

『 4과목 :』 데이터 분석과 인사이트 도출

- 통계와 확률 / 통계분석 -practice
- 머신러닝 기반 데이터 분석-지도
- 머신러닝 기반 데이터 분석-비지도
- 기타 데이터 마이닝

- 『4과목』 Self 점검



학습목표

- 이 워크샵에서는 와인 분석을 통해 통계분석과 통계분석 방법에 대해 알 수 있습니다.

눈높이 체크

- 통계분석의 방법에 대해 알고 계신가요?



1. 가설 검정을 통한 통계 분석 사례

와인 품질 예측하기

와인 품질 등급 예측하기

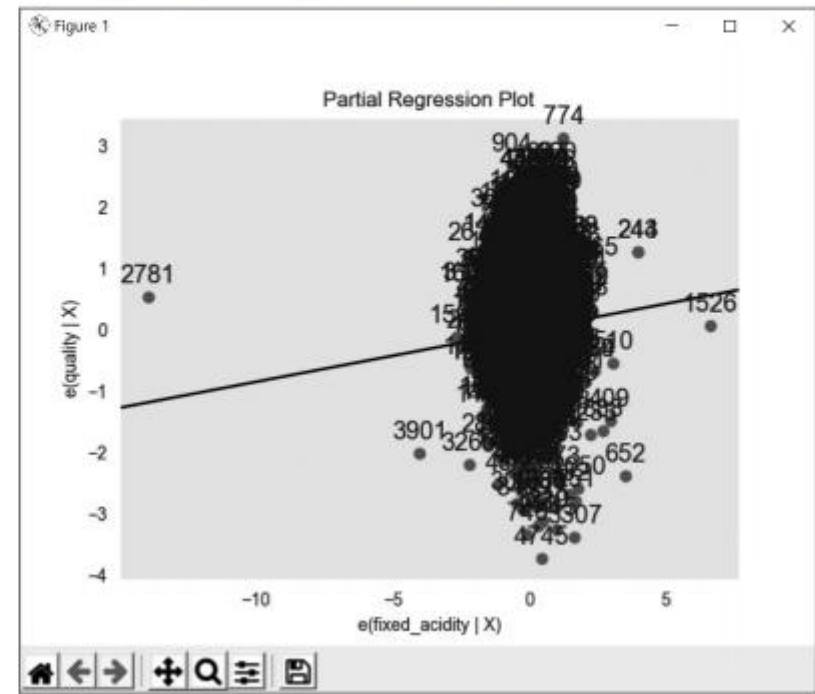
목표	와인 속성을 분석하여 품질 등급을 예측한다.	
핵심 개념	기술 통계, 회귀 분석, t-검정, 히스토그램	
데이터 수집	레드 와인/화이트 와인 데이터셋: 캘리포니아 어바인 대학의 머신러닝 저장소에서 다운로드	
데이터 준비	수집한 데이터 파일 병합	
데이터 탐색	1. 정보 확인: <code>info()</code> 2. 기술 통계 확인: <code>describe()</code> , <code>unique()</code> , <code>value_counts()</code>	
데이터 모델링	1. 데이터를 두 그룹으로 비교 분석 <ul style="list-style-type: none">• 그룹별 기술 통계 분석: <code>describe()</code>• t-검정: <code>scipy</code> 패키지의 <code>ttest_ind()</code>• 회귀 분석: <code>statsmodels.formula.api</code> 패키지의 <code>ols()</code>	2. 품질 등급 예측 <ul style="list-style-type: none">• 샘플을 독립 변수(x)로 지정 → 회귀 분석 모델 적용 → 종속 변수(y)인 품질 (quality) 예측



와인 품질 예측하기

결과 시각화

1. 히스토그램을 이용한 시각화

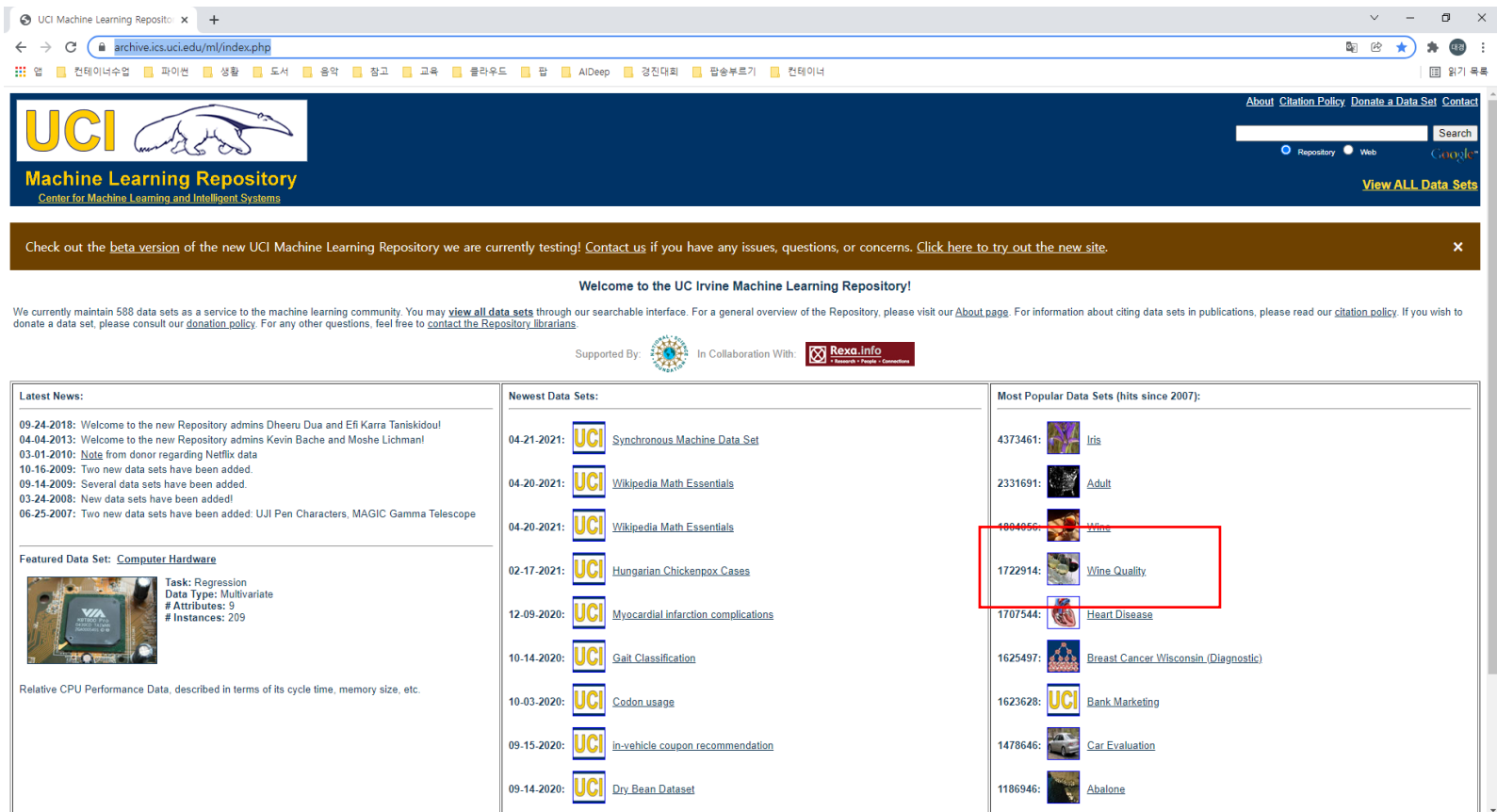


1. 가설 검정을 통한 통계 분석 사례

데이터 수집

- 캘리포니아 어바인 대학의 머신러닝 저장소에서 제공하는 오픈 데이터를 사용

<https://archive.ics.uci.edu/ml/index.php>























UCI Machine Learning Repository

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing! [Contact us](#) if you have any issues, questions, or concerns. [Click here to try out the new site.](#)

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 588 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#).

