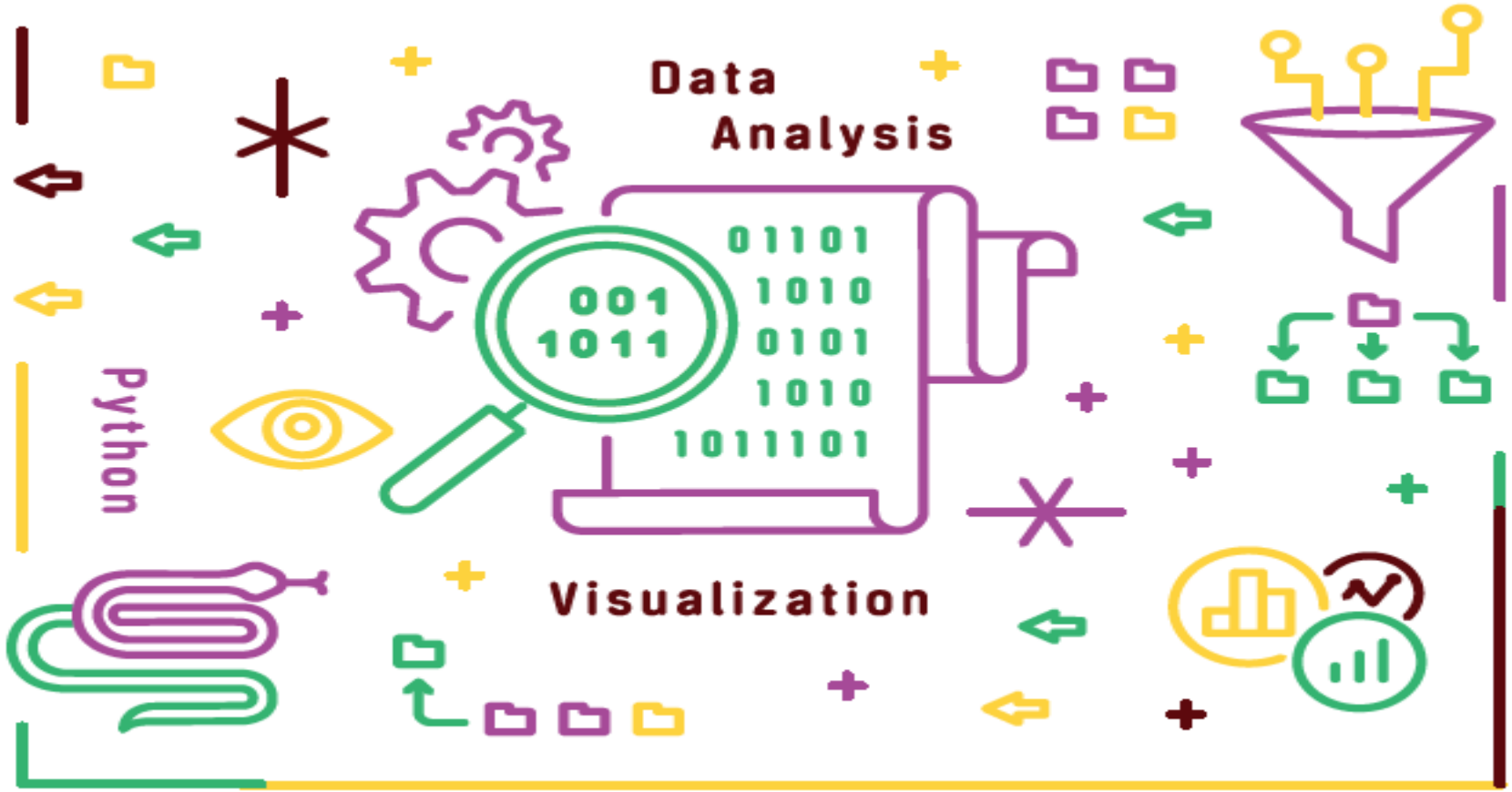

Data
Analysis
001
1011
01101
1010
0101
1010
1011101
Python
Visualization

<6장> 시계열 데이터 분석

학습 목표

- **시계열 데이터의 특징과 전처리 방법을 익히고 이를 바탕으로 실제 문제에서 결과를 예측한다.**

목차

01

시계열 데이터의 이해

1. 시계열 데이터의 이해

시계열 데이터 패턴

- 추세(Trend)
  - 시계열 데이터가 시간에 따라서 증가하거나 감소하는 경향을 보이는 패턴
- 계절성(Seasonality)
  - 시계열 데이터가 일정한 주기를 가지고 반복되는 패턴
- 순환성(Cyclical)
  - Trend와는 달리, 일정한 주기를 가지고 반복되지만 주기의 길이가 일정하지 않은 패턴
- 불규칙성(Irregular)
  - 지진, 홍수, 파업, 코로나와 같은 특수한 요인에 의해 발생하는 불규칙한 패턴

