2021 혁신성장 청년인재 집중양성 추경사업

Python 기반 AI를 활용한 빅데이터 분석가 양성 과정

머신러닝을 위한 Python 라이브러리

2021. 12. 01. ~ 2021. 12. 16.

메디치교육센터

CONTENTS

일차	주제	비고
1일차	머신러닝의 이해	
2일차	데이터 전처리, KNN 알고리즘의 이해	미니프로젝트 (코로나/경찰청/기상청)
3일차	회귀모델(KNN회귀, 선형)	
4일차	선형회귀(다중회귀, 리지, 라쏘)	
5일차	로지스틱 회귀(예제: 보스턴집값)	미니프로젝트 (심장질환/ 암진단/버스정류장 묶음과 파출소설치)
6일차	의사결정트리1	
7일차	의사결정트리2(예제: 클릭예측)	
8일차	나이브베이즈(스팸메일분류기)	
9일차	군집화	
10일차	앙상블	
11일차	예측 분석 관련 알고리즘	
12일차	추천 관련 알고리즘	

지난 시간 보충내용

1. 데이터 분석절차
2. 각 절차 별 챙겨야 할 내용들
3. 잊지 말고 확인할 사항

1. 데이터 분석 절차

1 데이터 준비

◆ 머신러닝에서는 데이터를 통해 모델이 도출되기 때문에 데이터의 중요성이 더 부곽된다.

◆ 분석을 위한 데이터를 확보한 후 체크할 사항

> 결측치 양과 분포

> 이상치 유무와 이상치의 분포

> 변수들의 스케일 상태 체크

> 분석 알고리즘 선정 대상 체크

2 데이터 전처리

◆ 머신러닝에서 데이터가 중요하기 때문에 분석결과를 왜곡시킬 데이터에 대한 처리나 고려가 반드시 필요하다.

◆ 전처리 과정에서 반드시 체크되고 처리해야 할 내용

> 결측값 처리

> 변수 스케일링

> 다중공선성 체크 및 처리

> 데이터 셋 나누기

1. 데이터 분석 절차

3 모델 만들기

- 분석 목적에 맞는 알고리즘을 선정한다.
- 알고리즘과 데이터 특성을 고려하여 모델 만들기 수행
  - > 알고리즘을 적용한다.
  - > training data를 사용하여 학습수행
  - > 검증데이터를 가지고 분석과정 및 성능을 체크
  - > 성능향상을 위해 데이터 전처리와 하이퍼 파라미터 적용으로 모델 최적화 수행
- 적절한 모델 평가방법을 적용하여 모델 평가 수행

4 분석 결과의 해석

- 분석 결과에 대한 해석에는 다양한 측면에서 각기 다른 접근법들이 존재한다. 본 과정에서는 다분히 모델 평가에서 나온 결과치를 가지고 분석 모형 적 해석을 한다.
- 분석 모형적 해석: 분석 후 적합한 모형을 도출하는 데 사용되는 지표는 각 모델마다 평가되는 기준이 다르다.
  - > 회귀모델
  - > 분류모델
  - > 군집모델
  - > 연관성모델

1 결측치 양과 분포 확인

- ◆ 결측치 시각화 (seaborn, missingno 라이브러리 사용)
- ◆ 결측치를 시각화 하여 양과 분포를 알아보는 것은 중요하다.

  대체로 사용되는 시각화 라이브러리 중 seaborn을 이용하는 1가지 방법과 missingno를 이용하는 4가지 방법

  > seaborn 결측치 시각화

  > missingno 결측치 시각화

  - ✓ Matrix
  - ✓ Barchart
  - ✓ HeatMap
  - ✓ Dendrogram

[결측치 시각화 및 처리 예제]

```
In [7]: #Ozone량 예측을 위한 데이터 시각화

        #사용할 라이브러리들을 import
        from sklearn.datasets import load_boston
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from scipy import stats

        #데이터 세트 로드
        df = pd.read_csv ("C:/anal_data/ozone.csv")

        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Ozone    116 non-null    float64
 1   Solar.R  146 non-null    float64
 2   Wind     153 non-null    float64
 3   Temp     153 non-null    int64
 4   Month    153 non-null    int64
 5   Day      153 non-null    int64
dtypes: float64(3), int64(3)
memory usage: 7.3 KB
```