Supported By:  In Collaboration With: 

Latest News:	Newest Data Sets:	Most Popular Data Sets (hits since 2007):
09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou! 04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman! 03-01-2010: Note from donor regarding Netflix data 10-16-2009: Two new data sets have been added. 09-14-2009: Several data sets have been added. 03-24-2008: New data sets have been added! 06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope	04-21-2021:  Synchronous Machine Data Set 04-20-2021:  Wikipedia Math Essentials 04-20-2021:  Wikipedia Math Essentials 02-17-2021:  Hungarian Chickenpox Cases 12-09-2020:  Myocardial infarction complications 10-14-2020:  Gait Classification 10-03-2020:  Codon usage 09-15-2020:  in-vehicle coupon recommendation 09-14-2020:  Dry Bean Dataset	4373461:  Iris 2331691:  Adult 1804056:  Wine 1722914:  Wine Quality 1707544:  Heart Disease 1625497:  Breast Cancer Wisconsin (Diagnostic) 1623628:  Bank Marketing 1478646:  Car Evaluation 1186946:  Abalone

Featured Data Set: [Computer Hardware](#)

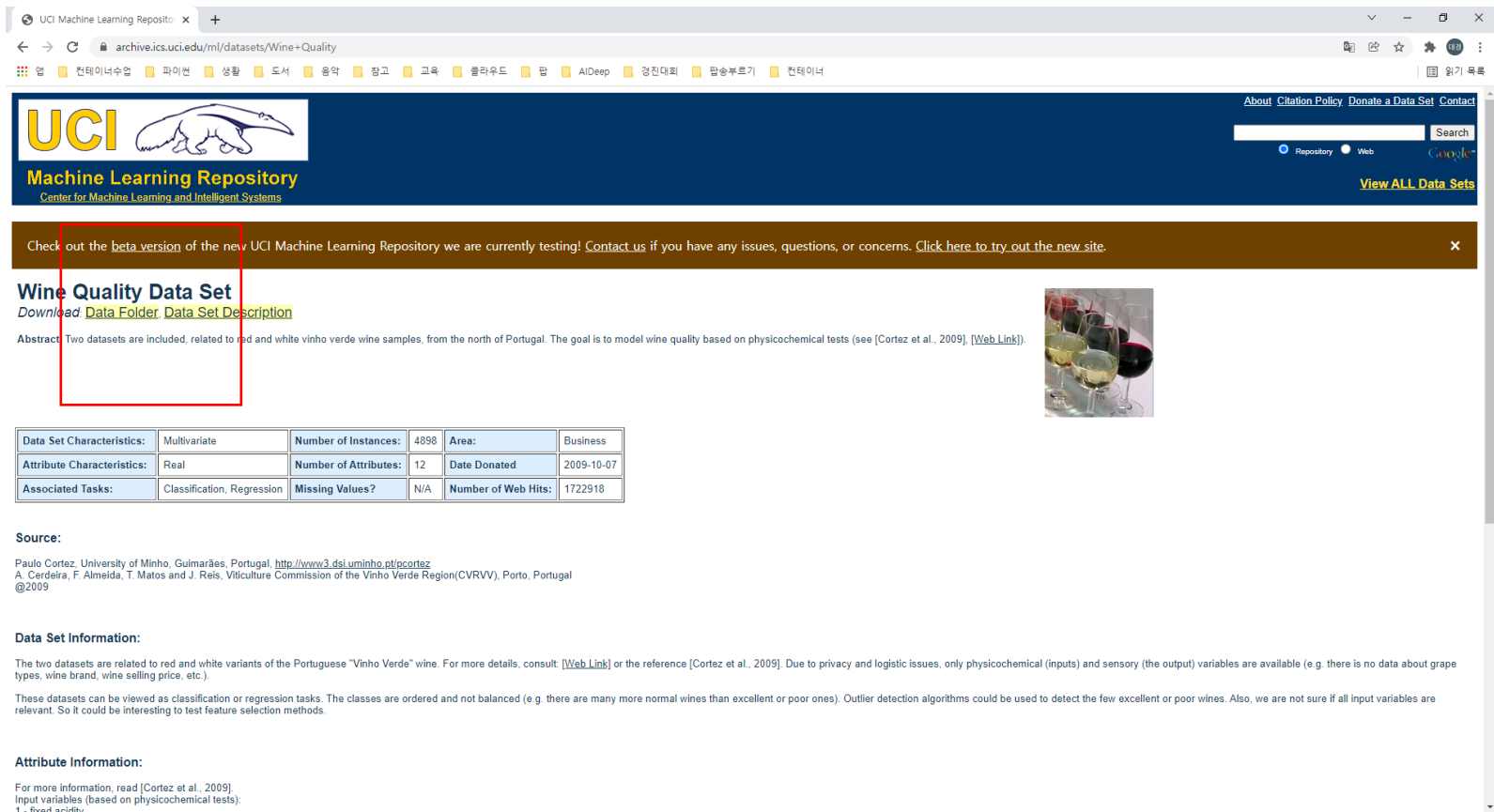
Task: Regression
Data Type: Multivariate
Attributes: 9
Instances: 209

Relative CPU Performance Data, described in terms of its cycle time, memory size, etc.

1. 가설 검정을 통한 통계 분석 사례

데이터 수집

- Download: Data Folder 클릭



UCI Machine Learning Repository

archive.ics.uci.edu/ml/datasets/Wine+Quality

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing! [Contact us](#) if you have any issues, questions, or concerns. [Click here to try out the new site.](#)

Wine Quality Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], [\[Web Link\]](#)).

Data Set Characteristics:	Multivariate	Number of Instances:	4898	Area:	Business
Attribute Characteristics:	Real	Number of Attributes:	12	Date Donated	2009-10-07
Associated Tasks:	Classification, Regression	Missing Values?	N/A	Number of Web Hits:	1722918

Source:

Paulo Cortez, University of Minho, Guimarães, Portugal, <http://www3.dsi.uminho.pt/cortez>
A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal
©2009

Data Set Information:

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult [\[Web Link\]](#) or the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

Attribute Information:

For more information, read [Cortez et al., 2009]
Input variables (based on physicochemical tests):
1 - fixed acidity



1. 가설 검정을 통한 통계 분석 사례

데이터 수집

● Download: Data Folder 클릭







1. 가설 검정을 통한 통계 분석 사례

데이터 수집

- 다운로드한 CSV 파일 정리하기
 - 엑셀은 CSV 파일을 열 때 쉼표를 열 구분자로 사용하므로 열이 깨진 것처럼 보임
 - 엑셀에서 세미콜론을 열 구분자로 인식하도록 파일을 다시 저장해야 함

› 내 PC › 로컬 디스크 (C:) › DEV › python_works › data ›

이름		수정한 날
 winequality-white.csv		2023-04-1
 winequality-red.csv		2023-04-1



1. 가설 검정을 통한 통계 분석 사례

데이터 수집

● 다운로드한 CSV 파일 정리하기

내 PC > 로컬 디스크 (C:) > DEV > python_works > data >	
이름	수정된 날짜
winequality-white2.csv	2023-04-06 오전 9:48
winequality-white.csv	2023-04-06 오전 9:42
winequality-red2.csv	2023-04-06 오전 9:48
winequality-red.csv	2023-04-06 오전 9:42

파일

소스코드

실습환경

py3_10_basic

```
import pandas as pd
```

```
red_df = pd.read_csv('datasets/winequality-red.csv', sep = ';', header = 0, engine = 'python')
```

```
white_df = pd.read_csv('datasets/winequality-white.csv', sep = ';', header = 0, engine='python')
```

소스코드

```
print(red_df.head())  
print("=" * 80)  
print(white_df.head())
```

```
red_df.to_csv('datasets/winequality-red2.csv', index = False)
```

```
white_df.to_csv('datasets/winequality-white2.csv', index = False)
```

비고

테이블 형태의 CSV 파일을 다루기 위해 pandas 라이브러리 패키지를 pd 이름으로 로드
pandas의 read_csv() 함수를 사용해 CSV 파일을 읽어온음 이때 CSV 파일 데이터의 열 구분자를 세미콜론으로 지정하기 위해 sep 매개변수 값을 ';'으로 지정
pandas로 읽은 CSV 데이터는 테이블 형태의 DataFrame 객체(red_df, white_df)에 있음. 이 상태 그대로 CSV 파일로 저장