1. 시계열 데이터의 이해

■ 시계열 데이터 분석의 이해

- 분해법
  - 시계열 데이터가 가지는 추세(Trend), 계절성(Seasonality), 주기(Cycle) 및 불규칙성(Irregularity)과 같은 구성요소를 분해하여 분석

- 시간 영역 분석법
  - AR(Auto Regressive Model) 모형
  - MA(Moving Average) 모형
  - ARMA 모형
  - ARIMA(Autoregressive Integrated Moving Average)

1. 시계열 데이터의 이해

시계열 데이터 전처리 방법

- to_datetime()을 이용하여 datetime 형 변환

info()로 데이터 살펴보기

```
import pandas as pd
d=pd.DataFrame({
'date':['2019-01-03','2021-11-22','2023-01-05'],
'name':['J','Y','O']})
d.info()
```

〈실행결과〉

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    3 non-null      object
 1   name    3 non-null      object
dtypes: object(2)
memory usage: 176.0+ bytes
None
```

datetime()으로 변환

```
d['date']=pd.to_datetime(d.date,
format='%Y-%m-%d')
d.info()
```

〈실행결과〉

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    3 non-null      datetime64[ns]
 1   name    3 non-null      object
dtypes: datetime64[ns](1), object(1)
memory usage: 176.0+ bytes
None
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

- set_index()를 이용하여 datetime 형의 컬럼을 인덱스로 설정

인덱스 설정

```
1
2
3
4
```

```
d.set_index(keys='date', inplace=True)
print(d)
```

〈실행결과〉

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3 entries, 2019-01-03 to 2023-01-05
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    3 non-null      object
dtypes: object(1)
memory usage: 48.0+ bytes
None
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

- 결측치 확인
  - isnull(), sum 함수 사용
- 결측치 처리
  - fillna(method='ffill') 함수 사용

```
import numpy as np
#예제 데이터 프레임 생성
d1=pd.DataFrame({'date':['2019-01-03',
'2021-11-22','2021-12-01','2023-01-05'],
'x1':[0.1,2.0,np.nan,1.2]})
d1
```

```
# datetime으로 변환
d1['date']=pd.to_datetime(d1.date,
format='%Y-%m-%d')
d1.info()
```

```
# datetime을 인덱스로 설정
d1.set_index('date', inplace=True)
d1
```

```
# null 값이 있는지 확인
print(d1.isnull().sum())

# 전체 데이터셋 출력
print(d1)
```

```
# 가장 최근 값으로 결측치 채우기
d1=d1.fillna(method='ffill')
print(d1)
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

- 결측치 처리
  - drop( ) 함수 사용
  - interpolate( ) 함수 사용

```
d2=pd.DataFrame({'date':['2019-01-03',
'2021-11-22','2021-12-01','2023-01-05'],
'x1':[0.1,2.0,np.nan,1.2]})
d2
```

```
# 결측치 제거
d2=d2.dropna()
print(d2)
```

```
d3=pd.DataFrame({'date':['2019-01-03',
'2021-11-22','2021-12-01','2023-01-05'],
'x1':[0.1,2.0,np.nan,1.2]})
d3
```

```
d3=d3.interpolate()
print(d3)
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

- 빈도 설정
  - index 속성 이용하여 빈도 설정 확인
  - asfreq() 함수를 사용해서 빈도 설정

index 속성으로 빈도 확인

```
import numpy as np
d=pd.DataFrame({'date':['2019-01-03','2021-11-
22','2021-12-01','2023-01-05'],
'x1':[0.1,2.0,1.6,1.2]})
print(d)
d['date']=pd.to_datetime(d.date, format='%Y-%m-%d')
d.set_index('date', inplace=True)
print(d.index)
```

〈실행결과〉

```
DatetimeIndex(['2019-01-03', '2021-11-22', '2021-12-01', '2023-01-05'],
dtype='datetime64[ns]', name='date', freq=None)
```

빈도 설정

```
print(d)
d2=d.asfreq('Y', method='ffill') #매년 마지막일
print(d2) # 설정한 주기로 생성된 데이터
```

〈실행결과〉

```
            x1
date
2019-01-03  0.1
2021-11-22  2.0
2021-12-01  1.6
2023-01-05  1.2
            x1
date
2019-12-31  0.1
2020-12-31  0.1
2021-12-31  1.6
2022-12-31  1.6
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