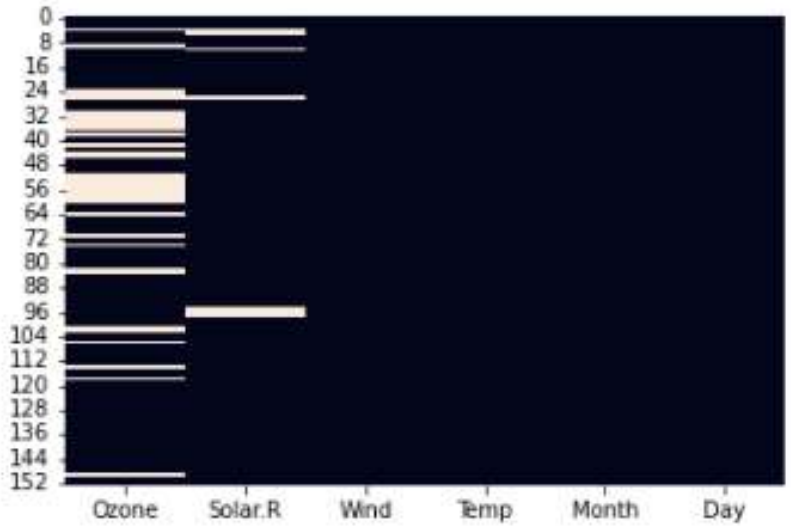
3. 결측치 시각화 방법 사용예

1 seaborn 결측치 시각화

In [9]:
```
# 1. seaborn 결측치 시각화

%matplotlib inline
sns.heatmap(df.isnull(),cbar=False)
```

Out[9]: <AxesSubplot:>

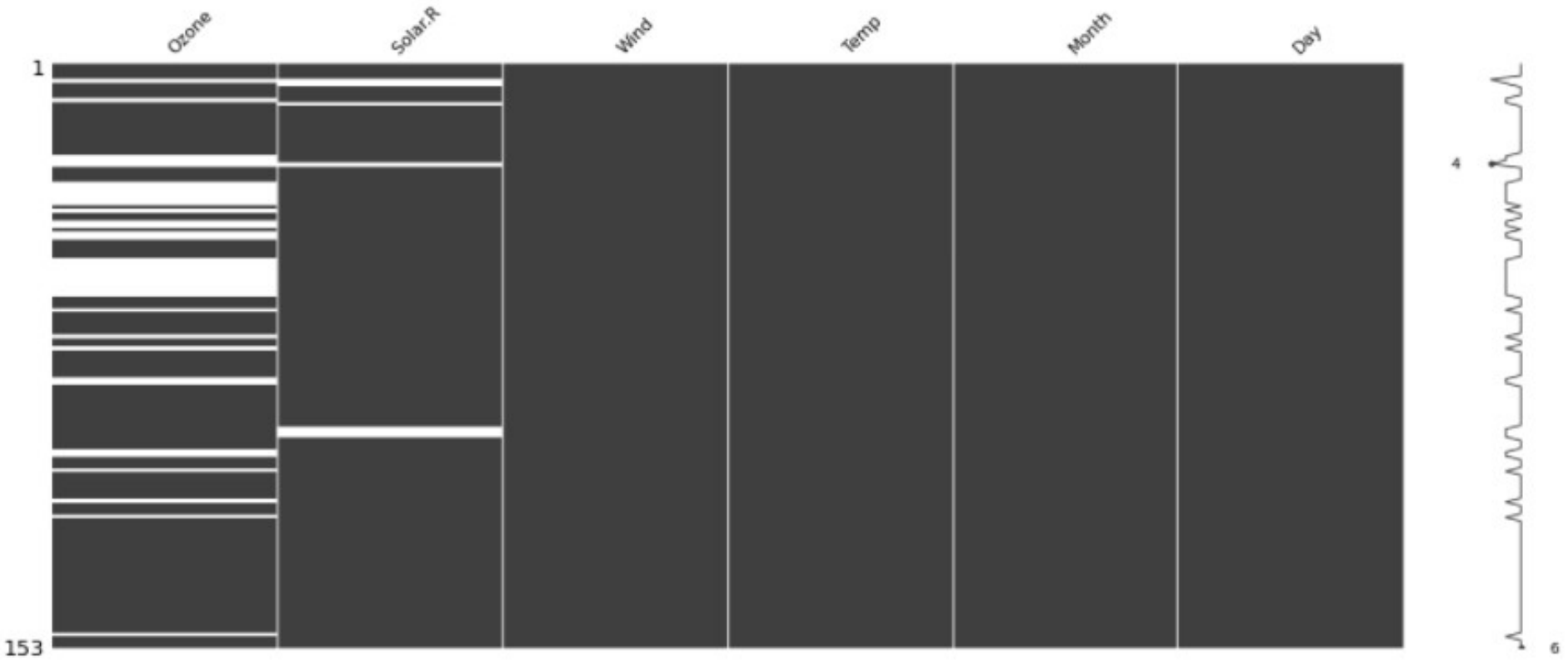


3. 결측치 시각화 방법 사용예

2 Missingno결측치 시각화 - Matrix

In [10]:
```
#2. Missingno결측치 시각화 - Matrix
import missingno as msno
%matplotlib inline
msno.matrix(df)
```