1. 가설 검정을 통한 통계 분석 사례

데이터 준비

- 데이터 병합하기
 - 레드 와인과 화이트 와인 파일 합치기

파일

소스코드

실습환경

py3_10_basic

소스코드

```
red_df.insert(0, column = 'type', value = 'red')
print(red_df.head())
print(red_df.shape)
print("=" * 80)
white_df.insert(0, column = 'type', value = 'white')
print(white_df.head())
print(white_df.shape)
print("=" * 80)
wine = pd.concat([red_df, white_df])
print(wine.head())
print(wine.shape)
wine.to_csv('datasets/wine.csv', index = False)
```

결과값1

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	
quality													
0	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	red	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	red	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	red	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

비고





1. 가설 검정을 통한 통계 분석 사례

데이터 탐색

● 기본 정보 확인하기

파일

소스코드

실습환경

py3_10_basic

소스코드

wine.info()

결과값1

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6497 entries, 0 to 4897
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   type                  6497 non-null  object  
1   fixed acidity          6497 non-null  float64  
2   volatile acidity       6497 non-null  float64  
3   citric acid            6497 non-null  float64  
4   residual sugar         6497 non-null  float64  
5   chlorides              6497 non-null  float64  
6   free sulfur dioxide    6497 non-null  float64  
7   total sulfur dioxide   6497 non-null  float64  
8   density                6497 non-null  float64  
9   pH                     6497 non-null  float64  
10  sulphates              6497 non-null  float64  
11  alcohol                6497 non-null  float64  
12  quality                6497 non-null  int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 710.6+ KB
```

비고

전체 샘플은 6,497개이고 속성을 나타내는 열은 13개, 각 속성의 이름은 type부터 quality까지 속성 중에서 실수 타입(float64)은 11개, 정수 타입(int64)은 1개(quality), 객체 타입(object)이 1개(type) 독립 변수(x)는 type부터 alcohol 까지 12개, 종속 변수(y)는 1개(quality)



1. 가설 검정을 통한 통계 분석 사례

데이터 탐색

● 함수를 사용해 기술 통계 구하기

파일

소스코드

실습환경

py3_10_basic

소스코드

```
wine.columns = wine.columns.str.replace(' ', '_')
wine.head()
```

wine.describe()

결과값1

	type	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates		
0	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	red	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	red	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	red	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

결과값2

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000

열이름 정렬하기

열 이름에 공백이 있으면 밑줄로 바꾼 뒤 한 단어로 연결

변경된 열 이름을 확인

비고

describe() 함수를 사용하여 속성별 개수, 평균, 표준편차, 최소값, 전체 데이터 백분율에 대한 25번째 백분위수(25%), 중앙값인 50번째 백분위수(50%), 75번째 백분위수(75%) 그리고 100번째 백분위수인 최대값max을 출력



1. 가설 검정을 통한 통계 분석 사례

데이터 탐색

● 함수를 사용해 기술 통계 구하기

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>sorted(wine.quality.unique()) wine.quality.value_counts()</pre>
결과값1	[3, 4, 5, 6, 7, 8, 9]
결과값2	<pre>6 2836 5 2138 7 1079 4 216 8 193 3 30 9 5 Name: quality, dtype: int64</pre>
비고	wine.quality.unique() 함수를 사용하여 quality 속성값 중에서 유일한 값을 출력. 이를 통해 와인 품질 등급quality은 3, 4, 5, 6, 7, 8, 9의 7개 등급이 있다는 것을 알 수 있음 quality.value_counts() 함수는 quality 속성값에 대한 빈도수를 보여줌. 6등급인 샘플이 2,826개로 가장 많고, 9등급인 샘플이 5개로 가장 적은 것을 알 수 있음



1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

● describe() 함수로 그룹 비교하기

파일	소스코드
실습환경	<pre>py3_10_basic wine.groupby('type')['quality'].describe()</pre>
소스코드	<pre>wine.groupby('type')['quality'].mean() wine.groupby('type')['quality'].std() wine.groupby('type')['quality'].agg(['mean', 'std'])</pre>
결과값1	<pre>count mean std min 25% 50% 75% max type red 1599.0 5.636023 0.807569 3.0 5.0 6.0 6.0 8.0 white 4898.0 5.877909 0.885639 3.0 5.0 6.0 6.0 9.0</pre>
결과값4	<pre>mean std type red 5.636023 0.807569 white 5.877909 0.885639</pre>
비고	<p>레드 와인과 화이트 와인을 구분하는 속성인 type을 기준으로 그룹을 나눈 뒤 그룹 안에서 quality 속성을 기준으로 기술 통계를 구함</p> <p>기술 통계 전부를 구할 때는 describe() 함수를 사용하지만 mean() 함수로 평균만 구하거나 std() 함수로 표준편차만 따로 구할 수도 있음</p> <p>mean() 함수와 std() 함수를 묶어서 한 번에 사용하려면 agg() 함수를 사용</p>



1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

- t-검정과 회귀 분석으로 그룹 비교하기
 - t-검정을 위해서는 scipy 라이브러리 패키지를 사용
 - 회귀 분석을 위해서는 statsmodels 라이브러리 패키지를 사용
 - 명령 프롬프트 창에서 다음과 같이 입력하여 statsmodels 패키지를 설치

파일	소스코드
실습환경	<pre>py3_10_basic from scipy import stats from statsmodels.formula.api import ols, glm red_wine_quality = wine.loc[wine['type'] == 'red', 'quality'] white_wine_quality = wine.loc[wine['type'] == 'white', 'quality']</pre>
소스코드	<pre>stats.ttest_ind(red_wine_quality, white_wine_quality, equal_var = False) Rformula = 'quality ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides + free_sulfur_dioxide + total_sulfur_dioxide + density + pH + sulphates + alcohol' regression_result = ols(Rformula, data = wine).fit() regression_result.summary()</pre>

결과값1

비고

1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

● t-검정과 회귀 분석으로 그룹 비교하기

파일

소스코드

실습환경

py3_10_basic

결과값1

```
OLS Regression Results

Dep. Variable: quality    R-squared: 0.292
Model: OLS              Adj. R-squared: 0.291
Method: Least Squares   F-statistic: 243.3
Date: Fri, 15 Nov 2024   Prob (F-statistic): 0.00
Time: 09:04:31          Log-Likelihood: -7215.5
No. Observations: 6497   AIC: 1.445e+04
Df Residuals: 6485      BIC: 1.454e+04
Df Model: 11
Covariance Type: nonrobust

               coef    std err          t      P>|t| [0.025   0.975]
-----
Intercept    55.7627    11.894     4.688    0.000   32.447    79.079
fixed_acidity  0.0677     0.016     4.346    0.000    0.037    0.098
volatile_acidity -1.3279    0.077   -17.162    0.000   -1.480   -1.176
citric_acid   -0.1097    0.080    -1.377    0.168   -0.266    0.046
residual_sugar  0.0436    0.005     8.449    0.000    0.033    0.054
chlorides     -0.4837    0.333    -1.454    0.146   -1.136    0.168
free_sulfur_dioxide 0.0060    0.001     7.948    0.000    0.004    0.007
total_sulfur_dioxide -0.0025    0.000    -8.969    0.000   -0.003   -0.002
density      -54.9669   12.137   -4.529    0.000  -78.760  -31.173
pH            0.4393    0.090     4.861    0.000    0.262    0.616
sulphates     0.7683    0.076    10.092    0.000    0.619    0.917
alcohol       0.2670    0.017    15.963    0.000    0.234    0.300