▪ 빈도 설정

- asfreq( ) 함수를 사용하여 빈도를 설정할 때 사용하는 옵션

옵션	설명	옵션	설명
B	공휴일과 주말을 제외한 매일(day)	BQS	공휴일과 주말을 제외한 매 분기 시작일
D	매일(day)	A,Y	매년 마지막일
W	매주 일요일	BA,BY	공휴일과 주말을 제외한 매년 마지막일
M	매월 마지막일	AS,YS	매년 시작일
SM	매월 15일	BAS,BYS	공휴일과 주말을 제외한 매년 시작일
CBM	주말을 제외한 매월 마지막일	BH	공휴일과 주말을 제외한 시(hour)
MS	매월 시작일	H	매시간(hour)
SMS	매월 1일과 15일	T,min	매분(minute)
BMS	공휴일과 주말을 제외한 매월 시작일	S	매초(second)
CBMS	주말을 제외한 매월 시작일	L,ms	매밀리초
Q	매분기 마지막일	U,us	매마이크로초
BQ	공휴일과 주말을 제외한 매 분기 마지막일	N	매나노초
QS	매 분기 시작일		

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

▪ 특징량 만들기

- rolling()을 사용
  - 사용법 : rolling(shift횟수).통계함수

인덱스	0	1	2	3	4
rolling(1)	5	4	3	2	7
rolling(2)	Null	5	4	3	2
결과	Null	(4+5)/2(rolling수)	(3+4)/2	(2+3)/2	(7+2)/2

```
# 특징량 만들기
import numpy as np
d=pd.DataFrame({'date':['2019-01-06', '2021-01-13',
'2021-01-20', '2021-01-27','2021-02-03'],
'x1':[5,4,3,2,7]})
d['date']=pd.to_datetime(d.date)
d.set_index(keys=['date'], inplace=True)
print('원본')
print('=='*20)
print(d)
print("=="*20)
tmp=d.rolling(1).mean()
print(tmp)
print()
print('=='*20)
tmp=d.rolling(2).mean()
print(tmp)
```

〈실행결과〉

```
원본
========================================
            x1
date
2021-01-06   5
2021-01-13   4
2021-01-20   3
2021-01-27   2
2021-02-03   7
========================================
            x1
date
2021-01-06  5.0
2021-01-13  4.0
2021-01-20  3.0
2021-01-27  2.0
2021-02-03  7.0
========================================
            x1
date
2021-01-06  NaN
2021-01-13  4.5
2021-01-20  3.5
2021-01-27  2.5
2021-02-03  4.5
```

1. 시계열 데이터의 이해

시계열 데이터 전처리 방법

- 이전 값과 차이 계산
  - diff() 함수 사용

```
print('원본')
print('==*20')
print(d)
y_dff=d.diff()
print(y_dff)
y_dff.columns=['diff']
tmp=pd.concat([d,y_dff], axis=1)
print('=='*20)
print(tmp)
```

```
원본
====================
            x1
date
2019-01-06   5
2021-01-13   4
2021-01-20   3
2021-01-27   2
2021-02-03   7
====================
             x1
date
2019-01-06  NaN
2021-01-13 -1.0
2021-01-20 -1.0
2021-01-27 -1.0
2021-02-03  5.0
====================
            x1  diff
date
2019-01-06   5   NaN
2021-01-13   4  -1.0
2021-01-20   3  -1.0
2021-01-27   2  -1.0
2021-02-03   7   5.0
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

- 지연값 추출 : shift() 사용

지연값 추출

```
d_temp=d
print('원본')
print('=='*10)
print(d_temp)
print('=='*10)
d_temp['shift']=d_temp['x1'].shift(2)
print(d_temp)
print('=='*10)
d_temp=d_temp.fillna(method='bfill')
print(d)
```

```
원본
====================
            x1
date
2019-01-06   5
2021-01-13   4
2021-01-20   3
2021-01-27   2
2021-02-03   7
====================
            x1  shift
date
2019-01-06   5    NaN
2021-01-13   4    NaN
2021-01-20   3    5.0
2021-01-27   2    4.0
2021-02-03   7    3.0
====================
            x1  shift
date
2019-01-06   5    NaN
2021-01-13   4    NaN
2021-01-20   3    5.0
2021-01-27   2    4.0
2021-02-03   7    3.0
```

1. 시계열 데이터의 이해

■ 시계열 데이터 전처리 방법

▪ 원-핫 인코딩

- 범주형 변수를 컴퓨터가 처리할 수 있는 형태로 변환하는 방법 중 하나
- get_dummies() 함수를 사용

원-핫 인코딩

```
1  #예시 데이터프레임 생성
2  df=pd.DataFrame({'color':
3  ['red','blue','green','blue','red']})
4
5  #원-핫 인코딩
6  one_hot=pd.get_dummies(df['color'],
7  dtype=np.int64)
8
9  #결과 출력
10 print(one_hot)
```

〈실행결과〉

	blue	green	red
0	0	0	1
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1

02

시계열 데이터 전처리 실습

2. 시계열 데이터 전처리 실습

문제 정의와 변수 설명

- 주식을 팔아야 할지, 팔지 말아야 할지 고민한다.
- 고민한 끝에 앞으로 상승한다면 계속 가지고 있기로 마음먹었다.
- 주식은 상승할 것인가 아니면 하강할 것인가?

번호	변수	설명
1	Date	날짜
2	open	거래가 시작되는 9시에 최초로 체결된 거래 가격
3	High	상한가(하루 중 주가 상승폭이 30%인 가격)
4	Low	하한가(하루 중 주가 하락폭이 30%인 가격)
5	Close	장이 마감하는 3시 30분에 마지막으로 체결된 가격
6	Adj Close	회사 측의 영향을 계산해서 반영하는 조정 종가
7	Volume	주식거래량

2. 시계열 데이터 전처리 실습

■ 주식 관련 라이브러리 설치

- FinanceDataReader 사용

FinanceDataReader 설치