Out[10]: <AxesSubplot:>

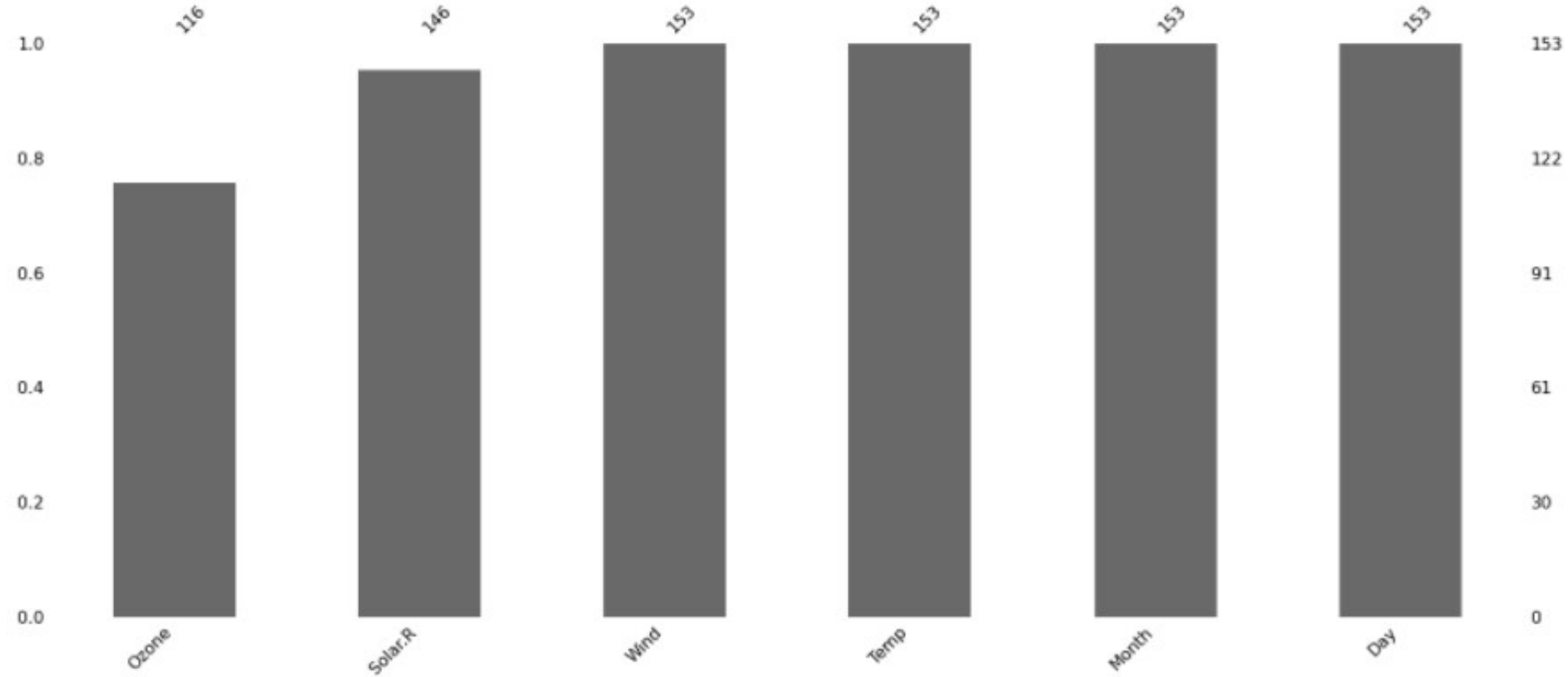


3. 결측치 시각화 방법 사용예

3 Missingno결측치 시각화 - Barchart

```
In [11]: #3. Missingno결측치 시각화 - Barchart
         msno.bar(df)
```

Out[11]: <AxesSubplot:>



3. 결측치 시각화 방법 사용예

4 Missingno결측치 시각화 - Heatmap

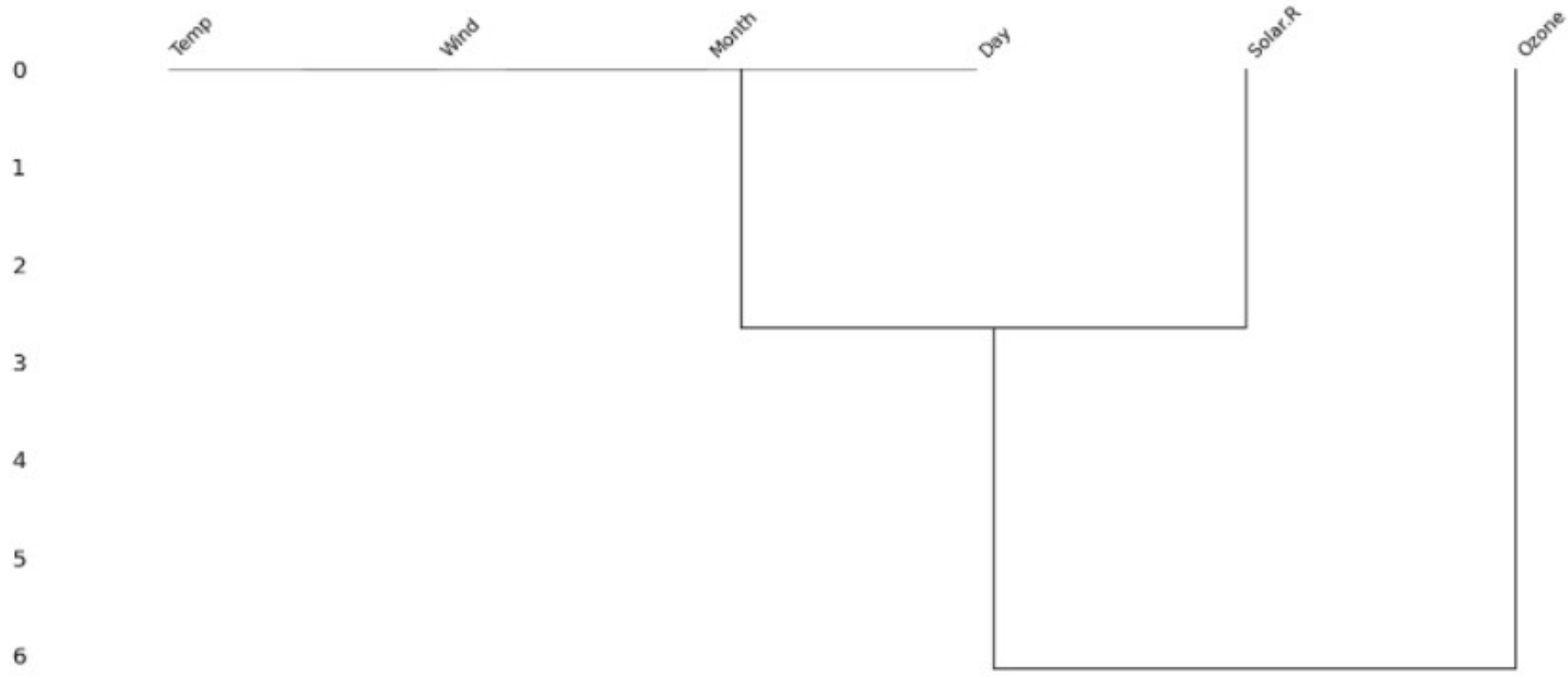
```
In [12]: #4. Missingno결측치 시각화 - Heatmap
         msno.heatmap(df)
Out[12]: <AxesSubplot:>
```

3. 결측치 시각화 방법 사용예

5 Missingno결측치 시각화 - Dendrogram

```
In [13]: #4. Missingno결측치 시각화 - Dendrogram
         msno.dendrogram(df)

Out[13]: <AxesSubplot:>
```



4. 결측치 처리

1 삭제: dropna( )

◆ 결측치를 가진 행을 모두 삭제하는 방법

```
In [14]: df = df.dropna()
```

◆ 특정 행에 결측치가 있을 경우 삭제하는 방법

```
In [15]: df = df.dropna(subset=["column_name"])
```

2 대체: fillna( )

◆ 결측치를 다른 값으로 대체하는 방안

```
In [16]: df['column_name'] = df['column_name'].fillna(value_to_fill_in)
```

◆ 대체할 수 있는 값

- ➢ 데이터가 정규분포일 때: 평균
- ➢ 데이터가 비정규분포일 때: 중위수
- ➢ 데이터가 범주형 변수 일 때: 최빈수

5. 스케일링 처리-sklearn에서 제공하는.

◆ 스케일 조정이 필요한 경우

- 분석때 변수들의 스케일이 너무 다를 경우
- 신경망 학습때, 데이터 셋의 값이 들쑥날쑥하거나,

  매우 큰 경우에는 cost의 값이 발산하여 정상적인 학습이 이루어지지 않아 문제가 생길 수 있다.

  ➔ 이런 경우에 스케일 조정으로 해결 할 수 있다.

