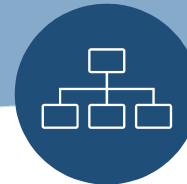
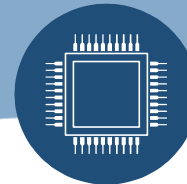


[일머리사관학교 디지털헬스케어]



# 시계열데이터분석

경남대학교 창의융합대학 교수 유현주  
[comjoo@kyungnam.ac.kr](mailto:comjoo@kyungnam.ac.kr)



# 강의 진행 안내



- 수업 자료 공유 URL
  - <https://github.com/yoohjoo/>
  - 수업에 필요한 강의 자료 다운로드 저장
- 소통을 위한 메일 주소: [comjoo@kyungnam.ac.kr](mailto:comjoo@kyungnam.ac.kr)  
[yoohjoo94@gmail.com](mailto:yoohjoo94@gmail.com)





## 시계열데이터분석

- 시계열데이터
- 시계열 데이터의 정상성 확보
- 인공지능 시계열 예측



# 시계열 데이터

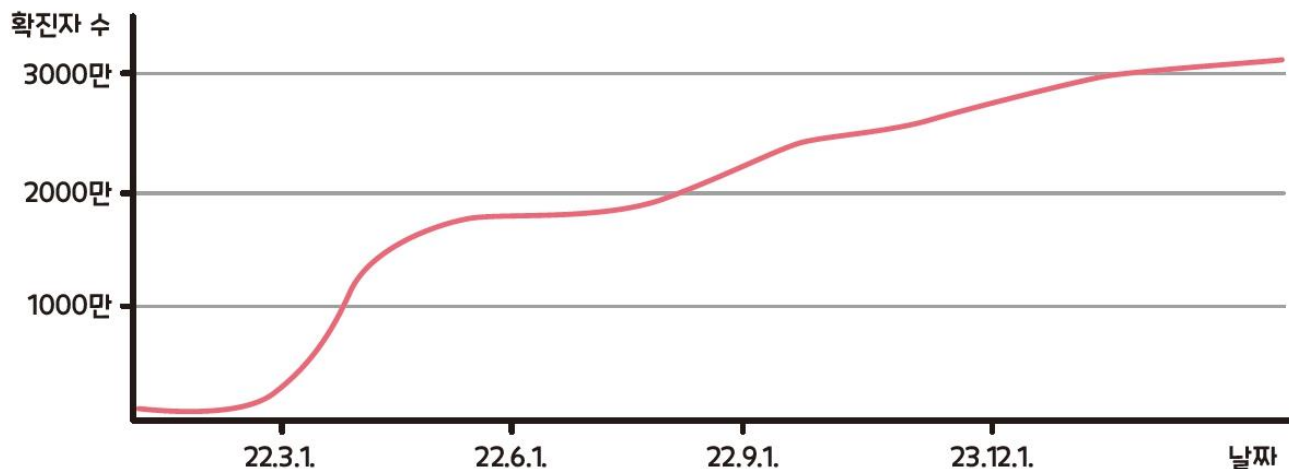


# 시계열 데이터



## ● 시계열 예측과 정상성

- 시계열 예측(Time series forecasting)은 시간 순서대로 수집하거나 정렬한 순차적인 (Sequential) 데이터를 활용하여 미래 시점의 상태를 예측하는 문제.
- 일별 코로나19 확진자 수와 같은 자료를 가지고 앞으로의 확진자 수를 예측하고 방역 정책을 수립하는 데 활용하기도 했음.
- 정상성(Stationarity) : 통계적 성질이 변하지 않고 일정한 상태를 의미





- 시계열 예측과 정상성

- 시계열 예측의 특징

- 시계열 예측은 분류 및 예측 문제와 다른 점이 몇 가지 있음.
      - 시계열 예측은 완벽할 수 없으며 변동 가능성이 있음.
      - 시계열 예측에서는 다변량인 경우가 흔함.
      - 시계열 데이터는 데이터셋의 구성을 적절히 변형하기 어려움.
    - 시계열 예측을 위해서는 데이터의 성질을 고려해야 함.
      - 데이터가 정상성을 띠 때 시계열 예측 정확도가 높음
      - 비정상성 시계열 데이터는 정상성을 갖도록 변환한 후에 예측에 활용.