Omnibus: 144.075   Durbin-Watson: 1.646
Prob(Omnibus): 0.000   Jarque-Bera (JB): 324.712
Skew: -0.006       Prob(JB): 3.09e-71
Kurtosis: 4.095     Cond. No.    2.49e+05
```

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 2.49e+05. This might indicate that there are strong multicollinearity or other numerical problems.

1. Dependent Variable (종속 변수):

- 이 모델에서는 'quality'가 종속 변수로 사용되었습니다.

2. R-squared (R제곱):

- R제곱 값은 모델이 종속 변수의 변동성을 얼마나 설명하는지를 나타냅니다. 여기서는 0.292로, 모델이 전체 데이터의 약 29.2%의 변동성을 설명한다고 볼 수 있습니다.

3. Adj. R-squared (조정된 R제곱):

- 조정된 R제곱 값은 독립 변수의 수와 관련하여 R제곱 값을 보정한 값입니다.

4. Method (방법):

- 이 모델에서는 최소 제곱법(Least Squares)을 사용했습니다.

5. F-statistic (F-통계량):

- F-통계량은 모델의 유의성을 검정하는데 사용됩니다. 이 값이 클수록 모델이 유의하다는 것을 의미합니다.

6. Prob (F-statistic) (F-통계량의 확률값):

- F-통계량의 확률값은 해당 통계량이 표본에서 관찰되기 힘들다는 가정하에, 실제로 표본에서 관찰된 통계량과 같거나 더 극단적인 통계량이 나타날 확률입니다. 이 값이 작을수록 모델이 유의합니다.

7. coef (계수):

- 각 독립 변수의 계수를 나타냅니다. 이는 해당 독립 변수의 단위 변화가 종속 변수에 어떻게 영향을 미치는지를 나타냅니다.

8. std err (표준 오차):

- 계수의 표준 오차를 나타냅니다.

9. t (t-통계량):

- t-통계량은 각 계수가 유의한지 여부를 검정하는데 사용됩니다.

10. P>|t| (유의확률):

- 각 계수에 대한 유의확률을 나타냅니다. 일반적으로 유의확률이 0.05보다 작으면 해당 계수는 통계적으로 유의합니다.

11. [0.025, 0.975] (신뢰구간):

- 계수의 95% 신뢰구간을 나타냅니다.

12. Omnibus, Prob(Omnibus), Jarque-Bera (JB), Skew, Kurtosis:

- 잔차에 대한 정규성, 왜도, 첨도 등을 검정하는 통계량입니다.

13. Durbin-Watson (더빈-왓슨)

- 잔차의 자기상관 여부를 검정합니다.

14. Warnings (경고):

- 이 부분에는 모델에 대한 경고 메시지가 포함될 수 있습니다. 여기서는 오차의 공분산 행렬이 올바르게 지정되었는지와 다중공선성 문제가 있는지 여부에 대한 경고가 포함될 수 있습니다.

ctly specified.
are



1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

● t-검정과 회귀 분석으로 그룹 비교하기

파일

소스코드

실습환경

py3_10_basic

비고

- t-검정에 필요한 scipy 패키지의 stats 함수와 회귀 분석에 필요한 statsmodels.formula.api 패키지의 ols, glm 함수를 로드
- 그룹 분리하기
- 레드 와인 샘플의 quality 값만 찾아서 red_wine에 저장
- 화이트 와인 샘플의 quality 값만 찾아서 white_wine에 저장
- scipy 패키지의 stats.ttest_ind() 함수를 사용하여 t-검정을 하고 두 그룹 간 차이를 확인
- 선형 회귀 분석 수행하기
- 선형 회귀 분석식의 종속 변수(y)와 독립 변수(x1~x10)를 구성
- 선형 회귀 모델 중에서 OLS Ordinary Least Squares 모델을 사용
- 선형 회귀 분석과 관련된 통계값을 출력



1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

- 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

파일	소스코드
실습환경	<pre>py3_10_basic sample1 = wine[wine.columns.difference(['quality', 'type'])] sample1 = sample1[0:5][:]</pre>
소스코드	<pre>sample1_predict = regression_result.predict(sample1) print(sample1_predict) print("=" * 80) print(wine[0:5]['quality'])</pre>
결과값1	<pre>0 4.997607 1 4.924993 2 5.034663 3 5.680333 4 4.997607 dtype: float64</pre>
결과값2	<pre>0 5 1 5 2 5 3 6 4 5 Name: quality, dtype: int64</pre>
비고	

1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

- 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>data = {"fixed_acidity" : [8.5, 8.1], "volatile_acidity": [0.8, 0.5], "citric_acid": [0.3, 0.4], "residual_sugar": [6.1, 5.8], "chlorides": [0.055, 0.04], "free_sulfur_dioxide": [30.0, 31.0], "total_sulfur_dioxide": [98.0, 99], "density": [0.996, 0.91], "pH": [3.25, 3.01], "sulphates": [0.4, 0.35], "alcohol": [9.0, 0.88]}</pre> <pre>sample2 = pd.DataFrame(data, columns= sample1.columns) print(sample2) print("=" * 80) sample2_predict = regression_result.predict(sample2) print(sample2_predict)</pre>
결과값1	<pre> alcohol chlorides citric_acid density fixed_acidity free_sulfur_dioxide pH residual_sugar sulphates total_sulfur_dioxide volatile_acidity 0 9.00 0.055 0.3 0.996 8.5 30.0 3.25 6.1 0.40 98.0 0.8 1 0.88 0.040 0.4 0.910 8.1 31.0 3.01 5.8 0.35 99.0 0.5</pre>
결과값2	<pre>0 4.809094 1 7.582129 dtype: float64</pre>
비고	



1. 가설 검정을 통한 통계 분석 사례

데이터 모델링

● 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

파일

소스코드

실습환경

py3_10_basic

예측에 사용할 첫 번째 샘플 데이터 만들기

- wine에서 quality와 type 열은 제외하고, 회귀 분석 모델에 사용할 독립 변수만 추출하여 sample1에 저장
- sample1에 있는 샘플 중에서 0번~4번 5개 샘플만 추출하고, sample1에 다시 저장하여 예측에 사용할 샘플을 제작

첫 번째 샘플의 quality 예측하기

- 샘플 데이터를 회귀 분석 모델 regression_result의 예측 함수 predict()에 적용하여 수행한 뒤 결과 예측값을 sample1_predict에 저장
- sample1_predict를 출력하여 예측한 quality를 확인
- wine에서 0번부터 4번까지 샘플의 quality 값을 출력하여 sample1_predict이 맞게 예측되었는지 확인

비고

예측에 사용할 두 번째 샘플 데이터 만들기

- 회귀식에 사용한 독립 변수에 대입할 임의의 값을 딕셔너리 형태로 제작
- 딕셔너리 형태의 값과 sample1의 열 이름만 뽑아 데이터프레임으로 묶은 sample2를 제작
- sample2를 출력하여 제대로 구성되었는지 확인

두 번째 샘플의 quality 예측하기

- 샘플 데이터를 회귀 분석 모델 regression_result의 예측 함수 predict()에 적용하여 수행한 뒤 결과 예측값을 sample2_predict에 저장
- sample2_predict를 출력하여 예측한 quality를 확인



1. 가설 검정을 통한 통계 분석 사례

결과 시각화

● 와인 유형에 따른 품질 등급 히스토그램 그리기

파일

소스코드

실습환경

py3_10_basic

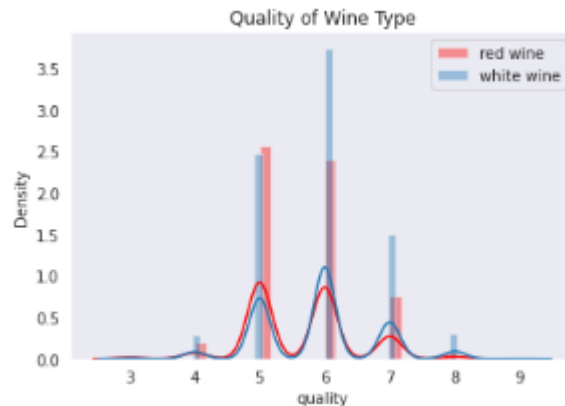
```
import matplotlib.pyplot as plt
import seaborn as sns
```

소스코드