◆ 사이킷런을 베이스로 하는

- SVM (서포트 벡터 머신: Support Vector Machine)
- 선형회귀(LinearRegression)
- 로지스틱 회귀(Logistic Regression)

이런 알고리즘들은 데이터가 가우시안 분포를 따르고 있다는 가정하에 만들어졌기 때문에
사전에 표준화를 적용하는 것은 예측 성능에 중요한 요소로 작용할 수 있다.
따라서, 사전 표준화를 반드시 고려해야 한다.

◆ 스케일 사용시 주의사항

- Scaler는 fit과 transform메서드를 가지고 있다.
- Fit 메서드는 훈련 데이터에만 적용해, 훈련 데이터의 분포를 먼저 학습
- 그 이후, transform메서드를 훈련 데이터와 테스트 데이터에 적용해 스케일 조정을 한다.
- 정리: 훈련데이터에는 fit()과 transform()을 적용하고

  테스트 데이터에는 transform() 메서드를 적용해야 한다.

  (cf. fit()과 transform()을 결합한 단축 메서드 fit_transform()를 사용하기도 함)

5. 스케일링 처리-sklearn에서 제공하는.

1 StandardScaler

◆ 필요한 라이브러리

:from sklearn.preprocessing import StandardScaler

◆ 사용법 및 특징

- ➢ 특성들의 평균을 0, 분산을 1로 스케일링 하는 것
- ➢ 최솟값과 최댓값의 크기를 제한하지 않음. 이상치에 민감. 회귀보다는 분류 알고리즘에 더 많이 사용됨

◆ 사용 예

```
from sklearn.model_selection import train_test_split
#lr: 알고리즘이 적용된 모델명
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

# 데이터 트레이닝: 80, 테스트: 20
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=0.2, random_state=42)

#데이터 스케일링 조절: 정규화
from sklearn.preprocessing import StandardScaler

#StandardScaler 객체 생성
scaler = StandardScaler()

#StandardScaler로 훈련데이터(독립변수만) 훈련
scaler.fit(Xtrain)

#훈련데이터 스케일링 적용(독립변수만 적용)
train_scaled = scaler.transform(Xtrain)

#테스트 데이터 스케일링(독립변수만 적용)
test_scaled = scaler.transform(Xtest)

# train 데이터를 사용하여 훈련()
lr.fit(train_scaled, ytrain)

#훈련 데이터의 정확도
print(lr.score(train_scaled, ytrain))

#테스트 데이터의 정확도
print(lr.score(test_scaled, ytest))
print(lr.intercept_,lr.coef_)
```

5. 스케일링 처리-sklearn에서 제공하는.

2 MinMaxScaler

- :from sklearn.preprocessing import StandardScaler사용법 및 특징
- 사용예

```
from sklearn.model_selection import train_test_split

# 데이터 트레이닝: 80, 테스트: 20
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=0.2, random_state=42)

#데이터 스케일링 조절: 정규화
from sklearn.preprocessing import MinMaxScaler

#MinMaxScaler 객체 생성
#MinMaxScaler로 데이터 셋트 변환, fit()과  transform() 호출
MinMax = MinMaxScaler()

MinMax.fit(Xtrain)
Min_train_scaled = MinMax.transform(Xtrain)
Min_test_scaled = MinMax.transform(Xtest)

# train 데이터를 사용하여 훈련
lr.fit(Min_train_scaled, ytrain)

#훈련 데이터의 정확도
print(lr.score(Min_train_scaled, ytrain))

#테스트 데이터의 정확도
print(lr.score(Min_test_scaled, ytest))
#테스트 데이터의 정확도
print(lr.score(test_scaled, ytest))
print(lr.intercept_,lr.coef_)
```

6. 데이터 모델만들기(선형회귀)

1 LinearRegresssion

2 OLS

◆ OLS모델을 통해 살펴볼 것들

- OLS Model은 선형 회귀분석에 있어서 각각의 독립변수 xi가 종속변수 y에 영향이 있는지 단적으로 확인 할 수 있다.
  (귀무가설과 대립가설 - t분포와 p value)
- 다른 독립변수들을 배제하고 특정 변수에 있어서 독립변수에 영향을 주는지 확인할 수 있다.(Regress out)
- 회귀 방정식에서 각 변수의 계수값을 알 수 있다.
- 해당 방정식으로 데이터들을 얼마나 설명할 수 있는지
- 명목변수의 encoding

1. LinearRegression

[1번]데이터 스케일링 없이 수행

[분석 조건]

- 알고리즘: LinearRegression
- 데이터 셋: 153 rows X 6 columns
- 결측량: 4.79%
- 데이터 셋: 훈련 80%, 테스트: 20%
- 데이터 셋: 스케일링 변환없이 진행
- 모델 평가방법: score 출력(R2값) / MSE값/ RMSE값 /기울기값과 절편값