```
1  !pip install finance-DataReader
```

2. 시계열 데이터 전처리 실습

애플 주식 데이터 가져오기

- DataReader( )의 파라미터에 '종목 코드', '시작 일자', '종료 일자'를 차례대로 추가
- 원하는 종목의 데이터를 가져오기

2024년 애플주식 데이터 로드하여 확인

```
import matplotlib.pyplot as plt
import seaborn as sns
import FinanceDataReader as fdr
df=fdr.DataReader('AAPL','2024')
df
```

	Open	High	Low	Close	Volume	Adj Close
2024-01-02	187.149994	188.440002	183.889999	185.639999	82488700	184.290421
2024-01-03	184.220001	185.880005	183.429993	184.250000	58414500	182.910507
2024-01-04	182.149994	183.089996	180.880005	181.910004	71983600	180.587540
2024-01-05	181.990005	182.759995	180.169998	181.179993	62303300	179.862823
2024-01-08	182.089996	185.600006	181.500000	185.559998	59144500	184.210999
...	...	...	...	...	...	...
2025-05-14	212.429993	213.940002	210.580002	212.330002	49325800	212.330002
2025-05-15	210.949997	212.960007	209.539993	211.449997	45029500	211.449997
2025-05-16	212.360001	212.570007	209.770004	211.259995	54737900	211.259995
2025-05-19	207.910004	209.479996	204.259995	208.779999	46140500	208.779999
2025-05-20	207.669998	208.470001	205.029999	206.860001	42443700	206.860001

347 rows × 6 columns

```
# 테슬라 주식 로드 예
df = fdr.DataReader('TSLA', start='2023-01-01',
end='2024-12-31')
```

2. 시계열 데이터 전처리 실습

■ 애플 주식 데이터 가져오기

- 국내, 미국 주식 관련 데이터 가져오는 방법

미국 주식 정보 가져오기

```
1  import FinanceDataReader as fdr
2  df=fdr.StockListing('NASDAQ')
3  print(df.head())
```

〈실행결과〉

```
100%|■■■■■■■■■■| 4393/4393 [00:36<00:00, 121.45it/s]
Symbol                    Name                Industry  IndustryCode
0   AAPL              Apple Inc  컴퓨터, 전화 및 가전제품        571060
1   MSFT         Microsoft Corp   소프트웨어 및 IT서비스        572010
2   AMZN         Amazon.com Inc          다양한 소매업        534020
3   TSLA              Tesla Inc   자동차 및 자동차 부품         31010
4  GOOGL  Alphabet Inc Class A   소프트웨어 및 IT서비스        572010
```

국내 주식 정보 가져오기