```
sns.set_style('dark')
sns.distplot(red_wine_quality, kde = True, color = "red", label = 'red wine')
sns.distplot(white_wine_quality, kde = True, label = 'white wine')
plt.title("Quality of Wine Type")
plt.legend()
plt.show()
```

x축: qualit
y축: 확률 밀도 함수값

결과값1



비고

시각화에 필요한 패키지를 로드
커널 밀도 추정(kde)을 적용한 히스토그램 그리기
히스토그램 차트의 배경색 스타일을 설정
레드 와인에 대한 distplot 객체를 생성
화이트 와인에 대한 distplot 객체를 생성
차트 제목을 설정
차트 범례를 설정
차트를 표시

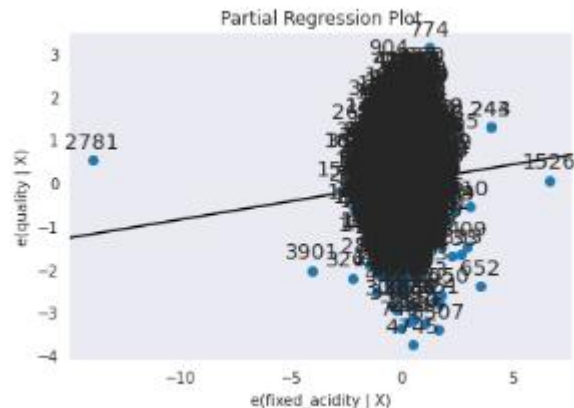


결과 시각화

- 부분 회귀 플롯으로 시각화하기
- 독립 변수가 2개 이상인 경우에는 부분 회귀 플롯을 사용하여 하나의 독립 변수가 종속

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>import statsmodels.api as sm others = list(set(wine.columns).difference(set(["quality", "fixed_acidity"]))) p, resid = sm.graphics.plot_partregress("quality", "fixed_acidity", others, data = wine, ret_coords = True) plt.show()</pre>

결과값1



비고

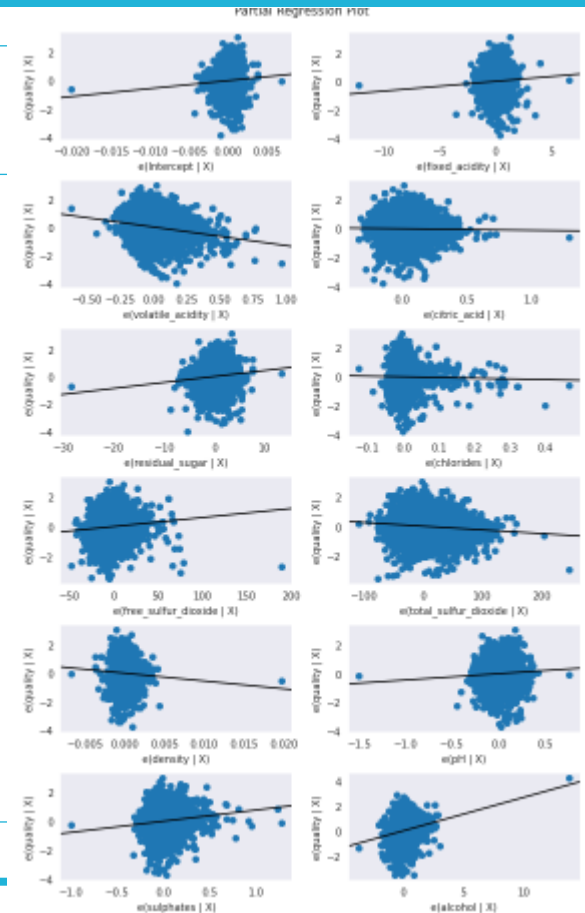
1. 가설 검정을 통한 통계 분석 사례

결과 시각화

● 부분 회귀 플롯으로 시각화하기

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>fig = plt.figure(figsize = (8, 13)) sm.graphics.plot_partregress_grid(regression_result, fig = fig) plt.show()</pre>

결과값1



비고



12. 앙상블 알고리즘

Random Forest

- 특성 중요도 분석 - `pip install mglearn`

라이브러리 импорт

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
```

데이터 로드 및 분할

```
iris = load_iris()

# 훈련/테스트 세트로 나누기
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, random_state=0)
```

...

1. `iris = load_iris()`: Iris 데이터셋을 로드합니다.
2. `train_test_split`: 데이터를 훈련 세트와 테스트 세트로 나눕니다. `random_state=0`은 데이터 분할의 재현성을 보장합니다.

12. 앙상블 알고리즘

Random Forest

모델 학습 및 평가

```
forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)

print("훈련 세트 정확도 : {:.3f}".format(forest.score(X_train, y_train)))
print("테스트 세트 정확도 : {:.3f}".format(forest.score(X_test, y_test)))
```

...

1. RandomForestClassifier(n_estimators=100, random_state=0): 100개의 트리로 구성된 랜덤 포레스트 분류 모델을 생성합니다.
2. forest.fit(X_train, y_train): 훈련 데이터를 사용하여 모델을 학습시킵니다.
3. forest.score(X_train, y_train): 훈련 세트의 정확도를 계산하고 출력합니다.
4. forest.score(X_test, y_test): 테스트 세트의 정확도를 계산하고 출력합니다.

특성 중요도 출력

특성 중요도

```
print("특성 중요도 : \n{}".format(forest.feature_importances_))
```

...

1. forest.feature_importances_: 학습된 모델에서 각 특성의 중요도를 반환합니다.
2. 중요도를 출력하여 각 특성이 모델 예측에 얼마나 기여했는지 확인할 수 있습니다.

Random Forest

특성 중요도 시각화

특성 중요도 시각화 하기

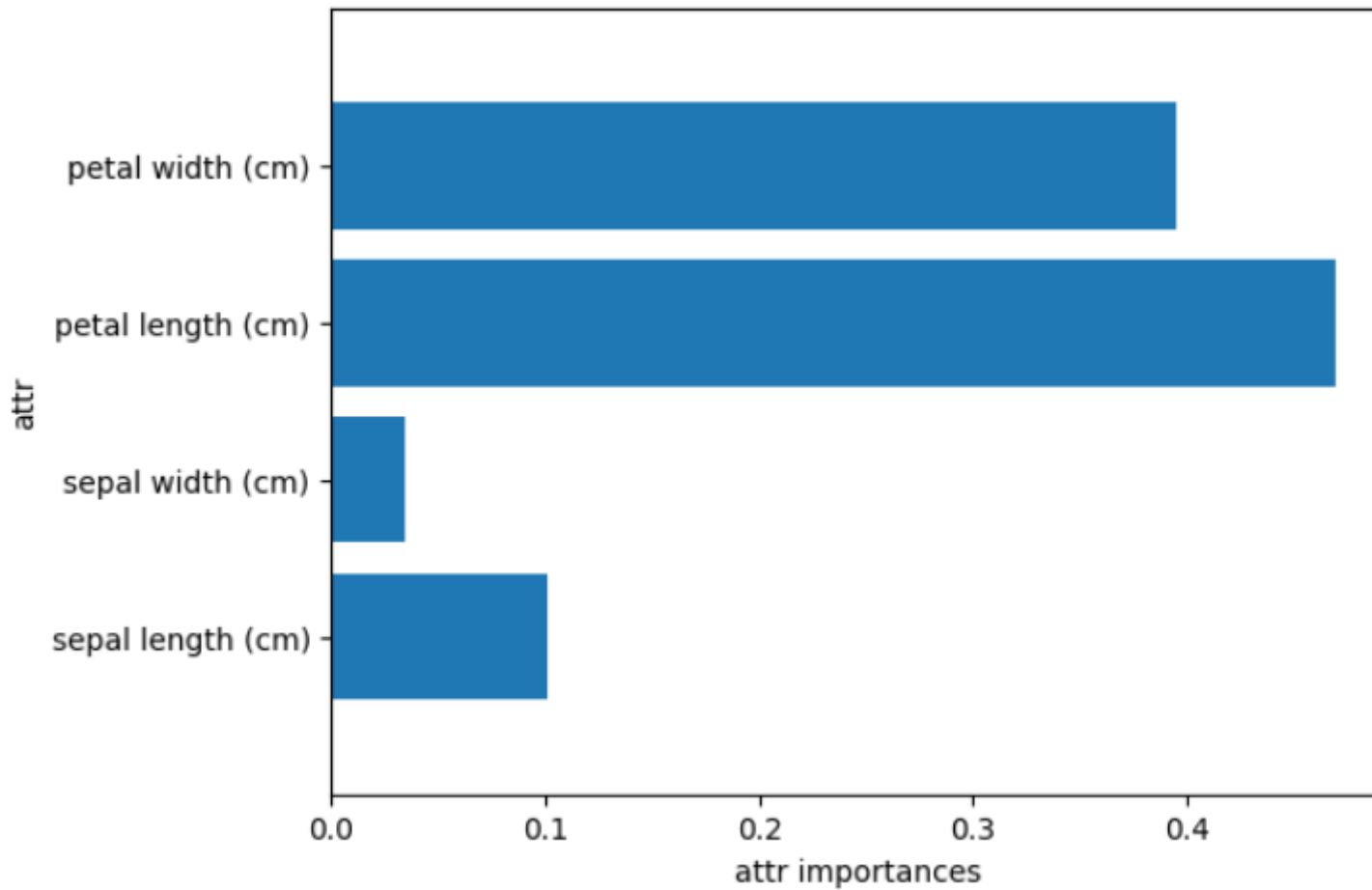
```
def plot_feature_importances_iris(model):  
    n_features = iris.data.shape[1]  
    plt.barh(range(n_features), model.feature_importances_, align='center')  
    plt.yticks(np.arange(n_features), iris.feature_names)  
    plt.xlabel("attr importances")  
    plt.ylabel("attr")  
    plt.ylim(-1, n_features)  
  
plot_feature_importances_iris(forest)  
plt.show()
```

...