33]:
```
from sklearn.model_selection import train_test_split
#lr: 알고리즘이 적용된 모델명
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

# 데이터 트레이닝: 80, 테스트: 20
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=0.2)

# train 데이터를 사용하여 훈련
lr.fit(Xtrain,ytrain)

#훈련 데이터의 정확도
print(lr.score(Xtrain, ytrain))

#테스트 데이터의 정확도
print(lr.score(Xtest, ytest))

#기울기와 절편 출력
print(lr.intercept_,lr.coef_)
```

```
0.48626716208026177
0.4093005381897121
-24.982259182522803 [ 0.04643547 -3.17940702  1.16877281]
```

1. LinearRegression

[2번]데이터 standardscaler 사용

[분석 조건]

- 알고리즘: LinearRegression
- 데이터 셋: 153 rows X 6 columns
- 결측량: 4.79%
- 데이터 셋: 훈련 80%, 테스트: 20%
- 데이터 셋: 스케일링:StandardScaler
- 모델 평가방법: score 출력(R2값) / MSE값/ RMSE값

In [48]:
```
from sklearn.model_selection import train_test_split
#lr: 알고리즘이 적용된 모델명
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

# 데이터 트레이닝: 80, 테스트: 20
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=0.2, random_state=42)

#데이터 스케일링 조절: 정규화
from sklearn.preprocessing import StandardScaler

#StandardScaler 객체 생성
scaler = StandardScaler()

#StandardScaler로 훈련데이터(독립변수만) 훈련
scaler.fit(Xtrain)

#훈련데이터 스케일링 적용(독립변수만 적용)
train_scaled = scaler.transform(Xtrain)

#테스트 데이터 스케일링(독립변수만 적용)
test_scaled = scaler.transform(Xtest)

# train 데이터를 사용하여 훈련()
lr.fit(train_scaled, ytrain)

#훈련 데이터의 정확도
print(lr.score(train_scaled, ytrain))

#테스트 데이터의 정확도
print(lr.score(test_scaled, ytest))
print(lr.intercept_,lr.coef_)
```

```
0.4735263074403241
0.4989947227530651
42.407786885245905 [ 4.4637745  -9.66293949 11.73824695]
```

1. LinearRegression

[3번]데이터 MinMaxScaler 사용

[분석 조건]

알고리즘: LinearRegression

데이터 셋: 153 rows X 6 columns

결측량: 4.79%

데이터 셋: 훈련 80%, 테스트: 20%

데이터 셋: 스케일링:MinMaxScaler

모델 평가방법: score 출력(R2값) / MSE값/ RMSE값

```
]: from sklearn.model_selection import train_test_split

# 데이터 트레이닝: 80, 테스트: 20
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=0.2, random_state=42)

#데이터 스케일링 조절: 정규화
from sklearn.preprocessing import MinMaxScaler

#MinMaxScaler 객체 생성
#MinMaxScaler로 데이터 셋트 변환, fit()과  transform() 호출
MinMax = MinMaxScaler()

MinMax.fit(Xtrain)
Min_train_scaled = MinMax.transform(Xtrain)
Min_test_scaled = MinMax.transform(Xtest)

# train 데이터를 사용하여 훈련
lr.fit(Min_train_scaled, ytrain)

#훈련 데이터의 정확도
print(lr.score(Min_train_scaled, ytrain))

#테스트 데이터의 정확도
print(lr.score(Min_test_scaled, ytest))
#테스트 데이터의 정확도
print(lr.score(test_scaled, ytest))
print(lr.intercept_,lr.coef_)
```

```
0.4735263074403242
0.4989947227530648
-2.9701305158192786
29.156945458644095 [ 16.85813209 -50.41746166  49.78748736]
```

2. OLS알고리즘

In [45]:
```
#OLS 알고리즘을 사용하기 위해 import
import statsmodels.api as sm
```

In [46]:
```
#전달이 사이킷런 알고리즘들과 다르다.
#알고리즘 OLS에 데이터 전달이 * (y,X) 주의
model = sm.OLS(y, X)
result = model.fit()
result.summary()
```

Out [46]:

OLS Regression Results

Dep. Variable:	Ozone	**R-squared (uncentered):**	0.831
Model:	OLS	**Adj. R-squared (uncentered):**	0.828
Method:	Least Squares	**F-statistic:**	246.7
Date:	Sun, 12 Dec 2021	**Prob (F-statistic):**	9.03e-58
Time:	23:27:24	**Log-Likelihood:**	-682.21
No. Observations:	153	**AIC:**	1370.
Df Residuals:	150	**BIC:**	1380.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Solar.R	0.0600	0.020	2.969	0.003	0.020	0.100
Wind	-3.4514	0.408	-8.459	0.000	-4.258	-2.645
Temp	0.8430	0.071	11.820	0.000	0.702	0.984