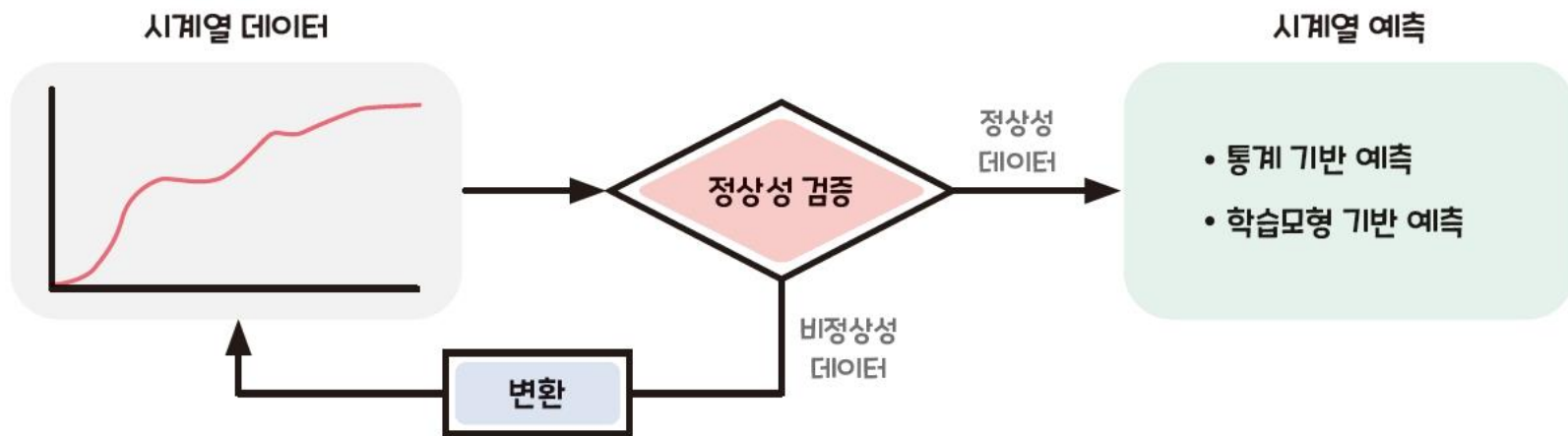
# 시계열 데이터



## ● 시계열 예측과 정상성

### ● 데이터의 정상성

- 데이터의 정상성(Stationarity)과 비정상성(Non-Stationarity).
- 정상성과 비정상성 판별: 확률 요소들이 예측 가능한 범위에 있는지 판단함.
- 비정상성 데이터는 정상성을 가지도록 변환해야 통계 기반 시계열 예측의 예측 정확도를 향상할 수 있음.



# 시계열 데이터



- 시계열 예측과 정상성

- 대표적인 시계열 패턴

- 시계열 데이터에서 주로 나타나는 패턴은 3가지.

① 추세(Trend): 장기적으로 증가하거나 감소하는 패턴.





# 시계열 데이터

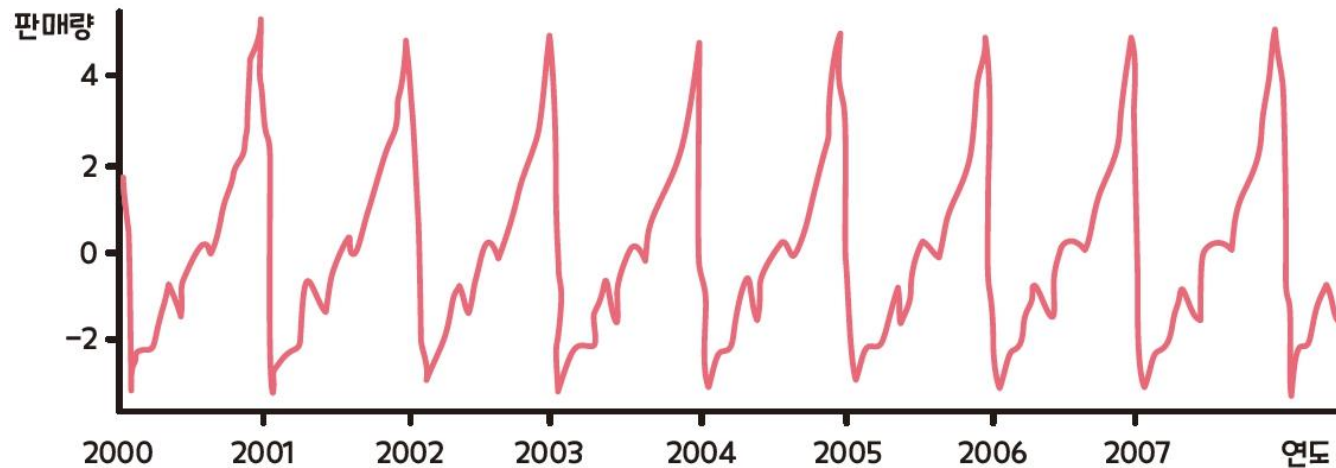


- 시계열 예측과 정상성

- 대표적인 시계열 패턴

- 시계열 데이터에서 주로 나타나는 패턴은 3가지.

② 계절성(Seasonality): 특정한 시기나 특정 요일, 주, 월 등에 반복되는 계절성 요인에 영향을 받는 패턴.