1. `plot_feature_importances_iris(model)`: 특성 중요도를 시각화하는 함수입니다.
2. `n_features = iris.data.shape[1]`: Iris 데이터셋의 특성 수를 구합니다.
3. `plt.barh(range(n_features), model.feature_importances_, align='center')`: 수평 막대 그래프로 특성 중요도를 시각화합니다.
4. `plt.yticks(np.arange(n_features), iris.feature_names)`: y축의 눈금을 특성 이름으로 설정합니다.
5. `plt.xlabel("attr importances")`: x축 레이블을 설정합니다.
6. `plt.ylabel("attr")`: y축 레이블을 설정합니다.
7. `plt.ylim(-1, n_features)`: y축의 범위를 설정합니다.
8. `plot_feature_importances_iris(forest)`: 학습된 랜덤 포레스트 모델의 특성 중요도를 시각화합니다.
9. `plt.show()`: 그래프를 화면에 표시합니다.

12. 앙상블 알고리즘

Random Forest



Random Forest 성능

- RandomForestClassifier를 사용하여 Iris 데이터셋을 학습하고 여러 가지 설정으로 모델을 평가합니다.

라이브러리 импорт 및 데이터 로드

```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd

# Iris 데이터셋 로드
iris = load_iris()
```

Random Forest 성능

데이터 분할

훈련 데이터 설정

```
x_train = iris.data[:-30]  
y_train = iris.target[:-30]
```

테스트 데이터 설정

```
x_test = iris.data[-30:] # 테스트 데이터의 특성 데이터  
y_test = iris.target[-30:] # 테스트 데이터의 타겟 데이터
```

```
print(y_train)  
print(y_test)
```

...

1. iris.data[:-30]와 iris.target[:-30]: 마지막 30개 샘플을 제외한 데이터를 훈련 데이터로 설정합니다.
2. iris.data[-30:]와 iris.target[-30:]: 마지막 30개 샘플을 테스트 데이터로 설정합니다.
3. print(y_train), print(y_test): 훈련 데이터와 테스트 데이터의 타겟 값을 출력합니다.

12. 앙상블 알고리즘

Random Forest 성능

모델 학습 및 예측

```
# RandomForestClassifier 라이브러리 import
from sklearn.ensemble import RandomForestClassifier
```

```
# tree의 개수로 Random Forest 분류 모듈 생성
rfc = RandomForestClassifier(n_estimators=10)
rfc
```

```
# 모델 학습
rfc.fit(x_train, y_train)
```

```
# 테스트 데이터로 타겟 데이터를 예측
prediction = rfc.predict(x_test)
```

```
# 예측 결과와 실제 테스트 데이터의 타겟을 비교
print(prediction == y_test)
```

```
# Random forest 정확도 측정
print(rfc.score(x_test, y_test))
```

```
...
```

1. RandomForestClassifier(n_estimators=10): 10개의 트리로 구성된 랜덤 포레스트 분류 모델을 생성합니다.
2. rfc.fit(x_train, y_train): 훈련 데이터를 사용하여 모델을 학습시킵니다.
3. rfc.predict(x_test): 테스트 데이터를 사용하여 타겟 데이터를 예측합니다.
4. print(prediction == y_test): 예측 결과와 실제 테스트 데이터의 타겟을 비교하여 일치 여부를 출력합니다.
5. print(rfc.score(x_test, y_test)): 모델의 정확도를 출력합니다.

12. 앙상블 알고리즘

Random Forest 성능

평가

```
from sklearn.metrics import classification_report
```

```
print("Accuracy is: ", accuracy_score(prediction, y_test))  
print("=====  
print(classification_report(prediction, y_test))
```

```
...
```

1. `accuracy_score(prediction, y_test)`: 예측 결과와 실제 결과를 비교하여 정확도를 출력합니다.
2. `classification_report(prediction, y_test)`: 예측 결과에 대한 정밀도, 재현율, F1 점수를 포함한 상세한 분류 보고서를 출력합니다.

12. 앙상블 알고리즘

Random Forest 성능

다른 모델 설정

다른 설정으로 모델 학습 및 평가

```
clf = RandomForestClassifier(n_estimators=10)
```

```
clf.fit(x_train, y_train)
```

```
prediction_1 = rfc.predict(x_test)
```

```
print("Accuracy is: ", accuracy_score(prediction_1, y_test))
```

```
print("=====")
```

```
print(classification_report(prediction_1, y_test))
```

...

1. `RandomForestClassifier(n_estimators=10)`: 10개의 트리로 구성된 랜덤 포레스트 분류 모델을 생성합니다.
2. `clf.fit(x_train, y_train)`: 훈련 데이터를 사용하여 모델을 학습시킵니다.
3. `prediction_1 = clf.predict(x_test)`: 테스트 데이터를 사용하여 타겟 데이터를 예측합니다.
4. `accuracy_score(prediction_1, y_test)`, `classification_report(prediction_1, y_test)`: 예측 결과와 평가 결과를 출력합니다.

12. 앙상블 알고리즘

Random Forest 성능

모델 설정 변경 및 평가

다른 설정으로 모델 초기화 및 학습

```
clf_2 = RandomForestClassifier(n_estimators=200, max_features=4, oob_score=True)
clf_2.fit(x_train, y_train)
prediction_2 = clf_2.predict(x_test)
```

```
print(prediction_2 == y_test)
print("Accuracy is: ", accuracy_score(prediction_2, y_test))
print("=====")
print(classification_report(prediction_2, y_test))
```

...

1. RandomForestClassifier(n_estimators=200, max_features=4, oob_score=True): 200개의 트리와 4개의 최대 특성을 사용하는 랜덤 포레스트 모델을 생성하며, OOB(out-of-bag) 샘플을 사용하여 모델을 평가합니다.
2. clf_2.fit(x_train, y_train): 훈련 데이터를 사용하여 모델을 학습시킵니다.
3. prediction_2 = clf_2.predict(x_test): 테스트 데이터를 사용하여 타겟 데이터를 예측합니다.
4. 예측 결과와 평가 결과를 출력합니다.

특성 중요도 출력

```
for feature, imp in zip(iris.feature_names, clf_2.feature_importances_):
    print(feature, imp)
```