```
1  import FinanceDataReader as fdr
2  df=fdr.StockListing('KRX')
3  print(df.head())
```

〈실행결과〉

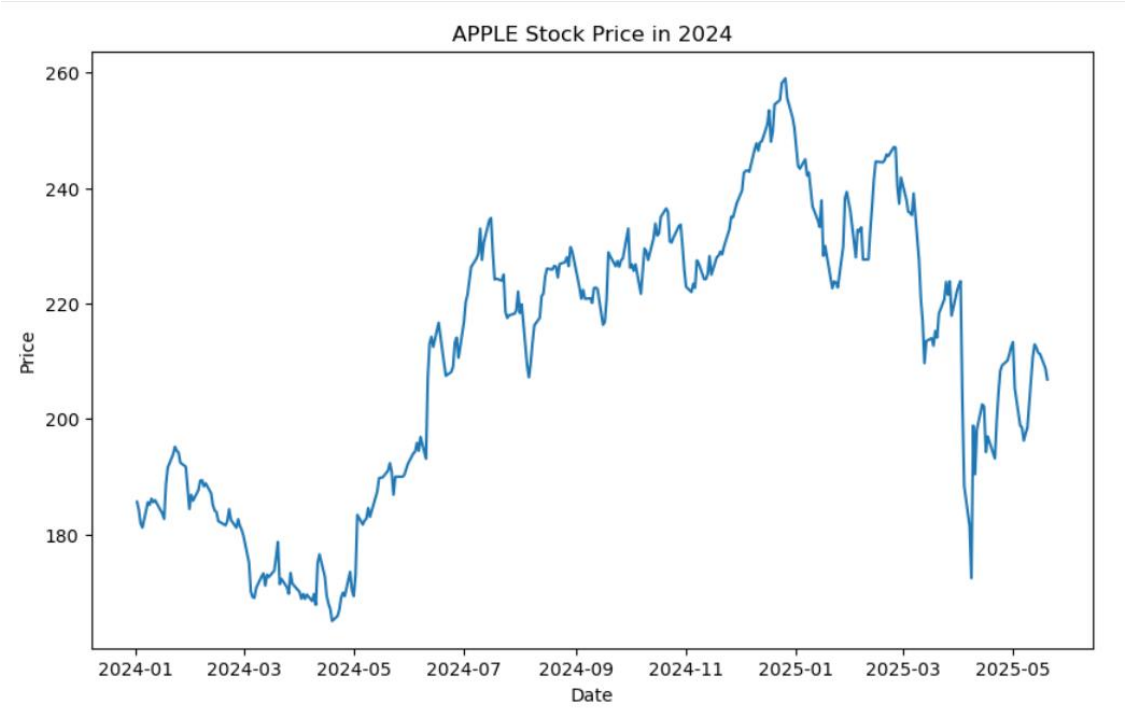
```
    Code       ISU_CD      Name Market Dept      Close ChangeCode  Changes ₩
0  005930  KR7005930003    삼성전자  KOSPI      61300          2     -700
1  373220  KR7373220003  LG에너지솔루션  KOSPI     511000          1     4000
2  000660  KR7000660001   SK하이닉스  KOSPI      91000          2     -1700
3  207940  KR7207940008  삼성바이오로직스  KOSPI    783000          2     -6000
4  051910  KR7051910008    LG화학  KOSPI      669000          1     1000

   ChagesRatio    Open    High     Low    Volume        Amount ₩
0        -1.13   62300   62600   61300  10308143  638634052687
1         0.79  508000  522000  507000    239646  123155230000
2        -1.83   93500   94000   91000   2496352  230723303704
3        -0.76  789000  790000  781000     39167   30725679000
4         0.15  668000  686000  666000    145162   97814598000

             Marcap      Stocks MarketId
0  365947670315000  5969782550      STK
1  119574000000000   234000000      STK
2   66248215215000   728002365      STK
3   55729242000000    71174000      STK
4   47226277467000    70592343      STK
```

2. 시계열 데이터 전처리 실습

주식 가격 시각

- matplotlib 패키지를 사용하여 데이터를 시각화

```
# 애플 주식 가격 시각화
plt.figure(figsize=(10,6))
sns.lineplot(x=df1.index, y=df1['Close'])
plt.title('APPLE Stock Price in 2024')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show(
```



2. 시계열 데이터 전처리 실습

■ 시계열 데이터 전처리 및 데이터 확인

- resample()을 이용하여 다운샘플링

```
# 다운샘플링
#날짜 기준으로 월말 평일(Business Month End) 단위로 데이터를 다시 샘플링
apple_month=df1.resample('BME').mean()
apple_month.head()
```

✓ 0.0s

	Open	High	Low	Close	Volume	Adj Close
2024-01-31	187.597143	188.999047	186.099524	187.724284	5.653425e+07	186.359552
2024-02-29	184.668000	186.043000	183.031001	184.775500	5.808135e+07	183.594860
2024-03-29	172.789000	174.319500	171.262498	172.696500	7.163914e+07	171.659779
2024-04-30	169.661363	171.292272	168.448183	169.604545	5.662350e+07	168.586384
2024-05-31	186.431817	187.820910	185.055909	186.285909	6.075171e+07	185.341726

2. 시계열 데이터 전처리 실습

시계열 데이터 전처리 및 데이터 확인

- pct_change()를 이용하여 수익률 구하기
- 변수 추가하기

