텍스트 파일을 저장합니다.
 - 파일명을 GraphicSetting.py로 저장합니다. *# [주의] py 파일을 Jupyter Notebook 파일과 같은 폴더에 저장해야 호출할 수 있습니다.*
- 시각화 설정 모듈을 호출합니다.

```
>>> from GraphicSetting import *
```

[참고] Python 파일 탐색 경로 확인

- 모듈(py 파일)을 호출하려면 아래 두 가지 조건 중 하나를 만족해야 합니다.
 - 현재 작업 중인 Jupyter Notebook 파일 경로에 py 파일을 저장했다.
 - Python 파일 탐색 경로 중 한 곳에 py 파일을 저장했다.
- 작업할 Jupyter Notebook 파일 경로가 바뀔 때마다 py 파일을 매번 옮겨야 하므로 첫 번째 조건은 불편합니다. 따라서 두 번째 조건을 따르는 것이 좋습니다.
- Python 파일 탐색 경로를 확인합니다.

```
>>> import sys # 관련 라이브러리를 호출합니다.
```

```
>>> sys.path # Python 파일 탐색 경로를 모두 출력합니다. 많은 경로 중 한 곳(마지막 경로 추천)에 py 파일을  
저장하면 해당 모듈을 항상 호출할 수 있습니다.
```

[참고] 시각화 함수 모듈 제공

- Python 통계 분석 및 머신러닝 강의에서 사용할 시각화 함수를 **HelloDataScience** 모듈(py 파일)로 제공합니다. 주요 함수는 아래와 같습니다.
 - 변수의 관계 확인: `plot_regression`, `plot_box_plot`, `plot_bar_dodge_freq` 등
 - 모형의 성능 평가: `regmetrics`, `plot_roc`, `clfmetrics`, `EpiROC` 등
- 통계 분석 및 머신러닝 관련 시각화 함수 모듈을 호출합니다.

```
>>> import HelloDataScience as hds
```

```
# HelloDataScience 모듈의 의존성 라이브러리를 설치합니다.
```

```
>>> !pip install statsmodels  
>>> !pip install graphviz
```

```
# (아나콘다) pip로 graphviz 라이브러리를 설치할 수 없으면 아래 코드를 실행하세요. 설치 시간이 길니다.
```

```
>>> !conda install python-graphviz
```

목표변수 분포 확인

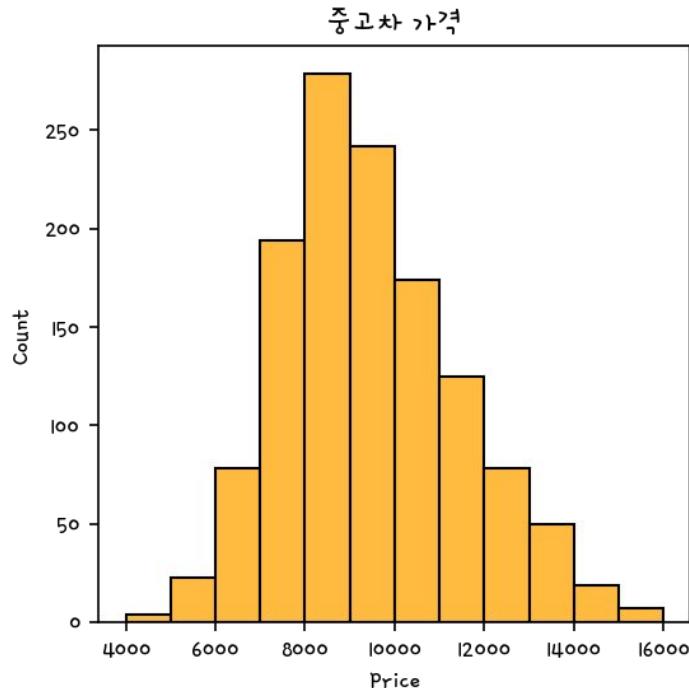
- 목표변수의 최솟값과 최댓값을 확인합니다.

```
>>> df['Price'].describe()[['min', 'max']]
```

- 히스토그램을 그립니다.

```
>>> sns.histplot(data = df, x = 'Price',
                 bins = 12, # 막대 개수와 범위를
                           # 지정합니다.
                 binrange = (4000, 16000),
                 color = 'orange')
```

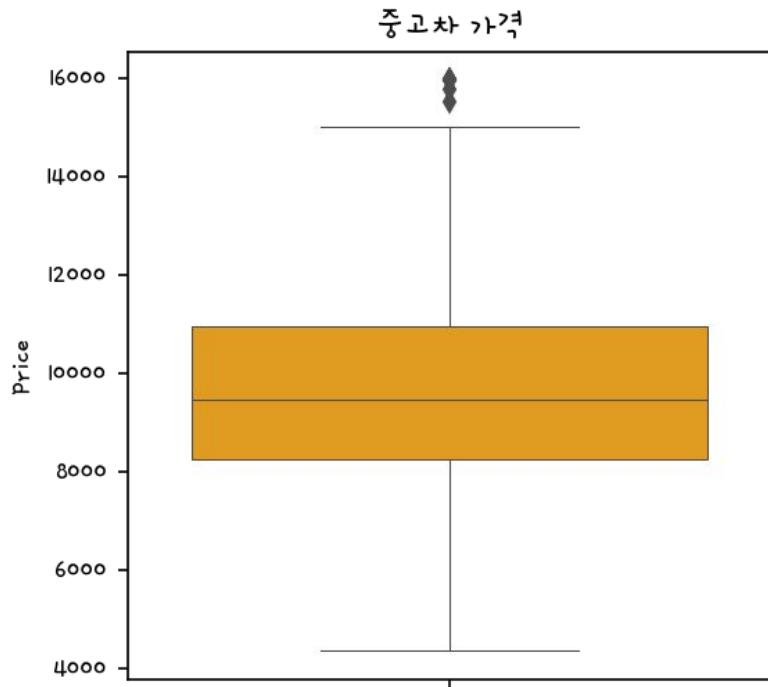
```
>>> plt.title(label = '중고차 가격');
```



목표변수 분포 확인(계속)

- 상자 수염 그림을 그립니다.

```
>>> sns.boxplot(data = df,  
                 y = 'Price',  
                 color = 'orange')  
  
>>> plt.title(label = '중고차 가격');
```



연속형 입력변수와 관계 파악: Age

- Age와 Price의 산점도를 그립니다.

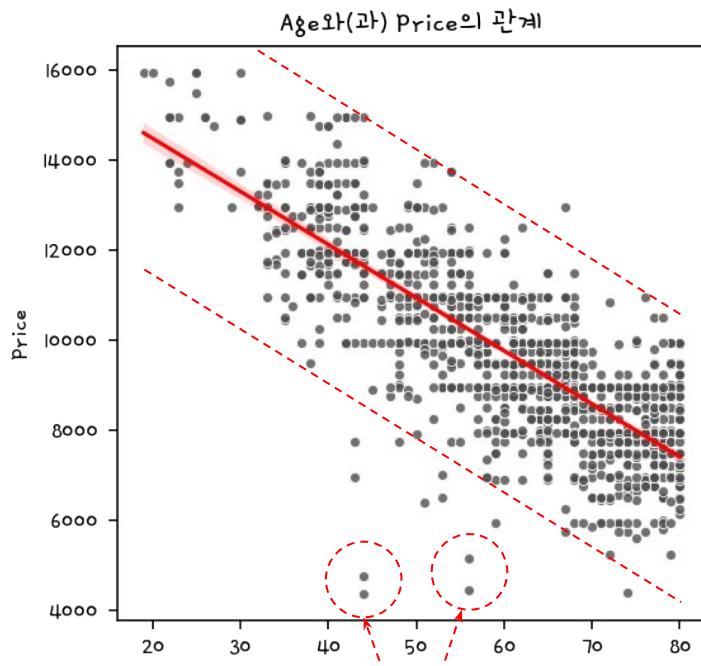
```
>>> hds.plot_regression(
```

```
    data = df,
```

```
    x = 'Age',
```

```
    y = 'Price'
```

```
)
```



이상치로 보이는 관측값이 있습니다.

연속형 입력변수와 관계 파악: KM

- KM와 Price의 산점도를 그립니다.

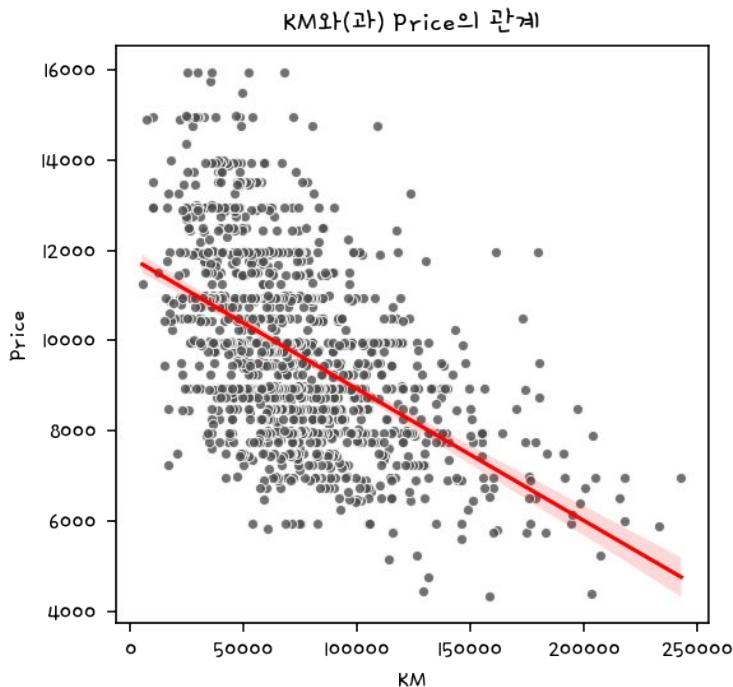
```
>>> hds.plot_regression(
```

```
    data = df,
```

```
    x = 'KM',
```

```
    y = 'Price'
```

```
)
```



연속형 입력변수와 관계 파악: HP

- HP와 Price의 산점도를 그립니다.

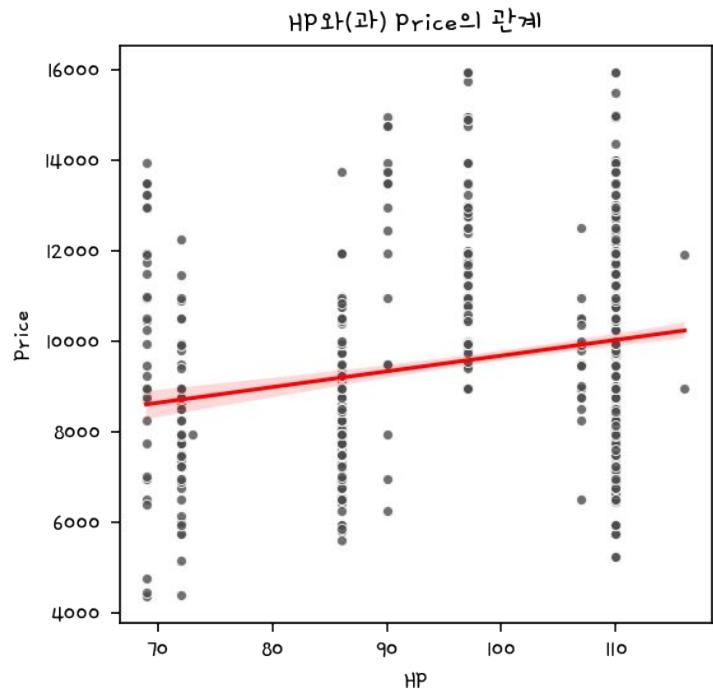
```
>>> hds.plot_regression(
```

```
    data = df,
```

```
    x = 'HP',
```

```
    y = 'Price'
```

```
)
```



연속형 입력변수와 관계 파악: CC

- CC와 Price의 산점도를 그립니다.

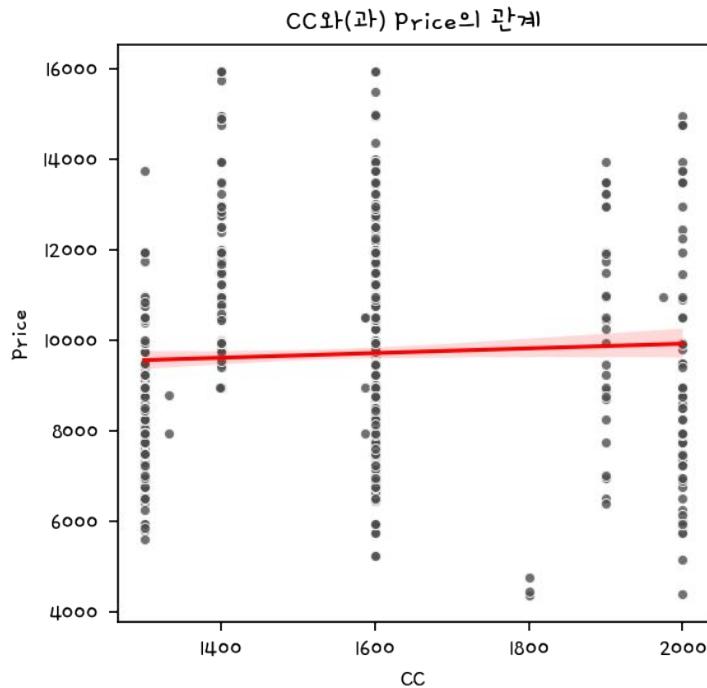
```
>>> hds.plot_regression(
```

```
    data = df,
```

```
    x = 'CC',
```

```
    y = 'Price'
```

```
)
```



연속형 입력변수와 관계 파악: Doors

- Doors와 Price의 산점도를 그립니다.

```
>>> hds.plot_regression(
```

```
    data = df,
```

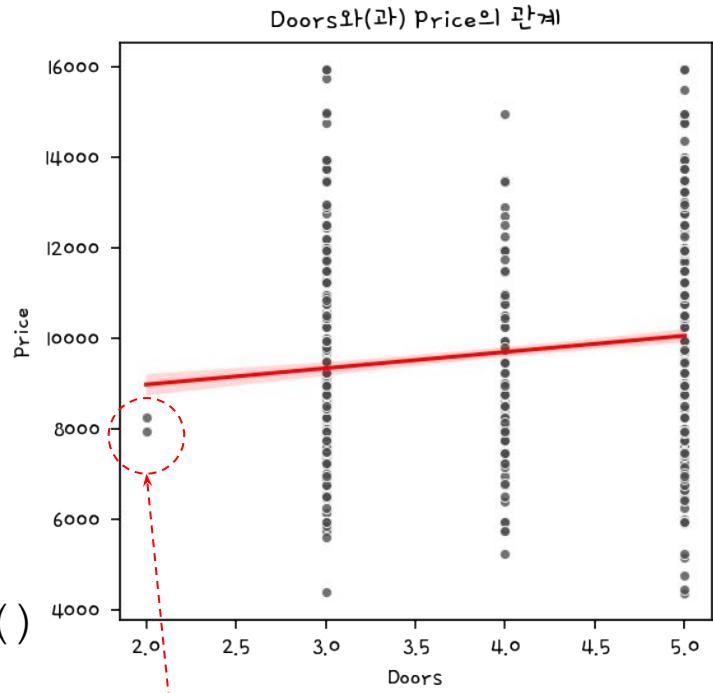
```
    x = 'Doors',
```

```
    y = 'Price'
```

```
)
```

```
>>> df['Doors'].value_counts().sort_index()
```

Doors의 원소별 빈도수를 확인합니다.



연속형 입력변수와 관계 파악: Weight

- Weight와 Price의 산점도를 그립니다.

```
>>> hds.plot_regression(
```

```
    data = df,
```

```
    x = 'Weight',
```

```
    y = 'Price'
```

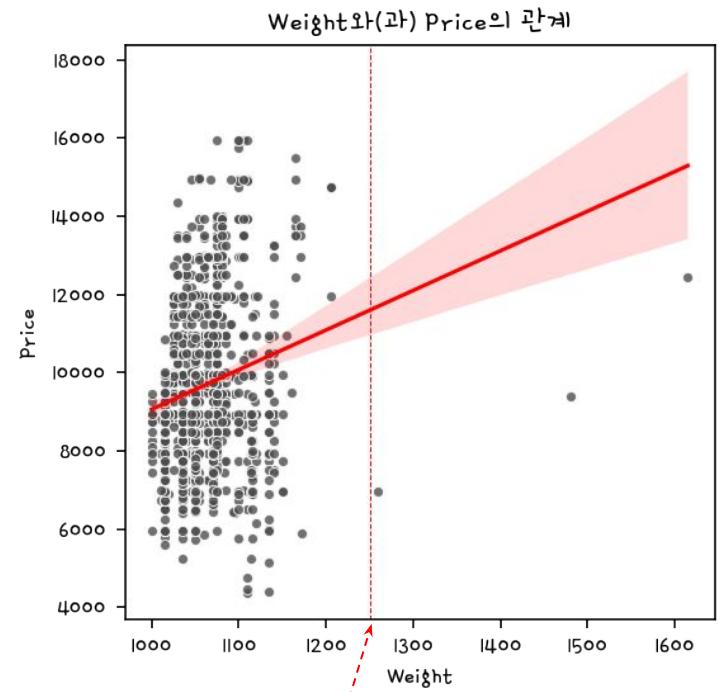
```
)
```

```
>>> plt.axvline(x = 1250, color = 'red',
```

세로선을

추가합니다.

```
    lw = 0.5, ls = '--');
```

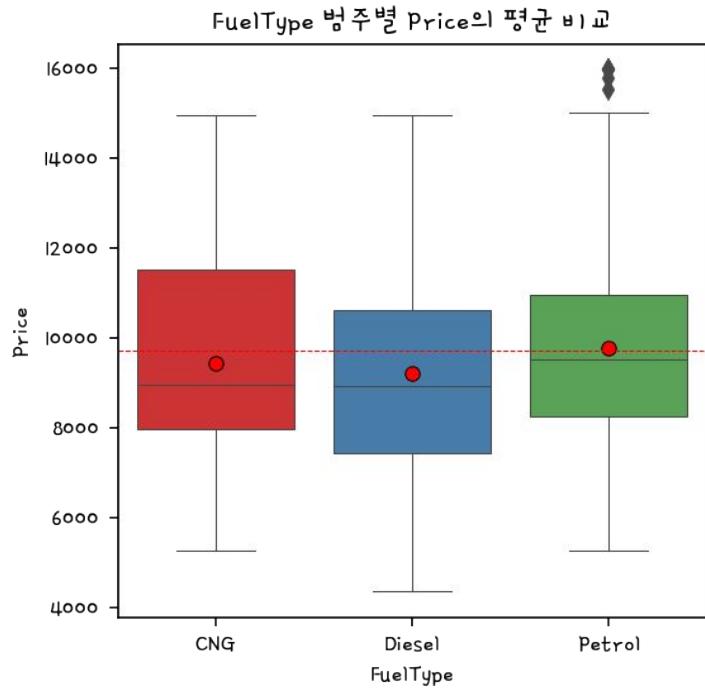


범주형 입력변수와 관계 파악: FuelType

- FuelType 범주별 Price의 상자 수염 그림을 그립니다.

```
>>> hds.plot_box_group(
```

```
    data = df,
    x = 'FuelType',
    y = 'Price',
    pal = 'Set1'
)
```



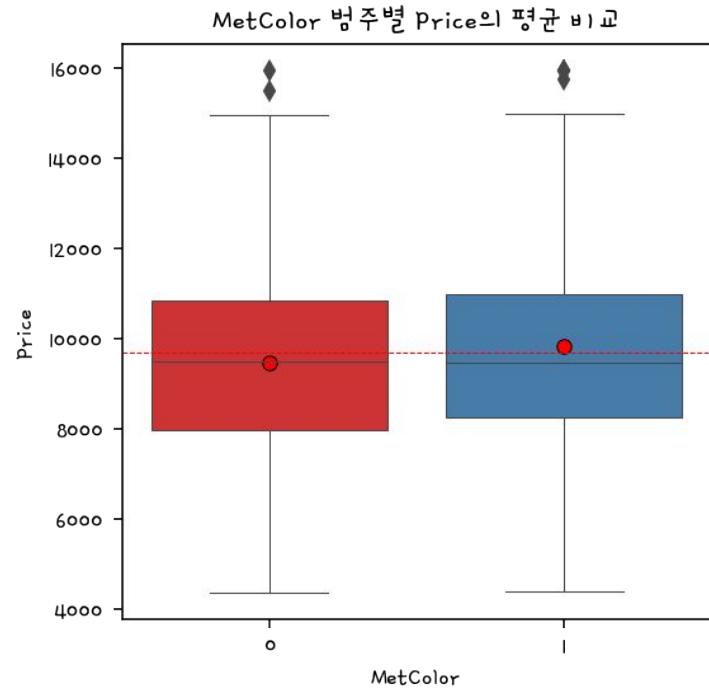
[참고] Price 범주별 평균(점)과 전체 평균(수평선)을 추가합니다.

범주형 입력변수와 관계 파악: MetColor

- MetColor 범주별 Price의 상자 수염 그림을 그립니다.

```
>>> hds.plot_box_group(
```

```
    data = df,
    x = 'MetColor',
    y = 'Price',
    pal = 'Set1'
)
```



범주형 입력변수와 관계 파악: Automatic

- Automatic 범주별 Price의 상자 수염 그림을 그립니다.

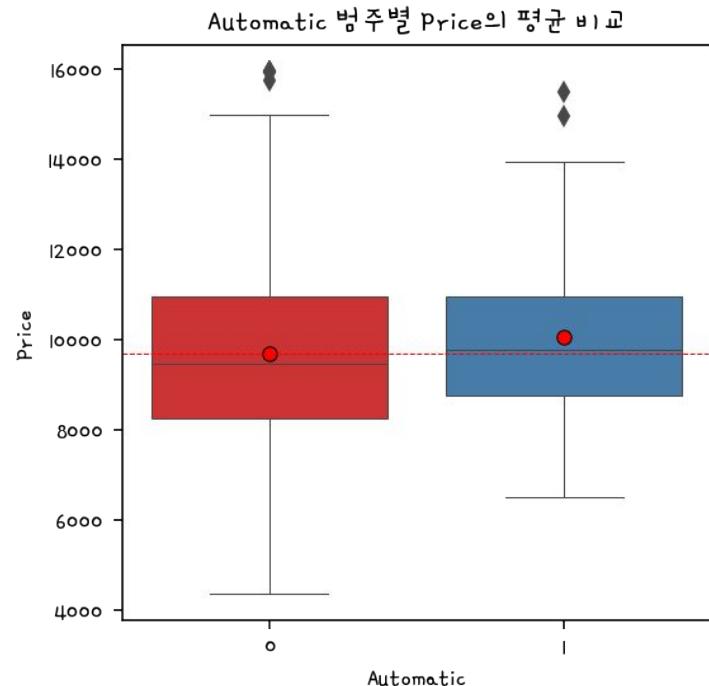
```
>>> hds.plot_box_group(
```

```
    data = df,  
    x = 'Automatic',
```

```
    y = 'Price',
```

```
    pal = 'Set1'
```

```
)
```



불필요한 행 삭제 및 외부 파일로 저장

- 시각화 결과 불필요하다고 판단하는 일부 행을 삭제합니다.

```
>>> df = df[df['Doors'].ne(2) & df['Weight'].le(1250)]
```

- df를xlsx 파일로 저장합니다.

```
>>> df.to_excel(excel_writer = 'Used_Cars_Price.xlsx', index = None)
```

- df를csv 파일로 저장합니다.

```
>>> df.to_csv(path_or_buf = 'Used_Cars_Price.csv', index = None)
```

- df를z 파일로 저장합니다.

```
>>> joblib.dump(value = df, filename = 'Used_Cars_Price.z')
```

기술통계 분석

기술통계

- 기술통계 Descriptive Statistics는 데이터의 주요 특징을 빠르게 파악할 때 사용합니다.
 - 'Descriptive'라는 단어에서 알 수 있듯이 데이터를 기술, 묘사한다는 것을 의미합니다.
- 기술통계의 주요 통계량에는 다음과 같습니다.
 - 대푯값: 평균, 절사평균, 중위수, 분위수, 사분위수, 최솟값, 최댓값, 최빈값
 - 산포: 범위, 사분범위, 분산, 표준편차, 중위수절대편차

관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3
```

```
>>> pd.options.display.precision = 3
```

관련 라이브러리 호출(계속)

- statsmodels 라이브러리를 설치합니다.

```
>>> !pip install statsmodels
```

- 통계 관련 라이브러리를 호출합니다.

```
>>> from scipy import stats # 절사평균 관련 함수를 포함하는 모듈을 호출합니다.
```

```
>>> from statsmodels import robust # 중위수절대편차 관련 함수를 포함하는 모듈을 호출합니다.
```

작업 경로 확인 및 변경

- 현재 작업 경로를 확인합니다.

```
>>> os.getcwd()
```

- data 폴더로 작업 경로를 변경합니다.

```
>>> os.chdir(path = '../data')
```

- 현재 작업 경로에 있는 폴더명과 파일명을 출력합니다.

```
>>> os.listdir()
```

실습 데이터셋 준비

- xlsx 파일을 읽고 데이터프레임 df를 생성합니다.

```
>>> df = pd.read_excel(io = 'Used_Cars_Price.xlsx')
```

- df의 열별 자료형을 출력합니다.

```
>>> df.dtypes
```

MetColor와 Automatic의 자료형이 numpy.int64입니다.
[참고] xlsx 파일로 저장되면서 문자열 '0'과 '1'을 정수로 변환합니다.

- z 파일을 읽고 데이터프레임 df를 생성합니다.

```
>>> df = joblib.load(filename = 'Used_Cars_Price.z')
```

- df의 열별 자료형을 출력합니다.

```
>>> df.dtypes
```

MetColor와 Automatic의 자료형이 object입니다.
[참고] z 파일은 Python 객체를 그대로 저장하므로 자료형을 그대로 유지합니다.

평균

- 평균^{mean}은 연속형 변수의 원소를 모두 더한 값을 원소 개수로 나눈 값입니다.
 - 평균은 이상치에 민감하게 영향을 받는다는 단점이 있습니다.
- 연속형 변수의 평균을 반환합니다.

```
>>> df['Price'].mean()
```

- 시리즈에 결측값^{NA}이 있으면 **mean()** 함수는 결측값을 제거한 평균을 반환합니다.

```
>>> nums = pd.Series(data = [1, 2, 3]) # 정수 1~3으로 시리즈를 생성합니다.
```

```
>>> nums.iloc[0] = np.nan; nums # nums의 0번 인덱스(첫 번째) 원소를 결측값으로 변경합니다.
```

```
>>> nums.mean() # nums의 평균을 반환합니다.
```

절사평균

- 절사평균^{trimmed mean}은 연속형 변수의 양 극단에서 일부를 제외한 평균입니다.
 - 아래 그림은 연속형 변수를 10등분한 십분위수를 표현한 것입니다.
 - 따라서 0%는 최솟값, 50%는 중위수, 그리고 100%는 최댓값입니다.
 - 10% 절사평균을 계산할 때 양 극단에서 10%씩 잘라내고 남은 80%로 평균을 계산합니다.



- 연속형 변수의 10% 절사평균을 반환합니다.

```
>>> stats.trim_mean(df['Price'], 0.1)
```

중위수

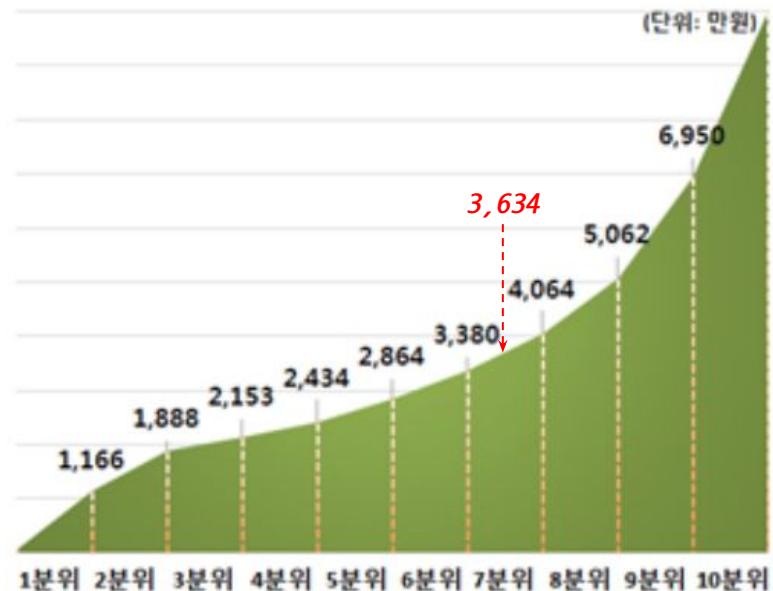
- 중위수 median는 연속형 변수를 오름차순 정렬했을 때 정가운데에 위치한 값입니다.
 - 원소 개수가 홀수면 정가운데 값을 반환합니다.
 - 원소 개수가 짝수면 정가운데에 있는 두 원소(수)의 평균을 반환합니다.
 - 중위수는 이상치에 민감한 평균에 비해 로버스트 robust한 통계량입니다.
 - 로버스트한 통계량은 이상치에 영향을 크게 받지 않는 값을 의미합니다.
- 연속형 변수의 중위수를 반환합니다.

```
>>> df['Price'].median()
```

```
>>> stats.trim_mean(df['Price'], 0.5) # [주의] 원소 개수가 홀수면 중위수, 짝수면 결측값(nan)을  
반환합니다.
```

[참고] 우리나라 근로자 연봉 평균 및 중위수(2018년)

- 2018년 기준 우리나라 근로자 평균 연봉은 약 3,634만원이었습니다.
- 그러나 오른쪽 그림에서 보이는 바와 같이 중위수는 약 2,864만원입니다.
- 이와 같이 이상치가 많은 데이터의 대표값으로는 평균 대신 중위수를 사용하는 것이 좋습니다.
 - 좌우대칭하는 분포는 중위수 대신 평균을 사용합니다.



분위수와 사분위수

- 분위수^{quantile}는 오름차순 정렬한 연속형 변수를 n 등분하는 경계 값입니다.
 - `quantile()` 함수의 q 매개변수에 0~1의 값을 지정하면 해당 분위수를 반환합니다.

- 백분위수^{percentile}는 전체 데이터를 100 등분하는 경계 값입니다.

```
>>> df['Price'].quantile(q = np.linspace(start = 0, stop = 1, num = 101))
```

- 십분위수^{decile}는 전체 데이터를 10 등분하는 경계 값입니다.

```
>>> df['Price'].quantile(q = np.linspace(start = 0, stop = 1, num = 11))
```

- 사분위수^{quartile}는 전체 데이터를 4 등분하는 경계 값입니다.

```
>>> df['Price'].quantile(q = np.linspace(start = 0, stop = 1, num = 5))
```

범위와 사분범위

- 범위^{range}는 연속형 변수의 최댓값과 최솟값의 간격입니다.

```
>>> df['Price'].max() - df['Price'].min()
```

```
>>> df['Price'].quantile(q = [0, 1]).diff().iloc[-1] # diff() 함수는 원소간 차이를 반환합니다. (최댓값 - 최솟값)
```

- 사분범위^{interquartile range}는 연속형 변수의 3사분위수와 1사분위수의 간격입니다.

```
>>> df['Price'].quantile(q = [0.25, 0.75]).diff().iloc[-1]
```



최빈값

- 최빈값^{mode}은 이산형 또는 범주형 변수에서 빈도수가 가장 높은 원소를 의미합니다.

```
>>> df['FuelType'].mode() # FuelType의 최빈값을 반환합니다.
```

- 범주형 변수의 빈도수를 내림차순 정렬한 결과를 반환합니다.

```
>>> df['FuelType'].value_counts() # [참고] value_counts() 함수의 ascending 매개변수에 전달하는  
인수의 기본값이 False입니다.
```

- 범주형 변수의 빈도수를 인덱스로 오름차순 정렬한 결과를 반환합니다.

```
>>> df['FuelType'].value_counts().sort_index()
```

- 범주형 변수의 빈도수 대신 상대도수를 내림차순 정렬한 결과를 반환합니다.

```
>>> df['FuelType'].value_counts(normalize = True)
```

평균, 분산 및 표준편차의 관계

- 아래는 A 중학교 학생 n명의 수학 시험 점수를 수집하여 표로 정리한 것입니다.

순번(i)	데이터(X_i)	편차($X_i - \bar{X}$)	편차 제곱
1	85	85 - 78	$(85 - 78)^2$
2	72	72 - 78	$(72 - 78)^2$
:	:	:	:
n	97	97 - 78	$(97 - 78)^2$
합계	$\sum_{i=1}^n X_i$	$\sum_{i=1}^n (X_i - \bar{X}) = 0$	$\sum_{i=1}^n (X_i - \bar{X})^2$
평균	$\frac{1}{n} \sum_{i=1}^n X_i = \bar{X}$	$\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X}) = 0$	$\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 = s^2$

편의상 X_i 의 평균을 78점으로 가정합니다.

분산 variance의 양의 제곱근은 표준편차 standard deviation입니다.

분산

- 분산^{variance}은 개별 관측값이 중심에서 떨어져 있는 정도를 나타내는 값입니다.
 - 개별 관측값이 중심(평균)에서 떨어져 있는 크기를 편차^{deviation}라고 합니다.
 - [참고] 편차를 단순하게 더하면 0이 되므로 제곱하여 부호를 통일해야 합니다.
 - 분산은 편차 제곱의 평균을 계산한 것입니다. 불편추정량이 되도록 $n-1$ 로 나눕니다.
 - 따라서 분산은 개별 관측값이 평균에서부터 평균적으로 떨어져 있는 정도를 나타냅니다.
 - 분산이 작을수록 개별 관측값이 평균에 밀집해 있다고 판단합니다.
- 연속형 변수의 분산을 반환합니다.

```
>>> df['Price'].var()
```

표준편차

- 표준편차 standard deviation 도 개별 관측값이 중심에서 떨어져 있는 정도를 나타냅니다.
 - 분산은 편차 제곱의 평균인데, 제곱했으므로 원래 변수의 척도와 다릅니다.
 - 예를 들어 단위가 cm인 연속형 변수의 분산을 계산하면 단위가 cm^2 로 바뀝니다.
 - 표준편차는 분산의 양의 제곱근이며, 원래 변수와 척도가 같아집니다.
 - 개별 관측값이 중심(평균)에서 떨어져 있는 정도를 파악할 때 표준편차를 확인합니다.
 - 표준편차도 작을수록 개별 관측값이 평균에 밀집해 있다고 판단합니다.
- 연속형 변수의 표준편차를 반환합니다.

```
>>> df['Price'].std()
```

중위수절대편차 median absolute deviation

- 중위수절대편차 median absolute deviation는 평균 대신 중위수를 사용하므로 표준편차보다 더욱 로버스트한 통계량이며 다음과 같이 계산합니다.
 - 연속형 변수의 중위수를 계산하고 개별 원소에서 중위수를 뺀 편차를 구합니다.
 - 중위수절대편차는 편차 절대값의 중위수를 계산하고 모수 추정 상수 1.4826을 곱합니다.

$$MAD = \text{median}(|X_i - \text{median}(X)|) \times 1.4826$$

- 연속형 변수의 중위수절대편차를 반환합니다.

```
>>> robust.mad(a = df['Price'])
```

여러 열의 기술통계량 확인

- 여러 연속형 변수의 평균, 표준편차 및 중위수를 반환합니다.

```
>>> df.apply(func = 'mean', numeric_only = True) # numeric_only는 pd.Series 통계 함수에 정의된 매개변수입니다.
```

```
>>> df.apply(func = 'std', numeric_only = True)
```

```
>>> df.apply(func = 'median', numeric_only = True)
```

- 여러 연속형/범주형 변수의 기술통계량을 반환합니다.