# 시계열 데이터

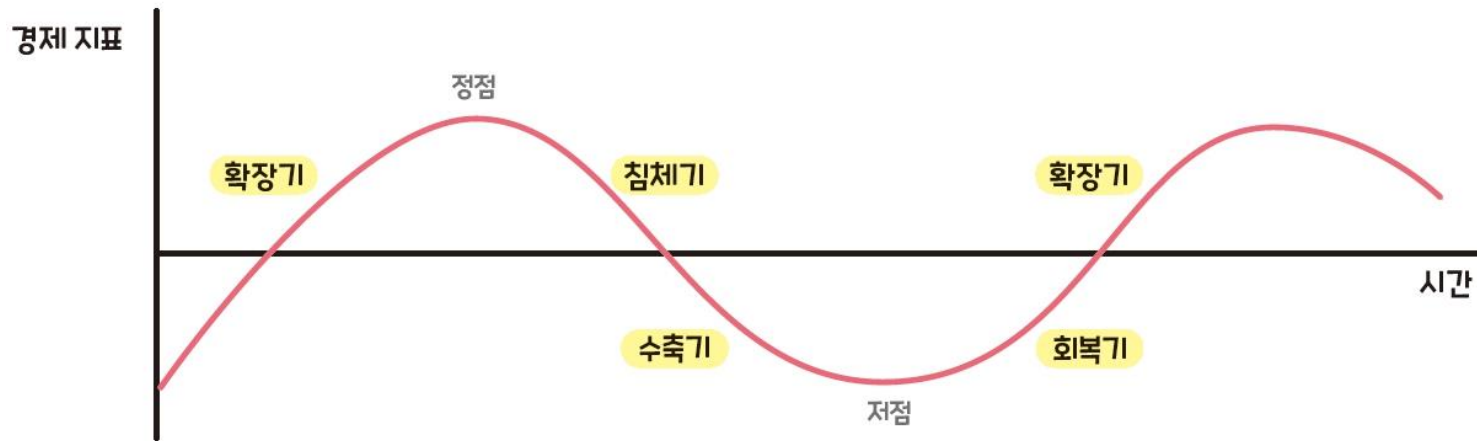


- 시계열 예측과 정상성

- 대표적인 시계열 패턴

- 시계열 데이터에서 주로 나타나는 패턴은 3가지.

③ 주기성(Cycle): 정해진 양이나 빈도 없이 값의 증가나 감소가 반복되는 패턴



# 시계열 데이터



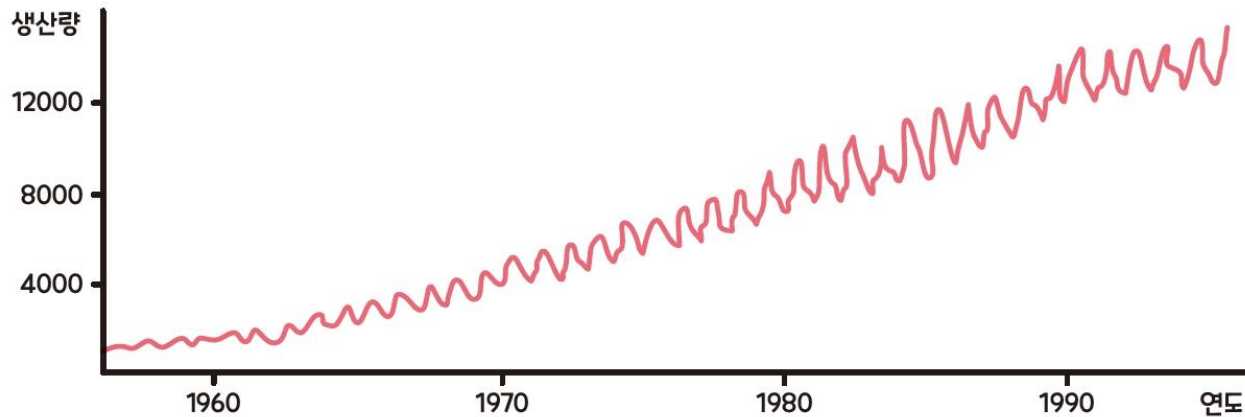
## ● 시계열 예측과 정상성

### ● 시계열 그래프와 정상성

#### • 정상성 데이터의 3가지 조건

- ① 그래프에 지속적인 상승 또는 하락 추세가 없다.
- ② 과거의 변동폭과 현재의 변동폭이 같다.
- ③ 계절성이 없다.

#### • 시계열의 패턴은 추세, 계절성, 주기성이 복합적으로 발생함





# 시계열 데이터

## ● 통계적 정상성 검증

- 일반적으로는 통계 기법으로 데이터의 정상성을 검증.
- 시각적 판단과 통계적인 판단은 사실은 동일한 방법.
- 시계열 데이터에서 서로 다른 시점의 값 사이 선형관계를 측정(자기상관)하거나 한 시점의 값이 결과에 미치는 영향을 측정(편자기상관)하여 조건을 만족하는지 판단.