```
# 수익률 추가
apple_month['rtn']=apple_month['Close'].pct_change()
df_month.head()
```

✓ 0.0s

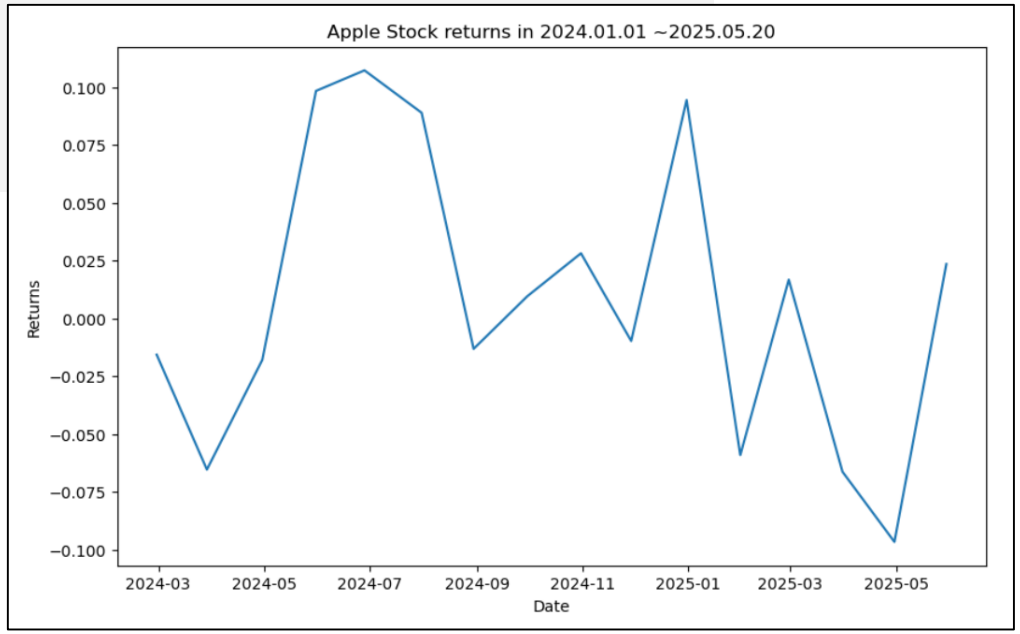
	Open	High	Low	Close	Volume	Adj Close	rtn
2024-01-31	187.597143	188.999047	186.099524	187.724284	5.653425e+07	186.359552	NaN
2024-02-29	184.668000	186.043000	183.031001	184.775500	5.808135e+07	183.594860	-0.015708
2024-03-29	172.789000	174.319500	171.262498	172.696500	7.163914e+07	171.659779	-0.065371
2024-04-30	169.661363	171.292272	168.448183	169.604545	5.662350e+07	168.586384	-0.017904
2024-05-31	186.431817	187.820910	185.055909	186.285909	6.075171e+07	185.341726	0.098354

2. 시계열 데이터 전처리 실습

시계열 데이터 전처리 및 데이터 확인

- 수익률 시각화

```
# 수익률 시각화
plt.figure(figsize=(10,6))
sns.lineplot(x=apple_month.index, y=apple_month['rtn'])
plt.title('Apple Stock returns in 2024.01.01 ~2025.05.20')
plt.xlabel('Date')
plt.ylabel('Returns')
plt.show()
```



2. 시계열 데이터 전처리 실습

시계열 데이터 전처리 및 데이터 확인

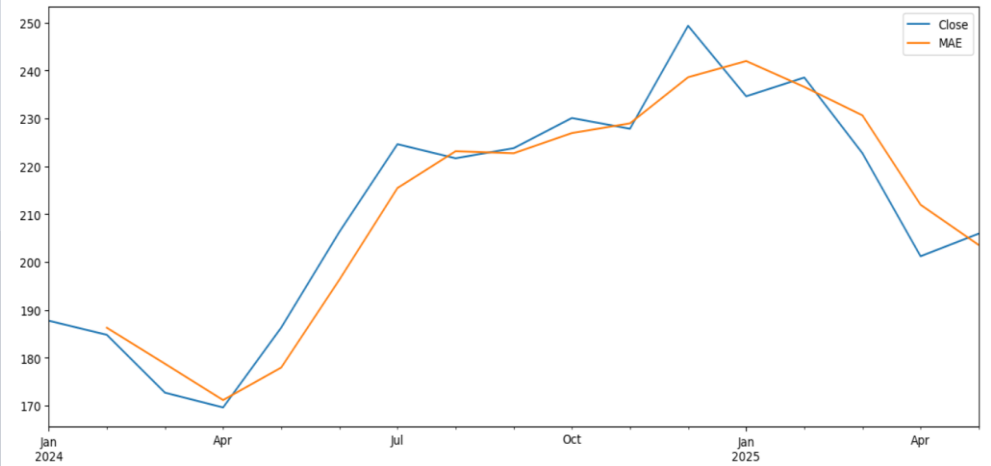
- 주가 흐름 파악하기

```
# 2달씩 종가로 이동평균 구하기
apple_month['MAE']=apple_month['Close'].rolling(2).mean()
apple_month.head()
```

✓ 0.0s