```
>>> df.describe() # 연속형 변수의 원소 개수, 평균, 표준편차, 최솟값, 사분위수 및 최댓값을 출력합니다.  
[참고] 표준편차를 계산할 때 분자를  $n-1$ 로 나눕니다.
```

```
>>> df.describe(include = object) # 범주형 변수의 원소 개수, 중복 제거한 원소의 종류, 최빈값 및 최빈값의 빈도수를 출력합니다.
```

확률분포의 이해

확률변수와 확률분포

- 정해진 확률로 값이 변하는 변수 X 를 확률변수라고 합니다. # [참고] 확률변수는 대문자로 표기합니다.
- 확률변수는 데이터의 형태에 따라 이산확률변수와 연속확률변수로 구분합니다.
 - 이산확률변수: 분리된(셀 수 있는) 정수를 갖는 변수입니다. # 주사위 눈금 1이 나올 확률
 - 연속확률변수: 연속한(셀 수 없는) 실수를 갖는 변수입니다. # 20대 남자 키가 170~175cm일 확률
 - 정밀도 문제로 연속형을 이산형으로 처리했다면 이산형을 연속형으로 다루어야 합니다.
- 확률분포는 확률변수가 가질 수 있는 값이 발생할 확률을 함수로 나타낸 것입니다.
 - 이산확률분포: 이산확률변수가 갖는 확률분포입니다.
 - 연속확률분포: 연속확률변수가 갖는 확률분포입니다.

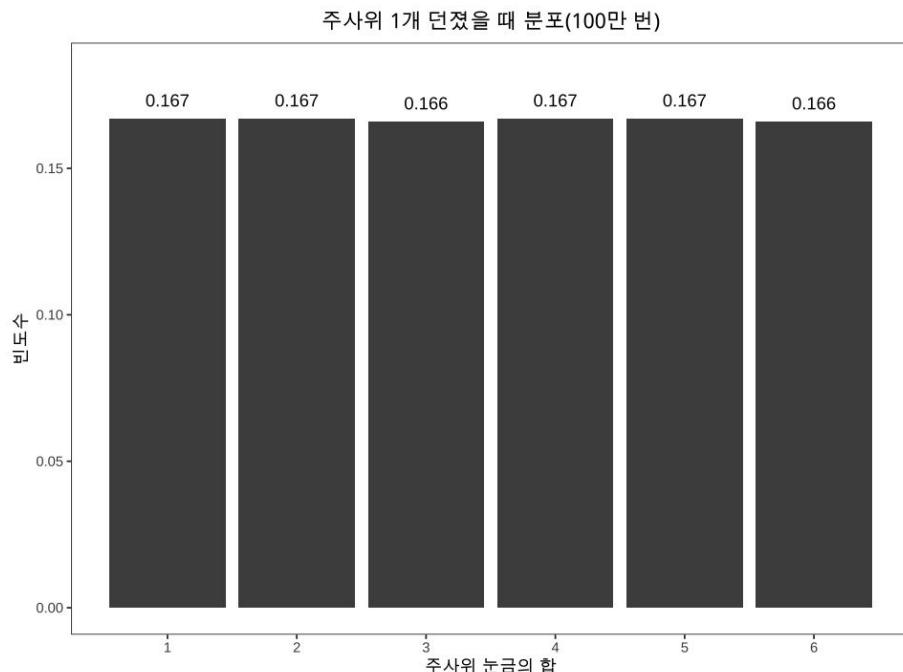
이산확률분포

- 이산확률변수 X 가 가질 수 있는 값이 발생할 확률을 계산할 수 있습니다.
 - 정상적인 주사위를 한 번 굴렸을 때 각 눈금이 나올 확률은 $1/6$ 입니다.

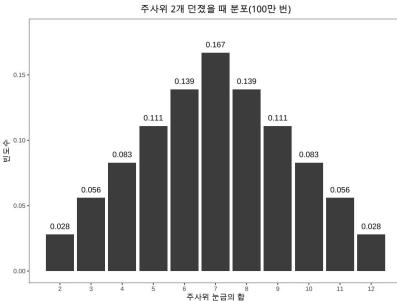
눈금	1	2	3	4	5	6
확률	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

- 이산확률변수의 분포를 나타내는 함수 $f(x)$ 를 확률질량함수라고 합니다.

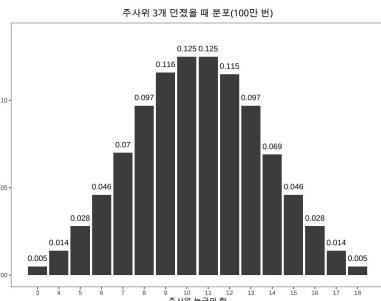
$$Pr(X) = f(x) \ # \text{ 함수값이 확률입니다.}$$



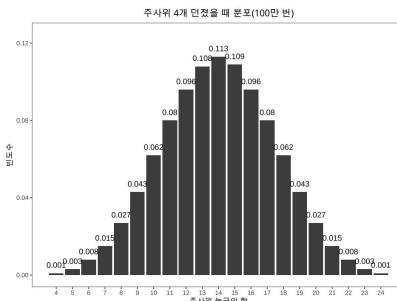
[참고] 주사위 던진 횟수를 늘리면?



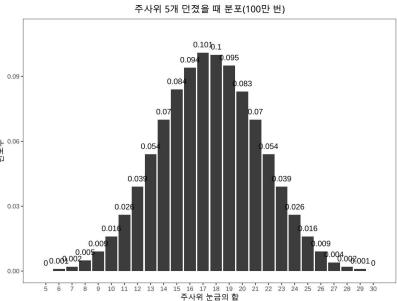
[주사위 2개 눈금의 합]



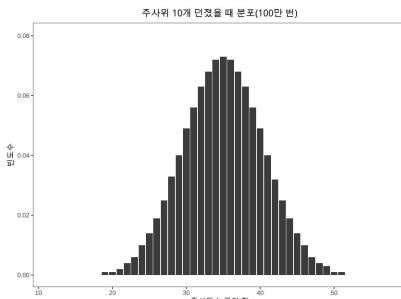
[주사위 3개 눈금의 합]



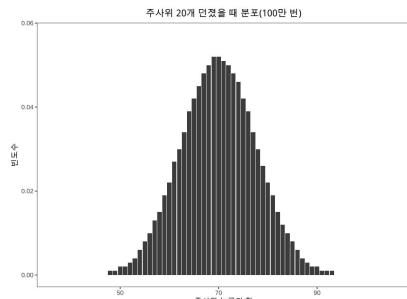
[주사위 4개 눈금의 합]



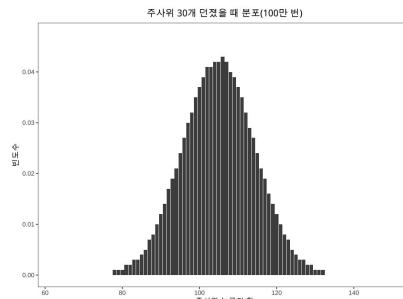
[주사위 5개 눈금의 합]



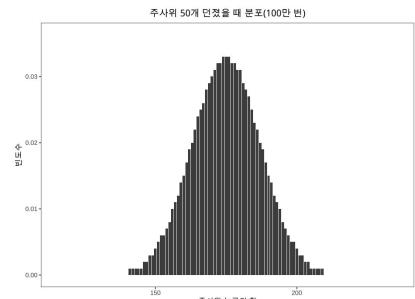
[주사위 10개 눈금의 합]



[주사위 20개 눈금의 합]



[주사위 30개 눈금의 합]



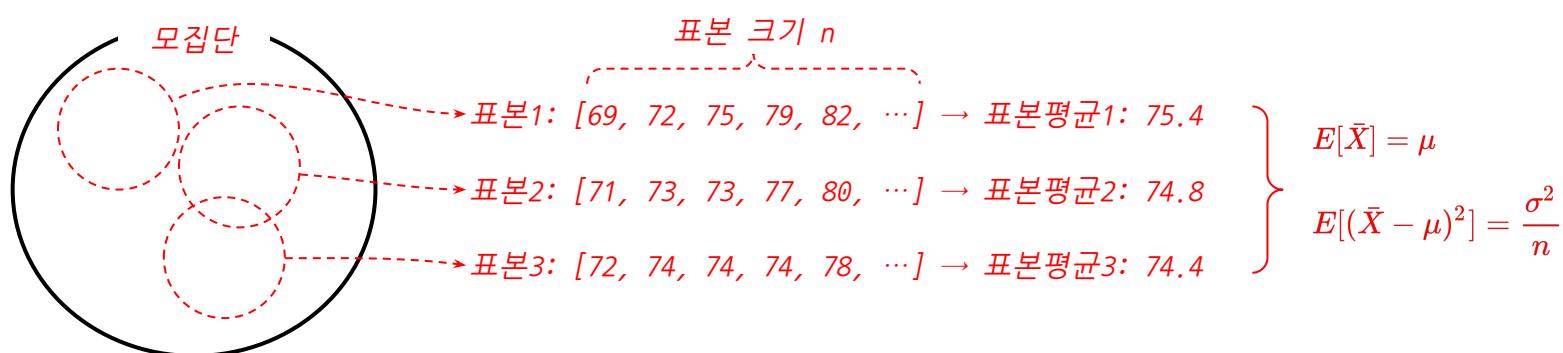
[주사위 50개 눈금의 합]

[참고] 중심극한정리

- 중심극한정리(Central Limit Theorem)는 모집단의 분포와 상관 없이 표본 크기가 증가하면 표본 통계량(예를 들어 표본평균)은 정규분포에 가까워진다는 내용입니다.
- 중심극한정리를 정리하면 다음과 같습니다.
 - 모집단은 평균이 μ 이고, 표준편차가 σ 인 분포를 따릅니다.
 - 모집단에서 n 개의 표본을 추출합니다. n 이 30개 이상이면 충분히 크다고 판단합니다.
 - 표본평균의 평균은 모평균, 표본평균의 분산은 모분산을 n 으로 나눈 값에 가까워집니다.
- 중심극한정리는 표본 통계량을 사용하는 가설검정(추론통계)에서 중요한 이론적인 근거를 제시하고 있으므로 매우 중요하며, 부트스트래핑을 확인할 수 있습니다.

[참고] 부트스트래핑

- 부트스트래핑은 단일표본으로 표본 통계량의 분포를 추정하는 방법입니다.
 - 단일표본을 모집단으로 가정하고 작은 표본(크기 n)을 여러 번 무작위 복원추출합니다.
 - 모집단 분포와 상관 없이 표본평균은 정규분포에 가깝습니다.(중심극한정리)
 - 작은 표본평균의 평균(기댓값)은 모평균, 표본평균의 분산은 모분산을 n 으로 나눈 값에 가까워집니다.

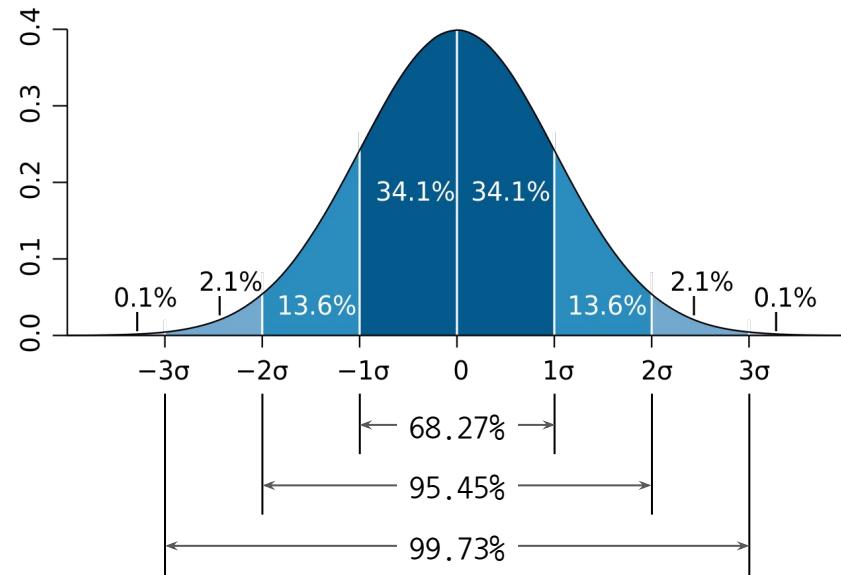


연속확률분포

- 연속확률변수의 분포를 나타내는 함수 $f(x)$ 를 확률밀도함수라고 합니다.
- 연속확률변수는 셀 수 없으므로 X 값의 확률 $Pr(X)$ 는 0이 됩니다.
- 연속확률변수 X 의 확률은 확률밀도함수 $f(x)$ 를 $a \sim b$ 구간에서 적분한 값(곡선 아래 면적)입니다.

$$Pr(a \leq X \leq b) = \int_a^b f(x) d(x) \quad \# \text{ 함수의 면적이 확률입니다.}$$

[평균이 0, 표준편차가 σ 인 정규분포]



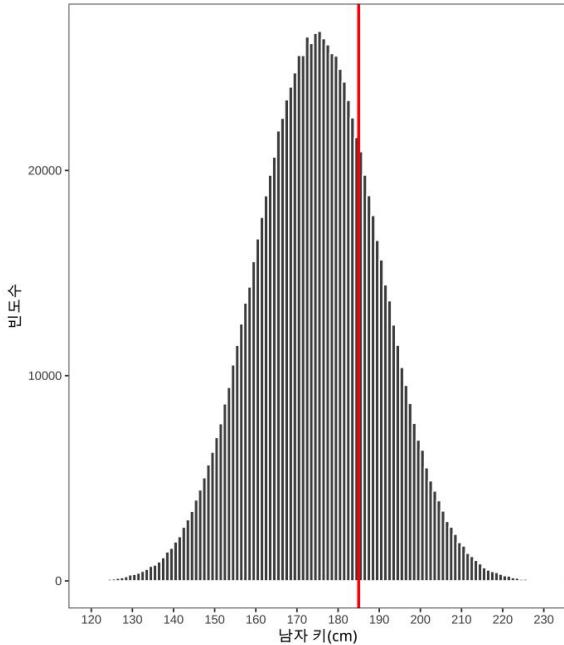
출처: https://ko.wikipedia.org/wiki/확률_분포

정규분포

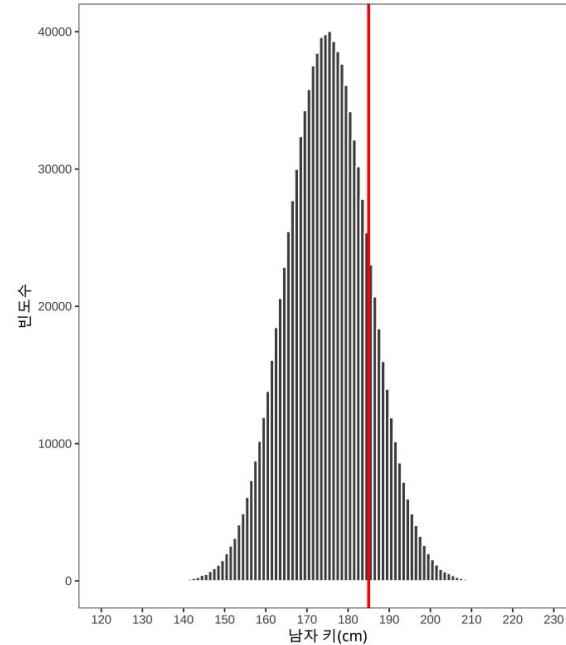
- 정규분포는 우리 주변에서 쉽게 발견할 수 있는 연속확률분포 중 하나입니다.
 - 분포의 형태는 좌우 대칭인 종모양의 곡선이고 분포의 중앙은 평균입니다.
 - 정규분포 확률밀도 곡선을 영어로는 Bell Curve라고 합니다.
 - 예를 들어 우리나라 20~59세 성인 남성을 모집단이라 가정하고 모집단에 속한 사람들의 몸무게를 측정하면 평균 몸무게 근처에 많은 사람들이 모여 있습니다.
- 모평균과 표준편차를 알면 정규분포 곡선을 그릴 수 있습니다.
 - 모평균은 중심이동, 표준편차는 분포의 폭을 결정합니다.
 - 표준편차가 작으면 평균 주변에 많은 원소가 있으므로 뾰족한 종모양을 그리지만 반대로 표준편차가 크면 넓게 퍼진 종모양으로 그려집니다.

정규분포의 예시

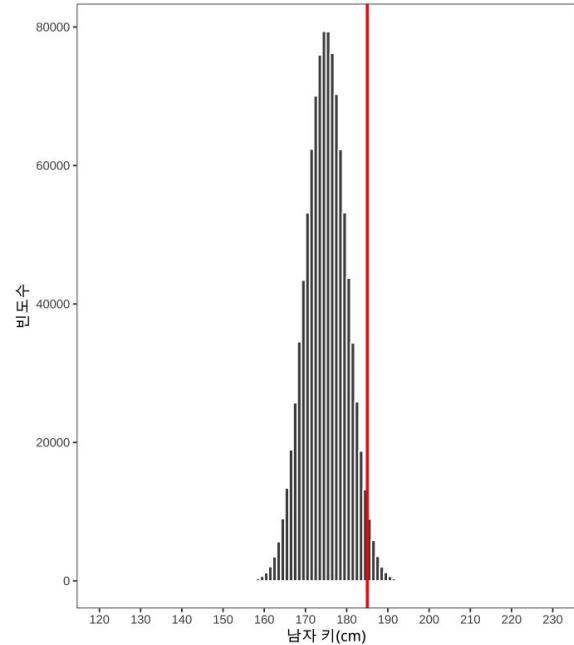
평균 175, 표준편차 15인 정규분포



평균 175, 표준편차 10인 정규분포



평균 175, 표준편차 5인 정규분포



정규분포를 따르는 세 집단의 평균은 같지만 표준편차가 다르기 때문에 정규분포의 폭이 다릅니다.

관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3
```

```
>>> pd.options.display.precision = 3
```

관련 라이브러리 호출(계속)

- 통계 관련 라이브러리를 호출합니다.

```
>>> from scipy import stats
```

```
>>> from statsmodels import robust
```

- 시각화 관련 모듈을 호출합니다.

```
>>> from GraphicSetting import *
```

```
>>> import HelloDataScience as hds
```

정규분포를 따르는 무작위 값 생성

- 시드를 고정합니다.

```
>>> np.random.seed(seed = 1234) # 시드에 같은 값을 지정하면 항상 같은 결과를 얻습니다.
```

- 평균이 175, 표준편차가 5인 정규분포를 따르는 무작위 표본을 생성합니다.

```
>>> heights = stats.norm.rvs(loc = 175, scale = 5, size = 5000)
```

- heights(표본)의 평균과 표준편차를 출력합니다.

```
>>> heights.mean() # heights(표본)의 평균은 모평균인 175에 가깝습니다.  
[참고] 표본 크기가 증가할수록 표본평균은 모평균에 더 가까워집니다.
```

```
>>> heights.std() # heights(표본)의 표준편차는 모표준편차인 5에 가깝습니다.  
[참고] 표본 크기가 증가할수록 표본표준편차도 모표준편차에 더 가까워집니다.
```

[참고] 정규분포 시각화

- 평균이 같고 표준편차가 다른 정규분포 확률밀도곡선을 시각화합니다.

```
>>> sp1 = stats.norm.rvs(loc = 175, scale = 5, size = 1000000)
```

```
>>> sp2 = stats.norm.rvs(loc = 175, scale = 10, size = 1000000)
```

```
>>> sp3 = stats.norm.rvs(loc = 175, scale = 15, size = 1000000)
```

```
>>> sns.kdeplot(x = sp1, label = 'Std: 5')
```

```
>>> sns.kdeplot(x = sp2, label = 'Std: 10')
```

```
>>> sns.kdeplot(x = sp3, label = 'Std: 15')
```

```
>>> plt.legend();
```

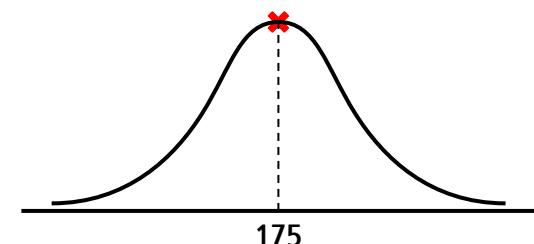
정규분포 확률밀도

- 다양한 정규분포에서 확률변수값 175의 확률밀도(높이)를 비교합니다.

```
>>> stats.norm.pdf(x = 175, loc = 175, scale = 15)
```

```
>>> stats.norm.pdf(x = 175, loc = 175, scale = 10)
```

```
>>> stats.norm.pdf(x = 175, loc = 175, scale = 5)
```



- 정규분포 확률밀도는 모집단에서 표본이 추출될 확률(가능도 likelihood)입니다.

```
>>> np.log(stats.norm.pdf(x = [174, 175, 176], loc = 173, scale = 5)).sum()
```

```
>>> np.log(stats.norm.pdf(x = [174, 175, 176], loc = 174, scale = 5)).sum()
```

```
>>> np.log(stats.norm.pdf(x = [174, 175, 176], loc = 175, scale = 5)).sum()
```

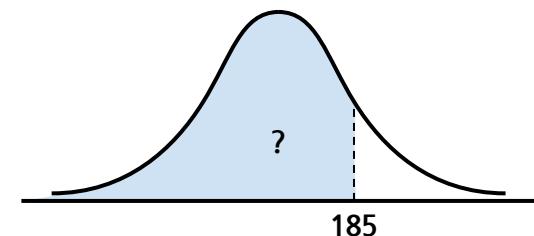
정규분포 누적확률

- 다양한 정규분포에서 확률변수값 185의 누적확률을 반환합니다.

```
>>> stats.norm.cdf(x = 185, loc = 175, scale = 15)
```

```
>>> stats.norm.cdf(x = 185, loc = 175, scale = 10)
```

```
>>> stats.norm.cdf(x = 185, loc = 175, scale = 5)
```



- 두 지점 간 확률을 계산하려면 x 매개변수에 원소가 2개인 리스트를 할당합니다.

```
>>> cdfs = stats.norm.cdf(x = [165, 185], loc = 175, scale = 5)
```

```
>>> np.diff(a = cdfs)[0]
```

평균(175)에서 ±2 표준편차(10)의 확률을 확인할 수 있습니다.
 [참고] cdfs는 numpy.ndarray으로 diff() 방식이 없습니다.

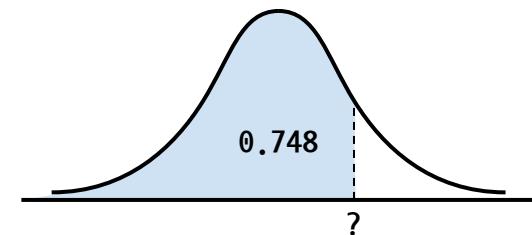
정규분포 확률변수값

- 다양한 정규분포에서 지정한 누적확률에 해당하는 확률변수값을 반환합니다.

```
>>> stats.norm.ppf(q = 0.748, loc = 175, scale = 15)
```

```
>>> stats.norm.ppf(q = 0.841, loc = 175, scale = 10)
```

```
>>> stats.norm.ppf(q = 0.977, loc = 175, scale = 5)
```



- (재미삼아) 20대 후반 남자 키가 평균이 175, 표준편차는 5인 정규분포를 따를 때 상위 5%인 남자 키는 몇 cm 이상일까요? 상위 1%인 남자 키도 알아봅시다.

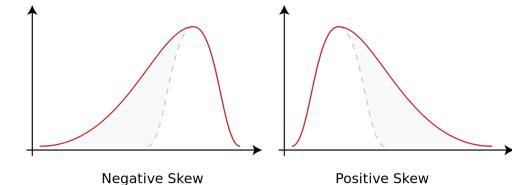
```
>>> stats.norm.ppf(q = 0.95, loc = 175, scale = 5)
```

```
>>> stats.norm.ppf(q = 0.99, loc = 175, scale = 5)
```

왜도와 첨도

- 왜도로 확률밀도곡선의 중심이 한 쪽으로 치우친 정도를 알 수 있습니다.

- 정규분포하는 데이터의 왜도는 0입니다.
- 표본 왜도가 ± 2 일 때 정규분포한다고 판단합니다.



`>>> stats.skew(a = heights)` # 왜도가 0보다 크면 왼쪽, 작으면 오른쪽으로 치우친 분포를 의미합니다.

- 첨도로 확률밀도곡선의 봉우리가 뾰족한지 완만한지 여부를 알 수 있습니다.

- 정규분포하는 데이터의 첨도는 3입니다.
- 표본 첨도가 1~5일 때 정규분포한다고 판단합니다.

`>>> stats.kurtosis(a = heights)` # 첨도가 3보다 크면 뾰족, 작으면 완만한 봉우리를 의미합니다.
 [참고] 이 함수는 정규분포하는 데이터의 첨도를 0으로 반환합니다.

정규성 검정

- 5천 건 이하인 데이터의 정규성 검정은 사피로-윌크 검정을 실행합니다.

```
>>> stats.shapiro(x = heights) # 정규성 검정의 귀무가설은 '데이터가 정규분포한다'입니다.  
# 유의확률 p-value이 0.05 보다 크면 정규분포한다고 판단합니다.
```

- 5천 건을 초과하는 가상의 키 데이터를 생성합니다.

```
>>> np.random.seed(seed = 1234)
```

```
>>> heights = stats.norm.rvs(loc = 175, scale = 5, size = 10000)
```

```
>>> stats.shapiro(x = heights) # 5천 건을 초과하는 데이터로 사피로-윌크 검정을 실행하면 유의확률이  
# 정확하지 않다는 경고를 출력합니다.
```

- 5천 건을 초과하는 데이터의 정규성 검정은 앤더슨-달링 검정을 실행합니다.

```
>>> stats.anderson(x = heights) # 앤더슨-달링 검정의 귀무가설도 '데이터가 정규분포한다'입니다.  
# [참고] 유의확률 대신 임계값 critical values 을 반환합니다.
```

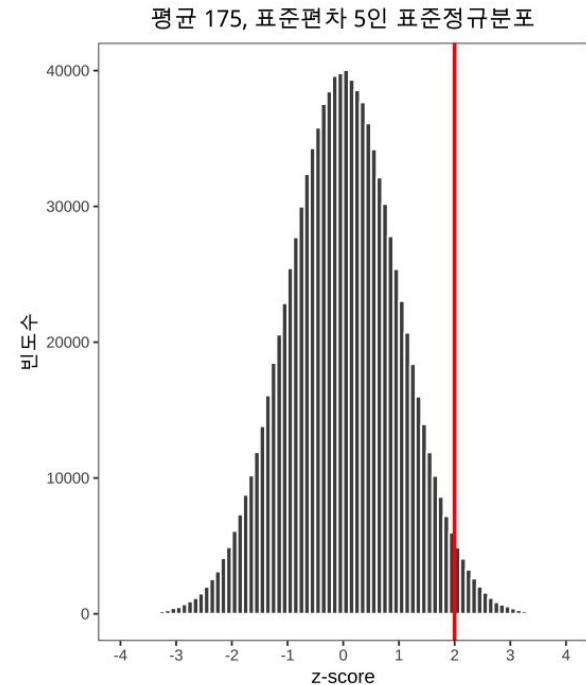
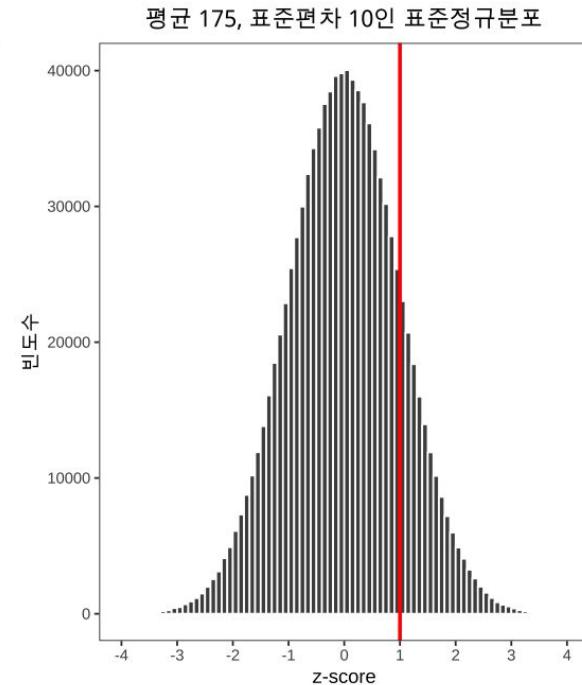
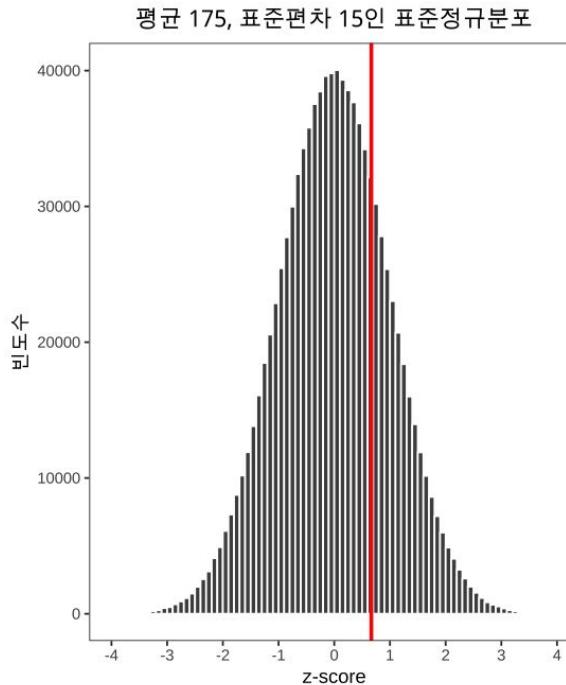
데이터 정규화

- 홍길동이 최근에 받은 수학과 영어 점수입니다. 어떤 과목을 더 잘하나요?
 - 수학: 90점(평균 75, 표준편차 15)
 - 영어: 55점(평균 40, 표준편차 10)
- 홍길동이 받은 두 과목의 점수 중에서 절대값은 수학이 크지만, 상대적인 크기를 비교할 수 없습니다.
 - 분포가 다른 두 값의 상대적인 크기를 비교하려면 척도를 통일해야 합니다.
 - 데이터 정규화^{normalization}는 두 값(스칼라, 벡터)의 척도를 같아지도록 변환하는 것입니다.
 - 다양한 머신러닝 알고리즘에서 입력 데이터셋의 정규화를 요구합니다.

표준정규분포

- 만약 두 확률변수가 따르는 정규분포의 평균과 표준편차가 다르면 확률변수값을 비교하는 것이 어렵습니다.
 - 하지만 확률변수를 평균이 0, 표준편차가 1인 표준정규분포를 따르는 z-score로 변환하면 확률변수값을 비교할 수 있습니다.
 - z-score는 확률변수값에서 평균을 빼고 표준편차로 나눈 값입니다.
- $$z\text{-score} = \frac{X - \mu}{\sigma} \quad \text{# 데이터를 표준화하면 개별 원소가 표준편차의 몇 배에 해당하는지 쉽게 알 수 있습니다.}$$
- 확률변수값을 z-score로 변환하는 것을 데이터 표준화^{standardization}라고 합니다.
 - 데이터 표준화는 데이터를 정규화하는 대표적인 방법 중 하나입니다.

표준정규분포의 예시



정규분포를 따르는 세 집단을 데이터 표준화한 결과, 185cm인 사람의 z-score가 서로 다른 위치에 있음을 알 수 있습니다.

데이터 표준화

- 데이터를 표준화하는 함수를 생성합니다.

```
>>> def scale(x, loc, scale):  
    return (x - loc) / scale
```

- 다양한 정규분포를 따르는 관측값을 표준화하고 크기를 비교합니다.

```
>>> scale(x = 185, loc = 175, scale = 15)  
>>> scale(x = 185, loc = 175, scale = 10)  
>>> scale(x = 185, loc = 175, scale = 5)
```

[참고] 이상치 탐지 방법 비교

- [std] 표준화된 데이터의 절대값이 3을 초과할 때 이상치로 판단합니다.

```
>>> locs = np.where(np.abs(stats.zscore(heights)) > 3) # 표준화된 절대값이 3을 초과하는
                                                # 인덱스를 locs에 할당합니다.
```

```
>>> heights[locs] # 평균과 표준편차 기준에서 이상치로 보이는
                    # 원소를 출력합니다. # np.where() 함수는 조건이 True인
                                                # 인덱스를 정수로 반환합니다.
```

- [mad] 평균과 표준편차 대신 중위수와 중위수절대편차를 이용한 방법입니다.

```
>>> med = np.median(a = heights) # 중위수를 med에 할당합니다.
```

```
>>> mad = robust.mad(a = heights) # 중위수절대편차를 mad에 할당합니다.
```

```
>>> locs = np.where(np.abs((heights - med) / mad) > 3) # 개별 값에서 중위수를 차감하고
                                                # 중위수절대편차로 나눕니다.
```

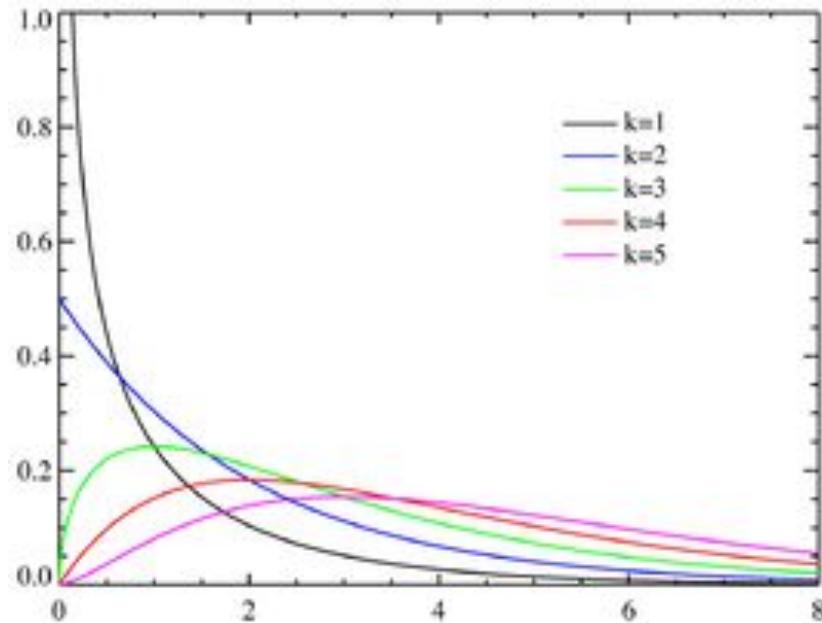
```
>>> heights[locs] # 중위수와 중위수절대편차 기준에서 이상치로 보이는 원소를 출력합니다.
```

χ^2 (카이제곱^{chi-square}) 분포

- 카이제곱 분포는 k 개의 서로 독립적인 z-score의 제곱을 모두 더한 분포이며 모집단의 분산을 추정할 때 사용합니다.

$$\chi^2_{(k)} = z_1^2 + z_2^2 + \cdots + z_k^2 = \sum_{i=1}^k z_i^2$$

- 자유도(k)에 따라 분포 모양이 달라지며 자유도가 증가할수록 중심이 오른쪽으로 이동합니다.
- 카이제곱 분포는 교차분석 및 로지스틱 회귀모형의 유의성 검정에서 사용합니다.

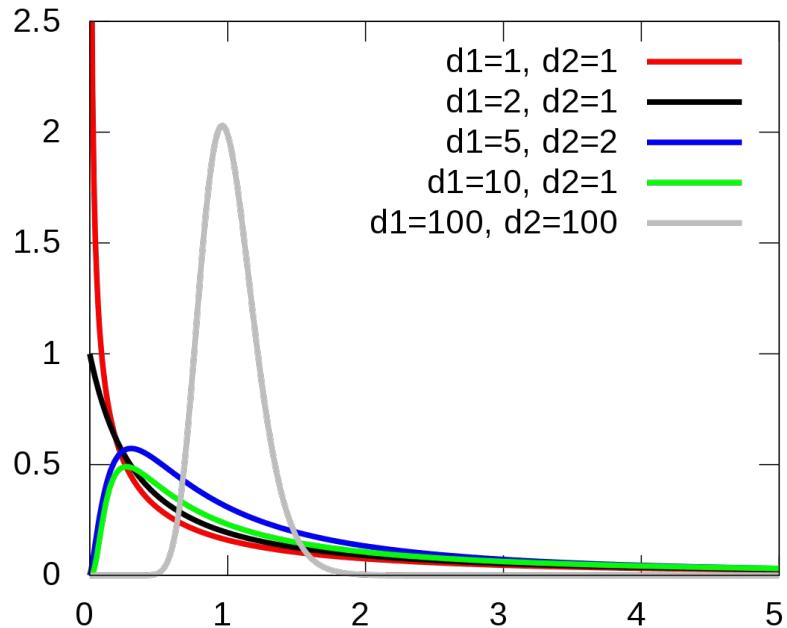


F 분포

- F 분포는 카이제곱 분포를 따르는 두 확률값을 각각의 자유도로 나눈 값의 비^{ratio}의 분포입니다.

$$F = \frac{\chi_1^2/k_1}{\chi_2^2/k_2}$$

- F 분포는 분산분석에서 집단 간 변동과 집단 내 변동의 비^{ratio}가 1인지(분산이 같은지) 확인할 때 사용합니다.
- 또한 선형 회귀모형의 유의성 검정에서 F 분포를 사용합니다.