시각적 판단	통계적 판단
그래프에 지속적인 상승 또는 하락 추세가 없음	평균이 일정
과거의 변동폭과 현재의 변동폭이 같음	분산이 시점에 독립적
계절성이 없음	공분산이 시차에 의존적이나 시점에 독립적

- 시계열에서 추세가 보인다면 평균이 일정하지 않을 것.
  - 차분(Differencing)하여 비정상성 요소를 제거.
- 분산이 일정하지 않다면 변환(Transformation)으로 데이터를 가공하여 정상성을 띠게 함.

# 시계열 데이터



- 통계적 정상성 검증

- 테슬라 사 주가 시계열의 정상성을 그래프와 ADF(Augmented Dickey-Fuller Test) 검정으로 판단함

야후 파이낸스 주가 데이터

```
import yfinance as yf
```

테슬라 주가 로드

```
tsla = yf.download('TSLA', start='2018-01-01', end='2022-12-31')  
tsla
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-01-02	20.799999	21.474001	20.733334	21.368668	21.368668	65283000
2018-01-03	21.400000	21.683332	21.036667	21.150000	21.150000	67822500
2018-01-04	20.858000	21.236668	20.378668	20.974667	20.974667	149194500
2018-01-05	21.108000	21.149332	20.799999	21.105333	21.105333	68868000
2018-01-08	21.066668	22.468000	21.033333	22.427334	22.427334	147891000
...	...	...	...	...	...	...
2022-12-23	126.370003	128.619995	121.019997	123.150002	123.150002	166989700
2022-12-27	117.500000	119.669998	108.760002	109.099998	109.099998	208643400
2022-12-28	110.349998	116.269997	108.239998	112.709999	112.709999	221070500
2022-12-29	120.389999	123.570000	117.500000	121.820000	121.820000	221923300
2022-12-30	119.949997	124.480003	119.750000	123.180000	123.180000	157777300

# 시계열 데이터



- 통계적 정상성 검증

- 그래프로 정상성 검증하기

- 일별 종가인 Close 열의 정상성을 판별.

테슬라 주식 종가

```
import pandas as pd
df_tsla = pd.DataFrame(tsla['Close'])
```

- Close 열에 종가가 있으므로 이 열만 데이터프레임 df\_tsla로 저장

테슬라 주가 시각화

```
import matplotlib.pyplot as plt
df_tsla.plot(figsize=(12.2, 6.4))
```

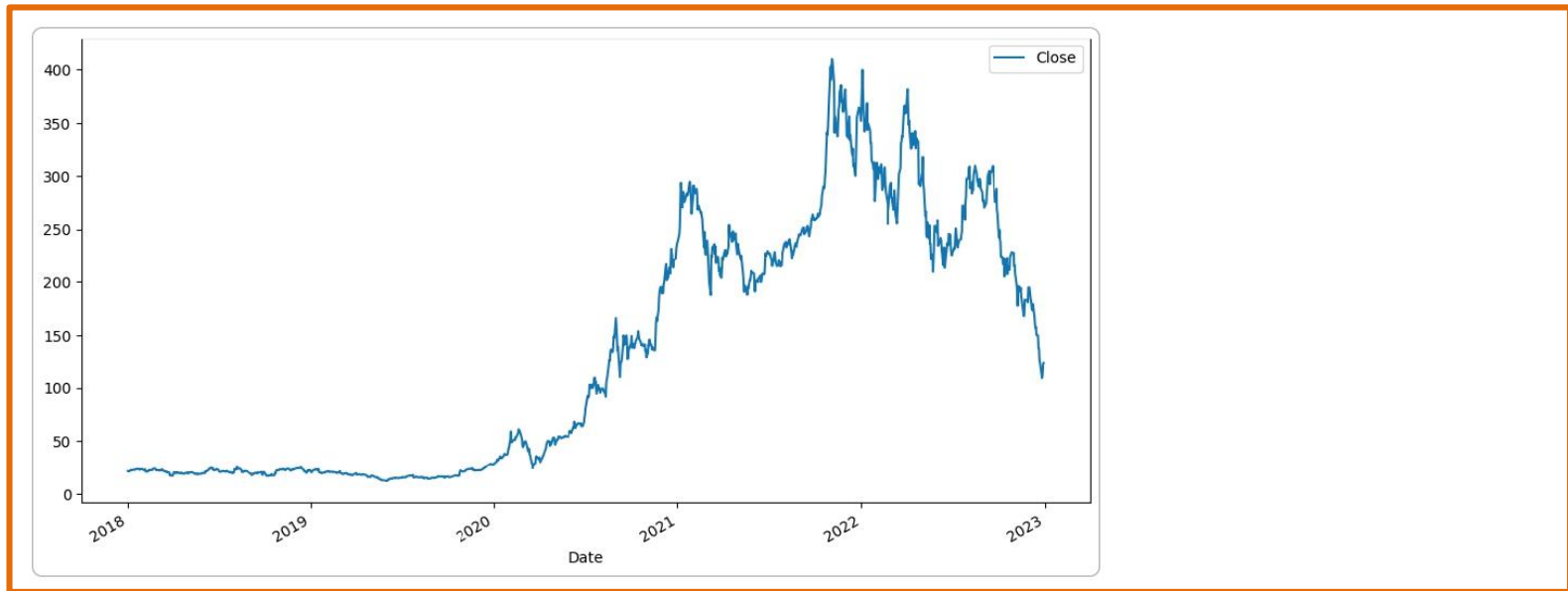
# 시계열 데이터



- 통계적 정상성 검증

- 그래프로 정상성 검증하기

실행결과



- 2018년부터 2022년까지 긴 상승 추세, 2022년 이후 하락 추세
- 상승과 하락의 주기성이 약하게 나타남.
- 그래프만으로는 정상성 데이터의 3가지 기준을 충족하는지 알아내기 어려움.

# 시계열 데이터



## ● 통계적 정상성 검증

### ● ADF 검정

- ADF(Augmented Dickey-Fuller) 검정은 대표적인 정상성 검증 기법.
- 가장 중요한 판단 기준인 p-값 확인만 진행.
- 데이터가 정상성을 띠는다는 대립가설을 두고, p-값이 0.05 이상일 때 가설을 기각.

#### ADF 검정