```
apple_month.iloc[:,[3,7]].plot(figsize=(15,6))
```

```
<Axes: >
```



	Open	High	Low	Close	Volume	Adj Close	rtn	MAE
2024-01-31	187.597143	188.999047	186.099524	187.724284	5.653425e+07	186.359552	NaN	NaN
2024-02-29	184.668000	186.043000	183.031001	184.775500	5.808135e+07	183.594860	-0.015708	186.249892
2024-03-29	172.789000	174.319500	171.262498	172.696500	7.163914e+07	171.659779	-0.065371	178.736000
2024-04-30	169.661363	171.292272	168.448183	169.604545	5.662350e+07	168.586384	-0.017904	171.150522
2024-05-31	186.431817	187.820910	185.055909	186.285909	6.075171e+07	185.341726	0.098354	177.945227

2. 시계열 데이터 전처리 실습

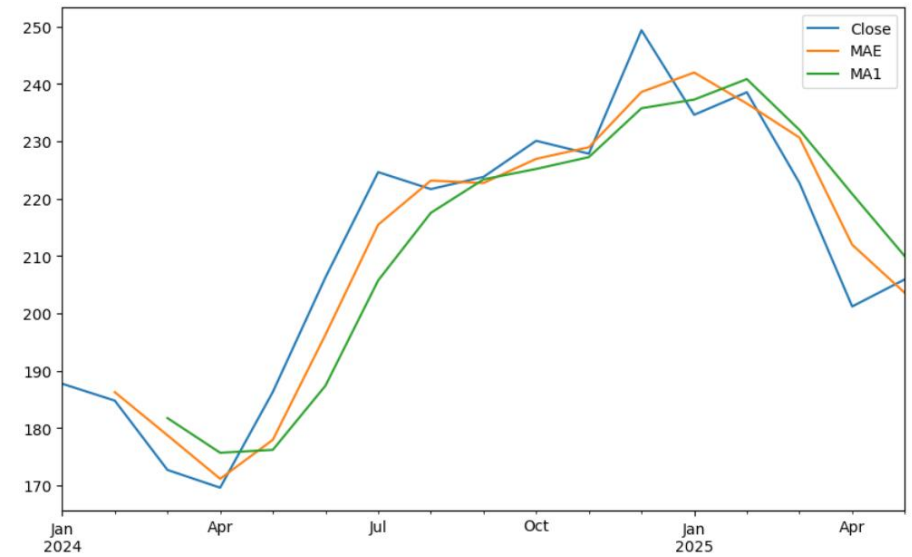
시계열 데이터 전처리 및 데이터 확인

- 주가 흐름 파악하기

이동평균선

```
apple_month['MA1']=apple_month['Close'].rolling(3).
mean()
apple_month.iloc[:,[3,7,8]].plot(figsize=(15,8))
```

<Axes: >



상승장인지 하락장인지 판별

```
last_close=apple_month['MAE'].iloc[-2] # 이동평균선 60일전 종가
print(last_close)
price=apple_month['Close'].iloc[-1] #오늘 종가
print(price)
if price>last_close:
    print('상승장')
elif price<last_close:
    print('하락장')
else:
    print('변화없음')
```

✓ 0.0s

```
211.9433343069894
205.90999930245536
하락장
```

2. 시계열 데이터 전처리 실습

시계열 데이터 전처리 및 데이터 확인

- 애플 데이터 저장하기

파일로 저장

```
1  apple_month.to_csv('data/apple_data.csv')
```

03

시계열 데이터 예측 분석

3. 시계열 데이터 예측 분석

ARIMA 사용

- 시세예측 분석에서 사용할 모델

ARIMA 기본 사용법

```
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
ARIMA(데이터, order=(AR,Difference,MA))
```

예) order = (3,1,3)은 ARIMA 모델의 파라미터 설정을 의미한다. 여기서 첫 번째 값인 3은 AR 모델에서 몇 번째 과거까지를 바라보는지 나타내는 파라미터이며, 두 번째 값인 1은 차분(Difference)에 대한 파라미터이고, 세 번째 값인 3은 MA(이동 평균) 모델에서 몇 번째 과거까지를 바라보는지 나타내는 파라미터이다. AR과 MA의 합이 2 미만인 경우, 혹은 곱이 0을 포함한 짝수인 경우가 좋은 파라미터 값으로 알려져 있다.

3. 시계열 데이터 예측 분석

ARIMA 모델 만들기

- 시세예측 분석에서 사용할 모델

```
# ARIMA 적용
from statsmodels.tsa.arima.model import ARIMA
#from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
#ARIMA(데이터, order=(AR, Difference, MA))
df=pd.read_csv('data/apple_data.csv')
#print(df)
# ARIMA 모델 만들기
model=ARIMA(df['Close'].values, order=(0,1,2))
model_fit=model.fit()
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   17
Model:                 ARIMA(0, 1, 2)   Log Likelihood                -62.054
Date:                Wed, 21 May 2025   AIC                            130.107
Time:                        17:19:41   BIC                            132.425
Sample:                             0   HQIC                           130.226
                                  - 17
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1          0.4444     63.762      0.007      0.994    -124.526     125.415
ma.L2          0.9998    286.838      0.003      0.997    -561.192     563.191
sigma2       103.6882   2.97e+04      0.003      0.997   -5.82e+04    5.84e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.70   Jarque-Bera (JB):                 1.30
Prob(Q):                              0.40   Prob(JB):                         0.52
Heteroskedasticity (H):               1.00   Skew:                             0.33
Prob(H) (two-sided):                  1.00   Kurtosis:                         1.76
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