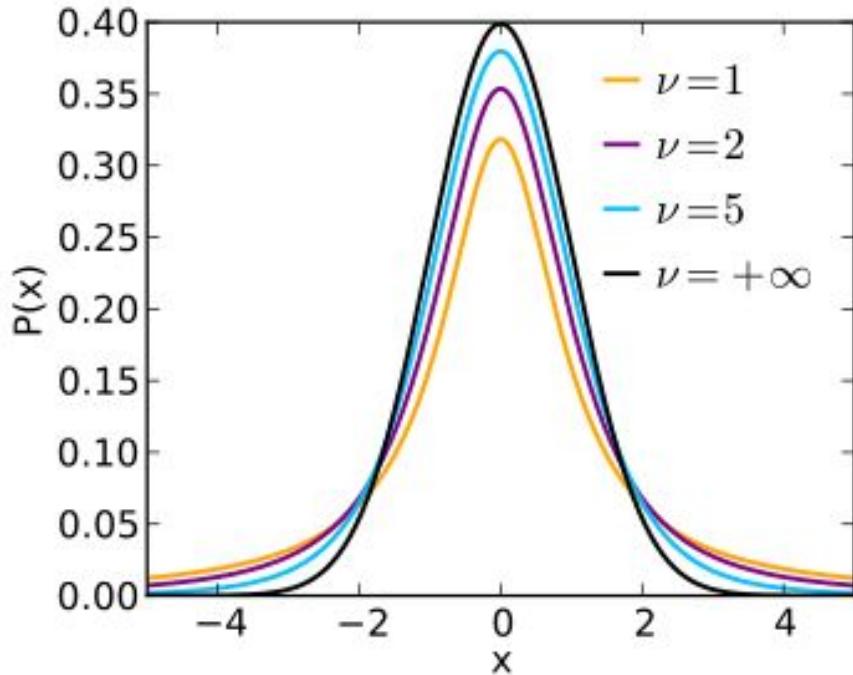


스튜던트 t 분포

- 스튜던트 t 분포는 모분산을 모르는 크기가 작은 표본평균의 분포입니다.

$$t_{\bar{X}} = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n-1}}}$$

- t 분포는 자유도의 크기가 증가할수록 표준정규분포에 가까워집니다.
- t 분포는 두 집단의 평균 차이가 서로 같은지 확인하는 t-검정 및 선형 회귀 계수의 유의성 검정에서 사용합니다.



가설검정의 이해

통계적 가설검정의 이해

- 통계적 가설검정은 어떤 주장(가설)에 대해 모수와 표본 통계량을 이용하여 해당 주장이 합당한 것인지를 판단하는 과정을 의미하며 아래 절차를 거칩니다.
 - 귀무가설^{null hypothesis}과 대립가설^{alternative hypothesis}을 설정합니다.
 - 유의수준(α)과 기각역을 설정합니다. 유의수준은 참인 귀무가설을 기각하는 위험입니다.
 - 모집단을 대표하는 표본 통계량과 모수의 통계적 거리인 검정통계량을 계산합니다.
 - 검정통계량으로 z-score(모분산 사용) 또는 t-value(표본분산 사용)를 주로 사용합니다.
 - 검정통계량으로부터 유의확률을 계산합니다. 유의확률은 귀무가설이 맞다는 가정 하에서 수집한 검정통계량보다 더욱 극단적인 값이 추출될 확률입니다.
 - 검정통계량이 기각역에 속하면(유의확률이 유의수준보다 작으면) 귀무가설을 기각합니다.

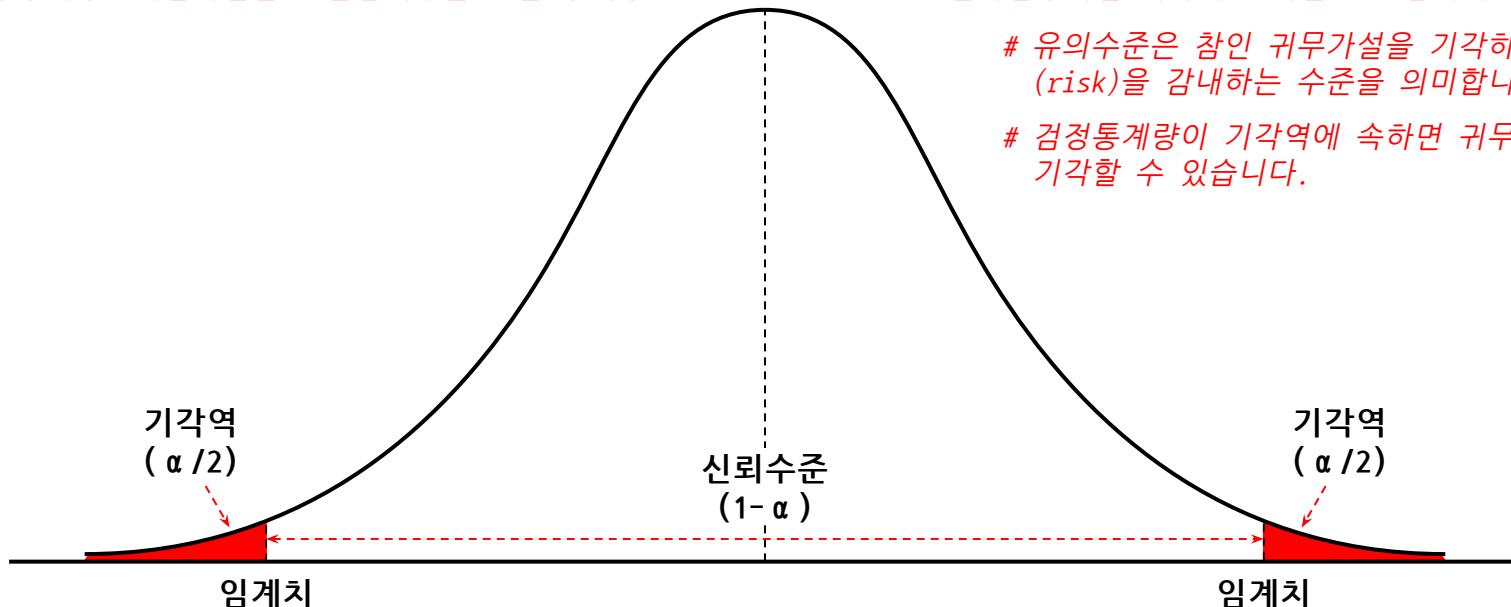
[참고] 가설검정 관련 용어 정리

- # 귀무가설: (차이 없는) 기각하기 위해 설정한 가설
- # 대립가설: 분석가가 채택하려는 가설
- # 검정통계량: 대립가설을 뒷받침해주는 표본 통계량

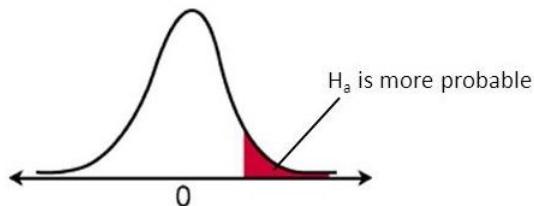
- # 유의수준(α)을 0.05로 설정할 때
 - 양측검정이면 기각역 크기는 0.025입니다.
 - 단측검정이면 기각역 크기는 0.05입니다.

유의수준은 참인 귀무가설을 기각하는 위험(risk)을 감내하는 수준을 의미합니다.

검정통계량이 기각역에 속하면 귀무가설을 기각할 수 있습니다.

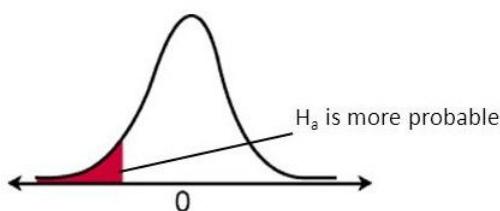


[참고] 단측검정과 양측검정의 차이



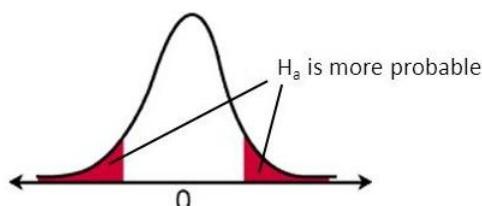
Right-tail test

$H_a: \mu > \text{value}$ # 한국 성인 남자의 평균 키는 173cm 보다 크다.



Left-tail test

$H_a: \mu < \text{value}$ # 한국 성인 남자의 평균 키는 173cm 보다 작다.



Two-tail test

$H_a: \mu \neq \text{value}$ # 한국 성인 남자의 평균 키는 173cm가 아니다.

출처: <https://www.fromthegenesis.com/why-hypothesis-testing/>

가설검정의 종류

구분		목표변수의 종류	
		연속형	범주형
입력변수의 종류	연속형	피어슨 상관분석	-
	범주형	t-검정 분산분석 ANOVA	교차분석 chisquared-test

관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3
```

```
>>> pd.options.display.precision = 3
```

관련 라이브러리 호출(계속)

- pingouin 라이브러리를 설치합니다.

```
>>> !pip install pingouin # [참고] 아나콘다 사용자는 터미널에서 아래 코드를 실행합니다.  
% conda install -c conda-forge pingouin
```

- 통계 관련 라이브러리를 호출합니다.

```
>>> from scipy import stats
```

```
>>> import pingouin as pg
```

작업 경로 확인 및 변경

- 현재 작업 경로를 확인합니다.

```
>>> os.getcwd()
```

- data 폴더로 작업 경로를 변경합니다.

```
>>> os.chdir(path = '../data')
```

- 현재 작업 경로에 있는 폴더명과 파일명을 출력합니다.

```
>>> os.listdir()
```

실습 데이터셋 준비

- z 파일을 읽고 데이터프레임 df를 생성합니다.

```
>>> df = joblib.load(filename = 'Used_Cars_Price.z')
```

- df의 정보를 확인합니다.

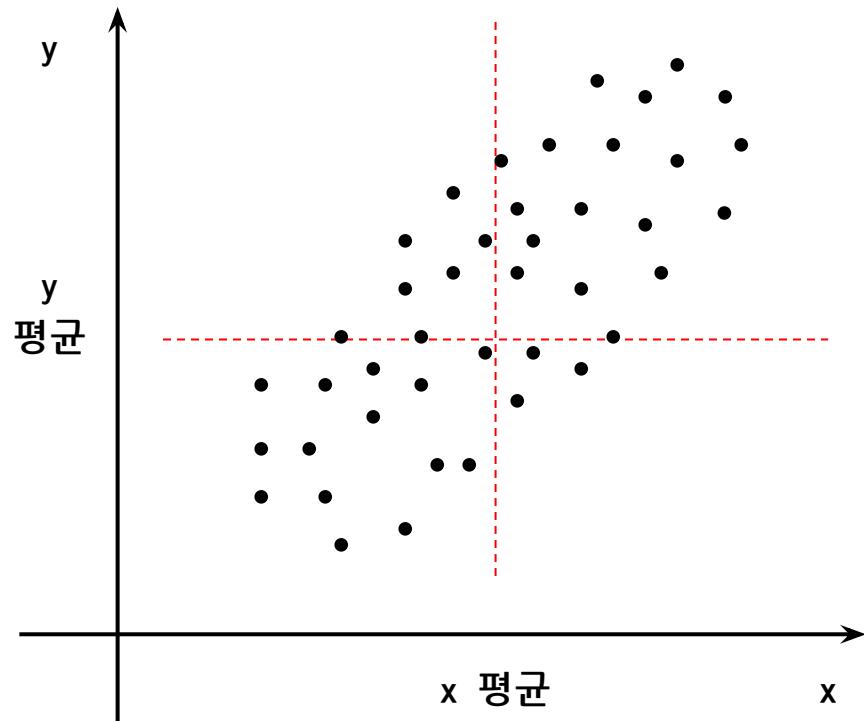
```
>>> df.info()
```

- df의 처음 5행을 출력합니다.

```
>>> df.head()
```

분산, 공분산 및 상관계수의 관계

구분	공식
분산	$\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$
공분산	$\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})$
상관계수	$\frac{1}{n-1} \sum_{i=1}^n \frac{(X_i - \bar{X})(Y_i - \bar{Y})}{s_X s_Y}$



공분산

- 공분산 covariance은 두 연속형 변수의 상관관계 방향을 나타내는 통계량입니다.
 - x가 증가할 때 대응하는 y가 증가하면 공분산은 양수입니다. # [참고] 상관관계는 직선의 관계를 의미합니다.
 - x가 증가할 때 대응하는 y는 감소하면 공분산은 음수입니다.
 - x와 y의 척도가 다르므로 공분산은 상관관계의 방향만 제시합니다.
- 두 연속형 변수의 공분산을 반환합니다.

```
>>> df['Age'].cov(df['Price'])
```

```
>>> df.cov() # df의 연속형 변수로 분산-공분산 행렬을 반환합니다.
```

상관계수

- 상관계수^{correlation coefficient}는 공분산을 각 변수의 표준편차로 나눈 것입니다.
 - 상관계수는 두 연속형 변수에 대한 상관관계의 방향과 강도를 함께 나타냅니다.
 - 상관계수는 -1~1의 값을 갖는데 ±1에 가까울수록 강한 상관관계가 있다고 판단합니다.

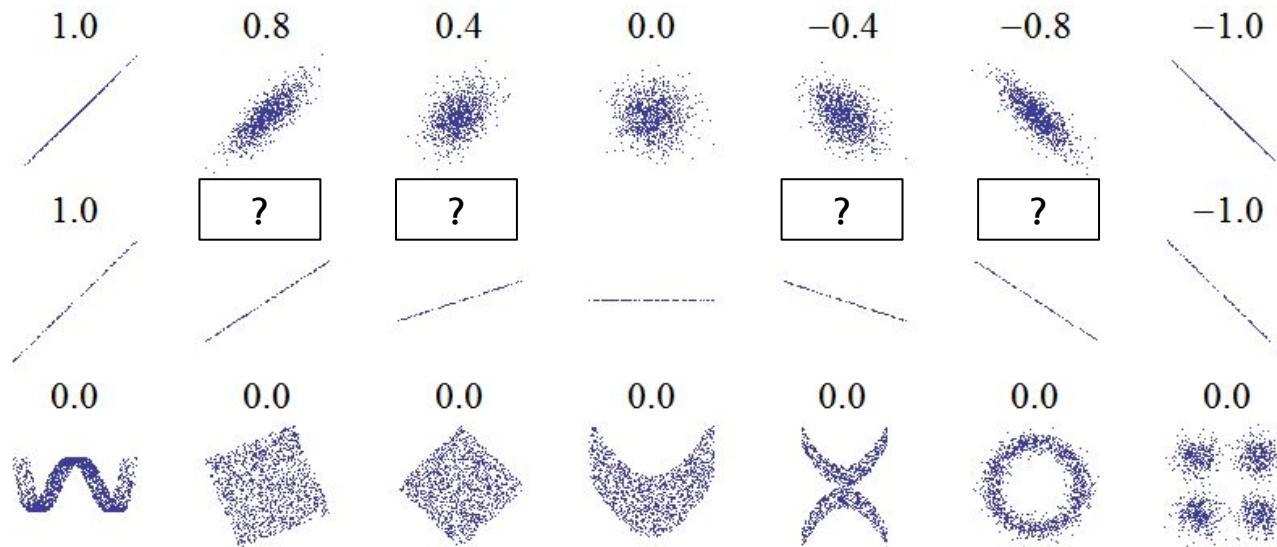
상관계수	$ r < 0.10$	$0.10 \leq r < 0.29$	$0.29 \leq r < 0.46$	$0.46 \leq r $
해석	상관관계 거의 없음	약한(낮은) 상관관계	보통 상관관계	강한(높은) 상관관계

- 두 연속형 변수의 피어슨 상관계수를 반환합니다.

```
>>> df['Age'].corr(df['Price']) # [참고] method 매개변수에 상관계수 종류를 문자열로 지정합니다.  
기본값은 'pearson'이고, 'spearman'과 'kendall'도 가능합니다.
```

```
>>> df.corr() # df의 연속형 변수로 피어슨 상관계수 행렬을 반환합니다.
```

[참고] 상관계수 관련 그래프



출처: https://en.wikipedia.org/wiki/Distance_correlation

피어슨 상관분석

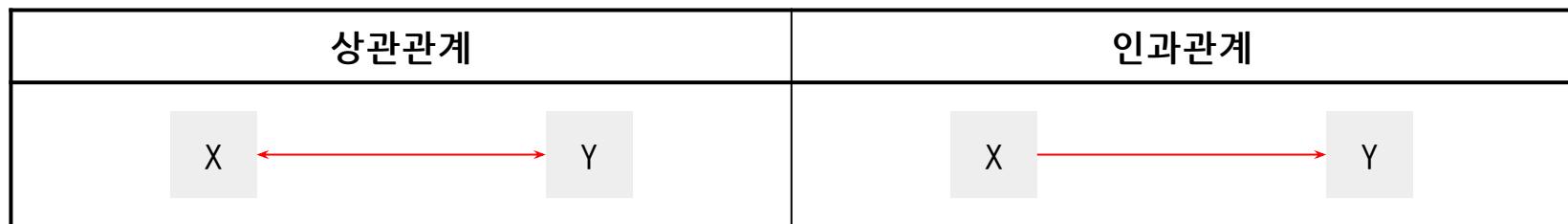
- 피어슨 상관분석은 두 연속형 변수에 상관관계가 있는지 측정합니다.
 - 피어슨 상관계수는 정규분포를 따르는 두 연속형 변수의 공분산을 각 표준편차로 나누어 표준화한 값입니다.(모수적 방법)
 - 예를 들어 국어시험 점수와 영어시험 점수와의 관계를 의미합니다.
 - 피어슨 상관계수는 $-1 \leq \rho \leq 1$ 을 만족합니다.
 - 피어슨 상관계수가 -1 또는 1에 가까워질수록 강한 상관관계를 갖습니다.
 - 피어슨 상관분석 관련 함수는 상관계수와 유의확률을 반환합니다.
 - 유의확률이 0.05 보다 작으면 두 변수에 상관관계가 있다고 판단합니다.

스피어만 상관분석

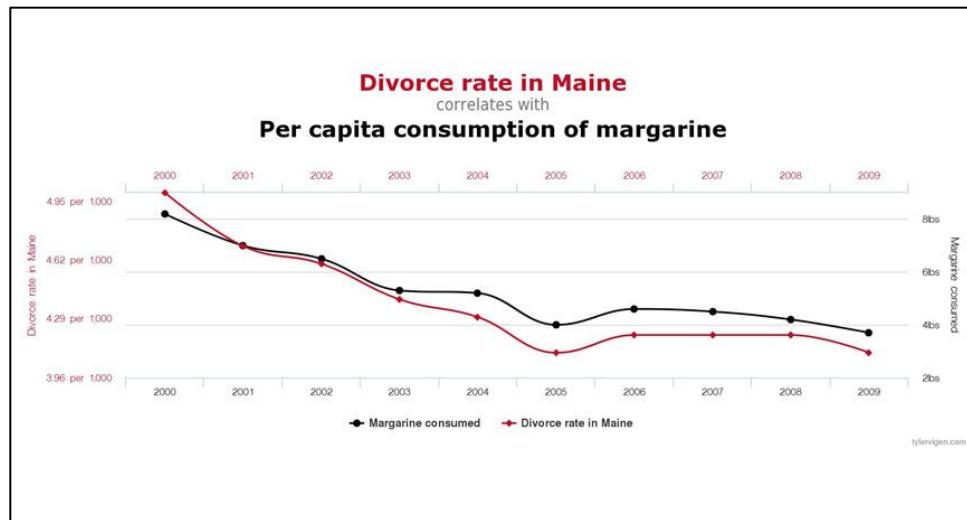
- 스피어만 상관분석은 이산형/순서형 변수의 의존성을 측정합니다.
 - 스피어만 상관계수는 정규분포하지 않는 두 변수에서 한 변수가 증가할 때 다른 변수도 증가하는지 확인합니다.(순위를 고려한 비모수적인 방법)
 - 예를 들어 국어시험 등수와 영어시험 등수와의 관계를 의미합니다.
 - 스피어만 상관계수의 공식은 피어슨 상관계수의 공식과 비슷합니다.
 - 데이터의 순서로 피어슨 상관계수를 계산합니다.
 - 스피어만 상관계수가 1이면 한 변수의 값이 증가할 때 다른 변수의 값도 증가한다는 것을 의미합니다.
 - 유의확률이 0.05 보다 작으면 두 변수의 순서에 상관관계가 있다고 판단합니다.

상관관계 vs 인과관계

- 두 변수에 상관관계가 있다는 것이 인과관계가 있다는 것을 의미하지 않습니다.
 - 상관관계는 두 변수가 서로 영향을 주고 받는 관계입니다. 우연의 일치도 있습니다.
 - 같은 시점에 측정한 두 변수의 크기가 같은 방향으로 움직일 때 상관관계가 높게 나타납니다.
 - 인과관계는 원인 변수가 결과 변수에 영향을 주는 관계입니다.
 - 두 변수에 시점의 차이가 있습니다. 예를 들어 식사량에 따라 체중 증가량이 달라집니다.



[참고] 인과관계 판단 오류



출처: <http://www.ssacstat.com/>

출처: <https://www.sciencenewsforstudents.org>

피어슨 상관분석의 가설 및 검정통계량

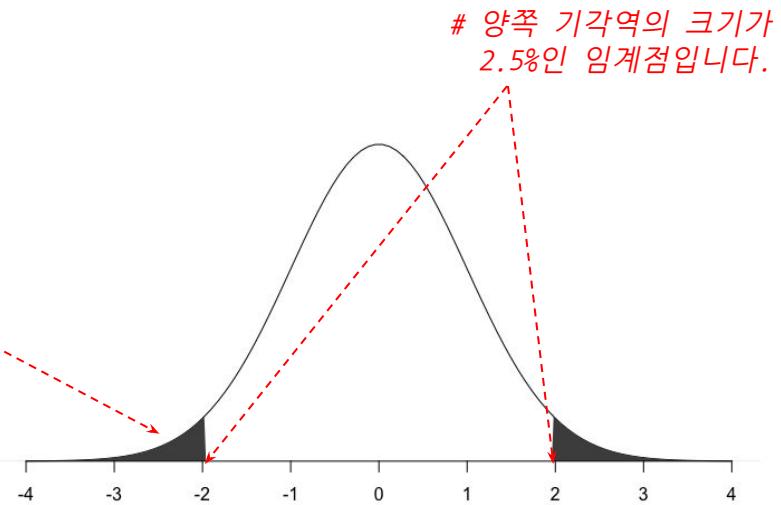
- 피어슨 상관분석의 가설 및 검정통계량은 다음과 같습니다.

- 귀무가설(H_0): $\rho = 0$ (모집단의 상관계수는 0이다)
- 대립가설(H_1): $\rho \neq 0$ (모집단의 상관계수는 0이 아니다)

- 유의수준: 5% \rightarrow 양측검정이므로 각 2.5%

- 검정통계량: $t = \frac{r - \rho}{\sqrt{\frac{1-r^2}{n-2}}}$

검정통계량이 기각역(검정색)에
있으면 귀무가설을 기각합니다.



피어슨 상관분석

- 두 연속형 변수의 피어슨 상관분석을 실행하고, 유의확률 $p\text{-value}$ 을 확인합니다.

```
>>> pg.corr(x = df['Age'], y = df['Price']) # method 매개변수의 기본 인수는 'pearson'입니다.
```

```
>>> pg.corr(x = df['KM'], y = df['Price'])
```

```
>>> pg.corr(x = df['HP'], y = df['Price'])
```

```
>>> pg.corr(x = df['CC'], y = df['Price'])
```

```
>>> pg.corr(x = df['Doors'], y = df['Price'])
```

```
>>> pg.corr(x = df['Weight'], y = df['Price'])
```

[참고] 피어슨 상관분석 유의확률 출력 함수 생성

- 변수 x에 df의 연속형 변수를 할당합니다.

```
>>> x = df[ 'Age' ]
```

- 변수 x와 Price의 피어슨 상관분석 실행 결과에서 유의확률만 출력합니다.

```
>>> pg.corr(x = x, y = df[ 'Price' ])['p-val']
```

- 연속형 입력변수와의 상관분석 유의확률을 출력하는 람다 표현식을 생성합니다.

```
>>> corr = lambda x: pg.corr(x = x, y = df[ 'Price' ])['p-val']
```

- 람다 표현식 함수로 피어슨 상관분석 유의확률을 출력합니다.

```
>>> corr(x = df[ 'Age' ])
```

[참고] apply() 함수를 활용한 상관분석 실행

- df의 열별 자료형을 확인합니다.

```
>>> df.dtypes # [참고] Windows는 정수를 numpy.int32로 생성합니다.
```

- 열별 자료형이 정수형 또는 실수형이면 True, 아니면 False인 벡터를 생성합니다.

```
>>> locs = df.dtypes.astype(str).isin(values = ['float64', 'int64']); locs
```

- df에서 정수형 또는 실수형 변수만 선택하고 열별로 람다 표현식을 실행합니다.

```
>>> df.loc[:, locs].apply(func = corr)
```

```
>>> df.loc[:, locs].apply(func = corr).lt(0.05) # 유의확률이 0.05보다 작은지 여부를 데이터프레임으로 반환합니다.
```

t-검정의 종류

- t-검정은 다음과 같이 세 가지 종류가 있습니다.

구분	단일 표본 t-검정	대응 표본 t-검정	독립 2 표본 t-검정
내용	표본 통계량과 모평균 비교	짝을 이루는 두 집단의 평균 비교	서로 독립인 두 집단의 평균 비교
예시	A 중학교 남학생의 평균 키는 170cm 이상인가?	다이어트 약 복용 전/후 평균 체중이 다른가?	B 중학교 남/여학생 간 평균 체중이 다른가?
기본 가정	정규성	정규성	독립성, 정규성, 등분산성
정규성 가정 만족 안함	윌콕슨 부호순위 검정	윌콕슨 부호순위 검정	윌콕슨 순위합 검정

독립 2 표본 t-검정

- 독립 2 표본 t-검정은 두 집단의 평균이 같은지 확인할 때 실행합니다.
 - 세 개 이상의 집단 간 평균 비교는 분산분석을 실행합니다.
- 독립 2 표본 t-검정은 두 집단 내에서의 변화량을 고려하여 두 집단 평균 차이가 통계적으로 유의한지 확인합니다.
 - 두 집단 평균 차이가 작을수록 두 집단의 평균이 같다고 판단할 수 있습니다.
 - 두 집단 평균 차이의 표준오차가 작을수록 두 집단의 평균은 서로 다르다고 판단할 수 있습니다.
 - t-검정은 두 집단의 분산이 같다는 가정 하에 합동표본분산을 사용합니다.

[참고] t-검정 자료구조

- 범주형 입력변수와 연속형 목표변수를 표로 나타내면 아래와 같습니다.

MetColor	Price(y_{ij})	범주별 평균 (\bar{y}_i)	잔차 ($y_{ij} - \bar{y}_i$)
0	8500	9460.8	-960.8
0	10450		989.2
:	:		:
1	9450	9814.1	-364.1
1	10050		235.9
:	:		:

[참고] 목표변수에서 범주별 평균을 차감한 잔차가 정규분포해야 합니다.

독립 2 표본 t-검정의 가설 및 검정통계량

- 독립 2 표본 t-검정의 가설 및 검정통계량은 다음과 같습니다.
 - 귀무가설(H_0): $\mu_1 = \mu_2$
 - 대립가설(H_1): $\mu_1 \neq \mu_2$
 - 유의수준: 5%(양측검정)
 - 검정통계량: $t = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{\sqrt{s_p^2 \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}}$

두 표본평균의 차이가 작을수록 t 통계량은 0에 수렴합니다.

귀무가설에 의해 두 모평균의 차이는 0입니다.

분모는 두 표본으로부터 계산한 편차 제곱합을 자유도 합계로 나눈 합동표본분산 pooled sample variance을 사용한 표준오차입니다.
- $$s_p^2 = \frac{(n_1 - 1) s_1^2 + (n_2 - 1) s_2^2}{(n_1 - 1) + (n_2 - 1)}$$
- # 합동표본분산은 모분산의 불편추정량이며 두 표본분산이 동일하다는 가정 하에서 계산된 가중평균입니다.

독립 2 표본 t-검정의 가정

- 입력변수가 두 가지 범주를 갖는 범주형이고 목표변수가 연속형이면 독립 2 표본 t-검정을 실행하여 입력변수 범주별 목표변수 평균이 같은지 확인합니다.
 - 입력변수 범주별 목표변수 평균이 다르면 선형 회귀모형의 입력변수로 추가합니다.
- 독립 2 표본 t-검정을 실행할 표본은 잔차의 독립성, 정규성 및 등분산성 가정을 만족해야 합니다.
 - 독립성 가정은 데이터를 수집하는 과정에서 두 집단 간 차이에 영향을 주는 외부 요인이 없어야 한다는 것을 의미합니다. 즉, 수집한 데이터는 독립표본이어야 합니다.
 - [참고] 실험 데이터는 무작위화^{randomization}를 통해 외부 요인을 상쇄시킬 수 있습니다.
 - 입력변수 범주에 관계 없이 잔차는 정규분포하고 분산이 같아야 합니다.

정규성 검정

- t-검정은 잔차의 정규성 가정을 만족해야 합니다.
 - 목표변수 개별 값에서 범주별 평균을 차감한 잔차가 정규분포하는지 확인합니다.
 - 귀무가설은 '데이터가 정규분포한다' 이므로 유의확률이 0.05 보다 크면 만족합니다.
- MetColor 범주별 Price의 정규성 검정을 실행합니다.

```
>>> pg.normality(data = df, # 데이터프레임을 지정합니다.
```

```
    dv = 'Price', # 연속형 목표변수를 지정합니다.
```

```
    group = 'MetColor', # 범주형 입력변수를 지정합니다.
```

```
    method = 'shapiro') # 정규성 검정 방식을 지정합니다.(기본값: 'shapiro')  
    [참고] 대용량 데이터는 'normaltest'를 지정합니다.
```

등분산성 검정

- 정규성 가정을 만족하는 두 집단의 등분산성 검정을 실행합니다.
 - 등분산성 검정의 귀무가설은 '두 집단의 분산을 나눈 비^{ratio}가 1이다' 이므로 유의확률이 0.05 보다 크면 두 그룹의 분산이 같다고 판단합니다.

- (정규성 가정 만족) MetColor 범주별 Price의 등분산성 검정을 실행합니다.

```
>>> pg.homoscedasticity(data = df, dv = 'Price', group = 'MetColor',  
                         method = 'levene') # 등분산 검정 방식을 지정합니다.(기본값: 'levene')  
                           [참고] 정규분포하면 'bartlett'을 지정합니다.
```

- 등분산이면 t-검정, 이분산이면 Welch의 t-검정을 실행합니다.
 - 실행 함수는 같지만 correction 매개변수에 전달하는 인수를 다르게 설정해야 합니다.

범주형 변수로 데이터프레임 분할

- MetColor 원소의 중복을 제거한 values를 생성합니다.

```
>>> values = df['MetColor'].unique(); values
```

- MetColor의 원소별 Price로 시리즈를 생성합니다.

```
>>> sp1, sp2 = [df['Price'][df['MetColor'].eq(v)] for v in values]
```

- 두 시리즈의 평균을 각각 확인합니다.

```
>>> print(sp1.mean()) # sp1은 MetColor가 '0'인 Price입니다.
```

```
>>> print(sp2.mean()) # sp2는 MetColor가 '1'인 Price입니다.
```

t-검정

- (정규성 가정 만족) 두 집단의 평균이 같은지 확인할 때 t-검정을 실행합니다.
 - 귀무가설은 '두 집단의 모평균이 같다' 이므로 유의확률이 0.05 보다 작으면 두 집단의 평균이 다르다고 판단합니다.
 - 등분산성 가정 만족 여부에 따라 correction 매개변수에 지정하는 값이 달라집니다.
- 등분산 가정된 t-검정을 실행합니다.

```
>>> pg.ttest(x = sp1, y = sp2, correction = False)
```

- 이분산 가정된 t-검정을 실행합니다.

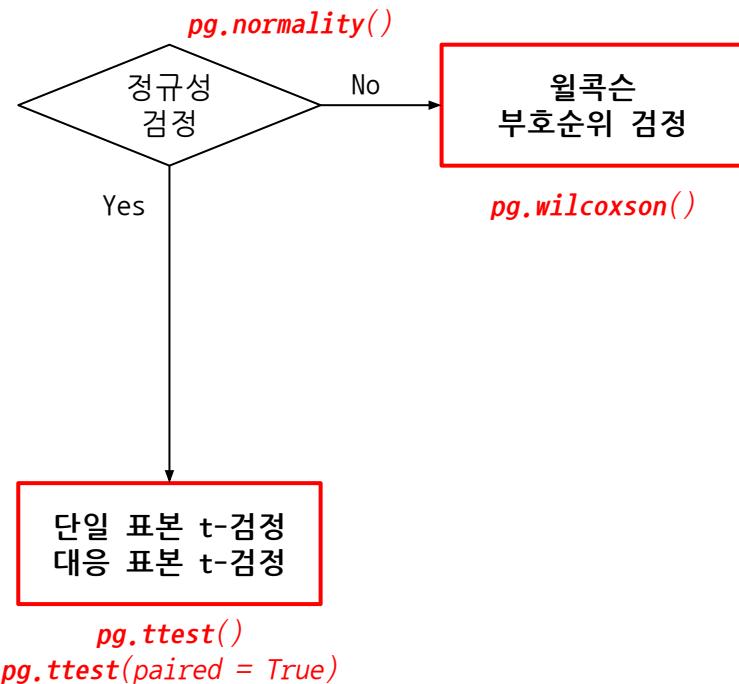
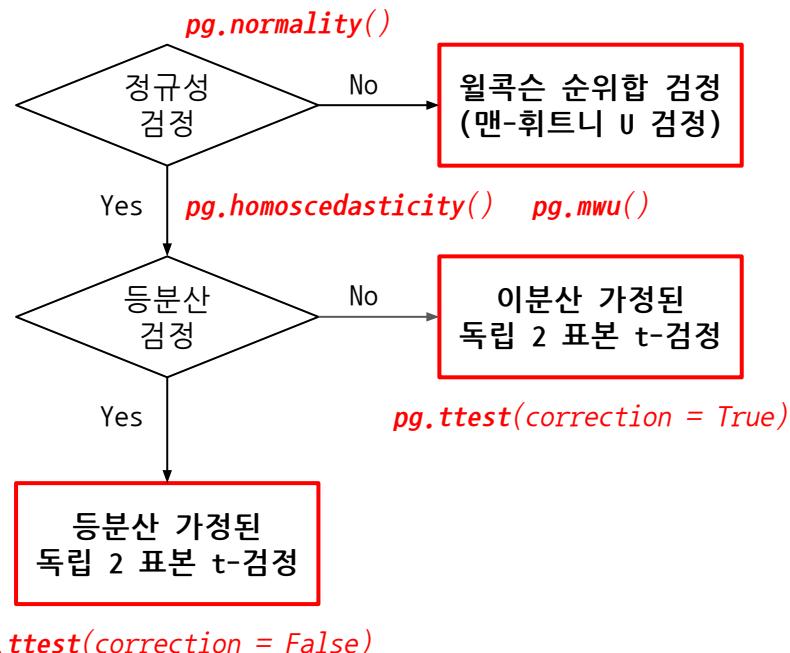
```
>>> pg.ttest(x = sp1, y = sp2, correction = True)
```

윌콕슨 순위합 검정 또는 맨-휘트니 U 검정

- 윌콕슨 순위합 검정은 데이터의 순위를 사용하여 두 집단의 중위수가 같은지 판단하는 비모수적인 검정 방법입니다. 맨-휘트니 U 검정과 같습니다.
 - 전체 데이터의 순위를 구하고 두 집단으로 분리합니다.
 - 둘 중 한 집단의 순위합을 계산합니다. 집단의 순위합으로 검정통계량을 계산합니다.
 - 귀무가설은 '두 집단의 중위수가 같다' 이므로 유의확률이 0.05 보다 작으면 두 집단의 중위수가 서로 다르다고 판단합니다.
- (정규성 가정 불만족) 맨-휘트니 U 검정을 실행합니다.