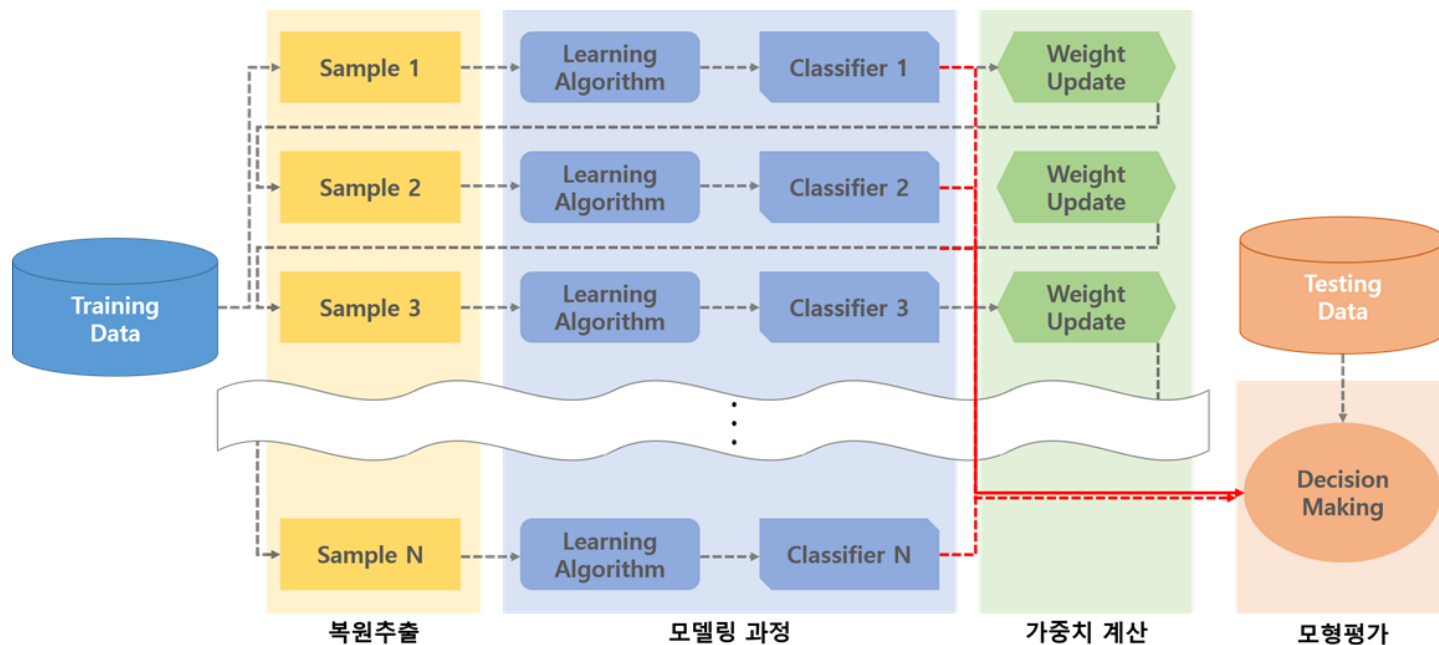
...

1. clf_2.feature_importances_: 학습된 모델에서 각 특성의 중요도를 반환합니다.
2. 각 특성과 그 중요도를 출력하여 어떤 특성이 모델에 가장 큰 영향을 미치는지 확인합니다.

12. 앙상블 알고리즘

Boosting

- 기존 학습 데이터에서 random sampling을 하고 1번 weak learner로 학습시킨다. 그 결과로 생긴 에러를 반영해 그 다음 데이터 샘플링과 2번 weak learner를 잡고 학습을 반복합니다. 이 과정을 N번 하면 iteration N번 돌린 부스팅 모델이 되는 것이다. 부스팅 계열 모델은 AdaBoost, Gradient Boost(GBM), XGBoost, LightGBM, CatBoost 등이 있습니다.



12. 앙상블 알고리즘

Boosting

- GradientBoostingClassifier를 사용하여 Iris 데이터셋에 대해 모델을 학습시키고, 학습된 모델의 특성 중요도를 출력하는 간단한 예제입니다.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import datasets

GradientBoostingClassifier: 그라디언트 부스팅 분류 모델을 사용하기 위한 클래스입니다.
datasets: 다양한 데이터셋을 제공하는 모듈로, 여기서는 Iris 데이터셋을 사용합니다.

# 데이터 로드 - 아이리스
iris = datasets.load_iris()

# 특성 초기화
data = iris.data
target = iris.target
```

12. 앙상블 알고리즘

Boosting

모델 생성 및 학습

부스팅 관련 분류기 객체 생성

```
gradientboost = GradientBoostingClassifier(random_state=0)
```

훈련

```
rs_gb = gradientboost.fit(data, target)
```

...

1. `GradientBoostingClassifier(random_state=0)`: 그라디언트 부스팅 분류기 객체를 생성합니다. `random_state=0`은 난수 생성을 위한 시드를 설정하여 결과의 재현성을 보장합니다.
2. `gradientboost.fit(data, target)`: 전체 Iris 데이터를 사용하여 모델을 학습시킵니다. `data`는 특성 값들이고, `target`은 각 샘플의 클래스 레이블입니다.
3. `rs_gb`: 학습된 모델 객체를 반환합니다.

특성 중요도 출력

```
rs_gb.feature_importances_
```

...

1. `rs_gb.feature_importances_`: 학습된 모델에서 각 특성의 중요도를 나타내는 속성입니다. 각 특성 중요도는 모델이 예측을 수행할 때 해당 특성이 얼마나 중요한지를 나타냅니다.
2. 이 속성은 특성 중요도를 포함하는 배열을 반환합니다. 예를 들어, Iris 데이터셋의 네 개 특성에 대해 네 개의 값이 포함된 배열이 반환됩니다. 각 값은 해당 특성의 중요도를 나타냅니다.

Boosting

- Gradient Boost(GBM) 사용해 보기
- GradientBoostingClassifier를 사용하여 Iris 데이터셋을 학습하고, 모델의 성능을 평가하며, 특성 중요도를 시각화하는 과정입니다.

라이브러리 임포트 및 데이터 로드

```
# 라이브러리 임포트
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import datasets

# 데이터 로드 - 아이리스
iris = datasets.load_iris()
```

12. 앙상블 알고리즘

Boosting

특성 초기화 및 모델 학습

```
# 특성 초기화
data = iris.data
target = iris.target

# 부스팅 관련 분류기 객체 생성
gradientboost = GradientBoostingClassifier(random_state=0)

# 훈련
rs_gb = gradientboost.fit(data, target)

# 특성 중요도 확인
rs_gb.feature_importances_
```

```

1. iris.data와 iris.target: Iris 데이터셋의 특성과 타겟을 각각 변수에 저장합니다.
2. GradientBoostingClassifier(random\_state=0): 그라디언트 부스팅 분류기 객체를 생성합니다.
3. gradientboost.fit(data, target): 전체 Iris 데이터를 사용하여 모델을 학습시킵니다.
4. rs\_gb.feature\_importances\_: 학습된 모델에서 각 특성의 중요도를 반환합니다.

# 12. 앙상블 알고리즘

## Boosting

훈련/테스트 데이터 분할 및 기본 모델 학습 및 평가

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np

iris = load_iris()

훈련/테스트 세트로 나누기
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, random_state=0)

gbrt = GradientBoostingClassifier(random_state=0)

gbrt.fit(X_train, y_train)

print("훈련 세트 정확도 : {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도 : {:.3f}".format(gbrt.score(X_test, y_test)))
```

...

1. train\_test\_split: 데이터를 훈련 세트와 테스트 세트로 나눕니다.
2. GradientBoostingClassifier(random\_state=0): 그라디언트 부스팅 분류 모델을 생성합니다.
3. gbrt.fit(X\_train, y\_train): 훈련 데이터를 사용하여 모델을 학습시킵니다.
4. gbrt.score(X\_train, y\_train), gbrt.score(X\_test, y\_test): 훈련 세트와 테스트 세트의 정확도를 출력합니다.



# 12. 앙상블 알고리즘

## Boosting

### 과대적합 문제 해결을 위한 사전 가지치기

```
사전 가지치기 설정 (max_depth=1)
gbrt = GradientBoostingClassifier(random_state=0, max_depth=1)

gbrt.fit(X_train, y_train)

print("훈련 세트 정확도 : {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도 : {:.3f}".format(gbrt.score(X_test, y_test)))
```

- ...
1. max\_depth=1: 각 트리의 최대 깊이를 1로 제한하여 과대적합을 막습니다.
  2. 모델 학습 및 평가 결과를 출력합니다.

### 과대적합 문제 해결을 위한 학습률 조정

```
학습률 낮추기 (learning_rate=0.01)
gbrt = GradientBoostingClassifier(random_state=0, learning_rate=0.01)

gbrt.fit(X_train, y_train)

print("훈련 세트 정확도 : {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도 : {:.3f}".format(gbrt.score(X_test, y_test)))
```

- ...
1. learning\_rate=0.01: 학습률을 낮추어 과대적합을 방지합니다.
  2. 모델 학습 및 평가 결과를 출력합니다.

## Boosting

### 특성 중요도 시각화

# 특성 중요도 시각화 함수 정의