Omnibus:	40.837	**Durbin-Watson:**	1.639
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	79.840
Skew:	1.210	**Prob(JB):**	4.60e-18
Kurtosis:	5.581	**Cond. No.**	52.5

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

2. OLS알고리즘 – 분석결과 해석하기

1번 표 영역

분석결과	설명
No.observations	“Number of observations“ 총표본수
Df Residuals:	잔차의 자유도, 전체 표본수에서 측정되는 변수들(종속변수/독립변수)의 개수를 뺀것
Df Model:	독립변수의 개수
R-squared (uncentered):	결정계수. 전체 데이터 중 해당 회귀 모델이 설명할 수 있는 데이터의 비율. 회귀식의 설명력을 나타내는 성능을 나타냄 (1에 가까울 수록 성능이 좋음)
Adj. R-squared (uncentered):	모델에 도움이 되는 데이터에 따라 조정된 결정계수 위 모델은 해당 회귀 모델은 데이터의 **% 를 설명한다. 라고 해석.
F-statistic:	F통계량으로 도출된 회귀식이 적절한지 볼 수 있다. 0에 가까울 수록 적절한 것
Prob (F-statistic):	회귀식이 유의미한지 판단. (0.05이하일 경우 변수 끼리 매우 관련 있다고 판단한다)
AIC:	표본의 개수와 모델의 복잡성을 기반을 모델을 평가하며, 수치가 낮을 수록 좋다
BIC:	AIC와 유사하나 패널티를 부여하여 AIC보다 모델 평가 성능이 더 좋으며, 수치가 낮을 수록 좋다

2. OLS알고리즘 – 분석결과 해석하기

2번 표 영역

분석결과	설명
coef	회귀계수
std err	계수 추정치의 표준오차, 값이 작을 수록 좋다.
t	T-test, 독립변수와 종속변수 사이의 상관관계. 값이 클수록 상관도가 크다
P>|t|	독립변수들의 유의 확률, 0.05보다 작아야 유의미하다. 변수들의 p-value를 살펴보면, INDUS, AGE가 0.05보다 크다. 이런 경우 유의하지 않다.
[0.025 0.975]	회귀계수의 신뢰구간

2. OLS알고리즘 – 분석결과 해석하기

3번 표 영역

분석결과	설명
Omnibus:	디아고스티노 검정(D'Angostino's Test), 비대칭도와 첨도를 결합한 정규성 테스트이며 값이 클수록 정규분포를 따름
Prob(Omnibus):	디아고스티노 검정이 유의한지 판단.(0.05이하일 경우 유의하다고 판단)
Skew(왜도)	평균 주위의 잔차들의 대칭하는지를 보는 것이며, 0에 가까울 수록 대칭이다.
Kurtosis(첨도)	잔차들의 분포 모양이며, 3에 가까울 수록 정규분도이다.(음수이면 평평한 형태, 양수이면 뾰족한 형태)
Durbin-Watson:	더빈왓슨 정규성 검정이며, 잔차의 동립성 여부를 판단, (1.5~2.5 사이일때 잔차는 독립적이라고 판단하며 0이나4에 가까울 수록 잔차들은 자기상관을 가지고 있다고 판단한다)
Jarque-Bera (JB):	자베트라 정규성 검정. 값이 클수록 정규분포의 데이터를 사용했다고 본다
Cond. No.	다중공선성 검정, 독립변수간 상관관계가 있는지 보는 것이며, 10이상이면 다중공선성이 있다고 판단한다.

범주형 변수 처리

- 범주형 변수 앞에 C를 붙여서 처리한다.

```
feature_names = list(boston.feature_names)
feature_names.remove("CHAS")
feature_names = [name for name in feature_names] + ["C(CHAS)"] #범주형 변수는 앞에 c를 붙여서 처리한다.
model2 = sm.OLS.from_formula("MEDV ~ 0 + " + "+".join(feature_names), data=df)
result2 = model2.fit()
print(result2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Sat, 19 Dec 2020   Prob (F-statistic):          6.72e-135
Time:                        18:25:14   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
C(CHAS)[0.0]  36.4595      5.103      7.144      0.000      26.432      46.487
C(CHAS)[1.0]  39.1462      5.153      7.597      0.000      29.023      49.270
CRIM          -0.1080      0.033     -3.287      0.001      -0.173      -0.043
ZN             0.0464      0.014      3.382      0.001       0.019       0.073
INDUS          0.0206      0.061      0.334      0.738      -0.100       0.141
NOX          -17.7666      3.820     -4.651      0.000     -25.272     -10.262
RM             3.8099      0.418      9.116      0.000       2.989       4.631
```