```
>>> pg.mwu(x = sp1, y = sp2)
```

[참고] t-검정 프로세스



분산분석

- 분산분석은 세 개 이상의 집단 간 평균이 같은지 여부를 확인할 때 실행합니다.
 - 하나의 요인(입력변수)에 수준이 2개 이상일 때 일원분산분석을 실행합니다.
 - 요인이 가질 수 있는 집단을 수준이라고 하고 모든 수준의 결합을 처리라고 합니다.
- 분산분석은 편차제곱합을 각각의 자유도로 나눈 평균제곱합의 비를 검정통계량으로 사용합니다.
 - 귀무가설(H_0): $\mu_1 = \mu_2 = \dots = \mu_k$ (모든 집단의 평균이 같다.)
 - 대립가설(H_1): 최소한 한 집단의 평균이 다른 집단의 평균과 다르다.
 - 분산분석의 검정통계량은 F분포를 따르며 F값이 커질수록 유의확률이 작아집니다.

[참고] 분산분석 자료구조

- i번째 FuelType(수준)의 j번째 Price(반응)를 표로 나타내면 아래와 같습니다.

FuelType	Price(y_{ij})	i번째 수준 평균(\bar{y}_i)	전체 평균(\bar{y})	총변동($y_{ij} - \bar{y}$)	설명 가능($\bar{y}_i - \bar{y}$)	설명 불가능($y_{ij} - \bar{y}_i$)
CNG	9450	9421.2	9694.7	-244.7	-273.5	28.8
CNG	9250			-444.7		-171.2
:	:			:		:
Diesel	9250	9227.7	9694.7	-444.7	-467.0	22.3
Diesel	9400			-294.7		172.3
:	:			:		:
Petrol	9795	9751.1	9694.7	100.3	56.4	43.9
Petrol	9799			104.3		47.9
:	:			:		:

분산분석표

- 관측값 y_{ij} 는 i번째 수준의 평균(\bar{y}_i)과 오차(ϵ_{ij})의 합으로 표현할 수 있습니다.
 - $y_{ij} = \bar{y}_i + \epsilon_{ij}$ (단, i는 수준, j는 관측값(행) 개수)
 - 오차는 서로 독립이고 평균이 0인 정규분포를 따르며 등분산성 가정을 만족해야 합니다.
- 총변동은 수준으로 설명 가능한 부분과 설명 불가능한 부분의 합계입니다.
 - 위 식의 양변에서 전체 평균(\bar{y})을 차감하면 아래와 같이 변경할 수 있습니다.
 - $y_{ij} - \bar{y} = (\bar{y}_i - \bar{y}) + (y_{ij} - \bar{y}_i)$ # 오른쪽 두 번째 항은 위 식을 오차항으로 정리한 것입니다.
 - $\bar{y}_i - \bar{y}$: 수준 i로 설명할 수 있는 부분입니다.
 - $y_{ij} - \bar{y}_i$: 수준 i로 설명할 수 없는 부분입니다.

분산분석표(계속)

- 각 항의 제곱합을 정리하면 아래 수식이 성립합니다.(뒷 페이지 참조)

$$\sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y})^2 = \sum_{i=1}^r n(\bar{y}_i - \bar{y})^2 + \sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y}_i)^2$$

SST
 $SSTR$
 SSE

- 처리와 오차의 제곱합을 나눈 비^{ratio}로 F-통계량과 유의확률을 확인합니다.

구분	제곱합(SS)	자유도(df)	평균제곱합(MS)	F-통계량	유의확률
처리(TR)	SSTR	r-1	$MSTR = SSTR \div (r-1)$	$MSTR / MSE$	p-value
오차(E)	SSE	n-r	$MSE = SSE \div (n-r)$		
전체(T)	SST	n-1			

[참고] 분산분석 증명

$$\sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y})^2 = \sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y}_i + \bar{y}_i - \bar{y})^2$$

$$= \sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y}_i)^2 + \sum_{i=1}^r \sum_{j=1}^n 2(y_{ij} - \bar{y}_i)(\bar{y}_i - \bar{y}) + \sum_{i=1}^r \sum_{j=1}^n (\bar{y}_i - \bar{y})^2$$

그런데,

$$\sum_{j=1}^n (y_{ij} - \bar{y}_i) = \sum_{j=1}^n y_{ij} - n\bar{y}_i = \sum_{j=1}^n y_{ij} - n \times \frac{1}{n} \sum_{j=1}^n y_{ij} = 0$$

따라서,

$$\sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y})^2 = \boxed{\sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y}_i)^2} + \boxed{\sum_{i=1}^r n(\bar{y}_i - \bar{y})^2}$$

수준 i 로 설명
가능한 부분

수준 i 로 설명
불가능한 부분

분산분석의 가정

- 분산분석은 잔차의 독립성, 정규성 및 등분산성 가정을 만족해야 합니다.
- 독립성 가정은 집단 간 차이에 영향을 주는 외부 요인이 없어야 한다는 것입니다.
- 정규성 가정은 잔차가 정규분포하는지 확인합니다.
 - 정규성 가정을 만족하면 등분산성 검정 결과를 확인하고 분산분석을 실행합니다.
 - 정규성 가정을 만족하지 못하면 크루스칼-왈리스 순위합 검정을 실행합니다.
- 범주가 세 개 이상인 데이터의 등분산성 검정은 바틀렛 검정을 실행합니다.
 - 바틀렛 검정은 정규분포하는 연속형 변수의 분산이 같은지 확인합니다.
 - 연속형 변수가 정규분포하지 않으면 레벤 검정을 대신 실행합니다.

정규성 검정

- 분산분석은 잔차의 정규성 가정을 만족해야 합니다.
 - 목표변수 개별 값에서 범주별 평균을 차감한 잔차가 정규분포하는지 확인합니다.
 - 귀무가설은 '데이터가 정규분포한다' 이므로 유의확률이 0.05 보다 크면 만족합니다.
- FuelType 범주별 Price의 정규성 검정을 실행합니다.

```
>>> pg.normality(data = df, # 데이터프레임을 지정합니다.
```

```
        dv = 'Price', # 연속형 목표변수를 지정합니다.
```

```
        group = 'FuelType', # 범주형 입력변수를 지정합니다.
```

```
        method = 'shapiro') # 정규성 검정 방식을 지정합니다.(기본값: 'shapiro')  
        [참고] 대용량 데이터는 'normaltest'를 대신 지정합니다.
```

등분산성 검정

- 정규성 가정을 만족하는 셋 이상 집단의 등분산성 검정을 실행합니다.
 - 바틀렛 검정의 귀무가설은 '모든 집단의 분산은 같다' 이므로 유의확률이 0.05 보다 크면 모든 집단의 분산이 같다고 판단합니다.
- (정규성 가정 만족) FuelType 범주별 Price의 등분산성 검정을 실행합니다.

```
>>> pg.homoscedasticity(data = df, dv = 'Price', group = 'FuelType',  
                         method = 'levene') # 등분산 검정 방식을 지정합니다. (기본값: 'levene')  
                           [참고] 정규분포하면 'bartlett'을 지정합니다.
```

- 등분산이면 분산분석, 이분산이면 Welch의 분산분석을 실행합니다.
 - [참고] 분산분석과 Welch의 분산분석을 실행하는 함수가 다릅니다.

분산분석

- (정규성 가정 만족) 셋 이상의 집단 간 평균이 같은지 분산분석을 실행합니다.
 - 귀무가설은 '모든 집단의 모평균이 같다' 이므로 유의확률이 0.05 보다 작으면 최소한 한 집단의 평균이 다른 집단과 다르다고 판단합니다.
 - 등분산성 가정 만족 여부에 따라 분산분석을 실행하는 함수가 달라집니다.
- 등분산 가정된 분산분석을 실행합니다.

```
>>> pg.anova(data = df, dv = 'Price', between = 'FuelType')
```

- 이분산 가정된 분산분석을 실행합니다.

```
>>> pg.welch_anova(data = df, dv = 'Price', between = 'FuelType')
```

크루스칼-왈리스 검정

- 크루스칼-왈리스 검정은 데이터의 순위를 사용하여 셋 이상 집단 간 중위수가 서로 같은지 판단하는 비모수적인 검정 방법이며 월록슨 순위합 검정의 확장판입니다.
 - 전체 데이터의 순위를 구하고 셋 이상의 집단으로 분리합니다.
 - 모든 집단의 순위합을 계산합니다. 모든 집단의 순위합으로 검정통계량을 계산합니다.
 - 귀무가설은 '모든 집단의 중위수가 같다' 이므로 유의확률이 0.05 보다 작으면 최소한 한 집단의 중위수가 다른 집단과 다르다고 판단합니다.
- (정규성 가정 불만족) 크루스칼-왈리스 순위합 검정을 실행합니다.

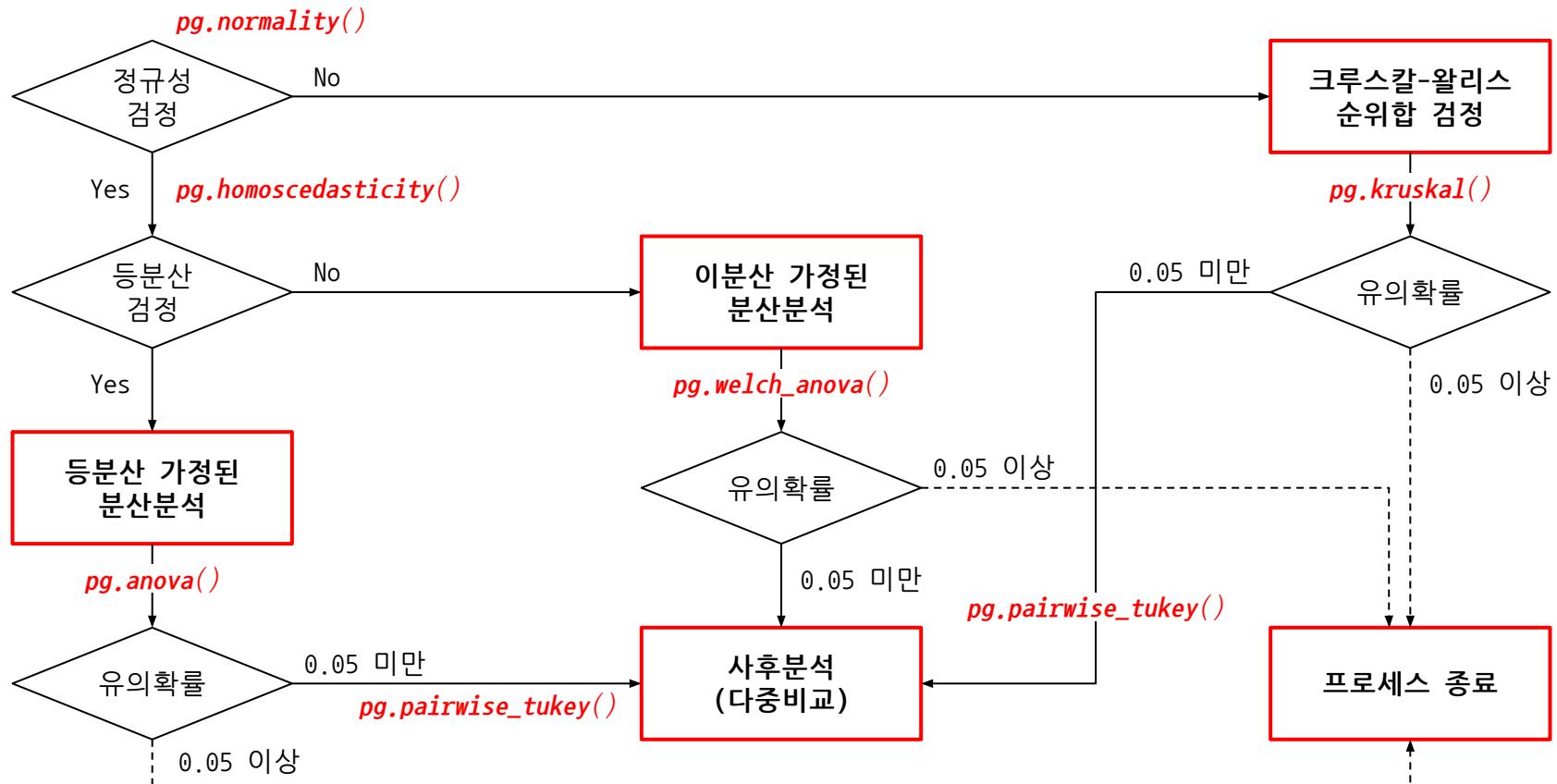
```
>>> pg.kruskal(data = df, dv = 'Price', between = 'FuelType')
```

사후분석

- 분산분석 실행 결과, 유의확률이 0.05 보다 작으면 귀무가설을 기각할 수 있으므로 최소한 한 집단의 평균이 다른 집단과 다르다고 판단합니다.
- 사후분석 Post-Hoc은 어떤 집단 간 평균이 다른지 확인하며 다중비교라고도 합니다.
- 사후분석 방식은 Tukey, Scheffe, Bonferroni, Duncan, Dunnett 등이 있습니다.
 - 수준별로 표본 크기가 같고 모든 경우의 수에서 집단 간 차이를 확인할 때 Tukey 방식을 사용합니다.
- Tukey 방식으로 사후분석을 실행합니다.

```
>>> pg.pairwise_tukey(data = df, dv = 'Price', between = 'FuelType')
```

[참고] 분산분석 프로세스



교차분석: 카이제곱 검정

- 입력변수 범주별로 목표변수의 빈도수 차이가 같은지 여부를 확인할 때 교차 분석(카이제곱 검정)을 실행합니다.
- 입력변수 범주별로 목표변수 실제값과 기대값의 차이가 크면 카이제곱 통계량 또한 커집니다.(오른쪽 표 참조)
 - 카이제곱 통계량이 증가하면 유의확률이 감소하며 유의확률이 0.05 보다 작을 때 귀무가설을 기각할 수 있습니다.

구분	목표변수		소계
	범주a	범주b	
입력	범주1	O_{1a}	$O_{1\cdot}$
변수	범주2	O_{2a}	$O_{2\cdot}$
소계	$O_{\cdot a}$	$O_{\cdot b}$	Total

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad \begin{cases} O_{ij} : \text{관측값 개수} \\ E_{ij} : \text{기댓값 개수} \end{cases}$$

단, $E_{ij} = \text{Total} \times \frac{O_{i\cdot}}{\text{Total}} \times \frac{O_{\cdot j}}{\text{Total}}$

교차테이블 출력

- 두 범주형 변수의 빈도수를 출력합니다.

```
>>> pd.crosstab(index = df['MetColor'],  
                 columns = df['Automatic']) # [참고] index에 입력변수, columns에 목표변수를  
                                            지정하면 결과를 해석하기 좋습니다.
```

- 두 범주형 변수의 상대도수를 출력합니다.

```
>>> pd.crosstab(index = df['MetColor'],  
                 columns = df['Automatic'],  
                 normalize = 'index', # normalize 매개변수에 'index'를 지정하면 각 행별 합계가  
                                         1이 되도록 상대도수를 반환합니다.  
                 margins = True) # margins 매개변수에 True를 지정하면 맨 아래에 합계를 추가합니다.  
                               # [참고] normalize 매개변수에 지정한 값에 따라 결과가 달라집니다.
```

카이제곱 검정

- 카이제곱 검정은 두 범주형 변수의 범주별로 빈도수 차이가 있는지 확인합니다.
 - 귀무가설이 '집단 간 빈도수 차이가 없다'이므로 유의확률이 0.05 보다 작으면 집단 간 빈도수 차이가 유의하다고 판단합니다.
- 두 범주형 변수로 교차분석(카이제곱 검정)을 실행합니다.

```
>>> pg.chi2_independence(data = df, x = 'MetColor', y = 'Automatic')
```

	Automatic	0	1					
	MetColor	0	1					
0	404.372	23.628						
1	793.628	46.372,						
Automatic	0	1						
MetColor	0	402.5	25.5					
1	795.5	44.5,						
		test	lambda	chi2	dof	pval	cramer	power
0	pearson	1.000	0.237	1.0	0.626	0.626	0.014	0.078
1	cressie-read	0.667	0.236	1.0	0.627	0.627	0.014	0.077
2	log-likelihood	0.000	0.234	1.0	0.628	0.628	0.014	0.077
3	freeman-tukey	-0.500	0.233	1.0	0.629	0.629	0.014	0.077
4	mod-log-likelihood	-1.000	0.232	1.0	0.630	0.630	0.014	0.077
5	neyman	-2.000	0.229	1.0	0.632	0.632	0.013	0.077)

위 코드를 실행하면 기댓값 행렬, 관측값 행렬 및 다양한 카이제곱 검정 결과를 반환합니다.

피어슨 검정의 유의확률로 귀무가설 기각 여부를 판단합니다.

변수 제거 및 z 파일로 저장

- 가설검정 결과를 반영하여 데이터프레임을 전처리합니다.

```
>>> df = df.drop(columns = ['CC', 'Automatic']) # CC와 Automatic을 삭제합니다.
```

```
>>> df = df[df['FuelType'].ne('CNG')] # FuelType이 'CNG'인 건수가 매우 적고 Petrol 및 Diesel과  
목표변수 평균에서 유의한 차이가 없으므로 삭제합니다.
```

- df의 행이름을 초기화합니다.

```
>>> df = df.reset_index(drop = True)
```

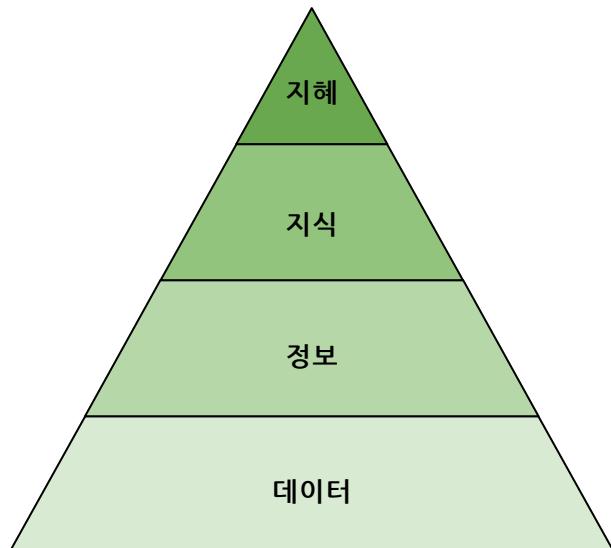
- df를 z 파일로 저장합니다.