```
from statsmodels.tsa.stattools import adfuller

print('ADF test with TSLA time-series')
ADF_result = adfuller(df_tsla.values)
#ADF 통계량
print('ADF Stats: %f' % ADF_result[0])
#p-값
print('p-value: %f' % ADF_result[1])
#임계값
print('Critical values: ' )
for key, value in ADF_result[4].items():
    print('\t%s: %.4f' % (key, value))
```



# 시계열 데이터



- 통계적 정상성 검증

- 그래프로 정상성 검증하기

## 실행결과

```
ADF test with TSLA time-series
ADF Stats: -1.301076
p-value: 0.628680
Critical values:
    1%: -3.4356
    5%: -2.8639
   10%: -2.5680
```

- ADF 검정에서 p-값이 0.05 이하이거나, ADF 검정 통계량(ADF Stats)이 임계값(Critical values) 보다 작으면 정상성을 가진다고 판단할 수 있음.
- p-값이 0.62여서 통계적으로 대립가설이 기각되어 이 데이터가 비정상성을 가진다고 판단.
- ADF 통계량 역시 임계값보다 크게 나타남. 따라서 미래의 테슬라 주가를 예측해 보고자 시계열 예측을 수행하려면 주가 데이터가 정상성을 갖도록 변환해야 함.

# 시계열 데이터



- 통계적 정상성 검증

- KPSS 검정

- KPSS 검정도 시계열 데이터의 정상성 판별 방법. 계절성에 민감한 통계량 분석에 유용.

## KPSS 검정

```
from statsmodels.tsa.stattools import kpss

print('KPSS test with TSLA time-series')
KPSS_result = kpss(df_tsla.values)
#KPSS 통계량
print('KPSS Stats: %f' % KPSS_result[0])
#p-값
print('p-value: %f' % KPSS_result[1])
#임계값
print('Critical values:' )
for key, value in KPSS_result[3].items():
    print('\t%s: %.4f' % (key, value))
```

# 시계열 데이터



- 통계적 정상성 검증

- 그래프로 정상성 검증하기

## 실행결과

```
KPSS test with TSLA time-series
KPSS Stats: 4.913280
p-value: 0.010000
Critical values:
    10%: 0.3470
    5%: 0.4630
    2.5%: 0.5740
    1%: 0.7390
```

- ADF 검정과 동일하게 정상성 판단을 수행.
- KPSS 검정 결과에서 p-값이 0.05 이하이므로 테슬라 시계열 데이터가 비정상성을 가짐.
- KPSS에서의 가설 설정은 기존 통계 검정과 반대.

# 시계열 데이터의 정상성 확보





# 시계열 데이터 정상성 확보

- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

- 이번 절에서는 비정상성을 띠는 시계열 데이터를 적절한 기법을 적용하여 정상성을 가지도록 변환.
    - 1949년부터 1960년까지의 월별 미국 비행기 탑승객 수 데이터를 사용.

## pydataset의 데이터셋 확인