```
def plot_feature_importances_iris(model):
 n_features = iris.data.shape[1]
 plt.barh(range(n_features), model.feature_importances_, align='center')
 plt.yticks(np.arange(n_features), iris.feature_names)
 plt.xlabel("attr importances")
 plt.ylabel("attr")
 plt.ylim(-1, n_features)
```

# 특성 중요도 시각화

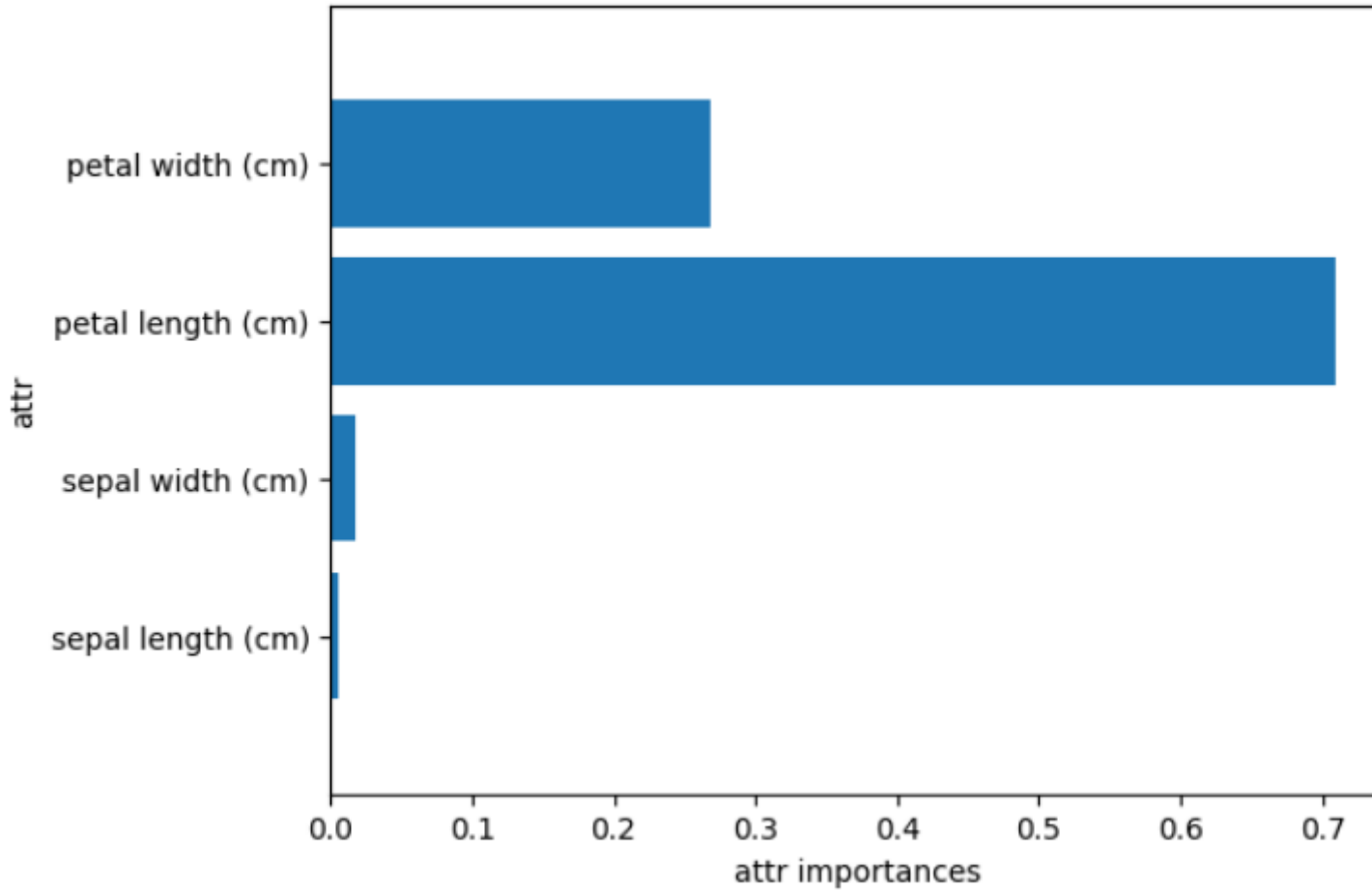
```
plot_feature_importances_iris(gbrt)
plt.show()
```

...

1. `plot_feature_importances_iris(model)`: 특성 중요도를 시각화하는 함수입니다.
2. `n_features = iris.data.shape[1]`: Iris 데이터셋의 특성 수를 구합니다.
3. `plt.barh(range(n_features), model.feature_importances_, align='center')`: 수평 막대 그래프로 특성 중요도를 시각화합니다.
4. `plt.yticks(np.arange(n_features), iris.feature_names)`: y축의 눈금을 특성 이름으로 설정합니다.
5. `plt.xlabel("attr importances")`: x축 레이블을 설정합니다.
6. `plt.ylabel("attr")`: y축 레이블을 설정합니다.
7. `plt.ylim(-1, n_features)`: y축의 범위를 설정합니다.
8. `plot_feature_importances_iris(gbrt)`: 학습된 그라디언트 부스팅 모델의 특성 중요도를 시각화합니다.
9. `plt.show()`: 그래프를 화면에 표시합니다.

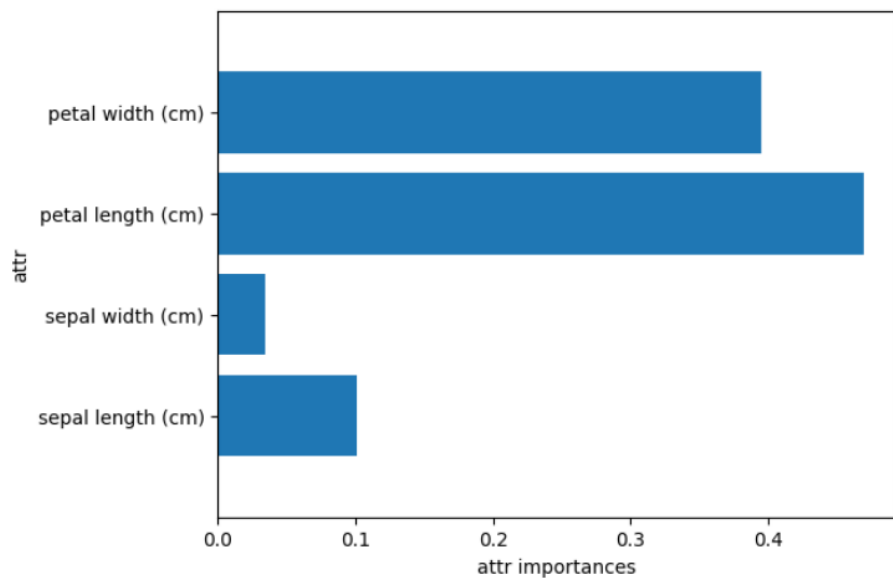
## 12. 앙상블 알고리즘

### Boosting

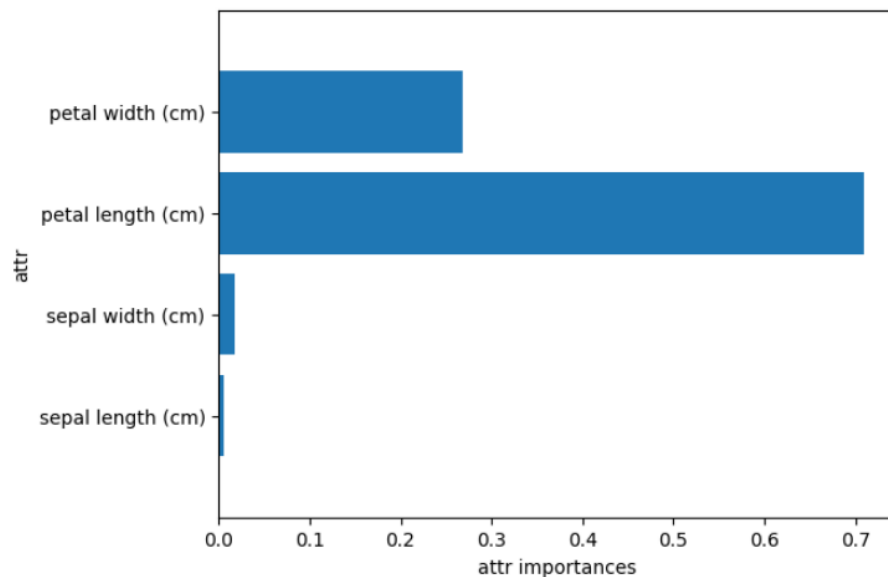


## Boosting

- Random Forest와 Gradient Boost(GBM) 특성 중요도 비교



Random Forest



Gradient Boost(GBM)