```
>>> joblib.dump(value = df, filename = 'Used_Cars_Price_Prep.z')
```

데이터 분석 개요

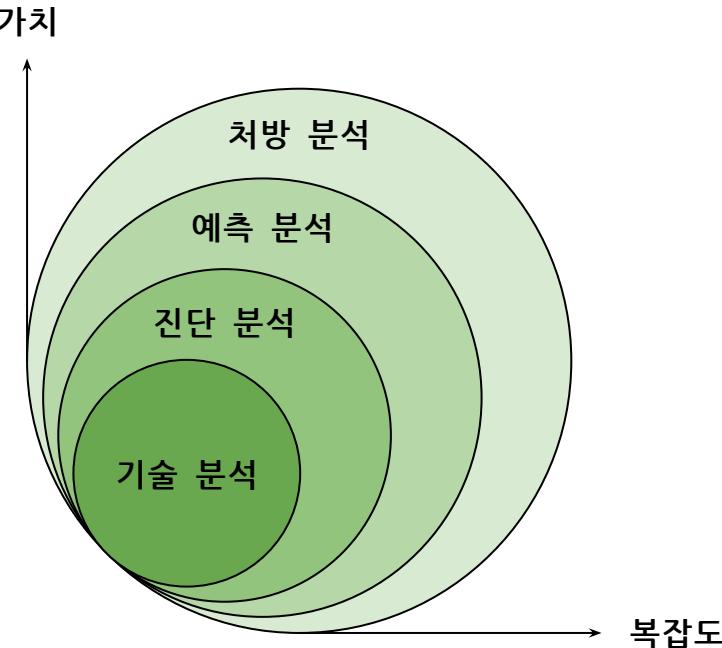
DIKW 피라미드

- DIKW 피라미드는 데이터^{data}, 정보^{information}, 지식^{knowledge}, 지혜^{wisdom}의 관계를 표현한 것입니다.



- 지혜: 축적한 지식 기반으로부터 도출한 창의적인 아이디어
- 지식: 유용한 정보에 개인의 경험을 결합하여 내재화한 것
- 정보: 데이터를 가공하여 패턴 및 의미를 확인한 것
- 데이터: 가공하기 전 상태의 객관적인 사실(수, 문자열, 기호)

데이터 분석의 4가지 유형



- 기술 분석: 무엇이 발생했는가?
 - 과거에 발생한 사건을 다양한 관점에서 정리
- 진단 분석: 원인이 무엇인가?
 - 과거에 발생한 사건의 인과관계 패턴에 초점
- 예측 분석: 앞으로 어떻게 될 것인가?
 - 데이터의 패턴을 파악하여 가까운 미래 예측
- 처방 분석: 무엇을 해야 하는가?
 - 미래 예측에 대한 다양한 대응 방안 검토

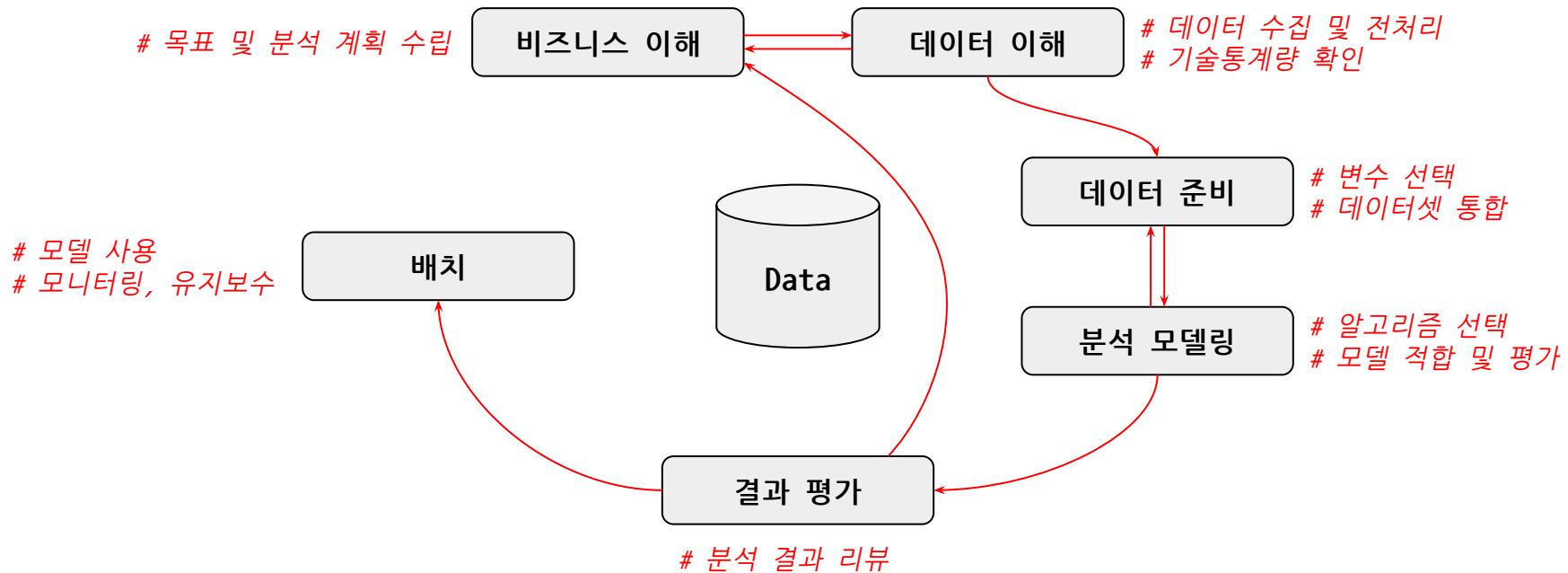
데이터 분석의 4가지 유형

구분	분석 방법	예시
기술 분석 <small>descriptive analysis</small>	<ul style="list-style-type: none"> 데이터 시각화, 기술통계 탐색적 데이터 분석 	<ul style="list-style-type: none"> 신규고객 유입 채널별 비율과 매출액 변화 확인 당사 고객의 인구통계학적 특성(성비, 연령 등)
진단 분석 <small>diagnostic analysis</small>	<ul style="list-style-type: none"> 상관관계 분석, 가설 검정 데이터 마이닝(원인 파악) 	<ul style="list-style-type: none"> 신규고객 매출액이 감소한 이유는 최근 시행한 이벤트로 유입된 고객 비중이 증가했기 때문
예측 분석 <small>predictive analysis</small>	<ul style="list-style-type: none"> 머신러닝(원인보다 예측력) 딥러닝 	<ul style="list-style-type: none"> 신규고객 유입 채널 비중을 현재 수준으로 유지한다면 신규고객 매출액은 계속 감소할 것임
처방 분석 <small>prescriptive analysis</small>	<ul style="list-style-type: none"> 예측 상황별 시뮬레이션 최적화 분석 	<ul style="list-style-type: none"> 매출액 감소를 예상하는 상황에서 현재 적용한 이벤트 예산을 어디에 지출해야 하는지 검토

데이터 분석 방법

구분	상세 내용
데이터 시각화	<ul style="list-style-type: none"> 일변량 데이터의 분포, 이변량 데이터의 관계를 시각적으로 확인함으로써 분석할 데이터에 대한 이해의 폭을 넓힐 수 있습니다.
기술통계 분석	<ul style="list-style-type: none"> 기술통계는 데이터의 주요 특징을 빠르게 파악할 때 사용하는 통계기법입니다. 평균, 중위수 등 대푯값과 분산, 표준편차 등 흩어진 정도를 파악합니다.
통계적 가설 검정	<ul style="list-style-type: none"> 통계적 가설검정은 모수를 이용하여 어떤 가설에 대해 합당한 것인지를 판단하는 과정입니다. 피어슨 상관분석, t-검정, 분산분석 및 교차분석 등을 실행합니다.
데이터 마이닝	<ul style="list-style-type: none"> 입력변수(원인)와 목표변수(결과)의 관계를 확인하는 분석방법입니다. 목표변수의 형태에 따라 선형 회귀분석(연속형)과 로지스틱 회귀분석(범주형)을 실행합니다.
머신러닝/딥러닝	<ul style="list-style-type: none"> 목표변수의 원인 규명보다 결과에 대한 정확한 예측에 중점을 둔 분석방법입니다. 대표적인 알고리즘으로 의사결정나무, 랜덤 포레스트 등이 있습니다.

데이터 마이닝 프로세스: CRISP-DM Framework



머신러닝의 이해

- 머신러닝은 컴퓨터로 데이터를 학습할 때 사용하는 다양한 알고리즘을 포함하는 기법을 의미합니다.

인공지능 Artificial Intelligence

인간에게는 어려운
과업을 컴퓨터가
지능적으로 수행하는
일련의 서비스

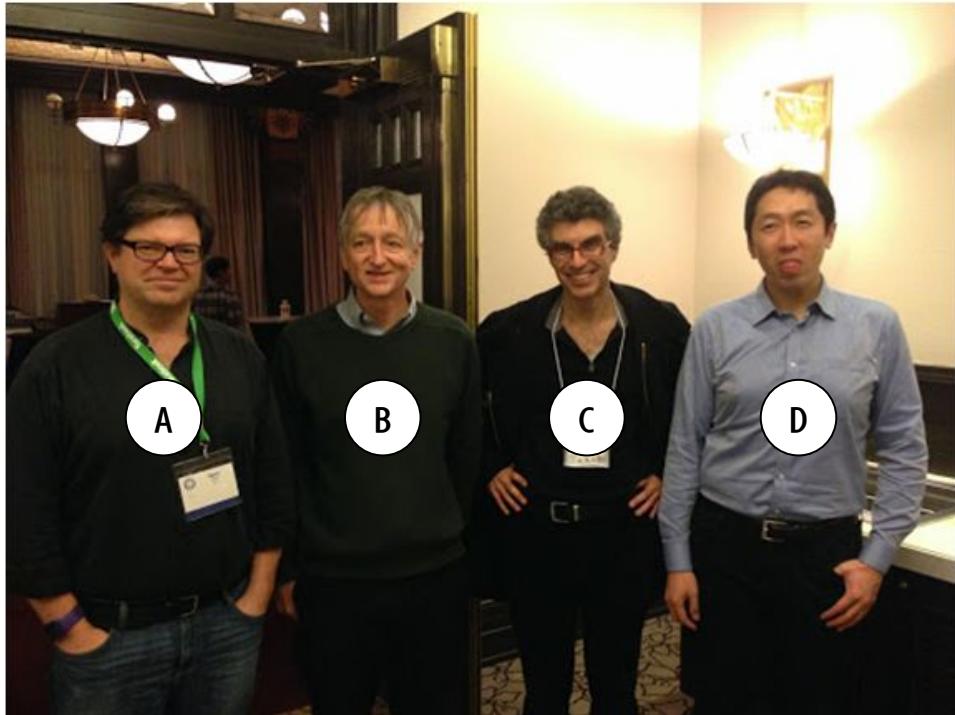
머신러닝 Machine Learning

컴퓨터가 다양한
알고리즘을 이용하여
데이터에 내재된 패턴을
학습하고, 정보를 제공

딥러닝 Deep Learning

인간의 뇌(뉴런)가
학습하는 방법을 모사한
인공신경망^{ANN} 알고리즘을
복잡한 형태로 개선한 것

[참고] 인공지능 4대천왕



A. 얀 르쿤

- 뉴욕대 교수
- 페이스북 AI 수장
- CNN 모델 개발

B. 제프리 힌튼

- 토론토대 교수
- 구글 AI 수장
- 딥러닝 개념 창시자

C. 요슈아 벤지오

- 몬트리올대 교수
- 밀라 MILA 연구소 설립
- 딥러닝 자연어처리 창안

D. 앤드류 응

- 스탠포드대 교수
- 구글 브레인, 바이두를 거쳐 랜динAI 창업

출처 : <https://www.kdnuggets.com/2015/03/talking-machine-deep-learning-gurus-p1.html>

머신러닝의 개념과 한계

- 머신러닝은 데이터에 잠재된 패턴을 학습함으로써 실생활에서의 문제를 해결하고 왔습니다.
 - 로지스틱 회귀분석은 제왕절개 추천 여부에 관한 0~1의 추정 확률을 제시합니다.
 - 나이브 베이즈는 이메일 제목/본문에 포함된 단어로 스팸 메일을 분류합니다.
- 분석가가 목표변수^{label}와 입력변수^{feature}를 지정해주어야 한다는 단점이 있습니다.
 - 머신러닝 알고리즘은 문제 해결을 위해 어떤 입력변수가 필요한지 모릅니다.
 - 머신러닝의 성과는 데이터 품질에 크게 의존합니다.
 - Garbage in, garbage out

머신러닝의 도전과 해결책

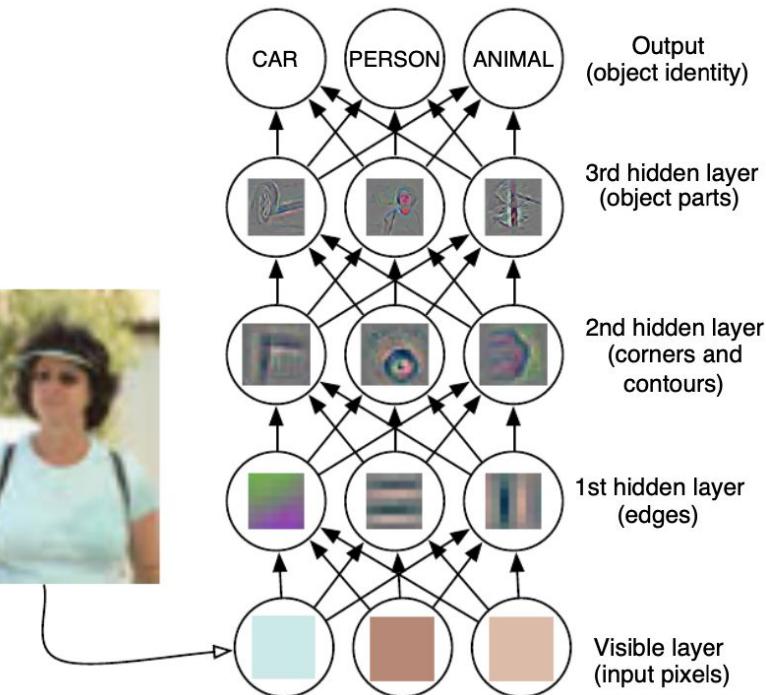
- 머신러닝은 수학적인 규칙을 통해 인간이 풀기 어려운 형식적인 문제를 컴퓨터로 쉽게 해결해왔습니다.
- 그런데 인간은 직관적으로 해결할 수 있지만 형식적으로 서술하기 어려운 문제들, 예를 들어 글자나 이미지를 인식하는 문제는 기호와 내용을 연결하지 못하는 특성 때문에 쉽게 해결하지 못했습니다.
- 하지만 컴퓨터가 개념의 계통구조를 이용하여 과거 데이터로부터 학습하는 구조를 설정함으로써 인간이 문제를 형식적으로 지정해주지 않아도 스스로 개념을 배울 수 있게 되었습니다.
 - 개념의 연결 관계를 도식화하면 여러 층^{layers}을 형성하는데 이를 딥러닝이라고 합니다.

딥러닝의 출현

- 딥러닝의 본질적인 모형은 순방향 심층 신경망 또는 다층 퍼셉트론입니다.
 - 다층 퍼셉트론은 인간의 뇌에 존재하는 뉴런을 모방한 것인데, 입력층과 출력층 사이에 다수의 은닉층을 설정하고 입력층 - 은닉층 - 출력층 간 연결고리를 간단한 수학 함수로 표현한 것입니다.
 - 각 연결고리마다 가중치^{weight}를 부여하며 이전 단계의 출력은 이후 단계의 입력이 됩니다.
 - 각 단계마다 가중치와 입력값으로 출력값을 계산하며 출력값이 활성함수^{activation function}에 미리 설정해놓은 임계점^{threshold}을 넘으면 다음 단계로 자극을 전달합니다.
- 딥러닝은 머신러닝의 한 종류이며 원시 데이터로부터 스스로 특징 집합을 추출할 수 있는 능력이 있으므로 머신러닝이 풀지 못한 많은 문제를 해결하고 있습니다.

딥러닝 모델의 예시

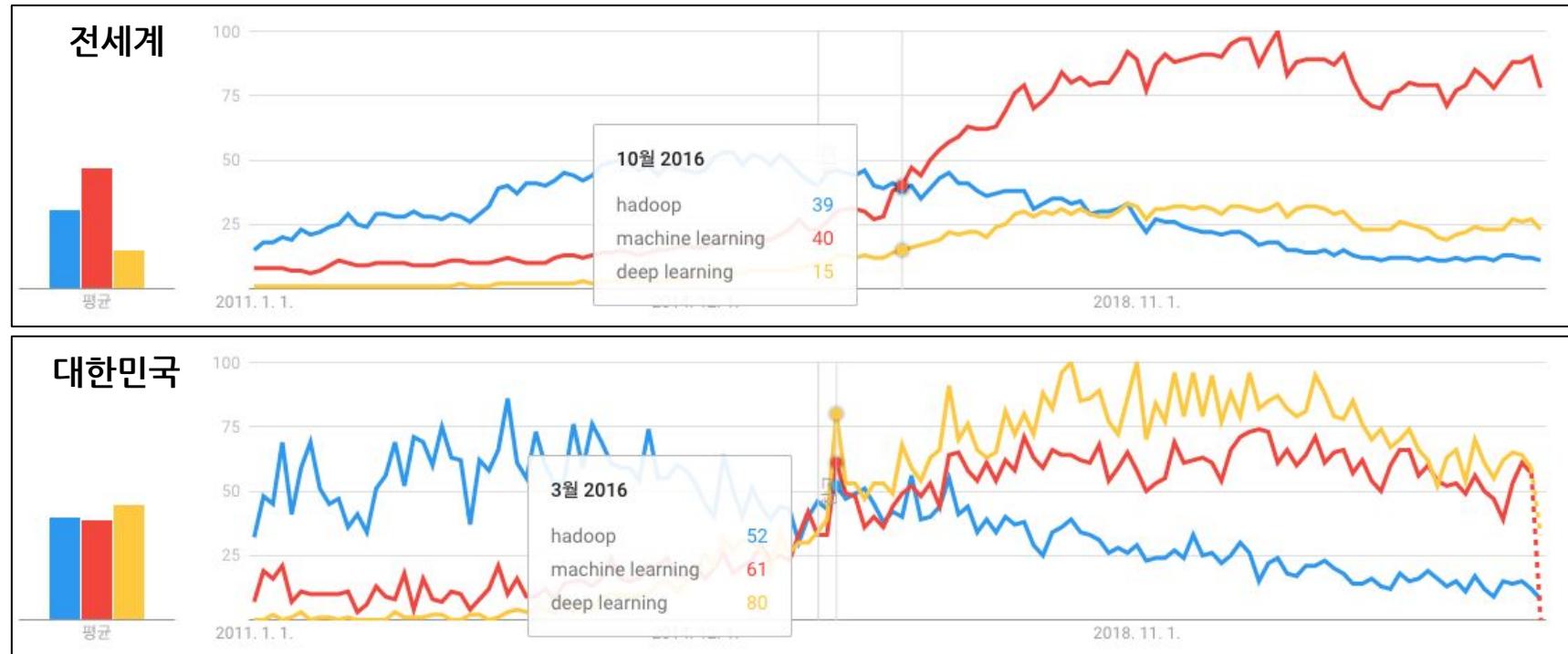
- 입력층에 이미지의 픽셀 데이터(1차원 배열)를 지정합니다.
 - 2차원 배열을 1차원으로 변환합니다.
- 여러 겹의 은닉층을 거치면서 추상화된 특징을 추출합니다.
- 각 연결고리에 부여한 가중치에 따라 출력값이 달라집니다.
- 활성함수가 출력값을 전달할지 여부를 결정합니다.



딥러닝 활용 사례

- 딥러닝은 음성 인식 분야에서 오류율을 크게 낮췄습니다.
- 보행자 인식 및 영상 분야에서 주목할만한 성공을 거두었습니다.
- 교통 표지판 분류에서 인간을 능가하는 성과를 보였습니다.
- 제시된 이미지에서 하나의 형체가 아닌 모든 문자를 인식합니다.
- 순환 신경망 Recurrent Neural network, RNN과 LSTM Long Short-Term Memory 알고리즘은 시퀀스 관계를 모형화하므로 시계열 분석 및 기계번역에서 혁신적인 발전을 가능하게 했습니다.
 - GPT-n 시리즈는 기계번역에 괄목할만한 혁신을 이루었습니다.
- 딥마인드는 Atari와 AlphaGo 등 게임 분야에서 딥러닝 강화학습을 활용했습니다.

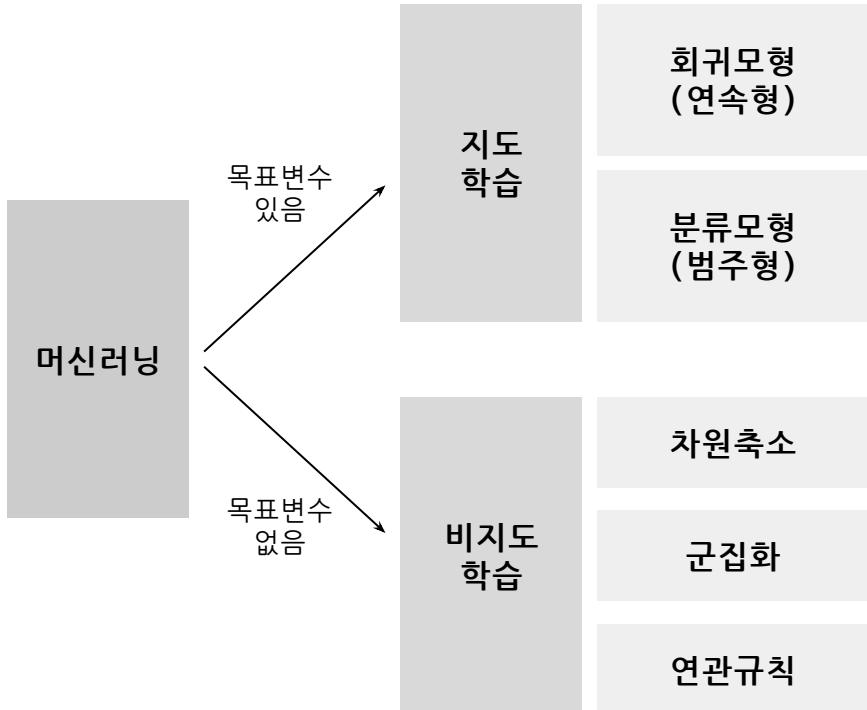
구글 트렌드로 살펴본 관심도 차이(전세계 vs 대한민국)



출처: 구글 트렌드(전세계/대한민국, 2022.6.1 기준)

머신러닝의 종류

- 머신러닝은 목표변수의 유무에 따라 지도학습과 비지도학습으로 구분하며 강화학습도 포함합니다.
- 지도학습은 목표변수의 형태에 따라 회귀모형(연속형)과 분류모형(범주형)으로 구분합니다.
- 비지도학습은 목표변수가 없으므로 데이터의 패턴을 파악하는 차원축소, 군집화 및 연관규칙 등이 있습니다.



[참고] 머신러닝 알고리즘의 종류

분류

- Logistic Regression
- Decision Tree
- Naive Bayes
- KNN(K-nearest neighbors)
- Random Forest
- GBM, XGBoost
- Support Vector Machine
- ANN(Artificial Neural Network)

회귀

- Linear Regression(Stepwise)
- Regularized Linear Regression
- Regression Tree
- KNN(K-nearest neighbors)
- Random Forest
- GBM, XGBoost
- Support Vector Machine
- ANN(Artificial Neural Network)

차원축소

- PCA(Principal Component Analysis)
- Factor Analysis
- MDS(Multi-Dimensional Scaling)

군집화

- Hierarchical Clustering
- K-means Clustering
- K-medoids Clustering
- SOM(Self-Organizing Map)

연관규칙

- MBA(Market Basket Analysis)
- Sequence MBA
- Collaborative Filtering

지도학습

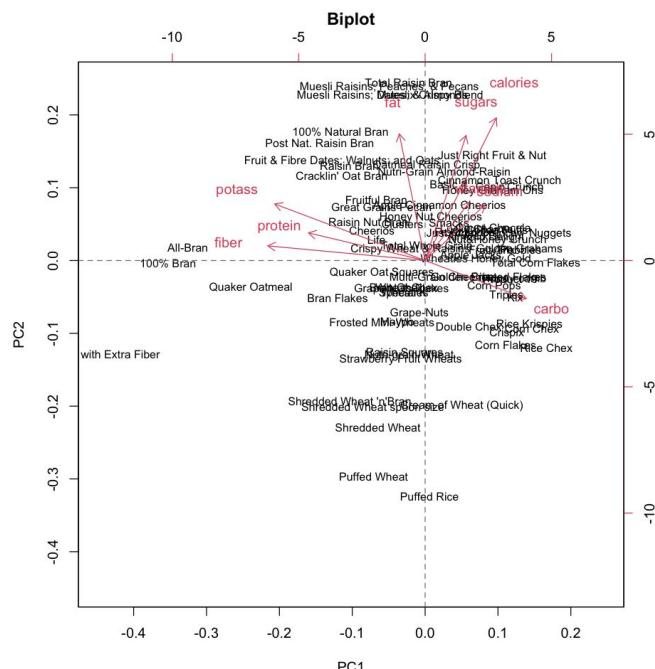
- 목표변수가 연속형이면 회귀모형 Regression을 적합합니다. # [참고] '적합한다'는 모형을 생성한다는 동사 'fit'을 번역한 것입니다.
 - 목표변수가 연속형이므로 회귀모형은 연속형 추정값을 반환합니다.
 - 목표변수를 범주형으로 변환하면 회귀모형을 분류모형으로 바꿀 수 있습니다.
 - 목표변수가 0~100인 실수형 변수를 구간화하여 '합격' 또는 '불합격'인 범주형 변수로 변환하면 분류모형을 적합하는 문제로 바뀝니다.
- 목표변수가 범주형이면 분류모형 Classification을 적합합니다.
 - 목표변수의 범주(레벨)가 두 가지(예를 들어 '예' 또는 '아니오')인 이진 분류와 범주가 여러 개인 다항 분류가 있습니다.
 - 분류모형은 실수형 추정확률(0~1) 및 범주형 추정값(라벨)을 반환합니다.

비지도학습

- 차원축소 Dimension Reduction는 p 열 column 을 m 개로 축소합니다. ($p > m$)[#] [참고] 차원은 열 개수를 의미합니다.
 - 차원축소는 주성분분석 PCA 을 많이 사용하며, $m = 2$ 이면 주성분점수로 산점도를 그립니다.
- 군집화 Clustering 는 n 행 row 을 k 개의 세부 군집으로 나눕니다.
 - 군집화는 행 row 을 축소한다는 점에서 차원축소와 다릅니다.
 - p 차원의 특성(열)이 유사한(거리가 짧은) 관측값(행)을 같은 군집으로 묶습니다.
- 연관규칙 Association Rule 은 조건부 확률을 이용하여 연관성이 높은 규칙을 발견함으로써 추천 recommendation 등 마케팅 활동에 활용합니다.
 - 아빠들이 대형마트에서 기저귀와 맥주를 함께 구매할 확률이 높다는 사례가 있습니다.

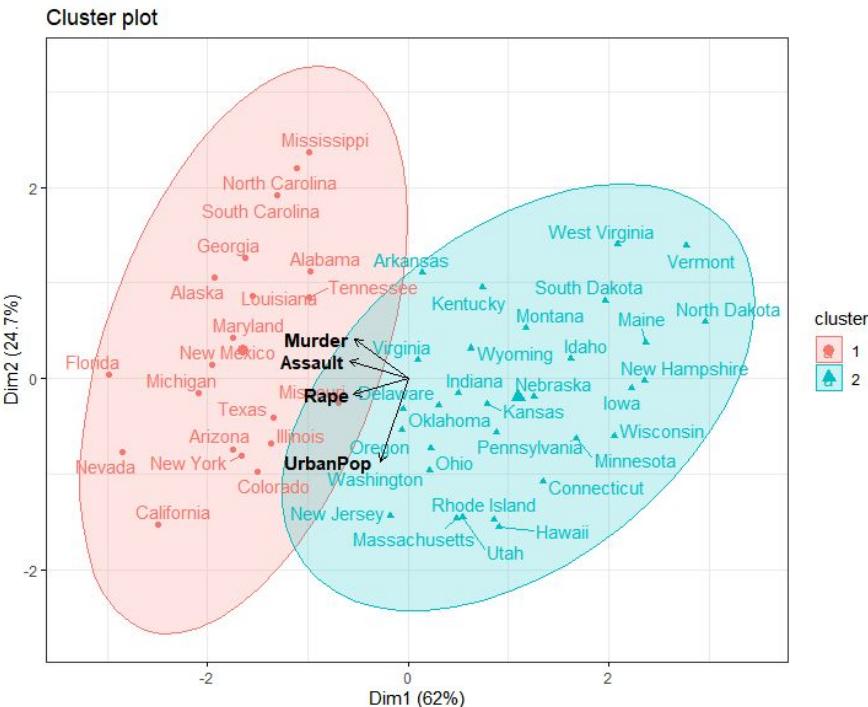
[참고] 차원축소: Principal Component Analysis

- 주성분분석은 p차원의 변수로 구성된 데이터셋을 m차원의 주성분으로 축소합니다.
 - 2차원으로 축소하면 산점도를 그릴 수 있으므로 전체 데이터를 한 눈에 파악할 수 있습니다.
 - 주성분은 상관관계가 높은 변수의 선형결합이며 계수 coefficients 가 큰 변수들로부터 주성분의 특징을 파악합니다.
 - 주성분은 직교하기 때문에 상관계수는 0입니다.
 - 회귀모형의 다중공선성 문제를 해결할 수 있습니다.



[참고] 군집분석: K-means Clustering

- k-means는 각 군집의 중심에서 가까운 관측값을 같은 군집으로 묶습니다.
 - 군집의 중심과 관측값 간 유클리드 거리를 계산합니다.
 - 거리를 계산하기 전 반드시 데이터 표준화를 실행해야 합니다.
- 군집별로 변수의 평균을 계산하거나 행렬도로 시각화하면 군집의 특징을 쉽게 파악할 수 있습니다.



[참고] 연관규칙: Market Basket Analysis

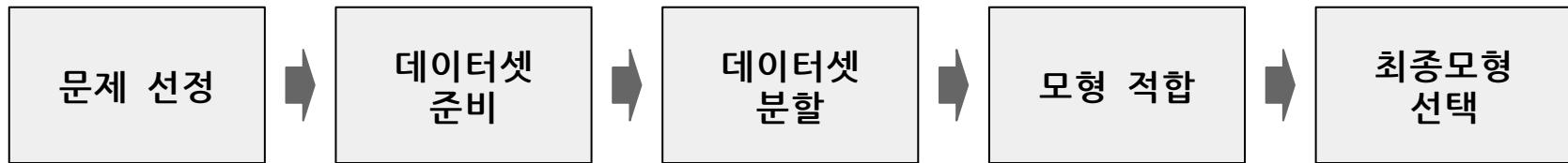
- 연관규칙은 같은 거래에서 함께 구매된 상품 정보로 동시 구매 패턴을 찾습니다.
- 연관규칙을 탐색하는 3가지 기준입니다.
 - 지지도: $P(A \cap B)$
 - 신뢰도: $\frac{P(A \cap B)}{P(A)} = P(B|A)$
 - 향상도: $\frac{P(A \cap B)}{P(A) \times P(B)} = \frac{P(B|A)}{P(B)}$
- 맥주와 기저귀 사례는 장바구니 분석의 대표적인 사례입니다.



출처: <https://www.korea.kr/news/top50View.do>

지도학습 프로세스

- 지도학습 알고리즘은 아래 과정을 거칩니다.



기업은 수익의 증대 및 비용의 감소 등 이익과 관련한 문제 해결에 많은 관심을 가지므로 관련 분석 주제 선정 및 목표 변수를 정의한 다음 다양한 분석 질문을 통해 분석의 범위를 결정합니다.

분석 질문으로부터 수집할 변수 목록을 선정하고 기업 내/외부에서 데이터를 확보합니다. 아울러 탐색적 데이터 분석으로 분석 데이터에 내재된 특징/패턴을 파악합니다.

분석할 데이터셋을 훈련셋, 검증셋 및 시험셋 등 세 가지 데이터셋으로 분할 합니다. 데이터셋의 크기에 따라 데이터 분할 방법을 다르게 적용합니다.

목표변수의 종류에 따라 분류/회귀 등 적합할 분석 모형을 결정합니다. 가능한 많은 머신러닝 알고리즘으로 많은 분석 모형을 적합합니다. 알고리�마다 같은 훈련셋을 적용해야 합니다.

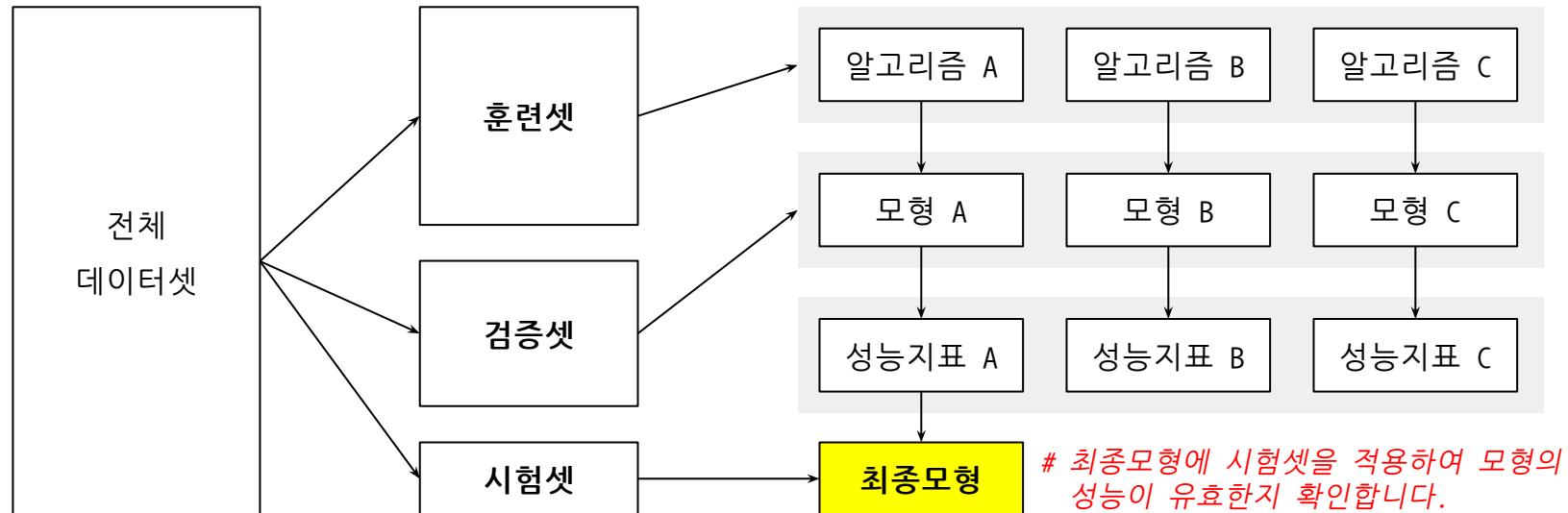
분석 모형에 검증셋으로 추정값과 추정 확률을 계산한 다음 실제값과의 차이를 비교하는 방식으로 모형의 성능지표를 계산합니다. 성능이 가장 우수한 모형을 최종모형으로 선정 합니다.

데이터셋 분할 방법

- 데이터셋이 충분히 크다고 판단하면 자료분할^{Hold-out Validation}을 사용합니다.
 - 전체 데이터셋을 훈련셋^{training}, 검증셋^{validation} 및 시험셋^{testing}으로 분할합니다.
 - 자료분할은 단순임의추출방식을 주로 사용하지만 데이터셋을 계층으로 분리해야 한다면 층화추출방식을 사용합니다.
- 데이터셋이 작다고 판단하면 k겹-교차검증^{k-folds Cross Validation}을 사용합니다.
 - 전체 데이터셋을 훈련셋과 시험셋으로 분할한 다음, 훈련셋을 k개로 등분합니다.
 - k-1개로 분류모형을 적합하고, 남은 1개로 추정값을 생성하여 성능지표를 계산합니다.
 - 위 과정을 k번 반복하여 얻은 성능지표의 평균을 해당 모형의 성능지표로 반영합니다.

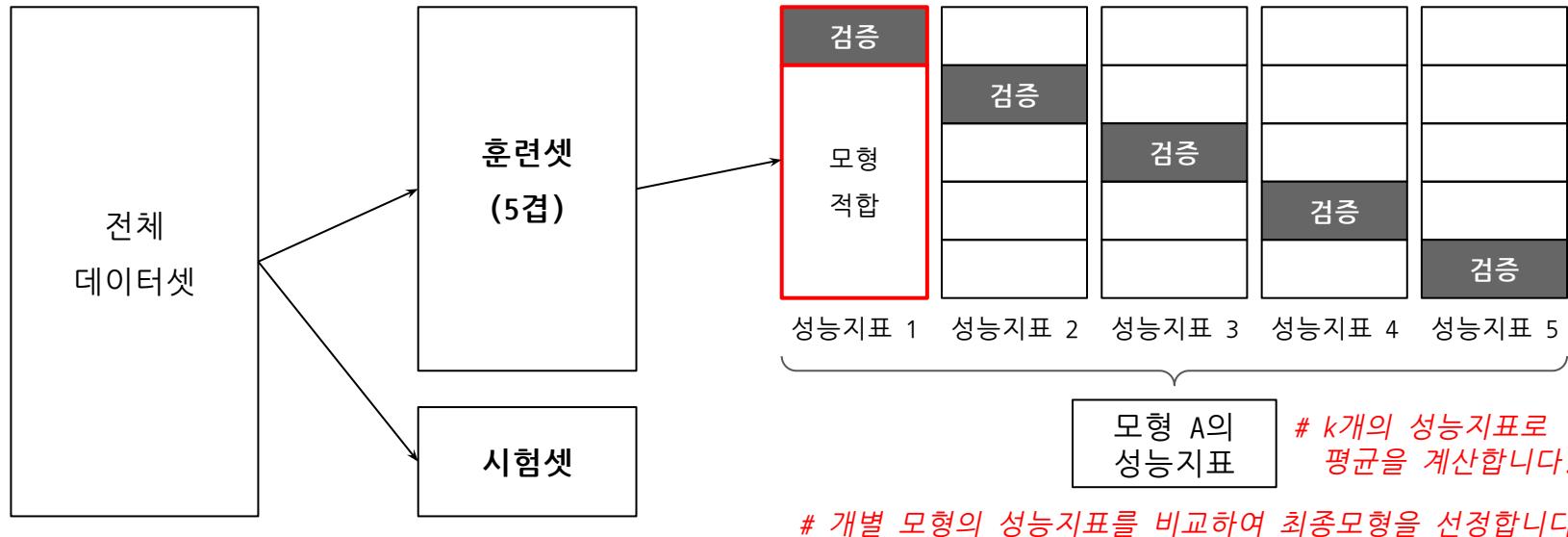
자료분할의 시각적 표현

- 자료분할은 전체 데이터셋을 훈련셋, 검증셋 및 시험셋으로 분할하는 방법입니다. 정해진 비율이 없으므로 스스로 결정합니다.



k-겹 교차검증의 시각적 표현

- 훈련셋을 k개로 등분하고 k-1개로 모형을 적합하고 남은 1개로 검증하는 것을 k번 반복하여 k개 성능지표의 평균이 우수한 최종모형을 선택합니다.



개별 모형의 성능지표를 비교하여 최종모형을 선정합니다.

회귀모형 성능 평가 기준

- 회귀모형은 목표변수가 연속형 벡터이므로 실제값과 추정값 간의 차이인 오차를 계산하여 모형의 성능을 평가합니다.

$$\text{오차}^{\text{Error}} = \text{실제값}^{\text{Actual Value}} - \text{추정값}^{\text{Predicted Value}}$$

- 모형 전체의 오차 크기를 계산할 때 개별 오차를 단순하게 합산하면 부호가 서로 다른 오차로 인해 0에 수렴하므로 오차를 제곱하거나 절대값을 취하는 방식으로 부호를 통일합니다.
- 회귀모형 성능을 평가하는 지표는 다양하지만 결과는 비슷하므로 한 가지 지표를 정하여 사용하는데 머신러닝 알고리즘은 MSE를, 사람은 RMSE를 많이 사용합니다.

회귀모형 성능 평가 지표(5종)

성능지표	세부 내용	공식
MSE Mean Squared Error	<ul style="list-style-type: none"> - 오차 제곱의 평균입니다. - 오차가 클수록 MSE는 크게 증가합니다. 	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
RMSE Root Mean Squared Error	<ul style="list-style-type: none"> - MSE의 양의 제곱근입니다. - 실제값과 척도가 같습니다. 	$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$
RMSLE Root Mean Squared Log Error	<ul style="list-style-type: none"> - 추정값과 실제값의 상대적 차이를 나타내며 과소추정에 높은 벌점을 부여합니다. 	$\sqrt{\frac{1}{n} \sum_{i=1}^n \log \left(\frac{\hat{y}_i + 1}{y_i + 1} \right)^2}$
MAE Mean Absolute Error	<ul style="list-style-type: none"> - 오차 절대값의 평균입니다. 	$\frac{1}{n} \sum_{i=1}^n y_i - \hat{y}_i $
MAPE Mean Absolute Percentage Error	<ul style="list-style-type: none"> - 오차를 실제값으로 나눈 절대값 평균입니다. 	$\frac{1}{n} \sum_{i=1}^n \frac{ y_i - \hat{y}_i }{ y_i }$

분류모형 성능 평가 기준

- 머신러닝 알고리즘은 우수한 분류모형을 선택할 때 실제값과 추정값이 같은 정도(정확도) 또는 다른 정도(오분류율)를 사용합니다.
 - 그런데 정확도와 오분류율은 다수 범주에 초점을 맞춘 가중평균된 값이므로 관심 있는 소수의 범주를 제대로 분류하지 못하는 분류모형도 높은 값을 가질 수 있습니다.
 - 따라서 정확도와 오분류율은 목표변수 범주별 백분율이 비슷할 때 효과적인 지표입니다.
- 해결해야 하는 문제의 종류와 데이터셋의 특징에 따라 성능 평가 지표가 다를 수 있다는 점에 유의해야 합니다.
 - 산업별로 분석 주제가 다양하므로 분류모형의 성능에 대한 요구사항도 달라집니다.
 - 목표변수 범주별 백분율이 크게 다르면 정확도보다 민감도와 정밀도가 더 중요합니다.

분류모형 성능 평가 지표(3종)

- 혼동행렬 Confusion Matrix 은 실제값 Actual Value 과 추정값 Predicted Value 의 빈도수를 성분으로 갖는 행렬이며, 분류모형이 실제값을 얼마나 다르게 추정했는지 확인합니다.
 - 정확도 accuracy , 민감도 sensitivity , 정밀도 precision , 특이도 specificity 등 성능지표를 계산합니다.
 - 민감도와 정밀도가 중요한데, 두 지표의 방향이 서로 엇갈리는 경우가 자주 발생합니다.
- F1 점수는 민감도와 정밀도의 조화평균이며, 두 지표가 높아야 큰 값을 갖습니다.
- ROC Receiver Operating Characteristic 곡선은 x축이 1-특이도, y축이 민감도인 그래프입니다.
 - ROC 곡선을 그렸을 때 왼쪽 위 모서리에 가장 가까운 분류모형을 찾습니다.
 - ROC 곡선 아래의 면적을 AUC Area Under Curve 라고 하며, AUC로 분류모형의 성능을 비교합니다.

혼동행렬

		실제값	
		긍정 Positive	부정 Negative
추정값	긍정 Positive	True Positive	False Positive
	부정 Negative	False Negative	True Negative

Positive는 관심 있는 소수 범주로 설정합니다.
예를 들어 '대출 연체', '고객 이탈' 등입니다.

- 혼동행렬의 4가지 성분입니다.

- TP: 모형이 긍정으로 추정, 실제값도 긍정인 건수입니다.
- FP: 모형은 긍정으로 추정, 실제값은 부정인 건수입니다.
- FN: 모형은 부정으로 추정, 실제값은 긍정인 건수입니다.
- TN: 모형이 부정으로 추정, 실제값도 부정인 건수입니다.

혼동행렬(계속)

- 실제값이 긍정인 건수

$$P = TP + FN$$

- 실제값이 부정인 건수

$$N = FP + TN$$

- 전체 건수

$$T = TP + FN + FP + TN$$

$$= P + N$$

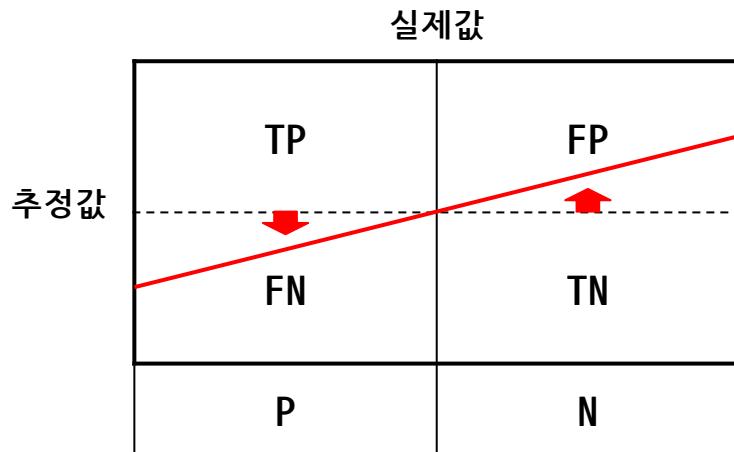
		실제값	
		TP	FP
추정값	TP		
	FN		TN
	P		N

분류모형의 성능에 따라 긍정과 부정으로 분류하는 빈도수가 달라집니다.

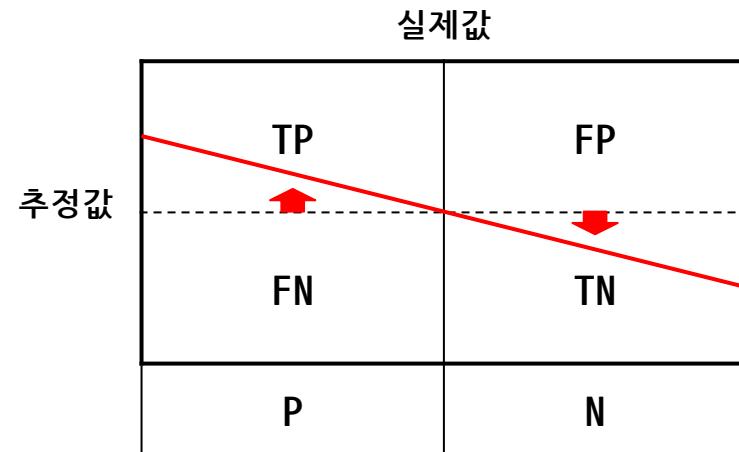
따라서 분류모형에 따라 가로선의 위치가 바뀝니다.

시험셋의 실제값 빈도수는 고정이므로 세로선의 위치는 바뀌지 않습니다.

[참고] 분류모형 성능에 따른 가로선의 변화



[분류 성능이 좋은 모형]



[분류 성능이 나쁜 모형]

혼동행렬 성능 평가 지표

성능지표	세부 내용	공식
정확도 (accuracy)	- 실제값과 추정값이 같은 건수를 전체 건수로 나눈 값입니다.	$\frac{TP + TN}{P + N}$
민감도 (sensitivity)	- 실제값이 긍정인 건에서 분류모형이 맞춘 비율입니다. 재현율 ^{recall} 또는 참긍정비율 ^{True Positive Rate, TPR} 이라고도 합니다.	$\frac{TP}{P}$
정밀도 (precision)	- 분류모형이 긍정이라고 한 건에서 분류모형이 맞춘 비율입니다.	$\frac{TP}{TP + FP}$
특이도 (specificity)	- 실제값이 부정인 건에서 분류모형이 맞춘 비율입니다.	$\frac{TN}{N}$
1-특이도	- 실제값은 부정인데 분류모형이 긍정으로 오분류한 비율입니다. 거짓긍정비율 ^{False Positive Rate, FPR} 이라고도 합니다.	$\frac{FP}{N}$

혼동행렬에서 중요한 지표

- 분류모형의 성능을 판단할 때 정확도는 좋은 지표일까요?

A

250	250	500
250	250	500
500	500	1000

B

5	45	50
45	905	950
50	950	1000

$$\text{정확도} = (250 + 250) \div 1000 = 0.50$$

$$\text{민감도} = 250 \div 500 = 0.50$$

$$\text{정밀도} = 250 \div 500 = 0.50$$

$$\text{정확도} = (5 + 905) \div 1000 = 0.91$$

$$\text{민감도} = 5 \div 50 = 0.10$$

$$\text{정밀도} = 5 \div 50 = 0.10$$

만약 이상거래 탐지모형이라면?

혼동행렬에서 중요한 지표(계속)

- 분류모형의 성능에 따라 민감도와 정밀도의 방향이 엇갈릴 수 있습니다.

A	40	20	60
	10	930	940
	50	950	1000

$$\text{민감도} = 40 \div 50 = 0.80$$

$$\text{정밀도} = 40 \div 60 = 0.67$$

$$\text{정확도} = (40 + 930) \div 1000 = 0.97$$

B	45	25	70
	5	925	930
	50	950	1000

$$\text{민감도} = 45 \div 50 = 0.90$$

$$\text{정밀도} = 45 \div 70 = 0.64$$

$$\text{정확도} = (45 + 925) \div 1000 = 0.97$$

혼동행렬에서 중요한 지표(계속)

- 민감도와 정밀도가 함께 높을수록 성능이 좋은 모형입니다.

A	40	20	60
	10	930	940
	50	950	1000

$$\text{민감도} = 40 \div 50 = 0.80$$

$$\text{정밀도} = 40 \div 60 = 0.67$$

$$\text{정확도} = (40 + 930) \div 1000 = 0.97$$

B	45	15	60
	5	935	940
	50	950	1000

$$\text{민감도} = 45 \div 50 = 0.90$$

$$\text{정밀도} = 45 \div 60 = 0.75$$

$$\text{정확도} = (45 + 935) \div 1000 = 0.98$$

혼동행렬에서 중요한 지표(계속)

- 민감도와 특이도는 반비례 관계일 수 있습니다.

A	0	0	0
50	950	1000	
50	950	1000	

$$\text{민감도} = \text{TP} \div \text{P} = 0.00$$

$$\text{특이도} = \text{TN} \div \text{N} = 1.00$$

$$1 - \text{특이도} = 0.00$$

ROC 곡선의
점(0, 0)을
의미합니다.

B	50	950	1000
0	0	0	
50	950	1000	

$$\text{민감도} = \text{TP} \div \text{P} = 1.00$$

$$\text{특이도} = \text{TN} \div \text{N} = 0.00$$

$$1 - \text{특이도} = 1.00$$

ROC 곡선의
점(1, 1)을
의미합니다.

F1 점수

- F1 점수는 민감도와 정밀도의 조화평균입니다.
 - 조화평균으로 계산하는 이유는 둘 중 하나라도 낮으면 F1 점수가 낮아지기 때문입니다.
- 조화평균은 평균속력을 계산할 때 사용합니다.
 - 같은 거리를 왕복할 때 가는 속력과 오는 속력의 차이가 클수록 평균속력이 낮아집니다.
 - 예를 들어 약 100km 거리(서울시청~천안시청)를 갈 때 100km/h, 올 때 25km/h 속력으로 왕복했다면 평균속력은 얼마가 될까요?
 - 분류모형의 민감도가 1.00이고 정밀도는 0.25라면 F1 점수는 얼마가 될까요?

[참고] 조화평균의 이해

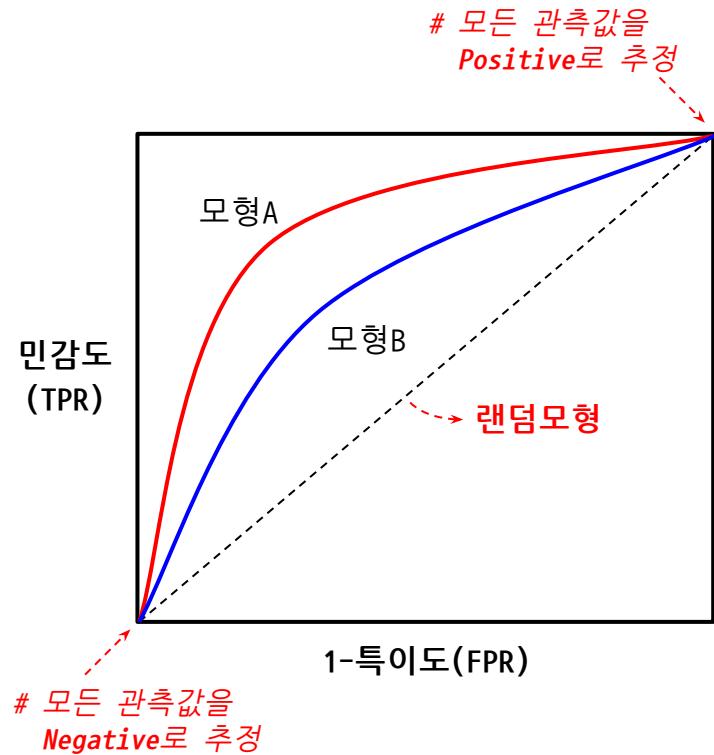
- 조화평균은 '역수로 계산한 산술평균의 역수'입니다.

$$\frac{2}{\frac{1}{a} + \frac{1}{b}} = \frac{2}{\frac{a+b}{ab}} = \frac{2ab}{a+b}$$

- 조화평균은 평균속력을 계산할 때 사용합니다.(속력 = 거리 ÷ 시간)
 - 같은 거리를 왕복할 때 만약 가는 속력과 오는 속력이 다르면 투입된 시간이 다르므로 평균 속력을 계산하려면 속력의 산술평균이 아닌 시간의 산술평균을 계산해야 합니다.
 - 시간(거리 ÷ 속력)의 산술평균은 '속력의 역수'로 계산한 평균입니다.
 - 평균시간에 역수를 취하면 원래 차원(평균속력)으로 되돌아옵니다.

ROC 곡선

- ROC 곡선은 X축에 1-특이도, Y축에 민감도를 놓고 분류모형의 성능을 그린 그래프입니다.
 - ROC 곡선은 항상 두 점 $(0, 0)$ 과 $(1, 1)$ 을 지납니다.
- 성능이 우수한 분류모형은 1-특이도가 낮고 민감도가 높은 모형입니다.
 - ROC 곡선이 왼쪽 위 모서리에서 가까울수록 성능이 좋은 모형입니다.
- [중요] ROC 곡선은 추정확률로 그립니다!



[참고] ROC 곡선을 그리는 원리

- 분류모형으로부터 추정확률을 생성하고 분리 기준점마다 추정값으로 변환합니다.
추정값으로 계산한 점(1-특이도, 민감도)들을 이어 붙이면 ROC 곡선을 그립니다.

행이름	추정확률		분리 기준점 <small>cut-off</small> 으로 추정값 생성					
	Negative	Positive	0.00	...	0.50	...	1.00	
1	0.0023	0.9977	Positive	...	Positive	...		Negative
2	0.9783	0.0217	Positive	...	Negative	...		Negative
3	0.8912	0.1088	Positive	...	Negative	...		Negative
4	0.1432	0.8568	Positive	...	Positive	...		Negative
:	:	:	:	:	:	:	:	:

FPR: 1

TPR: 1

FPR: 0.8

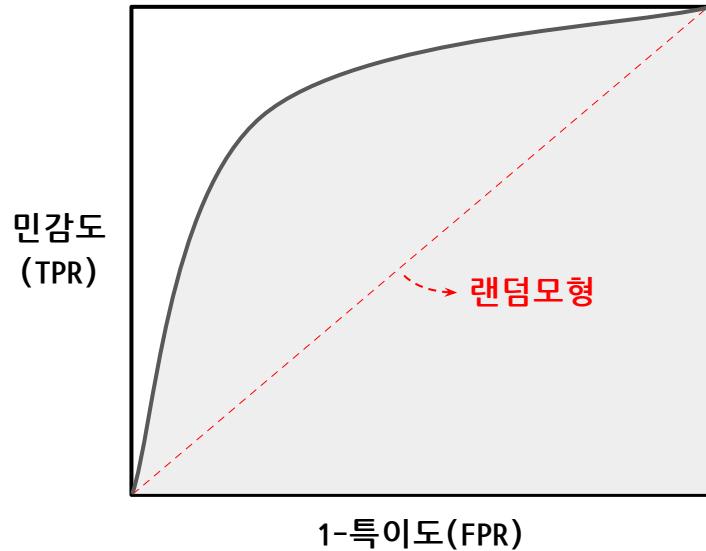
TPR: 0.2

FPR: 0

TPR: 0

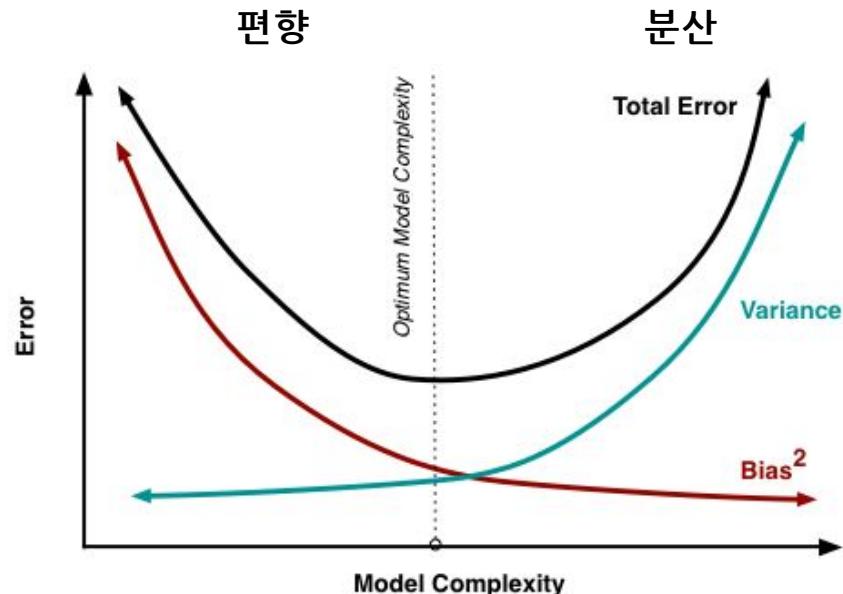
AUC

- AUC^{Area Under Curve}는 ROC 곡선 아래 면적입니다.
 - ROC 곡선이 왼쪽 모서리에 가까울수록 AUC는 1에 가까워집니다.
 - 추정값을 랜덤하게 반환하는 분류모형의 ROC 곡선은 오른쪽 그림의 빨간색 점선과 같고, 이 모형의 AUC는 0.5입니다.
 - AUC가 0.5 미만인 모형은 랜덤모형보다 성능이 낮으므로 가치가 없습니다.
- [중요] AUC도 추정확률로 계산합니다.



모형의 복잡도와 오차의 관계

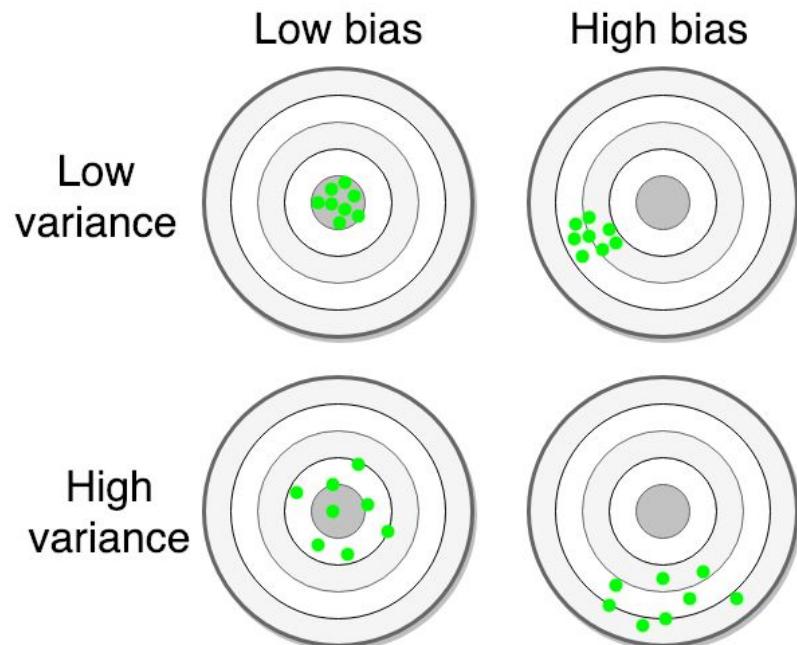
- 모형의 오차는 편향 제곱과 분산의 합입니다.
 - $\text{error} = \text{bias}^2 + \text{variance}$
- 학습 초기에 오차가 상당히 큰 이유는 과소적합에 의한 편향 때문입니다.
- 학습을 진행하면 총오차는 감소하다가 어느 시점을 지나면 다시 증가하는데 그 이유는 모델이 과적합하여 분산이 증가하기 때문입니다.



출처: <https://stats.stackexchange.com/questions/336433/bias-variance-tradeoff-math>

[참고] 편향과 분산

- 학습 초기 과소적합^{underfitting}된 모형은 충분하게 학습하지 않았으므로 편향을 가집니다.
 - 아직 모형이 단순하다는 의미입니다.
- 과적합^{overfitting}된 모형은 훈련셋을 외울 정도로 학습했으므로 모형을 적합할 때 사용하지 않은 데이터를 맞추지 못하기 때문에 분산이 증가합니다.
 - 모형이 복잡해졌다는 것을 의미합니다.



출처: <http://www.machinelearningtutorial.net/2017/01/26/the-bias-variance-tradeoff/>

과적합 및 과소적합 판단 기준

- 머신러닝 모형을 적합하면 훈련셋과 검증셋으로 성능지표를 계산하고 아래 표를 참고하여 과적합 또는 과소적합 여부를 판단합니다.

훈련셋의 성능	시험셋의 성능	판단 결과
높음	높음	성능 좋은 모형
높음	낮음	과적합 의심
낮음	낮음	과소적합 의심
낮음	높음	(알고리즘, 샘플링, 샘플 크기)

선형 회귀분석

선형 회귀분석의 개요

- 선형 회귀분석은 하나 이상의 입력변수와 목표변수 간 관계를 파악하기에 유용한 데이터 마이닝 기법의 일종이며 아래 관계식으로 표현합니다.
 - 모집단: $y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \epsilon_i$ (마지막 항은 오차)
 - 표본: $y_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + \cdots + b_p x_{pi} + e_i$ (마지막 항은 잔차)
 - 목표변수의 추정값(\hat{y}_i)은 입력변수의 선형결합에 y절편(b_0)을 더한 관계식으로 표현할 수 있으며 실제값과 추정값 간 차이를 잔차(e_i)라고 합니다.
 - 회귀분석은 입력변수별 회귀계수(b_j)를 추정합니다. 회귀계수는 다른 입력변수를 고정한 상태에서 해당 입력변수가 1단위 증가할 때 목표변수에 미치는 크기를 의미합니다.
 - 선형 회귀모형은 목표변수의 변동을 설명할 수 있는 크기를 결정계수(R^2)로 표현합니다.

선형 회귀분석의 기본 가정

- 목표변수와 입력변수는 직선의 관계를 가져야 합니다.
 - 두 변수로 산점도를 그렸을 때 직선의 관계가 있는지 확인합니다.
 - 피어슨 상관분석을 실행하여 두 변수 간 상관관계가 있는지 확인합니다.
 - 선형 회귀모형의 입력변수-잔차 산점도를 그려 잔차의 패턴을 확인합니다.
- 입력변수끼리 강한 상관관계가 없어야 합니다.
 - 다른 입력변수에 종속된 입력변수를 다중공선성 입력변수라고 합니다.
 - 다중공선성 입력변수는 목표변수에 대한 설명력을 다른 입력변수와 나눠가집니다.
 - 그 결과로 회귀계수의 설명력이 낮아지므로 다중공선성 변수를 제거해야 합니다.

선형 회귀모형 진단

- 회귀모형의 잔차항에 대한 가정은 아래와 같습니다.
 - 잔차는 평균이 0이고 표준편차가 σ인 정규분포를 따라야 합니다.[정규성]
 - 모든 입력변수 값에서 동일한 분산을 갖습니다.[등분산성]
 - 잔차끼리 자기 상관이 없어야 합니다.[독립성]
 - 잔차의 산점도를 그렸을 때 일정한 패턴이 없어야 합니다.[잔차의 패턴]
- 선형 회귀직선은 입력변수의 전체 영역에서 목표변수와의 관계를 설명하는 유일한 직선이므로 만약 관측값에 이상치가 있으면 전체 관계가 왜곡될 수 있습니다.
 - 따라서 이상치가 있으면 제거해야 합니다.[쿡의 거리로 이상치 진단]

[참고] 선형 회귀모형의 시각적 이해

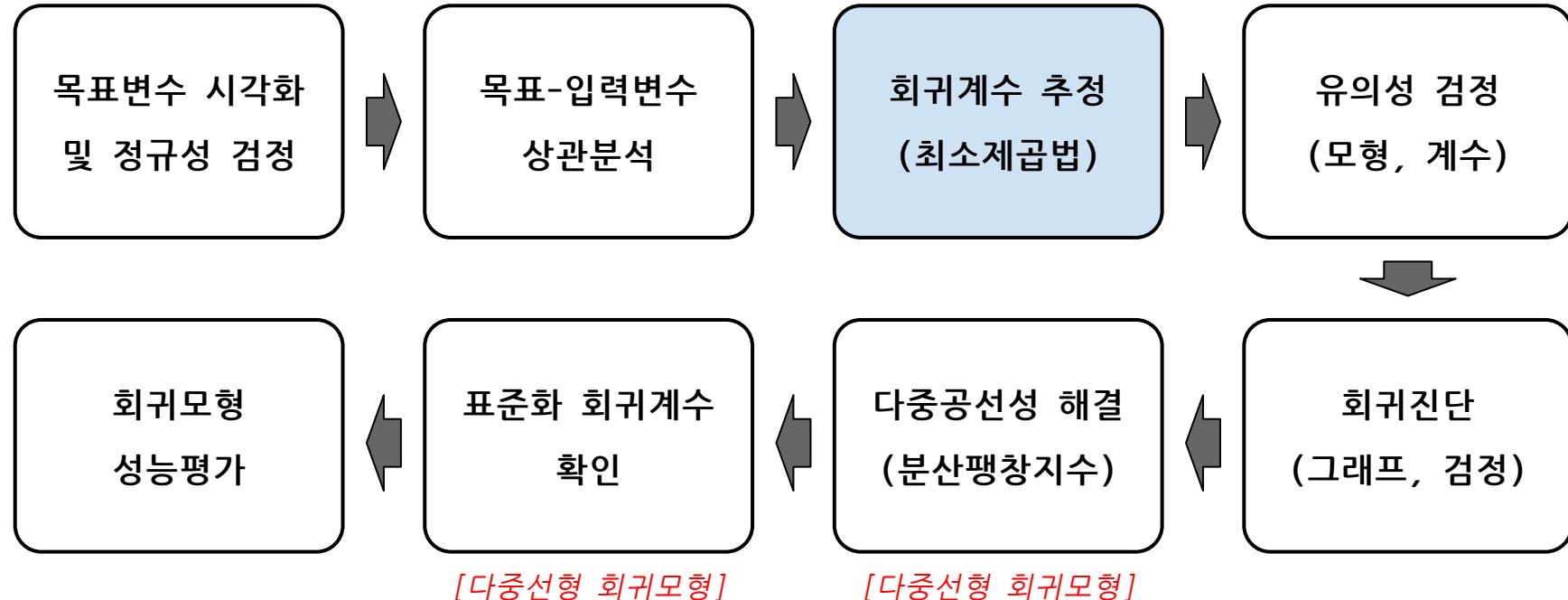
아들의
키

잔차

아버지의 키

- # 입력변수가 증가함에 따라 목표변수의 값도 지속적으로 증가하거나 감소해야 합니다.
- # 회귀직선은 입력변수의 값이 변함에 따라 목표변수의 평균을 이은 직선입니다.
- # 어떤 입력변수 값에서 목표변수 실제값(회색점)과 회귀직선 위 추정값(빨간점) 간 차이를 잔차라고 합니다.
- # 잔차는 평균이 0, 표준편차가 σ 인 정규분포를 따라야 합니다.(정규성, 등분산성)
- # 만약 잔차가 어떤 패턴을 보이면 입력변수를 추가해야 합니다.(선형성)

선형 회귀분석 프로세스



관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3
```

```
>>> pd.options.display.precision = 3
```