```
!pip install pydataset  
from pydataset import data  
data( )
```



# 시계열 데이터 정상성 확보

- 통계적 정상성 검증
  - 비정상성 시계열 데이터 준비

## 실행결과

	dataset_id	title
0	AirPassengers	Monthly Airline Passenger Numbers 1949-1960
1	BJsales	Sales Data with Leading Indicator
2	BOD	Biochemical Oxygen Demand
3	Formaldehyde	Determination of Formaldehyde
4	HairEyeColor	Hair and Eye Color of Statistics Students
...	...	...
752	VerbAgg	Verbal Aggression item responses
753	cake	Breakage Angle of Chocolate Cakes
754	cbpp	Contagious bovine pleuropneumonia
755	grouseticks	Data on red grouse ticks from Elston et al. 2001
756	sleepstudy	Reaction times in a sleep deprivation study
757 rows × 2 columns		



# 시계열 데이터 정상성 확보

- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

- pydataset에서 0번 Airpassengers 데이터셋을 불러와 설명을 확인.

pydataset의 데이터셋 확인

AirPassengers 데이터셋 로드

```
AirPassengers
```

```
PyDataset Documentation (adopted from R Documentation. The displayed examples are in R)
```

```
## Monthly Airline Passenger Numbers 1949-1960
```

```
### Description
```

```
The classic Box & Jenkins airline data. Monthly totals of international  
airline passengers, 1949 to 1960.
```

```
### Usage
```

```
AirPassengers
```

# 시계열 데이터 정상성 확보



- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

```
### Format
```

```
A monthly time series, in thousands.
```

```
### Source
```

```
Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976) _Time Series Analysis,  
Forecasting and Control._ Third Edition. Holden-Day. Series G.
```

```
### Examples
```

- 정상적으로 가져왔다면 show\_doc=True 인자 때문에 데이터셋 설명이 나타남.



# 시계열 데이터 정상성 확보



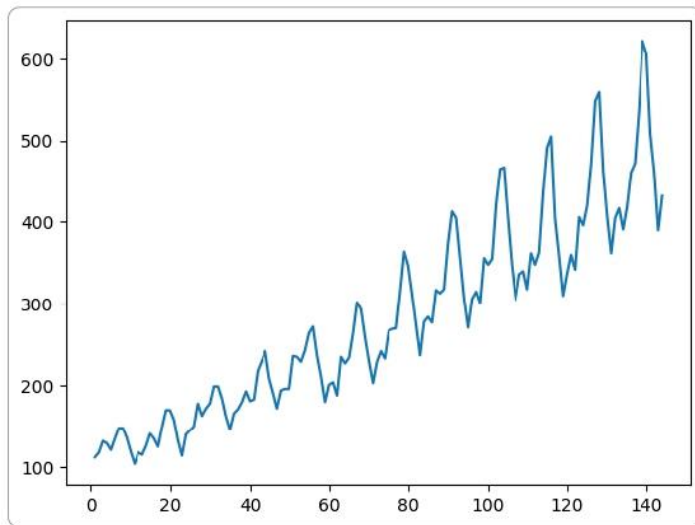
- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

- 가져온 데이터를 시각화해 어떤 시계열 패턴이 포함되었는지 확인.

그래프로 시계열 패턴 확인

```
import matplotlib.pyplot. as plt  
data = air['AirPassengers']  
plt.plot(data)  
plt.show( )
```



# 시계열 데이터 정상성 확보



- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

- 시각화로 비정상성을 확인했음. ADF와 KPSS 검정으로도 검증.

## AirPassengers 데이터셋 ADF 검증

```
from statsmodels.tsa.stattools import adfuller

print('ADF test with AirPassengers time-series')
ADF_result = adfuller(data)
#ADF 통계량
print('ADF Stats: %f' % ADF_result[0])
#p-값
print('p-value: %f' % ADF_result[1])
#임계값
print('Critical values:' )
for key, value in ADF_result[4].items():
    print('\t%s: %.4f' % (key, value))
```

# 시계열 데이터 정상성 확보



- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

## 실행결과

ADF test with AirPassengers time-series

ADF Stats: 0.815369

p-value: 0.991880

Critical values:

1%: -3.4817

5%: -2.8840

10%: -2.5788

- ADF 검정 결과 p-값이 0.99이고 ADF 통계량도 임계값보다 훨씬 큼.
- 따라서 AirPassengers 데이터셋은 비정상성 데이터.



# 시계열 데이터 정상성 확보

- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

AirPassengers 데이터셋 KPSS 검정

```
from statsmodels.tsa.stattools import kpss

print('KPSS test with AirPassengers time-series')
KPSS_result = kpss(data)
#KPSS 통계량
print('KPSS Stats: %f' % KPSS_result[0])
#p-값
print('p-value: %f' % KPSS_result[1])
#임계값
print('Critical values: ' )
for key, value in KPSS_result[3].items():
    print('\t%s: %.4f' % (key, value))
```

# 시계열 데이터 정상성 확보



- 통계적 정상성 검증

- 비정상성 시계열 데이터 준비

## 실행결과