3. OLS변수처리

연속형 변수 스케일링

- 변수에 scale()를 씌워서 처리한다.

```
feature_names = list(boston.feature_names)
feature_names.remove("CHAS")
feature_names = ["scale({})".format(name) for name in feature_names] + ["CHAS"]
model3 = sm.OLS.from_formula("MEDV ~ " + "+".join(feature_names), data=df)
result3 = model3.fit()
print(result3.summary())
```

OLS Regression Results

Dep. Variable:	MEDV	R-squared:	0.741
Model:	OLS	Adj. R-squared:	0.734
Method:	Least Squares	F-statistic:	108.1
Date:	Sat, 19 Dec 2020	Prob (F-statistic):	6.72e-135
Time:	18:25:34	Log-Likelihood:	-1498.8
No. Observations:	506	AIC:	3026.
Df Residuals:	492	BIC:	3085.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	22.3470	0.219	101.943	0.000	21.916	22.778
scale(CRIM)	-0.9281	0.282	-3.287	0.001	-1.483	-0.373
scale(ZN)	1.0816	0.320	3.382	0.001	0.453	1.710
scale(INDUS)	0.1409	0.421	0.334	0.738	-0.687	0.969
scale(NOX)	-2.0567	0.442	-4.651	0.000	-2.926	-1.188
scale(RM)	2.6742	0.293	9.116	0.000	2.098	3.251
scale(AGE)	0.0195	0.371	0.052	0.958	-0.710	0.749
scale(DIS)	-3.1040	0.420	-7.398	0.000	-3.928	-2.280
scale(RAD)	2.6622	0.577	4.613	0.000	1.528	3.796
scale(TAX)	-2.0768	0.633	-3.280	0.001	-3.321	-0.833
scale(PTRATIO)	-2.0606	0.283	-7.283	0.000	-2.617	-1.505
scale(B)	0.8493	0.245	3.467	0.001	0.368	1.331
scale(LSTAT)	-3.7436	0.362	-10.347	0.000	-4.454	-3.033
CHAS	2.6867	0.862	3.118	0.002	0.994	4.380

Omnibus:	178.041	Durbin-Watson:	1.078
Prob(Omnibus):	0.000	Jarque-Bera (JB):	783.126

3. OLS변수처리

3번 표 영역

EX) Cars93데이터에서 EngineSize,RPM,Weight의 독립변수들과 Price라는 종속변수의 다변량회귀분석 해석

```
mtcars = pd.read_csv('Cars93.csv', index_col=0)

model2 = sm.OLS.from_formula("Price ~ EngineSize+RPM+Weight", data=mtcars)
result2 = model2.fit()
print(result2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.561
Model:                            OLS   Adj. R-squared:                  0.547
Method:                 Least Squares   F-statistic:                     37.98
Date:                Sat, 05 Dec 2020   Prob (F-statistic):           6.75e-16
Time:                        20:29:04   Log-Likelihood:                -304.05
No. Observations:                  93   AIC:                             616.1
Df Residuals:                      89   BIC:                             626.2
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -51.7933      9.106     -5.688      0.000     -69.887     -33.699
EngineSize     4.3054      1.325      3.249      0.002       1.673       6.938
RPM            0.0071      0.001      5.208      0.000       0.004       0.010
Weight         0.0073      0.002      3.372      0.001       0.003       0.012
==============================================================================
Omnibus:                       62.441   Durbin-Watson:                   1.406
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              361.880
Skew:                           2.076   Prob(JB):                     2.62e-79
Kurtosis:                      11.726   Cond. No.                     8.27e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.27e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

- 상수항을 포함한 모두 회귀계수의 p-value가 0.05보다 낮아 다 통계적으로 유의함
- 수정된 결정계수가 0.547로 전체의 54.67%를 설명할 수 있다. -> 잘 설명하고 있다고 보기 힘듬
- F-statistic은 37.98, p-value는 6.75e-16로 통계적으로 유의하다.
- 결정계수가 낮기 때문에 모형이 데이터에 가지는 설명력은 낮지만, 회귀계수들이 통계적으로 유일마ㅡ로, 자동차의 가격을 엔진크기와 RPM,무게로 추정할 수 있다.

Python 기반 AI를 활용한 빅데이터 분석가 양성 과정

Q&A / 감사합니다.

메디치교육센터
서울특별시 금천구 가산디지털1로 168(우림라이온스밸리 A동 304 ~ 307호)
Tel 02-861-8568 · **Fax** 070-7525-8538