관련 라이브러리 호출(계속)

- 통계 관련 라이브러리를 호출합니다.

```
>>> from scipy import stats
```

```
>>> import pingouin as pg
```

- 시각화 관련 모듈을 호출합니다.

```
>>> from GraphicSetting import *
```

```
>>> import HelloDataScience as hds
```

작업 경로 확인 및 변경

- 현재 작업 경로를 확인합니다.

```
>>> os.getcwd()
```

- data 폴더로 작업 경로를 변경합니다.

```
>>> os.chdir(path = '../data')
```

- 현재 작업 경로에 있는 폴더명과 파일명을 출력합니다.

```
>>> os.listdir()
```

실습 데이터셋 준비

- z 파일을 읽고 데이터프레임 df를 생성합니다.

```
>>> df = joblib.load(io = 'Used_Cars_Price_Prep.z')
```

- df의 정보를 확인합니다.

```
>>> df.info()
```

- df의 처음 5행을 출력합니다.

```
>>> df.head()
```

- y절편 역할을 수행할 상수 1을 df의 두 번째 열로 삽입합니다.

```
>>> df.insert(loc = 1, column = 'const', value = 1)
```

범주형 입력변수의 더미변수 변환

- 범주형 입력변수는 범주 개수 - 1개의 더미변수로 변환합니다.
 - 왜냐하면 범주 개수만큼 더미변수를 생성하면 다중공선성 문제가 발생하기 때문입니다.
- 이번 예제에서는 FuelType으로 더미변수 Petrol을 생성합니다.

FuelType	Diesel	Petrol
Diesel	1	0
Petrol	0	1

FuelType이 'Diesel'일 때 더미변수는 0이므로 목표변수 추정값에 영향을 미치지 않지만 'Petrol'일 때는 회귀계수만큼 추정값에 영향을 미칩니다.

회귀계수가 양수면 'Petrol' 차량의 가격이 'Diesel'에 비해 평균적으로 회귀계수만큼 높다는 것을 의미합니다.

- MetColor도 범주형 입력변수지만 원소가 '0' 또는 '1'인 문자형이므로 정수형으로 변환하면 더미변수와 같은 형태입니다.

더미변수 생성

- 범주형 입력변수로 더미변수를 생성합니다.

```
>>> df = pd.get_dummies(data = df, columns = ['FuelType'], drop_first = True)
```

- df의 처음 10행을 출력합니다.

```
>>> df.head(n = 10)
```

- 더미변수명을 변경합니다.

```
>>> df = df.rename(columns = {'FuelType_Petrol': 'Petrol'})
```

- df의 열별 자료형을 확인합니다.

```
>>> df.dtypes # [참고] 더미변수의 자료형은 numpy.uint8(부호 없는 8비트 정수)입니다.
```

더미변수 생성(계속)

- 문자형 열이름으로 리스트를 생성합니다.

```
>>> cols = ['MetColor']
```

- 지정한 변수를 정수형으로 변환합니다.

```
>>> df[cols] = df[cols].astype(np.uint8) # [참고] 'uint8'로 지정해도 됩니다.
```

- df의 열별 자료형을 확인합니다.

```
>>> df.dtypes
```

실습 데이터셋 분할

- 관련 라이브러리를 호출합니다.

```
>>> from sklearn.model_selection import train_test_split
```

- 전체 데이터셋의 70%를 훈련셋, 30%를 시험셋으로 분할합니다.

```
>>> trSet, teSet = train_test_split(df, test_size = 0.3, random_state = 0)
```

- 훈련셋의 목표변수 평균을 출력합니다.

```
>>> trSet['Price'].mean()
```

- 시험셋의 목표변수 평균을 출력합니다.

```
>>> teSet['Price'].mean()
```

입력변수와 목표변수 분리

- 목표변수명을 변수에 할당합니다.

```
>>> yvar = 'Price'
```

- 훈련셋을 목표변수 벡터와 입력변수 행렬로 분리합니다.

```
>>> trReal = trSet[yvar].copy()
```

```
>>> trSetX = trSet.drop(columns = [yvar])
```

- 시험셋을 목표변수 벡터와 입력변수 행렬로 분리합니다.

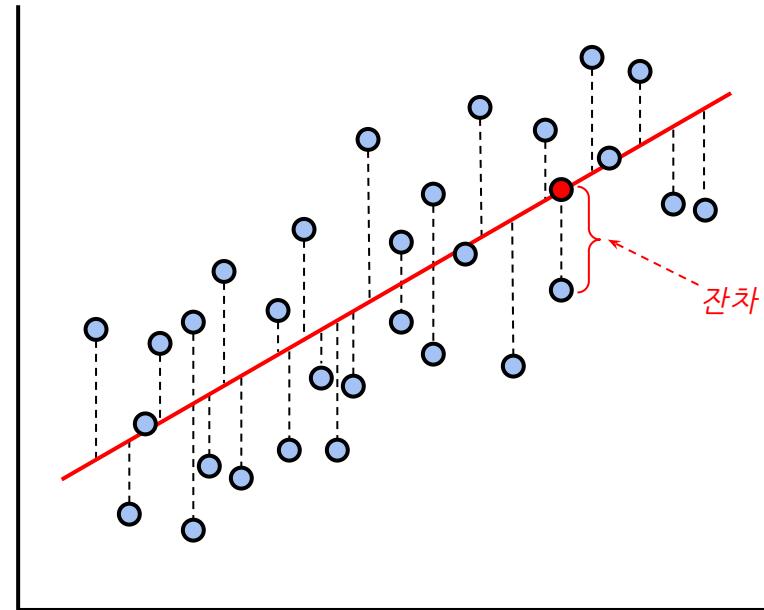
```
>>> teReal = teSet[yvar].copy()
```

```
>>> teSetX = teSet.drop(columns = [yvar])
```

선형 회귀계수 추정: 최소제곱법

- x축에 입력변수, y축에 목표변수를 놓고 산점도를 그립니다.
 - 산점도는 상관관계를 시각화한 것입니다.
- 산점도에서 두 변수의 관계를 표현하는 회귀직선을 찾습니다.

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$
- 회귀직선의 기울기와 y절편은 목표변수의 실제값과 추정값의 차이를 최소로 만드는 모델 파라미터입니다.



선형 회귀계수 추정: 최소제곱법(계속)

- 입력변수가 1개인 단순선형 회귀모형은 아래와 같습니다.

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- 회귀계수를 추정하기 위해 최소제곱법(잔차 제곱합) 공식으로 변형합니다.

$$\sum (y_i - \hat{y}_i)^2 = \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

- 위 식을 각 회귀계수로 편미분한 방정식을 풁니다.

$$\frac{\partial}{\partial \beta_0} \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = -2 \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

$$\frac{\partial}{\partial \beta_1} \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = -2 \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)x_i = 0$$

선형 회귀계수 추정(계속)

- β_0 으로 편미분 방정식을 풀면 다음과 같습니다.

$$\frac{\partial}{\partial \beta_0} \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = -2 \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

$$\sum y_i - \sum \hat{\beta}_0 - \sum \hat{\beta}_1 x_i = 0$$

$$n\hat{\beta}_0 = \sum y_i - \hat{\beta}_1 \sum x_i$$

$$\hat{\beta}_0 = \frac{1}{n} \left(\sum y_i - \hat{\beta}_1 \sum x_i \right) = \bar{y} - \hat{\beta}_1 \bar{x}$$

y 의 평균 - (기울기) x 의 평균

선형 회귀계수 추정(계속)

- β_1 으로 편미분 방정식을 풀면 다음과 같습니다.

$$\frac{\partial}{\partial \beta_1} \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = -2 \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) x_i = 0$$

$$\sum x_i y_i = \hat{\beta}_0 \sum x_i + \hat{\beta}_1 \sum x_i^2$$

$\hat{\beta}_1 \sum x_i^2 = \sum x_i y_i - \hat{\beta}_0 \sum x_i$ # $\hat{\beta}_0 = \frac{1}{n} (\sum y_i - \hat{\beta}_1 \sum x_i)$ 을 대입하고 $\hat{\beta}_1$ 으로 정리합니다.

$$\hat{\beta}_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

분자: x 와 y 의 공분산
분모: x 의 분산

선형 회귀모형의 유의성 검정

- 회귀모형이 의미를 가지려면 최소 한 개 이상의 회귀계수는 0이 아니어야 합니다.
 - 만약 모든 회귀계수가 0이면 회귀모형은 추정값으로 목표변수의 평균을 반환합니다.
 - 입력변수의 값이 바뀌어도 목표변수의 추정값은 바뀌지 않는다는 것을 의미합니다.
- 회귀모형의 유의성 검정은 분산분석을 실행합니다.
 - 귀무가설은 '모든 회귀계수가 0이다'입니다.
 - 대립가설은 '최소한 하나 이상의 회귀계수는 0이 아니다'입니다.
 - 선형 회귀모형의 총분산^{SST}은 회귀제곱합^{SSR}과 잔차제곱합^{SSE}의 합입니다.
 - 평균회귀제곱합^{MSR}을 평균잔차제곱합^{MSE}로 나눈 F 통계량의 유의확률로 판단합니다.

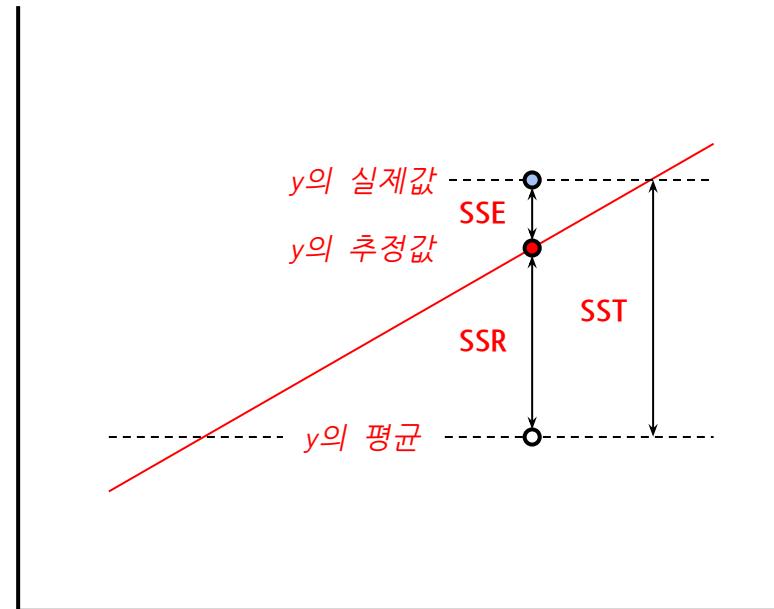
선형 회귀모형의 유의성 검정(계속)

- 회귀제곱합 SSR 은 추정값과 평균 차이를 제곱하여 모두 더한 것입니다.

$$SSR = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

- 잔차제곱합 SSE 은 실제값과 추정값 차이를 제곱하여 모두 더한 것입니다.

$$SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



목표변수에 대한 입력변수의 설명력이 증가할수록
잔차가 감소하므로 SSR 은 증가, SSE 는 감소합니다.

선형 회귀모형의 유의성 검정(계속)

- 회귀모형의 유의성 검정통계량은 회귀제곱합과 잔차제곱합을 각각의 자유도로 나눈 평균제곱합의 비^{ratio}입니다.
 - 평균회귀제곱합^{MSR}은 회귀제곱합을 자유도인 p 로 나눈 것입니다. (p 는 입력변수 개수)
 - 평균잔차제곱합^{MSE}은 잔차제곱합을 자유도인 $n-p-1$ 로 나눈 것입니다. (n 은 행 개수)
- 회귀모형의 분산분석표는 아래와 같습니다.

구분	제곱합(ss)	자유도(df)	평균제곱(MS)	F-통계량	유의확률
모형(R)	SSR	p	$MSR = SSR \div p$	MSR / MSE	$p\text{-value}$
잔차(E)	SSE	$n-p-1$	$MSE = SSE \div (n-p-1)$		
전체(T)	SST	$n-1$			

선형 회귀모형의 유의성 검정(계속)

- 선형 회귀모형을 적합했다면 가장 먼저 확인해야 할 것은 F-통계량의 유의확률이 유의수준(α)보다 작은지 여부입니다.
- F-통계량의 유의확률이 유의수준(0.05) 보다 작으면 '모든 회귀계수가 0'이라는 귀무가설을 기각할 수 있으므로 회귀모형에 있는 회귀계수 중 최소한 하나는 0이 아니라고 판단합니다.
- 만약 F-통계량의 유의확률이 유의수준(0.05) 보다 크면 모든 회귀계수가 0이라고 판단할 수 있으므로 회귀모형을 사용할 수 없습니다.
- 관측값(행)을 늘리거나 입력변수(열)를 추가하고 회귀모형을 다시 적합합니다.

선형 회귀계수의 유의성 검정

- 회귀모형에서 어떤 입력변수가 의미를 가지려면 회귀계수가 0이 아니어야 합니다.
 - 만약 어떤 입력변수의 회귀계수가 0이면 입력변수의 값이 바뀌어도 목표변수의 추정값은 바뀌지 않는다는 것을 의미합니다.
- 회귀계수의 유의성 검정은 t-검정으로 실행합니다.
 - 귀무가설은 '모집단의 회귀계수(β_j)가 0이다'입니다.
 - 대립가설은 '모집단의 회귀계수(β_j)는 0이 아니다'입니다.
 - 선형 회귀모형에서 추정한 회귀계수로 모집단의 회귀계수가 0인지 여부를 판단합니다.
 - 유의확률이 0.05 보다 작으면 귀무가설을 기각하고 회귀계수는 0이 아니라고 판단합니다.

선형 회귀계수의 유의성 검정(계속)

- 선형 회귀계수의 t-검정통계량은 다음과 같습니다.

$$t = \frac{b_j - \beta_j}{s_{b_j}}$$

b_j 는 회귀모형에서 추정한 회귀계수입니다.
 # 귀무가설에서 모집단의 회귀계수(β_j)를 0으로 가정합니다.
 # s_{b_j} 는 회귀계수의 표준오차입니다.

- 선형 회귀계수의 표준오차는 다음과 같이 추정합니다.

$$s_{b_j} = \frac{s_\epsilon}{\sqrt{\sum(x_i - \bar{x})^2}}$$

분자는 회귀모형 잔차의 표준오차입니다.

$$s_\epsilon = \sqrt{\frac{\text{SSE}}{n - p - 1}} = \sqrt{\text{MSE}}$$

잔차의 표준오차는 실제값과 추정값이 서로 가까울수록 작아집니다.
 이 값이 작아질수록 회귀계수의 표준오차가 작아지며 t-검정통계량의 절대값은 커지므로 유의확률은 결국 감소합니다.

결정계수

- 결정계수 R-squared는 선형 회귀모형의 적합도 goodness of fit를 나타내는 지표입니다.
 - 결정계수는 총변동 SST에서 회귀모형에 의해 설명할 수 있는 변동 SSR의 비율을 의미합니다.

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

- 결정계수는 0~1의 값을 가지며 1에 가까울수록 적합도가 좋은 모형입니다.
- 단순선형 회귀모형의 결정계수는 입력변수와 목표변수 간 피어슨 상관계수를 제곱입니다.
- 따라서 입력변수와 목표변수 간 강한 상관관계를 가질수록 결정계수는 증가합니다.

조정된 결정계수

- 선형 회귀모형의 결정계수는 입력변수의 개수가 많아질수록 계속 증가합니다.
 - 최적의 회귀모형을 선택하는 기준으로 결정계수를 사용하지 않습니다.
- 조정된 결정계수^{adjusted R-squared}는 입력변수의 개수로 벌점^{penalty}을 부여합니다.

$$\text{adj.}R^2 = 1 - \frac{(n - 1)(1 - R^2)}{n - p - 1}$$

n : 훈련셋의 행(관측값) 개수
 p : 훈련셋의 열(입력변수) 개수

- 조정된 결정계수는 0~1의 값을 가지며, 분수가 0으로 수렴하면 1에 가까워집니다.
- 입력변수를 추가하면 분모는 감소하고, 결정계수 증가로 분자도 감소합니다.
- 만약 목표변수와 상관계수가 낮은 입력변수를 추가하면 결정계수가 증가하지 않으므로 분자는 고정인데 분모가 감소했기 때문에 조정된 결정계수는 오히려 감소합니다.

선형 회귀모형 함수

- statsmodels 라이브러리에 선형 회귀모형을 적합하는 두 가지 함수를 소개합니다.
 - `statsmodels.api.OLS()` 함수에 목표변수와 입력변수를 각각 지정합니다.
 - `statsmodels.formula.api.ols()` 함수에 '목표변수 ~ 입력변수' 관계식을 문자열로 지정합니다. 입력변수가 여러 개일 때 입력변수 사이에 '+' 기호를 추가합니다.
- 선형 회귀모형을 적합할 때 `statsmodels.api.OLS()` 함수를 사용합니다.
 - endog: 목표변수를 지정합니다.
 - exog: 입력변수를 지정합니다. 함수에 지정하기 전에 상수 1을 추가해야 합니다.

선형 회귀모형 적합 함수 생성

- 관련 라이브러리를 호출합니다.

```
>>> import statsmodels.api as sm
```

- 선형 회귀모형을 반환하는 함수를 생성합니다.

```
>>> def ols(y, X):  
  
    model = sm.OLS(endog = y, exog = X)  
  
    return model.fit()
```

선형 회귀모형 적합 및 결과 확인

- 훈련셋으로 선형 회귀모형을 적합하고 결과를 확인합니다.

```
>>> fit1 = ols(y = trReal, X = trSetX)
```

```
>>> fit1.summary() # Prob(F-statistic), 회귀계수의 P>|t|, Adj.R-squared를 차례대로 확인합니다.
```

Dep. Variable:	Price	R-squared:	0.733	coef	std err	t	P> t	[0.025	0.975]
Model:	OLS	Adj. R-squared:	0.730	const	-4832.6730	2260.499	-2.138	0.033	-9269.362 -395.984
Method:	Least Squares	F-statistic:	339.3	Age	-94.4704	2.961	-31.902	0.000	-100.283 -88.658
Date:	Sun, 26 Jun 2022	Prob (F-statistic):	2.57e-243	KM	-0.0170	0.001	-12.892	0.000	-0.020 -0.014
Time:	13:21:49	Log-Likelihood:	-7326.4	HP	2.2961	4.971	0.462	0.644	-7.460 12.052
No. Observations:	875	AIC:	1.467e+04	MetColor	14.4447	75.854	0.190	0.849	-134.434 163.324
Df Residuals:	867	BIC:	1.471e+04	Doors	-5.0942	46.181	-0.110	0.912	-95.733 85.545
Df Model:	7			Weight	18.9523	2.256	8.401	0.000	14.524 23.380
Covariance Type:	nonrobust			Petrol	1283.4546	325.808	3.939	0.000	643.990 1922.919

회귀진단: 잔차 가정 확인

- fit1 모형의 잔차 가정을 확인합니다.

Omnibus 값이 0에 가까우면 정규분포한다고 판단합니다.

왜도가 0이고 첨도가 3이면 정규분포한다고 판단합니다.

Omnibus: 54.450	Durbin-Watson: 1.962
Prob(Omnibus): 0.000	Jarque-Bera (JB): 140.503
Skew: -0.314	Prob(JB): 3.09e-31
Kurtosis: 4.860	Cond. No. 5.13e+06

Durbin-Watson 값이 1~3이면 독립성 가정을 만족합니다.

Jarque-Bera 값이 크면 정규성 가정을 만족하지 못합니다.

Condition number가 큰 값이면 다중공선성을 의심합니다.

- fit1 모형 잔차의 등분산성 검정을 실행합니다.

```
>>> hds.breushpagan(model = fit1) # Breush-Pagan 검정은 회귀모형 잔차의 이분산성을 확인합니다.  
F-통계량 유의확률이 0.05 이상이면 등분산성 가정을 만족합니다.
```

회귀진단: 잔차 그래프

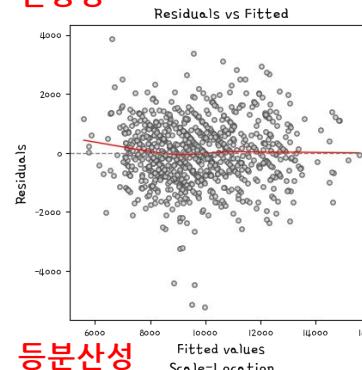
- fit1 모형의 잔차 그래프를 그립니다.

```
>>> hds.regressionDiagnosis(model = fit1)
```

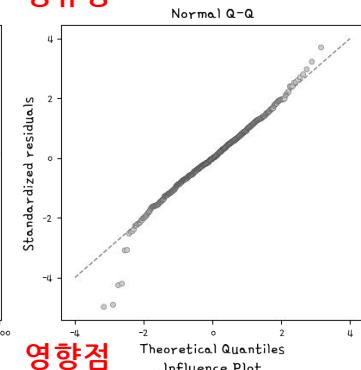
- 오른쪽 그래프로 잔차 가정을 확인합니다.

- 선형성: 빨간 직선에서 패턴이 없어야 합니다.
- 정규성: 모든 점이 대각선 위에 놓여야 합니다.
- 등분산성: 빨간 직선에서 폭이 같아야 합니다.
- 영향점: 표준화 잔차와 레버리지 값이 큰 점이 없어야 합니다.

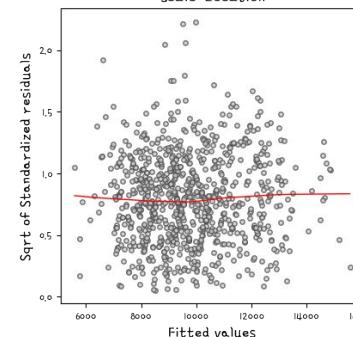
선형성



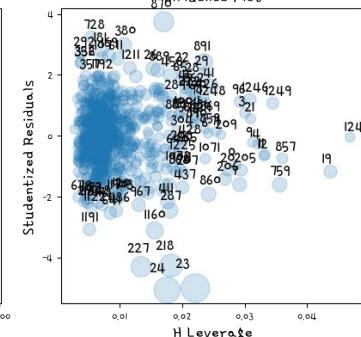
정규성



등분산성



영향점



회귀진단: 잔차의 정규성 검정

- Normal Q-Q Plot을 그렸을 때 모든 점이 45도 점선 근처에 있으면 정규성 가정을 만족하는 것으로 판단합니다.
 - 오른쪽 위 꼬리가 올라가고 왼쪽 아래 꼬리가 내려간 것은 정규분포의 양 극단 관측값이 이상적인 분포에 비해 많다는 것을 의미합니다.
- 잔차의 히스토그램을 그려서 분포를 확인합니다.

```
>>> sns.histplot(x = fit1.resid, bins = 50, stat = 'density');
```

- 잔차의 정규성 검정을 실행합니다.

```
>>> stats.shapiro(x = fit1.resid) # 유의확률이 0.05 보다 크면 잔차가 정규분포한다고 판단합니다.
```

회귀진단: 잔차의 독립성 검정

- 잔차의 독립성 가정은 잔차끼리 상관관계가 없어야 한다는 것을 의미합니다.
 - 잔차끼리 상관관계가 있으면 유의성 검정통계량과 결정계수를 과대추정할 수 있습니다.
- 대표적인 잔차의 독립성 문제는 시계열 자료를 회귀분석으로 다룰 때 나타납니다.
 - 이전 시점의 잔차항이 다음 시점의 잔차항에 영향을 미칠 수 있습니다.
 - 시차를 두고 잔차항이 상관관계를 보이는 것을 자기상관^{autocorrelation}이라고 합니다.
- 자기상관 여부를 확인할 수 있는 방법은 더빈-왓슨^{Durbin-Watson} 검정입니다.
 - 더빈-왓슨 검정통계량은 0~4의 값을 가지며 2에 가까우면 자기상관이 없고, 0 또는 4에 가까우면 자기상관이 있는 것으로 판단합니다. [참고] 유의확률은 확인하지 않습니다.

[참고] 영향점 판단 기준

구분	세부 내용
스튜던트 잔차	<ul style="list-style-type: none"> - 잔차는 목표변수의 실제값과 추정값의 차이가 클수록 큰 값을 갖습니다. - 잔차의 크기는 입력변수에 영향을 받으므로, 입력변수의 영향을 제거하려면 표준화해야 합니다. - 스튜던트 잔차는 잔차를 표준편차와 레버리지로 나눈 것입니다. - 스튜던트 잔차가 2~4보다 크면 이상치로 판단합니다.
레버리지	<ul style="list-style-type: none"> - 레버리지는 목표변수의 실제값이 추정값에 미치는 영향을 나타낸 값인데, 목표변수의 추정값은 Hat Matrix의 대각원소와 실제값의 내적으로 계산합니다. - 레버리지는 입력변수의 평균에서 멀어질수록 큰 값을 가지며, 회귀모형에 영향을 미칩니다. - 레버리지의 합계는 회귀모형의 입력변수 개수(p)와 같으므로, 평균은 p/n과 같습니다. - 레버리지가 평균의 2~4배일 때 영향점으로 판단합니다.
쿡의 거리	<ul style="list-style-type: none"> - 잔차와 레버리지를 모두 고려하여 이상치를 판단하는 지표가 쿡의 거리입니다. - 쿡의 거리가 $4/n$(행 개수)보다 큰 관측값 중에서 이상치 여부를 확인합니다. - (행 개수에 따라 달라지지만) 쿡의 거리가 0.5 또는 1보다 크면 이상치로 판단합니다.