```
KPSS test with AirPassengers time-series
```

```
KPSS Stats: 1.651312
```

```
p-value: 0.010000
```

```
Critical values:
```

```
10%: 0.3470
```

```
5%: 0.4630
```

```
2.5%: 0.5740
```

```
1%: 0.7390
```

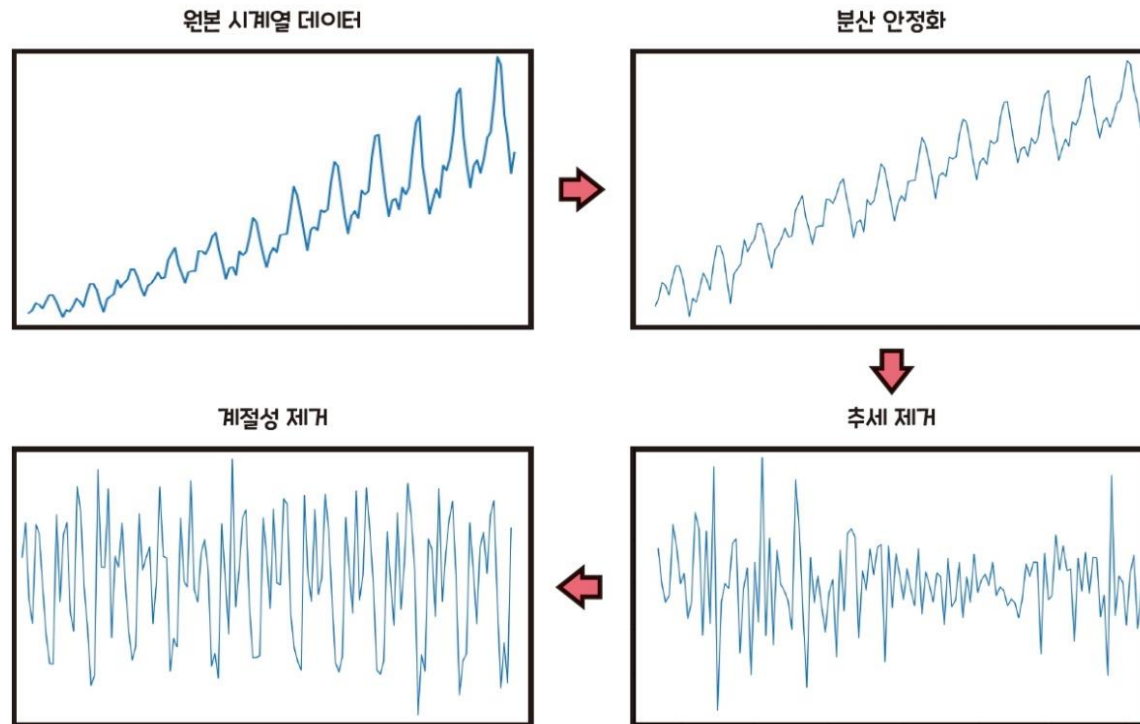
- p-값은 0.01
- AirPassengers 데이터셋은 비정상성을 가진 것으로 결론.



# 시계열 데이터 정상성 확보

## ● 데이터 정상성 확보

- AirPassengers의 정상성을 확보해야 시계열 데이터 예측에 사용할 수 있음.
- 분산 안정화(Variance stabilization), 추세 제거(De-trend by differencing), 계절성 제거(De-seasonality by differencing) 순서로 적용.



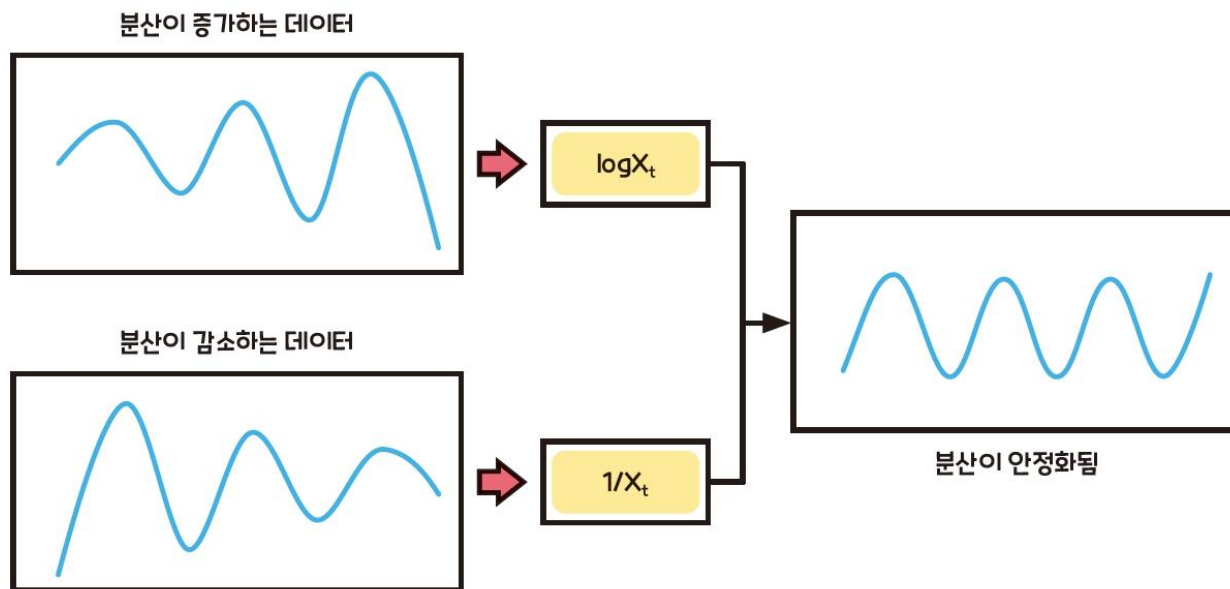
# 시계열 데이터 정상성 확보



## ● 데이터 정상성 확보

### ● 분산 안정화

- 시계열 데이터의 정상성 판별 기준 3가지 중에 두 번째 항목은 '과거의 변동폭과 현재의 변동폭이 같다'이므로 분산이 시점에 의존하지 않아야 함.
- 분산 안정화(Variance stabilizing): 데이터를 멱 변환(Power Transformation).



# 시계열 데이터 정상성 확보



## ● 데이터 정상성 확보

### ● 분산 안정화

#### 로그 변환