상수가 유효하지 않은 경우

```
1    model_fit(trend='nc')
```

3. 시계열 데이터 예측 분석

■ 모델을 이용하여 예측

- predict( )를 사용하여 시각화

ARIMA 모델사용 예측

```
predict=model_fit.predict()
predict
```

〈실행결과〉

```
array([  0.        , 187.74157774, 183.59527611, 168.47415692,
       164.11239295, 194.86746133, 226.31695361, 232.06001786,
       216.2886123 , 218.69321624, 240.6250247 , 231.89828715,
       245.69508498, 244.91653062, 226.3247165 , 215.60255116,
       191.93192509])
```

미래의 값 예측

```
1  forecast_data = model_fit.forecast(steps=5)
2  print(forecast_data)
```

〈실행결과〉

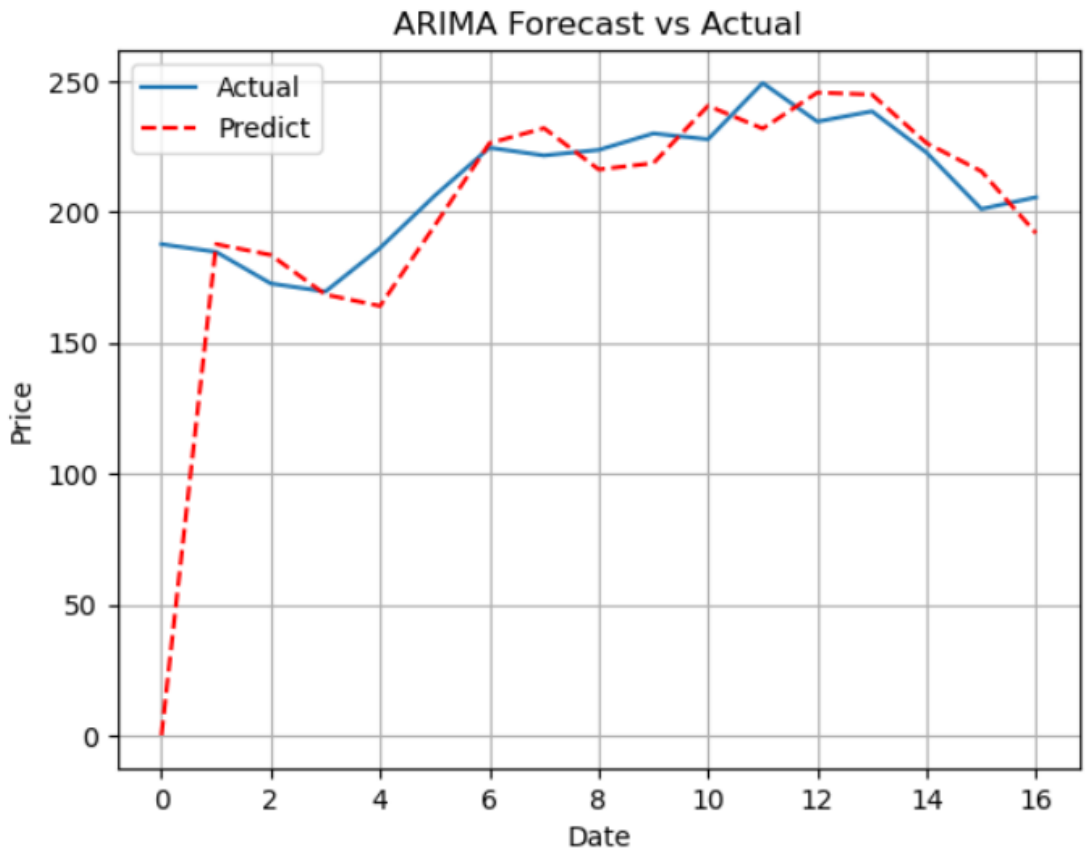
```
(array([145.965245  , 136.42198279, 134.53326575, 132.64454871,
        130.75583167]),
 array([7.6338443 , 8.88415336, 8.88415336, 8.88415336, 8.88415336]),
 array([[131.0031851 , 160.9273049 ],
        [119.00936217, 153.83460341],
        [117.12064513, 151.94588637],
        [115.23192809, 150.05716932],
        [113.34321105, 148.16845228]]))
```

3. 시계열 데이터 예측 분석

모델을 이용하여 예측

- 예측값과 실제 값을 사용하여 시각화

```
import matplotlib.pyplot as plt

# 실제 데이터
plt.plot(df['Close'], label='Actual')
index=range(17)
# 예측 결과 (10일 미래 예측 예시)
#forecast = model_fit.forecast(steps=5)
#print(forecast)
plt.plot(index, predict, label='Predict',
         linestyle='--', color='r')
plt.title('ARIMA Forecast vs Actual')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```



Q&A