[참고] 영향점 확인

- 훈련셋의 관측값마다 영향점 정보를 갖는 데이터프레임을 생성합니다.

```
>>> aug = hds.augment(model = fit1)
```

```
>>> aug.head() # 목표변수 실제값, 추정값(fitted), 잔차(resid), 레버리지(hat), 해당 관측값을 제거한 모형의  
잔차 표준편차 추정값(sigma), 쿡의 거리(cooksd) 및 표준화잔차(std_resid)를 출력합니다.
```

- 스튜던트 잔차의 절대값이 3을 초과하는 행 개수를 확인합니다.

```
>>> aug['std_resid'].abs().gt(3).sum()
```

- 레버리지 평균의 3배를 초과하는 행 개수를 확인합니다.

```
>>> hatAvg = aug['hat'].mean(); hatAvg
```

```
>>> aug['hat'].gt(hatAvg * 3).sum()
```

훈련셋에서 이상치 제거

- 쿡의 거리가 $4/n$ (행 개수)를 초과하는 행 개수를 확인합니다.

```
>>> n = trSet.shape[0] # 훈련셋의 행 개수 n을 생성합니다.
```

```
>>> locs = aug['cooksd'].gt(4/n) # 쿡의 거리가 4/n(행 개수)를 초과하면 True, 미만이면 False인 원소를 갖는 부울형 시리즈 locs를 생성합니다.
```

```
>>> locs.sum() # 쿡의 거리가 4/n(행 개수)를 초과하는 원소 개수를 확인합니다.
```

- 훈련셋에서 locs가 True(이상치)인 행을 제거합니다.

```
>>> trSetX = trSetX.loc[~locs] # 훈련셋의 입력변수 행렬에서 locs가 False인 원소를 남깁니다.  
[참고] ~ 기호는 논리부정이므로 True/False를 반전합니다.
```

```
>>> trSetX.shape[0] # 이상치를 제거한 훈련셋의 행 개수를 확인합니다.
```

```
>>> trReal = trReal.loc[~locs] # 훈련셋의 목표변수 벡터에서 locs가 False인 원소를 남깁니다.
```

선형 회귀모형 재적합 및 결과 확인

- 이상치를 제거한 훈련셋으로 선형 회귀모형을 적합합니다.

```
>>> fit2 = ols(y = trReal, X = trSetX)
```

- fit2 모형의 적합 결과를 확인합니다.

```
>>> fit2.summary()
```

- fit2 모형 잔차의 등분산성 검정을 실행합니다.

```
>>> hds.breushpagan(model = fit2)
```

- fit2 모형의 잔차 그래프를 그립니다.

```
>>> hds.regressionDiagnosis(model = fit2)
```

[참고] 더미변수의 시각적 이해

- Age와 Price의 회귀직선은 FuelType에 따라 달라집니다.

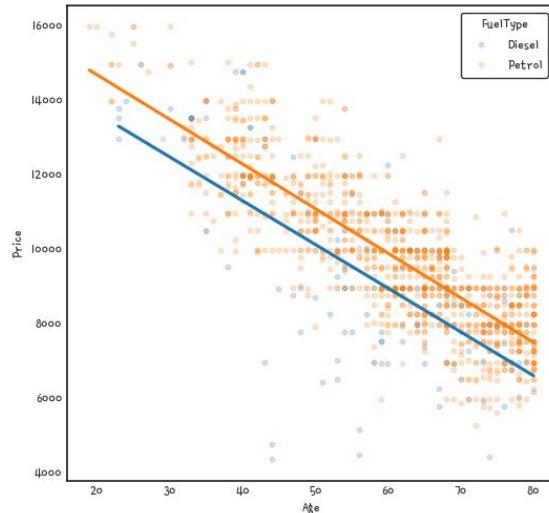
```

>>> labels = ['Diesel', 'Petrol']

>>> for i, v in enumerate(labels):

    sns.regplot(data = df[df['Petrol'].eq(i)],
                 x = 'Age', y = 'Price',
                 ci = None, label = v,
                 scatter_kws = dict(s = 10, alpha = 0.2))

>>> plt.legend(loc = 'best', title = 'FuelType');
  
```



목표변수의 추정값 생성

- 훈련셋으로 fit1과 fit2 모형의 추정값을 생성하고 실제값과 비교합니다.

```
>>> trPred1 = fit1.predict(exog = trSetX)
```

```
>>> trPred2 = fit2.predict(exog = trSetX) # fit1의 추정값보다 fit2의 추정값이 실제값에 가까운  
                                         (오차가 작은) 관측값이 더 많습니다.
```

```
>>> pd.DataFrame(data = {'real': trReal, 'pred1': trPred1, 'pred2': trPred2})
```

- 시험셋으로 fit1과 fit2 모형의 추정값을 생성하고 실제값과 비교합니다.

```
>>> tePred1 = fit1.predict(exog = teSetX)
```

```
>>> tePred2 = fit2.predict(exog = teSetX)
```

```
>>> pd.DataFrame(data = {'real': teReal, 'pred1': tePred1, 'pred2': tePred2})
```

선형 회귀모형 성능 평가

- 훈련셋으로 fit1과 fit2 모형의 성능지표를 출력합니다.

```
>>> hds.regmetrics(y_true = trReal, y_pred = trPred1)
```

```
>>> hds.regmetrics(y_true = trReal, y_pred = trPred2)
```

- 시험셋으로 fit1과 fit2 모형의 성능지표를 출력합니다.

```
>>> hds.regmetrics(y_true = teReal, y_pred = tePred1)
```

```
>>> hds.regmetrics(y_true = teReal, y_pred = tePred2)
```

[참고] 회귀모형 성능지표 관련 함수

- 관련 라이브러리를 호출합니다.

```
>>> from sklearn import metrics
```

- 회귀모형 성능지표를 출력합니다. # [참고] metrics 모듈에는 RMSE와 RMSLE를 반환하는 함수는 없으므로 MSE와 MSLE의 제곱근을 계산합니다.

```
>>> metrics.mean_squared_error(y_true = teReal, y_pred = tePred2)
```

```
>>> metrics.mean_squared_log_error(y_true = teReal, y_pred = tePred2)
```

```
>>> metrics.mean_absolute_error(y_true = teReal, y_pred = tePred2)
```

```
>>> metrics.mean_absolute_percentage_error(y_true = teReal, y_pred = tePred2)
```

다중공선성

- 현실세계에서 관찰하는 데이터에는 정도의 차이가 있지만 입력변수 간 상관관계를 피할 수 없습니다.
- 입력변수 간 강한 상관관계가 있으면 다중공선성 Multicollinearity 문제가 발생합니다.
 - 다중선형 회귀계수는 다른 입력변수를 고정한 상태에서 해당 입력변수가 1단위 증가할 때마다 목표변수에 미치는 크기를 의미합니다.
 - 상관관계가 높은 입력변수로 다중선형 회귀모형을 적합하면 목표변수에 대한 설명력을 입력변수가 나누어 가지므로 회귀계수의 절대값과 부호가 달라질 수 있습니다.
 - 또한 회귀계수의 표준오차가 증가하므로 회귀계수의 설명력도 낮아집니다.
 - 결국 회귀모형을 해석할 수 없는 문제가 발생합니다.

분산팽창지수

- 다중선형 회귀모형에서 사용한 p 개의 입력변수마다 다른 입력변수에 종속되었는지 여부를 판단할 때 분산팽창지수 Variance Inflation Factor를 계산합니다.

- j 번째 입력변수를 목표변수, 나머지 $p-1$ 개를 입력변수로 설정하여 회귀모형을 적합하고 반환되는 결정계수로 j 번째 입력변수의 분산팽창지수를 계산합니다.

$$vif_j = \frac{1}{1 - R_j^2}$$

- 만약 j 번째 입력변수를 목표변수로 설정한 회귀모형의 결정계수가 0.9이면 위 공식으로 계산한 분산팽창지수는 10입니다.
- 분산팽창지수가 10 이상이면 다중공선성 입력변수로 판단하고 삭제합니다.
 - [참고] 다중공선성을 판단하는 절대적인 기준은 아닙니다.

다중공선성 확인

- 분산팽창지수를 출력하고 다중공선성 입력변수를 확인합니다.

```
>>> hds.vif(X = trSetX)
```

- vif() 함수 실행 결과, 분산팽창지수가 10 이상인 입력변수가 없으므로 이번 예제에서는 다중공선성 입력변수는 없는 것으로 판단합니다.
- 만약 분산팽창지수가 10 이상인 입력변수가 다수일 때 분산팽창지수가 가장 큰 입력변수 하나만 훈련셋에서 삭제하고 회귀모형을 다시 적합합니다.
- 새로 적합한 회귀모형의 분산팽창지수를 확인합니다.
- 분산팽창지수가 10 이상인 변수가 없을 때까지 위 과정을 반복합니다.

다중공선성 입력변수 제거

- 다중공선성 입력변수가 있다고 가정하고 훈련셋에서 삭제합니다.

```
>>> trSetX1 = trSetX.drop(columns = ['Petrol'])
```

- 입력변수별 분산팽창지수를 다시 출력합니다.

```
>>> hds.vif(X = trSetX1) # [참고] 모든 입력변수의 분산팽창지수가 감소했습니다.
```

변수선택법

- 입력변수가 p개일 때 모든 경우의 수를 고려하여 성능이 우수한 최종 회귀모형을 선택한다고 가정하면 총 2^p 개의 후보모형을 생성해야 합니다.
 - 예를 들어 입력변수가 세 개(x_1, x_2, x_3)일 때 가능한 조합은 다음과 같습니다.
 - $y = x_1, y = x_2, y = x_3, y = x_1 + x_2, y = x_1 + x_3, y = x_2 + x_3, y = x_1 + x_2 + x_3$
(모든 입력변수를 추가한 Full 모형)과 $y = \beta_0$ (입력변수 없는 Null 모형)이 있습니다.
- 입력변수 개수가 많아질수록 모든 후보모형을 고려하는 것이 어렵습니다.
 - 입력변수 개수가 10개일 때 가능한 조합은 1024개로 증가합니다.
- 따라서 전진선택법 forward selection, 후진소거법 backward elimination, 단계적방법 stepwise method 을 사용하여 여러 후보모형 중에서 AIC가 가장 작은 모형을 선택합니다.

최적의 회귀모형 선택 기준: AIC

- AIC^{Akaike Information Criteria}는 로그 SSE(잔차제곱합)에 $2p$ (벌점)를 더한 값입니다.

$$AIC = n \ln \left(\frac{SSE}{n} \right) + 2p$$

- AIC의 특징은 아래와 같습니다.
 - 목표변수와 상관계수가 큰 입력변수를 추가하면 실제값과 추정값의 차이가 작아지므로 SSE(잔차제곱합)은 감소하는데, $2p$ (벌점)보다 감소량이 크면 AIC는 감소합니다.
 - 목표변수와 상관계수가 낮거나 상관관계가 없는 입력변수를 추가하면 SSE(잔차제곱합)은 거의 감소하지 않는데, $2p$ (벌점)보다 감소량이 작으면 AIC는 오히려 증가합니다.
 - 따라서 AIC가 작을수록 좋은 모형이라고 판단합니다.

전진선택법

- 전진선택법은 입력변수를 하나씩 추가하면서 최적의 회귀모형을 탐색합니다.
 - Null 모형에서 입력변수를 하나씩 추가한 후보모형을 적합한 다음, Null 모형보다 AIC가 작은 후보모형 중에서 AIC가 최소인 후보모형을 선택합니다.
 - 이전 단계에서 선택한 회귀모형으로 위 과정을 반복하면서 Full 모형까지 전진합니다.
 - 이전 단계에서 선택한 회귀모형보다 AIC가 작은 후보모형이 없으면 중단합니다.
- 전진선택법의 장점과 단점은 다음과 같습니다.
 - 장점: 입력변수가 p 개일 때 최대 $\frac{p(p+1)}{2}$ 개의 후보모형을 적합합니다.
 - 단점: 한 번 선택한 입력변수를 제거하지 않습니다.

후진소거법

- 후진소거법은 입력변수를 하나씩 제거하면서 최적의 회귀모형을 탐색합니다.
 - Full 모형에서 입력변수를 하나씩 제거한 후보모형을 적합한 다음, Full 모형보다 AIC가 작은 후보모형 중에서 AIC가 최소인 후보모형을 선택합니다.
 - 이전 단계에서 선택한 회귀모형으로 위 과정을 반복하면서 Null 모형까지 후진합니다.
 - 이전 단계에서 선택한 회귀모형보다 AIC가 작은 후보모형이 없으면 중단합니다.
- 후진소거법의 장점과 단점은 다음과 같습니다.
 - 장점: 입력변수가 p개일 때 최대 $\frac{p(p+1)}{2}$ 개의 후보모형을 적합합니다.
 - 단점: 한 번 제거한 입력변수를 추가하지 않습니다.

단계적 방법

- 단계적 방법은 전진선택법과 후진소거법의 단점을 보완한 방법입니다.
 - 시작 모형과 범위(Null 모형 ~ Full 모형)를 지정합니다.
 - 시작 모형에서 입력변수를 하나씩 추가한 후보모형과 하나씩 제거한 후보모형을 적합하고 시작 모형보다 AIC가 작은 후보모형 중에서 AIC가 최소인 후보모형을 선택합니다.
 - 이전 단계에서 선택한 회귀모형보다 AIC가 작은 후보모형이 없으면 중단합니다.
- 단계적 방법의 장점과 단점은 다음과 같습니다.
 - 장점: 전진선택법과 후진소거법보다 성능이 좋은 회귀모형을 탐색할 가능성이 있습니다.
 - 단점: 전진선택법과 후진소거법보다 적합해야 할 후보모형 개수가 많습니다.

단계적방법으로 선형 회귀모형 적합

- 단계적방법으로 선형 회귀모형을 적합합니다.

```
>>> fit3 = hds.stepwise(y = trReal, X = trSetX, direction = 'both')
```

direction 매개변수에 변수선택법 방향('forward',
'backward', 'both')을 문자열로 지정합니다.

- fit3 모형의 적합 결과를 확인합니다.

```
>>> fit3.summary()
```

- fit3 모형 잔차의 등분산성 검정을 실행합니다.

```
>>> hds.breushpagan(model = fit3)
```

표준화 회귀계수

- 입력변수가 두 개 이상인 다중선형 회귀모형은 입력변수별 회귀계수를 추정하지만 회귀계수로는 입력변수의 상대적인 영향력을 비교할 수 없습니다.
 - 왜냐하면 입력변수 간 척도가 서로 다르기 때문입니다.
- 표준화 회귀계수^{Standardized Regression Coefficient}로 목표변수에 대한 입력변수의 상대적인 영향력을 비교합니다.
- 표준화 회귀계수의 절대값이 클수록 목표변수에 대한 영향력이 크다고 판단합니다.
 - 표준화 회귀계수의 부호는 방향을 의미할 뿐 상대적인 영향력과는 무관합니다.

$$\beta_j^* = \beta_j \times \frac{\sigma_{x_i}}{\sigma_y}$$

표준화 회귀계수 확인

- fit3 모형의 회귀계수를 출력합니다.

```
>>> fit3.params
```

- 표준화 회귀계수를 생성합니다.

```
>>> beta_z = hds.std_coefs(model = fit3)
```

```
>>> beta_z
```

- 표준화 회귀계수의 절대값을 오름차순 정렬한 결과를 출력합니다.

```
>>> beta_z.abs().sort_values()
```

회귀모형 성능 평가

- 시험셋으로 fit3 모형의 추정값을 생성합니다.

```
>>> tePred3 = fit3.predict(exog = teSetX)
```

- 시험셋으로 fit3 모형의 성능지표를 출력합니다.

```
>>> hds.regmetrics(y_true = teReal, y_pred = tePred3)
```

- 시험셋으로 fit2 모형의 성능지표와 비교합니다.

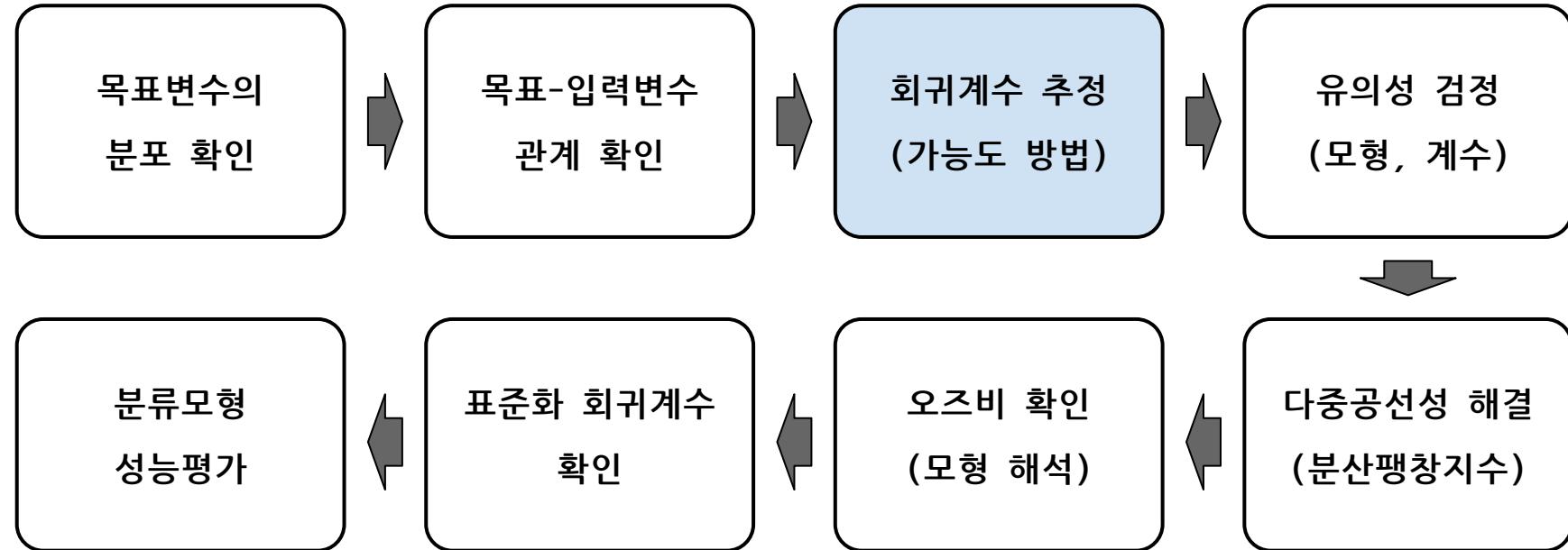
```
>>> hds.regmetrics(y_true = teReal, y_pred = tePred2)
```

로지스틱 회귀분석

로지스틱 회귀분석의 개요

- 로지스틱 회귀분석은 목표변수가 범주형일 때 사용하는 알고리즘입니다.
- 로지스틱 회귀분석은 목표변수를 입력변수 간 선형결합으로 표현합니다.
 - 로지스틱 회귀모형은 입력변수마다 목표변수에 영향을 미치는 크기를 제시합니다.
 - 로지스틱 회귀모형은 해석하기 쉬운 모형이므로 의사결정나무 알고리즘과 함께 분류모형 적합에서 많이 사용합니다.
- 로지스틱 회귀분석은 0~1의 추정확률을 반환합니다.
 - 분석가는 분리 기준점^{cut-off}을 정하여 추정확률을 추정값으로 변환해야 합니다.
 - 분리 기준점에 따라 혼동행렬, F1 점수 등 분류모형의 성능지표가 달라집니다.

로지스틱 회귀분석 프로세스



실습 데이터셋 소개

- 미국의 한 대학교에 지원한 고등학생들의 영어 성적, 학점, 출신 고등학교 등급 및 합격 여부를 포함하는 텍스트 데이터입니다.
 - admit: 합격 여부('Fail', 'Pass')
 - gre: 영어 성적
 - gpa: 학점
 - rank: 출신 고등학교 등급(1~4)

관련 라이브러리 호출

- 관련 라이브러리를 호출합니다.

```
>>> import os, joblib
```

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

- 실수를 출력할 소수점 자리수를 설정합니다.

```
>>> %precision 3
```

```
>>> pd.options.display.precision = 3
```

관련 라이브러리 호출(계속)

- 통계 관련 라이브러리를 호출합니다.

```
>>> from scipy import stats
```

```
>>> import pingouin as pg
```

- 시각화 관련 모듈을 호출합니다.

```
>>> from GraphicSetting import *
```

```
>>> import HelloDataScience as hds
```

실습 데이터셋 준비

- 인터넷에 공유 중인 텍스트 데이터를 읽고 데이터프레임 df를 생성합니다.

```
>>> df = pd.read_csv(filepath_or_buffer = 'https://bit.ly/Univ_Admit')
```

- df의 정보를 확인합니다.

```
>>> df.info()
```

- df의 처음 5행을 출력합니다.

```
>>> df.head()
```

- y절편 역할을 수행할 상수 1을 df의 두 번째 열로 삽입합니다.

```
>>> df.insert(loc = 1, column = 'const', value = 1)
```

실습 데이터셋 전처리

- rank를 문자형으로 변환합니다.

```
>>> df['rank'] = df['rank'].astype(str)
```

- 연속형 변수의 기술통계량을 확인합니다.

```
>>> df.describe()
```

- 범주형 변수의 기술통계량을 확인합니다.

```
>>> df.describe(include = object)
```

- rank의 원소별 빈도수를 출력합니다.

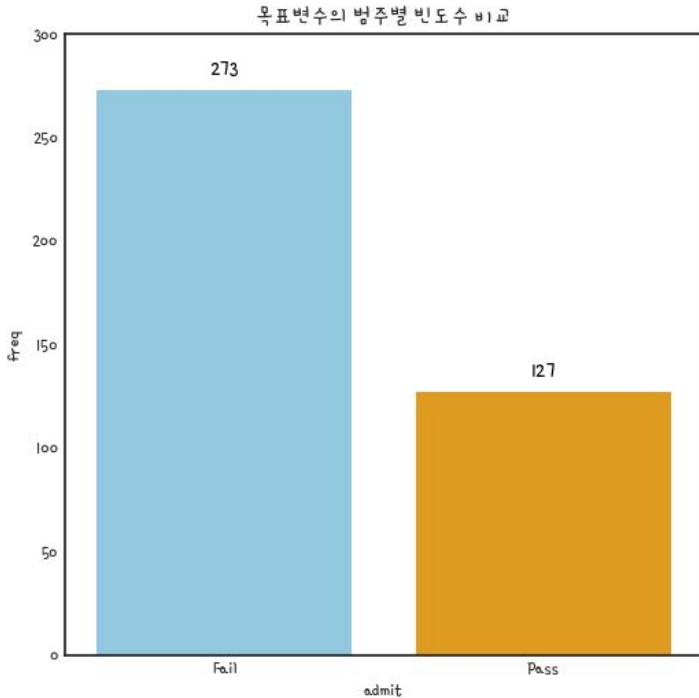
```
>>> df['rank'].value_counts().sort_index()
```

목표변수 시각화

- 목표변수 범주별 빈도수로 일변량 막대 그래프를 그립니다.

```
>>> hds.plot_bar_freq(
```

```
    data = df,
    y = 'admit',
    pal = [ 'skyblue', 'orange' ]
)
```

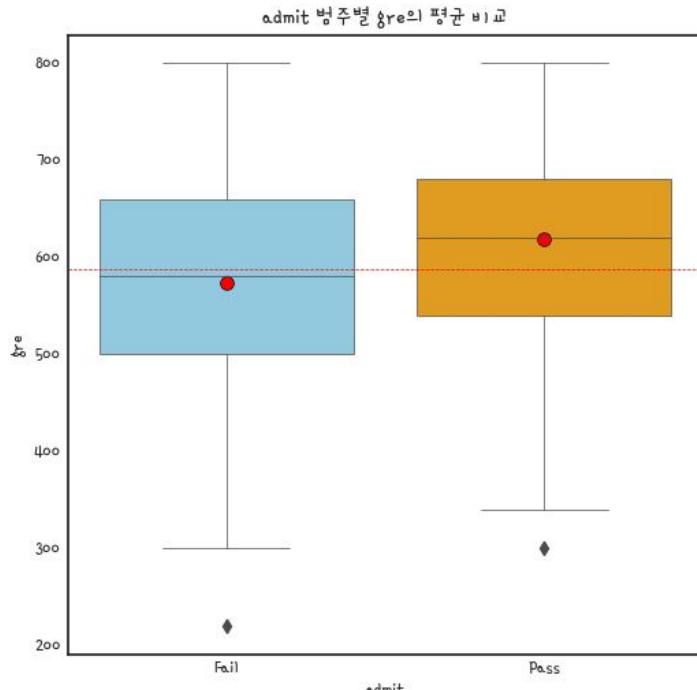


입력변수와 관계 파악: gre

- admit 범주별 gre의 상자 수염 그림을 그립니다.

```
>>> hds.plot_box_group(
```

```
    data = df,
    x = 'admit',
    y = 'gre',
    pal = ['skyblue', 'orange']
)
```

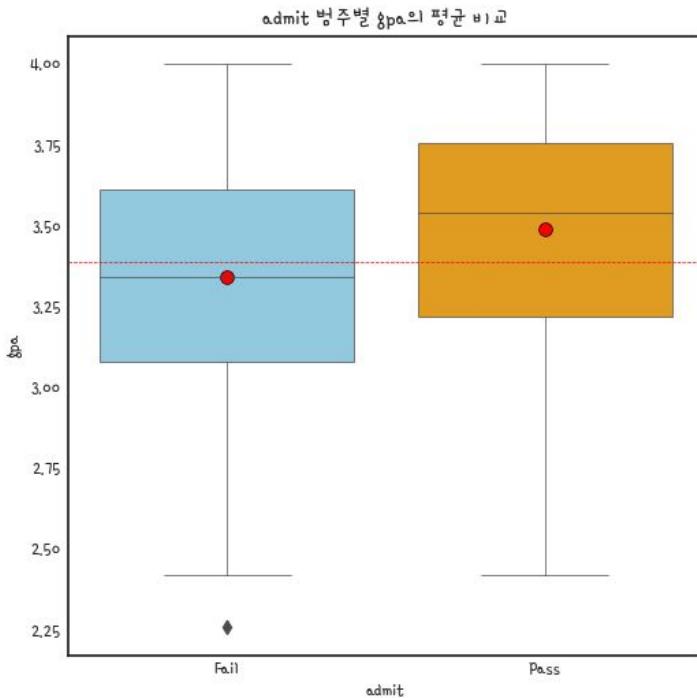


입력변수와 관계 파악: gpa

- admit 범주별 gpa의 상자 수염 그림을 그립니다.

```
>>> hds.plot_box_group(
```

```
    data = df,
    x = 'admit',
    y = 'gpa',
    pal = ['skyblue', 'orange']
)
```

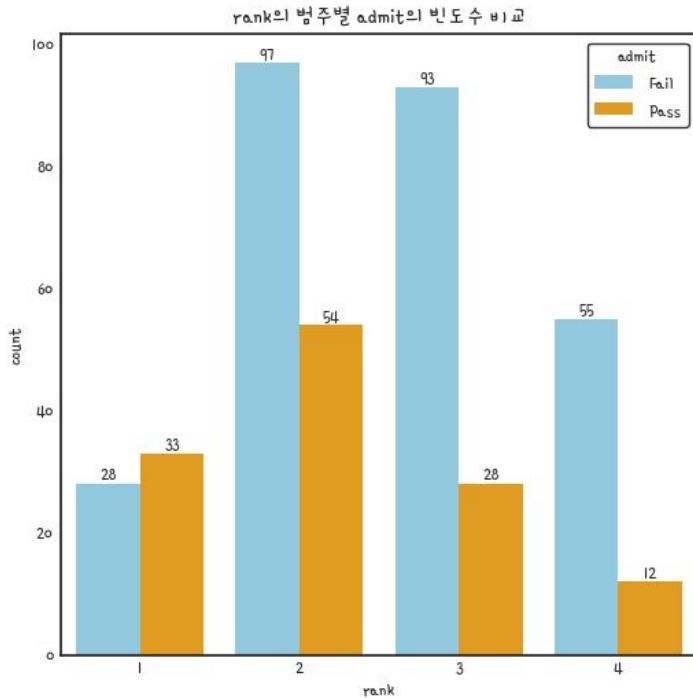


입력변수와 관계 파악: rank

- rank 범주별 admit의 빈도수로 묶음 막대 그래프를 그립니다.

```
>>> hds.plot_bar_dodge_freq
```

```
data = df,
x = 'rank',
y = 'admit',
pal = ['skyblue', 'orange']
)
```

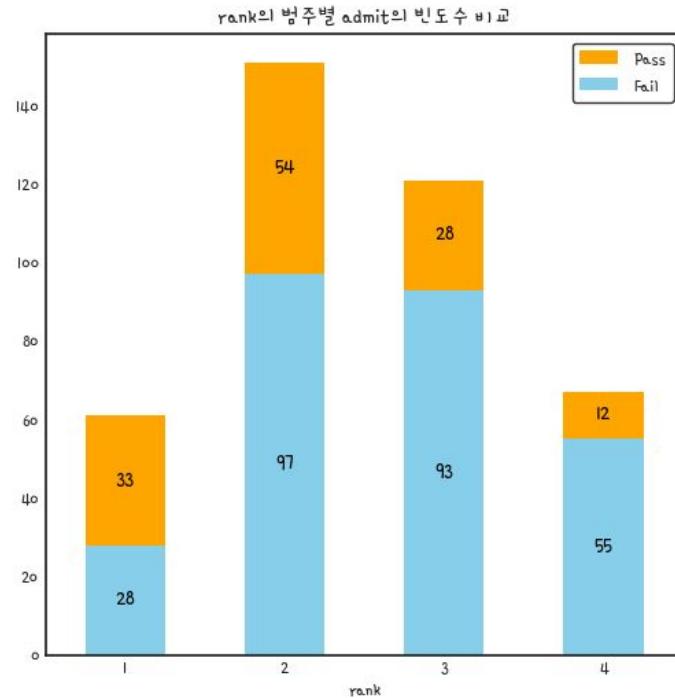


입력변수와 관계 파악: rank(계속)

- rank 범주별 admit의 빈도수로 쓰은 막대 그래프를 그립니다.

```
>>> hds.plot_bar_stack_freq
```

```
data = df,  
  
x = 'rank',  
  
y = 'admit',  
  
pal = ['skyblue', 'orange']  
)
```

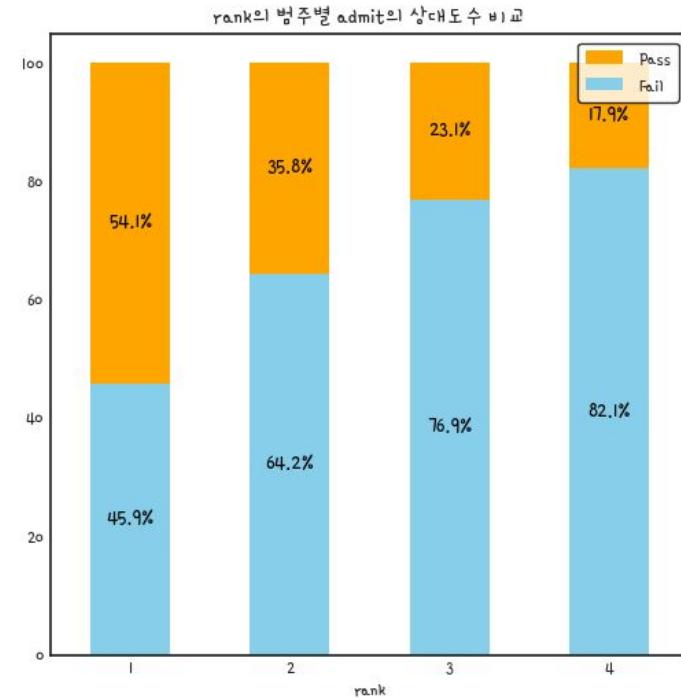


입력변수와 관계 파악: rank(계속)

- rank 범주별 admit의 상대도수로 쓰은 막대 그래프를 그립니다.

```
>>> hds.plot_bar_stack_prop()
```

```
data = df,  
  
x = 'rank',  
  
y = 'admit',  
  
pal = ['skyblue', 'orange']  
)
```



t-검정: gre

- 정규성 검정을 실행합니다.

```
>>> pg.normality(data = df, dv = 'gre', group = 'admit')
```

- admit 범주별 gre로 시리즈를 생성합니다.

```
>>> sp1 = df['gre'][df['admit'].eq('Fail')]
```

```
>>> sp2 = df['gre'][df['admit'].eq('Pass')]
```

- (정규성 가정 불만족) 맨-휘트니 U 검정을 실행합니다.

```
>>> pg.mwu(x = sp1, y = sp2)
```

t-검정: gpa

- 정규성 검정을 실행합니다.

```
>>> pg.normality(data = df, dv = 'gpa', group = 'admit')
```

- admit 범주별 gpa로 시리즈를 생성합니다.

```
>>> sp1 = df['gpa'][df['admit'].eq('Fail')]
```

```
>>> sp2 = df['gpa'][df['admit'].eq('Pass')]
```

- (정규성 가정 불만족) 맨-휘트니 U 검정을 실행합니다.

```
>>> pg.mwu(x = sp1, y = sp2)
```

교차분석: rank

- 범주형 입력변수 rank와 목표변수의 교차테이블을 출력합니다.

```
>>> pd.crosstab(index = df['rank'],  
                 columns = df['admit'],  
                 normalize = 'index',  
                 margins = True)
```

- 교차테이블 빈도수로 교차분석(카이제곱 검정)을 실행합니다.

```
>>> pg.chi2_independence(data = df, x = 'rank', y = 'admit')  
# 피어슨 검정의 유의확률이 0.05 보다 작으면 집단 간 빈도수에 유의한 차이가 있다고 판단합니다.
```

범주형 입력변수의 더미변수 변환

- 범주형 입력변수는 범주 개수 - 1개의 더미변수를 생성합니다.
 - 왜냐하면 범주 개수만큼 더미변수를 생성하면 다중공선성 문제가 발생하기 때문입니다.
- 이번 예제에서는 rank로 더미변수 rank_2, rank_3 및 rank_4를 생성합니다.

rank	rank_1	rank_2	rank_3	rank_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

rank가 1이면 모든 더미변수는 0이므로 목표변수 추정확률에 아무런 영향을 미치지 않습니다.

rank_2의 회귀계수가 음수이면 rank가 1인 학생 대비 rank가 2인 학생의 합격할 확률이 낮다는 것을 의미합니다.

더미변수 생성

- 범주형 입력변수로 더미변수를 생성합니다.

```
>>> df = pd.get_dummies(data = df,  
                      columns = ['rank', 'admit'],  
                      drop_first = True)  
# [참고] columns 매개변수를 생략하면  
# 전체 범주형 변수를 더미변수로 일괄  
# 변환합니다.
```

- df의 처음 10행을 출력합니다.

```
>>> df.head(n = 10)
```

- 목표변수명을 'admit'으로 변경합니다.

```
>>> df = df.rename(columns = {'admit_Pass': 'admit'})
```

실습 데이터셋 분할

- 관련 라이브러리를 호출합니다.

```
>>> from sklearn.model_selection import train_test_split
```

- 전체 데이터셋의 70%를 훈련셋, 30%를 시험셋으로 분할합니다.

```
>>> trSet, teSet = train_test_split(df, test_size = 0.3, random_state = 0)
```

[참고] stratify 매개변수에 총화추출할
변수명을 지정할 수 있습니다.

- 훈련셋의 목표변수 범주별 상대도수를 확인합니다.

```
>>> trSet['admit'].value_counts(normalize = True)
```

- 시험셋의 목표변수 범주별 상대도수를 확인합니다.

```
>>> teSet['admit'].value_counts(normalize = True)
```

입력변수와 목표변수 분리

- 목표변수명을 변수에 할당합니다.

```
>>> yvar = 'admit'
```

- 훈련셋을 목표변수 벡터와 입력변수 행렬로 분리합니다.

```
>>> trReal = trSet[yvar].copy()
```

```
>>> trSetX = trSet.drop(columns = [yvar])
```

- 시험셋을 목표변수 벡터와 입력변수 행렬로 분리합니다.

```
>>> teReal = teSet[yvar].copy()
```

```
>>> teSetX = teSet.drop(columns = [yvar])
```

추정확률

- 로지스틱 회귀분석에서는 목표변수의 실제값이 1인 것을 p , 0인 것을 $1-p$ 로 설정합니다.
 - 이렇게 p 와 $1-p$ 로 설정하면 모든 관측값에 대해 p 를 1에 가깝게 추정하는 문제로 변환할 수 있습니다.
- 입력변수 행렬 X 가 주어졌을 때 p 는 목표변수 y 가 1인 확률, $1-p$ 는 목표변수 y 가 0인 확률로 표현할 수 있습니다.
 - 따라서 p 와 $1-p$ 를 조건부 확률로 표현하면 아래와 같습니다.

$$p = Pr(y = 1|X), \quad 1 - p = Pr(y = 0|X)$$

[참고] 추정확률의 설정

- 로지스틱 회귀모형을 적합할 데이터셋의 입력변수(영어점수~학교등급), 목표변수(합격여부) 및 추정확률 간 관계를 아래 표로 설명할 수 있습니다.

행이름	영어점수	학점	학교등급	합격여부	추정확률
1	380	3.61	3	0	$1-p$
2	660	3.67	3	1	p
3	800	4.00	1	1	p
4	520	2.93	4	0	$1-p$
5	760	3.00	2	1	p
:	:	:	:	:	:

오즈

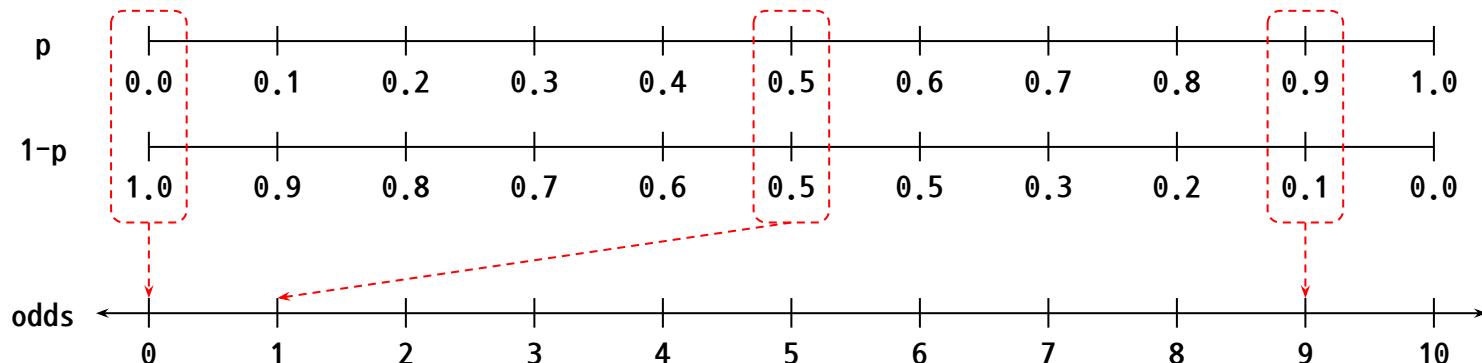
- 로지스틱 회귀분석은 오즈^{odds}(승산, 이길 가능성)에서 시작합니다.

- 오즈는 성공 확률(p)을 실패 확률($1-p$)로 나눈 값입니다.

$$\text{odds} = \frac{p}{1-p} = \frac{Pr(y=1|X)}{Pr(y=0|X)}$$

오즈는 성공 확률이 실패 확률의 배수이므로
오즈가 1보다 크면 y 는 1일 확률이 커집니다.

- p 가 $1-p$ 보다 큰 경우와 작은 경우의 범위가 같지만, 오즈 값의 범위는 다릅니다.

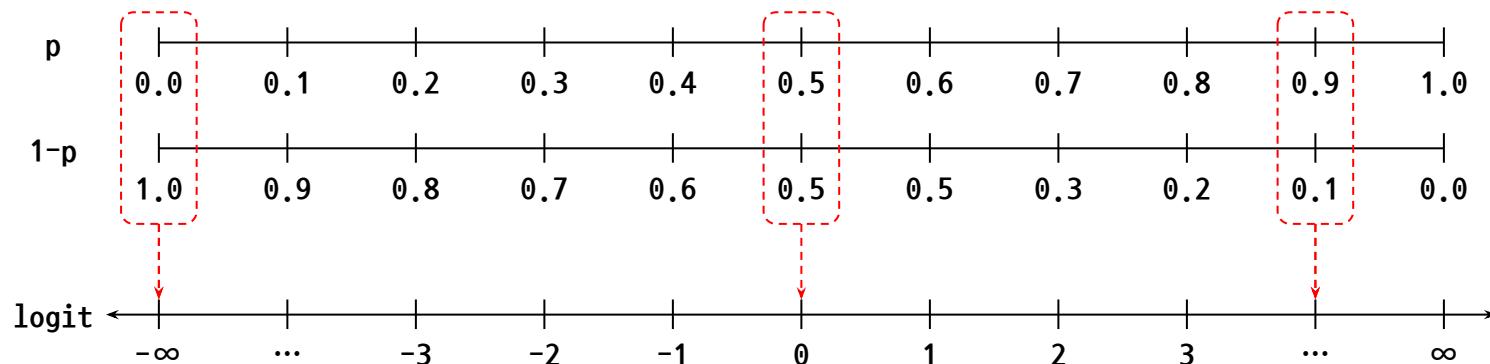


로짓

- 오즈에 로그를 써운 것을 로그오즈 또는 로짓^{logit}이라고 합니다.

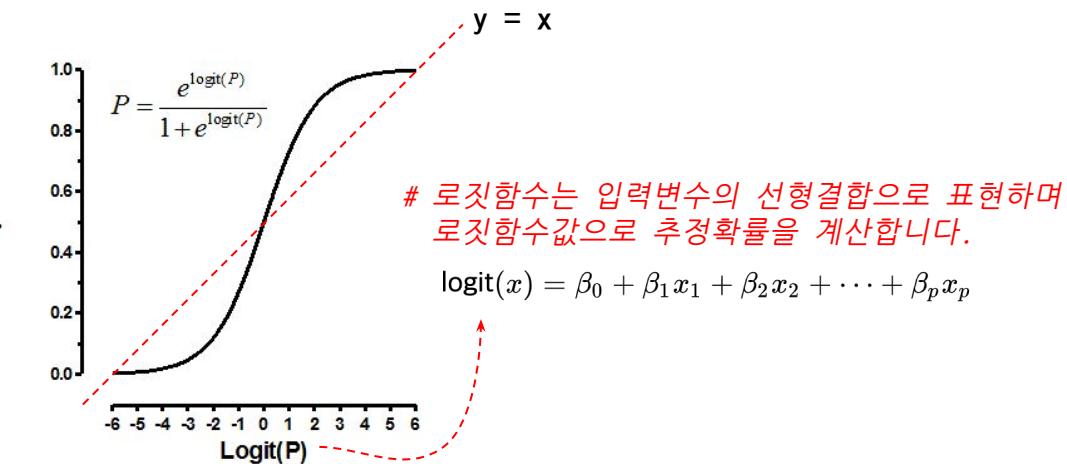
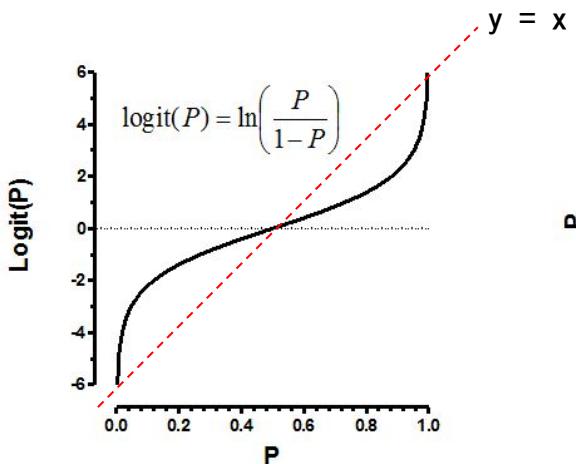
$$\ln \left(\frac{p}{1-p} \right) = \text{logit}(p)$$

- 로짓은 p 가 $1-p$ 보다 큰 경우와 작은 경우에 따른 값의 범위가 같습니다.



로짓 변환

- 로짓함수^{logit function}를 그래프로 그리면 왼쪽 아래 이미지와 같습니다.
 - 이 함수의 x축은 0~1, y축은 $-\infty \sim \infty$ 값을 갖습니다.
 - 로지스틱 회귀모형의 추정확률은 0~1이므로, 로짓함수를 $y = x$ 축대칭합니다.



[참고] 로짓과 확률 p의 관계

- 로짓을 p로 정리하면 아래와 같은 관계식이 성립합니다.

$$\ln\left(\frac{p}{1-p}\right) = \text{logit}(x) \quad \# \text{ 오즈에 로그를 씌운 것을 로짓이라고 합니다.}$$

$$\frac{p}{1-p} = e^{\text{logit}(x)} \quad \# \text{ 왼쪽 항의 로그를 벗기기 위해 양변을 자연상수 } e \text{를 밑으로 하는 지수로 변환합니다.}$$

$$p = (1 - p) \times e^{\text{logit}(x)} \quad \# \text{ 왼쪽 항의 분모를 양변에 곱합니다.}$$

$$(1 + e^{\text{logit}(x)}) \times p = e^{\text{logit}(x)} \quad \# \text{ 전체 식을 } p \text{로 정리합니다.}$$

$$p = \frac{e^{\text{logit}(x)}}{1 + e^{\text{logit}(x)}} = \frac{e^{\ln(\text{odds})}}{1 + e^{\ln(\text{odds})}} \quad \# \text{ 로짓은 오즈에 로그를 씌운 것이므로 오즈를 알면 확률을 계산할 수 있습니다.}$$

가능도 함수

- 표본으로 모수를 추정할 때 가능도 함수 Likelihood Function 를 사용합니다.
 - 모집단의 분포에서 표본이 발생할 가능성(확률밀도)을 의미합니다.

$$L(\theta|x) = \Pr(X = x|\theta) \quad \text{# } x\text{는 표본, } \theta\text{는 모수입니다.}$$

- 서로 독립적인 사건이 함께 발생할 확률은 모두 곱합니다.

$$L(\theta|x) = \Pr(X = x_1|\theta) \times \Pr(X = x_2|\theta) \times \cdots \Pr(X = x_n|\theta) = \prod_{i=1}^n \Pr(X = x_i|\theta)$$

- 양변에 자연로그를 써우면 곱셈을 덧셈으로 변형할 수 있습니다.

$$\ln L(\theta|x) = \sum_{i=1}^n \ln \Pr(X = x_i|\theta)$$

로지스틱 회귀분석의 가능도 함수

- 로지스틱 회귀모형은 다음을 만족해야 합니다.
 - 목표변수의 실제값 y 가 1일 때 성공할 확률 p 를 1에 가깝게 추정합니다.
 - 목표변수의 실제값 y 가 0일 때 실패할 확률 $1-p$ 를 0에 가깝게 추정합니다.
 - 모든 관측값에 대해 p 를 1에 가깝게 추정하는 문제로 바뀝니다.

y	p
0	$1-p$
1	p
:	:

$$L(\beta|X, y) = \Pr(y|X; \theta) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}$$

목표변수의 실제값 y 가 1인 행 개수만큼
성공할 확률 p 를 곱한 것

목표변수의 실제값 y 가 0인 행 개수만큼
실패할 확률 $1-p$ 를 곱한 것

최대 로그 가능도 방법

- 양변에 자연로그를 씌우면 곱셈을 덧셈으로 변형합니다.
 - 가능도 함수는 0~1의 값을 갖지만, 로그 가능도 함수는 $-\infty \sim 0$ 의 값을 갖습니다.
 - 따라서 0에 가까울수록 적합도가 높은 모형입니다.

$$\begin{aligned}\ln L(\beta|X, y) &= \sum_{i=1}^n \left[y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \right] \\ &= \sum_{i=1}^n \left[y_i \ln p_i - y_i \ln(1 - p_i) + \ln(1 - p_i) \right] \\ &= \sum_{i=1}^n \left[y_i \ln \left(\frac{p_i}{1 - p_i} \right) + \ln(1 - p_i) \right]\end{aligned}$$

최대 로그 가능도 방법(계속)

- 앞 페이지 마지막 식에서 대괄호 안을 입력변수의 선형결합으로 치환합니다.

$$\ln\left(\frac{p_i}{1-p_i}\right) = \text{logit}(p_i) = \beta^T X_i$$

$$\ln(1-p_i) = -\ln(1+e^{\beta^T X_i})$$

오른쪽 항을 정리하는 과정은 다음 페이지를
참조하시기 바랍니다.

- 아래는 로지스틱 회귀모형의 로그 가능도 함수를 입력변수의 선형결합으로 정리한 것이고, 이 함수를 최대로 만드는 회귀계수(β)를 추정합니다.

$$\ln L(\beta|X, y) = \sum_{i=1}^n \left[y_i \beta^T X_i - \ln(1 + e^{\beta^T X_i}) \right]$$

[참고] 로그 가능성도 함수의 오른쪽 항 정리

$$\ln\left(\frac{p_i}{1-p_i}\right) = \text{logit}(p_i) = \beta^T X_i$$

$$\ln p_i - \ln(1-p_i) = \beta^T X_i$$

$$\boxed{\ln(1-p_i)} = \ln p_i - \beta^T X_i$$

$$= \ln p_i - \ln e^{\beta^T X_i}$$

$$= \ln\left(\frac{p_i}{e^{\beta^T X_i}}\right) \quad \text{단, } p_i = \frac{e^{\beta^T X_i}}{1 + e^{\beta^T X_i}} \text{ 이므로}$$

$$= \ln\left(\frac{1}{1 + e^{\beta^T X_i}}\right)$$

$$= \boxed{-\ln(1 + e^{\beta^T X_i})}$$

로지스틱 회귀분석 함수

- 로지스틱 회귀분석은 목표변수가 이항분포^{binomial distribution}를 따릅니다.
 - 로지스틱 회귀분석은 일반화 선형모형^{Generalized Linear Model}으로 적합합니다.
 - 연결함수는 로짓함수를 사용합니다.
- 로지스틱 회귀모형을 적합할 때 `statsmodels.api.GLM()` 함수를 사용합니다.
 - `endog`: 목표변수를 지정합니다.
 - `exog`: 입력변수를 지정합니다. 함수에 지정하기 전에 상수 1을 추가해야 합니다.
 - `family`: 목표변수가 이항분포를 따르므로 `sm.families.Binomial()` 함수를 지정합니다.
 - `sm.families.Binomial()` 함수의 `link` 매개변수에 연결함수를 지정할 수 있습니다.

로지스틱 회귀모형 적합 함수 생성

- 관련 라이브러리를 호출합니다.

```
>>> import statsmodels.api as sm
```

- 로지스틱 회귀모형을 반환하는 함수를 생성합니다.

```
>>> def glm(y, X):  
    model = sm.GLM(endog = y, exog = X, family = sm.families.Binomial())  
    return model.fit()
```

로지스틱 회귀모형 적합 및 결과 확인

- 훈련셋으로 로지스틱 회귀모형을 적합하고 결과를 확인합니다.

```
>>> fit1 = glm(y = trReal, X = trSetX)
```

```
>>> fit1.summary() # Deviance, Df Model 및 회귀계수의 P>|z| 등을 차례대로 확인합니다.  

[참고] Cox & Snell의 Pseudo R2를 제공합니다. 공식: 1 - exp((llnull - llf) * 2/nobs)
```

Dep. Variable:	admit	No. Observations:	280	coef	std err	z	P> z	[0.025	0.975]	
Model:	GLM	Df Residuals:	274	const	-3.3721	1.353	-2.492	0.013	-6.024	-0.720
Model Family:	Binomial	Df Model:	5	gre	0.0013	0.001	0.999	0.318	-0.001	0.004
Link Function:	Logit	Scale:	1.0000	gpa	0.8190	0.390	2.103	0.035	0.056	1.582
Method:	IRLS	Log-Likelihood:	-160.27	rank_2	-0.7190	0.373	-1.930	0.054	-1.449	0.011
Date:	Sat, 11 Jun 2022	Deviance:	320.53	rank_3	-1.4775	0.416	-3.555	0.000	-2.292	-0.663
Time:	10:34:16	Pearson chi2:	279.	rank_4	-1.8588	0.519	-3.584	0.000	-2.875	-0.842
No. Iterations:	4	Pseudo R-squ. (CS):	0.1003							

로지스틱 회귀모형의 유의성 검정

- 입력변수를 추가한 residual 모형과 입력변수가 없는 null 모형 간 유의한 차이가 있는지 확인합니다.
 - `statsmodels.api.GLM()` 함수는 residual 모형과 null 모형의 이탈도를 제공합니다.
 - 이탈도는 작을수록 로지스틱 회귀모형의 성능이 좋다는 것을 의미합니다.
- 두 모형의 이탈도 차이를 검정통계량으로 하는 카이제곱 검정을 실행합니다.
 - 두 모형의 성능 차이가 클수록 검정통계량인 이탈도의 차이가 증가합니다.
 - 검정통계량의 자유도는 두 모형의 자유도 차이로 지정합니다.
 - `stats.chi2.cdf()` 함수에 검정통계량과 자유도를 지정하여 유의확률을 계산합니다.

[참고] 이탈도 공식

- 로지스틱 회귀모형의 이탈도 deviance는 아래 공식으로 계산합니다.
 - 상수항만 있는 null 모형: $-2(\ln L_0 - \ln L_s)$ # L_0 는 Null 모형, L_s 는 포화모형의 가능도입니다.
[참고] 포화모형은 완벽한 모형입니다.
 - 입력변수를 추가한 residual 모형: $-2(\ln L_p - \ln L_s)$ # L_p 는 입력변수가 p 개인 모형의 가능도입니다.
- 두 모형 간 이탈도 차이(D)는 아래 공식으로 계산합니다.
 - $D = \left[-2(\ln L_0 - \ln L_s) \right] - \left[-2(\ln L_p - \ln L_s) \right] = -2(\ln L_0 - \ln L_p)$
 - 이탈도 차이(D)는 표본 크기(n)가 무한대로 갈 때 카이제곱 분포를 따릅니다.
 - 카이제곱 분포의 자유도는 두 모형의 자유도 차이입니다.
 - residual 모형의 자유도는 $n-1$ 이고 null 모형의 자유도는 $n-p-1$ 이므로, 두 모형의 자유도 차이는 residual 모형의 입력변수 개수(p)와 같습니다.

로지스틱 회귀모형의 유의성 검정

- 두 모형의 이탈도 차이를 생성합니다.(검정통계량)

```
>>> devGap = fit1.null_deviance - fit1.deviance
```

```
>>> devGap
```

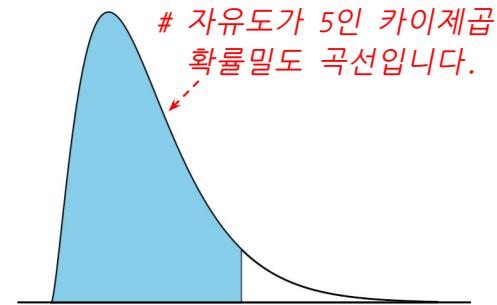
- 두 모형의 자유도 차이를 생성합니다.(카이제곱 분포의 자유도)

```
>>> dfGap = fit1.df_model
```

```
>>> dfGap
```

- 검정통계량과 자유도로 유의확률을 출력합니다.

```
>>> 1 - stats.chi2.cdf(x = devGap, df = dfGap)
```



로지스틱 회귀계수의 유의성 검정

- 로지스틱 회귀계수의 유의성 검정은 z-검정 또는 Wald 검정을 이용합니다.

$$z = \frac{\hat{\beta} - \beta_0}{\sigma_{\hat{\beta}}} \sim N(0, 1^2), W^2 = \frac{(\hat{\beta} - \beta_0)^2}{\sigma_{\hat{\beta}}^2} \sim \chi^2_{(1)}$$

- 두 검정의 귀무가설은 '회귀계수가 0이다' 이므로 분자항의 β_0 는 0입니다.
 - z-검정통계량은 회귀계수를 표준오차로 나눈 값으로 표준정규분포를 따릅니다.
 - Wald 통계량은 회귀계수를 표준오차로 나누어 제곱한 값으로 카이제곱 분포를 따릅니다.
 - `stats.chi2.cdf()` 함수로 자유도가 1인 카이제곱 분포의 오른쪽 확률을 계산합니다.

다중공선성 확인

- 분산팽창지수를 출력하고 다중공선성 입력변수를 확인합니다.

```
>>> hds.vif(X = trSetX)
```

- **vif()** 함수 실행 결과, 분산팽창지수가 10 이상인 입력변수가 없으므로 이번 예제에서는 다중공선성 입력변수는 없는 것으로 판단합니다.
- 만약 분산팽창지수가 10 이상인 입력변수가 다수일 때 분산팽창지수가 가장 큰 입력변수 하나만 훈련셋에서 삭제하고 회귀모형을 다시 적합합니다.
- 새로 적합한 회귀모형의 분산팽창지수를 확인합니다.
- 분산팽창지수가 10 이상인 변수가 없을 때까지 위 과정을 반복합니다.

오즈비

- 로지스틱 회귀계수의 유의성 검정까지 완료하면 회귀모형을 해석할 단계입니다.
입력변수가 1개인 로지스틱 회귀모형으로 오즈를 표현하면 다음과 같습니다.

$$\text{odds} = \frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1}$$

- 오즈비 odds ratio는 두 오즈 간 비율입니다.

$$\text{odds ratio} = \frac{e^{\beta_0 + \beta_1(x_1+1)}}{e^{\beta_0 + \beta_1 x_1}} = e^{\beta_1}$$

- 입력변수가 1단위 증가하면 회귀모형의 추정확률이 오즈비만큼 배수로 증가한다고 해석합니다.

오즈비 확인

- 입력변수별 회귀계수의 오즈비를 출력합니다.

```
>>> np.exp(fit1.params)
```

- y절편 intercept 은 의미가 없으므로 제외하고 나머지 다섯 개의 회귀계수만 확인합니다.
- gre의 오즈비는 1.001이므로 1이라고 봐도 무방합니다.
 - 이것이 의미하는 것은 다른 입력변수의 값을 고정했을 때 gre가 1단위 증가할 때마다 불합격할 확률($1-p$) 대비 합격할 확률(p)이 약 1.001배 증가한다는 것입니다.
- gpa의 오즈비는 2.268이므로 gpa가 1단위 증가할 때 불합격할 확률 대비 합격할 확률이 약 2.268배 증가한다는 것입니다.
 - gre와 gpa는 척도가 다르므로 어떤 변수가 목표변수에 더 영향을 미치는지 아직 알 수 없습니다.

오즈비 확인(계속)

- 더미변수에 대한 오즈비 해석입니다.
 - rank2, rank3, rank4 등 더미변수는 범주형인 rank 대신 생성한 것입니다.
 - 이번 예제에서 더미변수의 의미는 rank가 1(첫 번째 범주)인 학생에 비해 rank가 2~4인 학생의 합격할 배수입니다.
 - 즉, rank2의 오즈비가 0.487이므로 rank가 1인 학생에 비해 rank가 2인 학생은 합격할 확률이 절반 수준으로 낮아진다는 것을 의미합니다.
 - rank3의 오즈비는 0.228이고 rank4의 오즈비는 0.156이므로 rank가 1인 학생에 비하여 rank가 3 또는 4인 학생의 합격할 확률이 지속적으로 낮아지는 현상을 보입니다.

표준화 회귀계수 확인

- fit1 모형의 회귀계수를 출력합니다.

```
>>> fit1.params
```

- 표준화 회귀계수를 생성합니다.

```
>>> beta_z = hds.std_coefs(model = fit1)
```

```
>>> beta_z
```

- 표준화 회귀계수의 절대값을 오름차순 정렬한 결과를 출력합니다.

```
>>> beta_z.abs().sort_values()
```

목표변수의 추정확률 생성

- 훈련셋으로 fit1 모형의 추정확률을 생성하고 실제값과 비교합니다.

```
>>> trProb = fit1.predict(exog = trSetX)
```

```
>>> pd.DataFrame(data = {'real': trReal, 'prob': trProb})
```

- 시험셋으로 fit1 모형의 추정확률을 생성하고 실제값과 비교합니다.

```
>>> teProb = fit1.predict(exog = teSetX)
```

```
>>> pd.DataFrame(data = {'real': teReal, 'prob': teProb})
```

분류모형 성능 평가: ROC 곡선

- 훈련셋과 시험셋의 추정확률로 ROC 곡선을 그리고 AUC를 확인합니다.

```
>>> hds.plot_roc(y_true = trReal,
```

[중요] 추정확률로
지정해야 합니다!

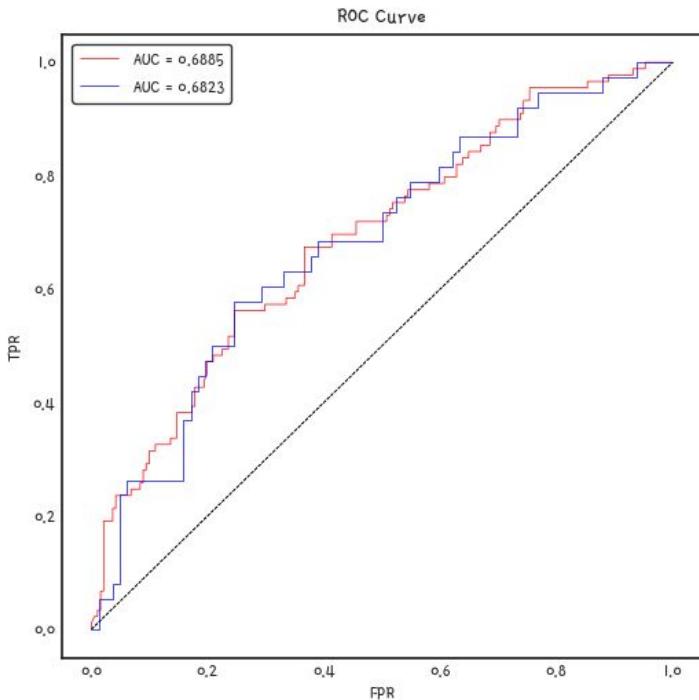
```
          y_prob = trProb,
```

```
          color = 'red')
```

```
>>> hds.plot_roc(y_true = teReal,
```

```
          y_prob = teProb,
```

```
          color = 'blue')
```



목표변수의 추정값 생성

- 분리 기준점을 0.5로 설정합니다.

```
>>> cutoff = 0.5
```

- 훈련셋의 목표변수 추정값(라벨)을 생성합니다.

```
>>> trPred1 = np.where(trProb >= cutoff, 1, 0)
```

- 시험셋의 목표변수 추정값(라벨)을 생성합니다.

```
>>> tePred1 = np.where(teProb >= cutoff, 1, 0)
```

분류모형 성능 평가: 혼동행렬 리포트

- 훈련셋 추정값으로 혼동행렬 리포트를 출력합니다. 시험셋으로도 실습해보세요.

```
>>> hds.clfmetrics(y_true = trReal, y_pred = trPred1)
```

▶ Confusion Matrix

175	16
67	221

추정값
실제값

TP	FN
FP	TN

▶ Classification Report

	precision	recall	f1-score	support
0	0.7231	0.9162	0.8083	191
1	0.5789	0.2472	0.3465	89
accuracy			0.7036	280
macro avg	0.6510	0.5817	0.5774	280
weighted avg	0.6773	0.7036	0.6615	280

[참고] 혼동행렬 관련 함수

- 관련 라이브러리를 호출합니다.

```
>>> from sklearn import metrics
```

- 시험셋 추정값으로 혼동행렬을 출력합니다.

```
>>> print(metrics.confusion_matrix(y_true = teReal, y_pred = tePred1))
```

- 시험셋 추정값으로 분류모형 리포트를 출력합니다.

```
>>> print(metrics.classification_report(y_true = teReal, y_pred = tePred1))
```

- 시험셋 추정값으로 F1 점수를 출력합니다.

```
>>> metrics.f1_score(y_true = teReal, y_pred = tePred1, pos_label = 1)
```

분리 기준점 설정

- 혼동행렬과 F1 점수는 목표변수 추정값(라벨)으로 계산합니다.
 - 추정확률로 추정값을 생성하려면 분리 기준점^{cut-off}이 필요합니다.(보통 0.5로 설정)
 - 불균형 데이터로 학습한 분류모형은 다수 범주를 잘 맞추도록 추정확률을 반환합니다.
 - 이런 상황에서 분리 기준점을 0.5로 설정하면 혼동행렬과 F1 점수는 크게 낮아집니다.
- 불균형된 데이터로 발생하는 문제를 해결하는 방법입니다.
 - ROC 곡선에서 왼쪽 위 모서리에서 가장 가까운 분리 기준점을 사용합니다.
 - 매튜의 상관계수^{Matthew's Correlation Coefficient}가 최대일 때의 분리 기준점을 사용합니다.
 - 목표변수 범주별 백분율이 5:5로 균형화된 훈련셋을 사용합니다.(SMOTE 활용)

[참고] 목표변수 범주별 추정확률 분포

- 목표변수 범주별 추정확률 분포를 확인합니다.

```
>>> sns.boxplot(x = trReal, y = trProb)
```

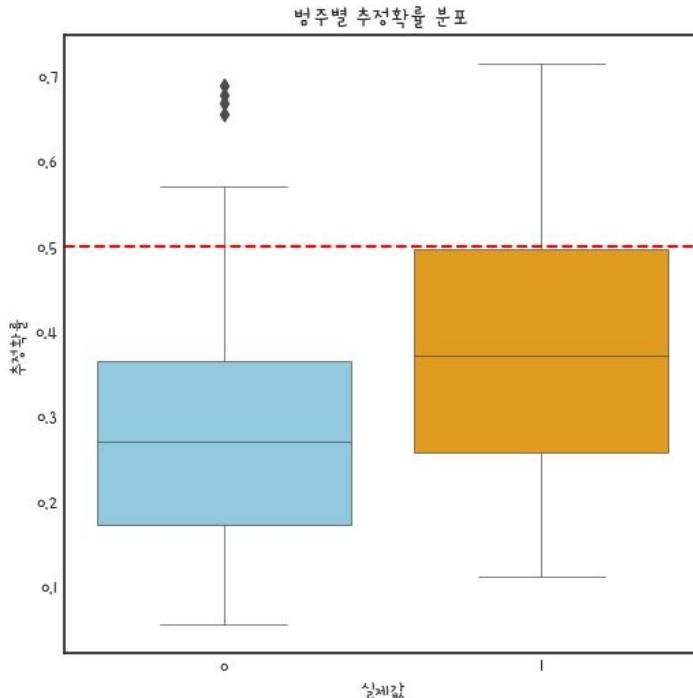
```
>>> plt.title(label = '범주별 추정확률 분포')
```

```
>>> plt.xlabel(xlabel = '실제값')
```

```
>>> plt.ylabel(ylabel = '추정확률')
```

```
>>> plt.axhline(y = 0.5, color = 'red',
```

```
lw = 1.5, ls = '--');
```



[참고] 최적의 분리 기준점 탐색

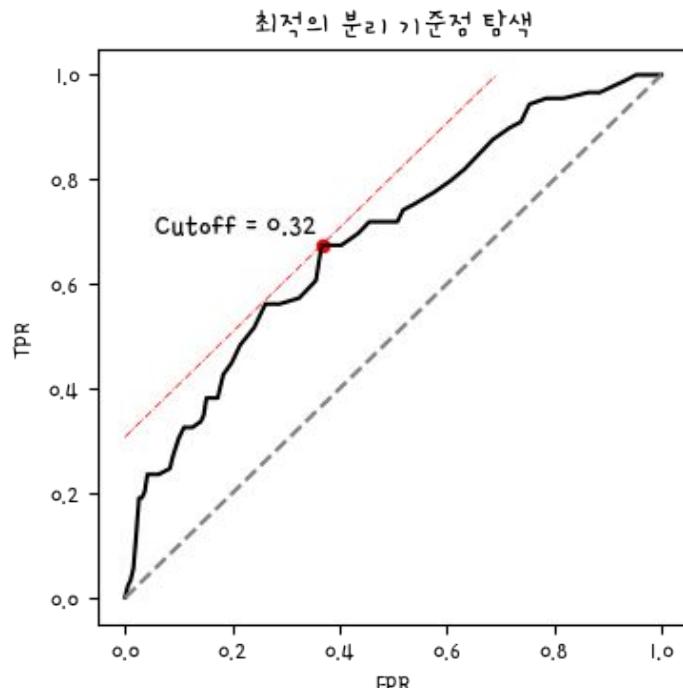
- 분리 기준점마다 분류모형의 성능지표를 계산한 데이터프레임을 생성합니다.

```
>>> cfm = hds.clfCutoffs(y_true = trReal,
                           y_prob = trProb)
```

>>> cfm # 분리 기준점마다 FPR, TPR 및 MCC 값을 계산한 데이터프레임입니다.

- 오른쪽 그림에서 민감도와 특이도의 합계가 최댓값일 때의 분리 기준점을 확인합니다.

```
>>> hds.EpiROC(obj = cfm)
```



[참고] 최적의 분리 기준점으로 성능지표 확인

- 최적의 분리 기준점을 설정합니다.

```
>>> cutoff = 0.32
```

- 분리 기준점 변경 후 시험셋의 목표변수 추정값(라벨)을 생성합니다.

```
>>> tePred2 = np.where(teProb >= cutoff, 1, 0)
```

- 분리 기준점 변경 후 시험셋 추정값으로 혼동행렬과 F1 점수를 확인합니다.

```
>>> hds.clfmetrics(y_true = teReal, y_pred = tePred2)
```

- 분리 기준점 변경 전 시험셋 추정값으로 혼동행렬과 F1 점수를 확인합니다.

```
>>> hds.clfmetrics(y_true = teReal, y_pred = tePred1)
```

[참고] 매트의 상관계수

- 이진 데이터의 분류 성능을 비교하는 척도로 매트의 상관계수를 사용합니다.
 - 매트의 상관계수는 추정값과 실제값이 완벽하게 같으면 1, 완벽하게 다르면 -1입니다.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- 분리 기준점 변경 전/후 시험셋 추정값으로 매트의 상관계수를 비교합니다.

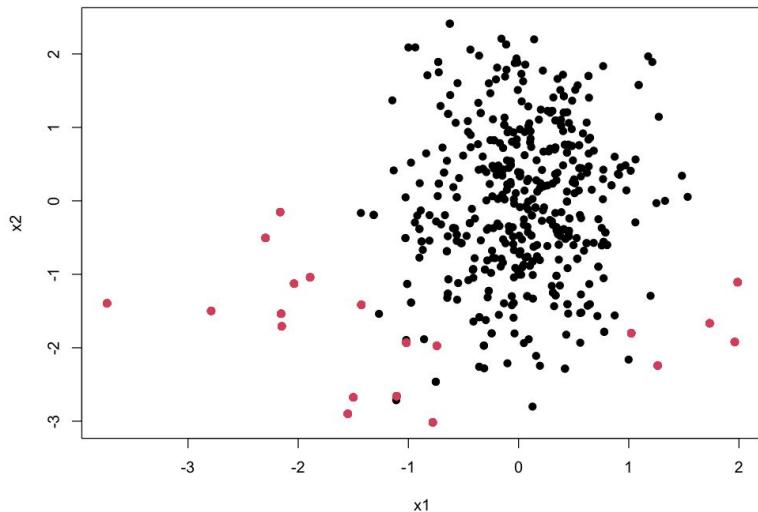
```
>>> metrics.matthews_corrcoef(y_true = teReal, y_pred = tePred1)
```

```
>>> metrics.matthews_corrcoef(y_true = teReal, y_pred = tePred2)
```

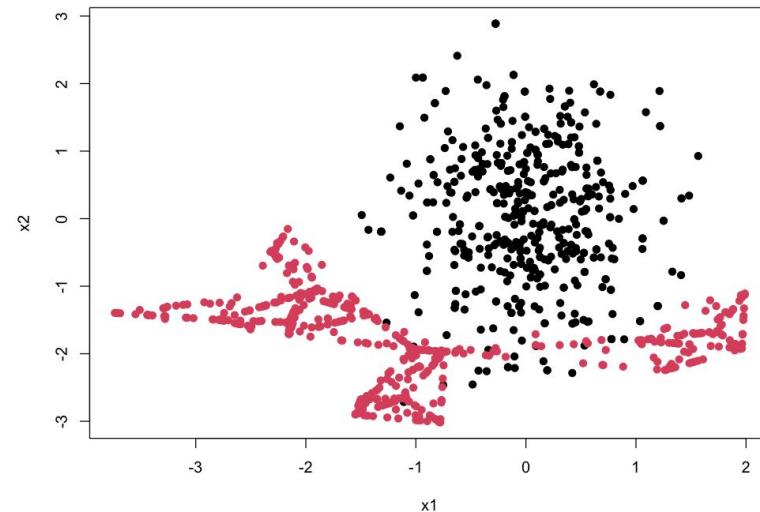
[참고] SMOTE

- SMOTE^{Synthetic Minority Oversampling Technique}는 소수 범주 관측값(빨간점)과 가까운 k개 이웃 중에서 한 점을 선택하고 두 점의 직선 위에 있는 임의의 값을 생성합니다.

[Over & Under]



[SMOTE(k=5)]



End of Document