```
import numpy as np
df_log_air = np.log(air['AirPassengers'])
df_log_air.head( )
```

```
1 4.718499
2 4.770685
3 4.882802
4 4.859812
5 4.795791
Name: AirPassengers, dtype: float64
```

- 넘파이의 `log( )` 함수로 AirPassengers 열의 로그를 취하고 이를 데이터프레임 `df_log_air`에 저장.
- `df_log_air`의 처음 다섯 행을 출력해 확인.
- 이 숫자만으로는 정말 분산이 평활화되었는지 확인하기 어려우니 다시 시각화하여 확인.



# 시계열 데이터 정상성 확보

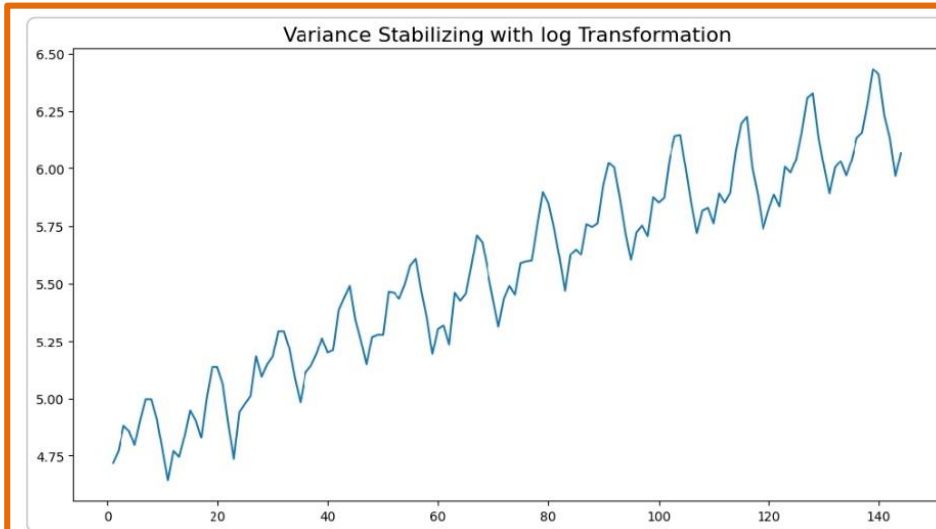


- 데이터 정상성 확보

- 분산 안정화

분산 안정화 결과

```
df_log_air.plot(figsize=(12.2, 6.4))  
plt.title('Variance Stabilizing with log Transformation', fontsize=16)  
plt.show( )
```



- 비교하면 주기별 변동폭이 일정하여 분산이 안정화됨.



# 시계열 데이터 정상성 확보

- 데이터 정상성 확보

- 차분과 추세 제거

- 차분(Differencing)이란 현재 시점의 시계열 값에서 시간  $t$  이전의 값을 빼서 데이터를 값의 증감으로 변환하는 기법.

차분 변환으로 추세 제거

```
df_log_air_diff = df_log_air.diff(1).dropna( )

df_log_air_diff.plot(figsize=(12.2, 6.4))
plt.title('De-trend by Differencing(1st order)', fontsize=16)
plt.show( )
```

- `diff( )` 함수만으로 간단히 차분 변환을 할 수 있음.
- 차분 변환하여 생긴 모든 결측을 `dropna( )` 함수로 제거.

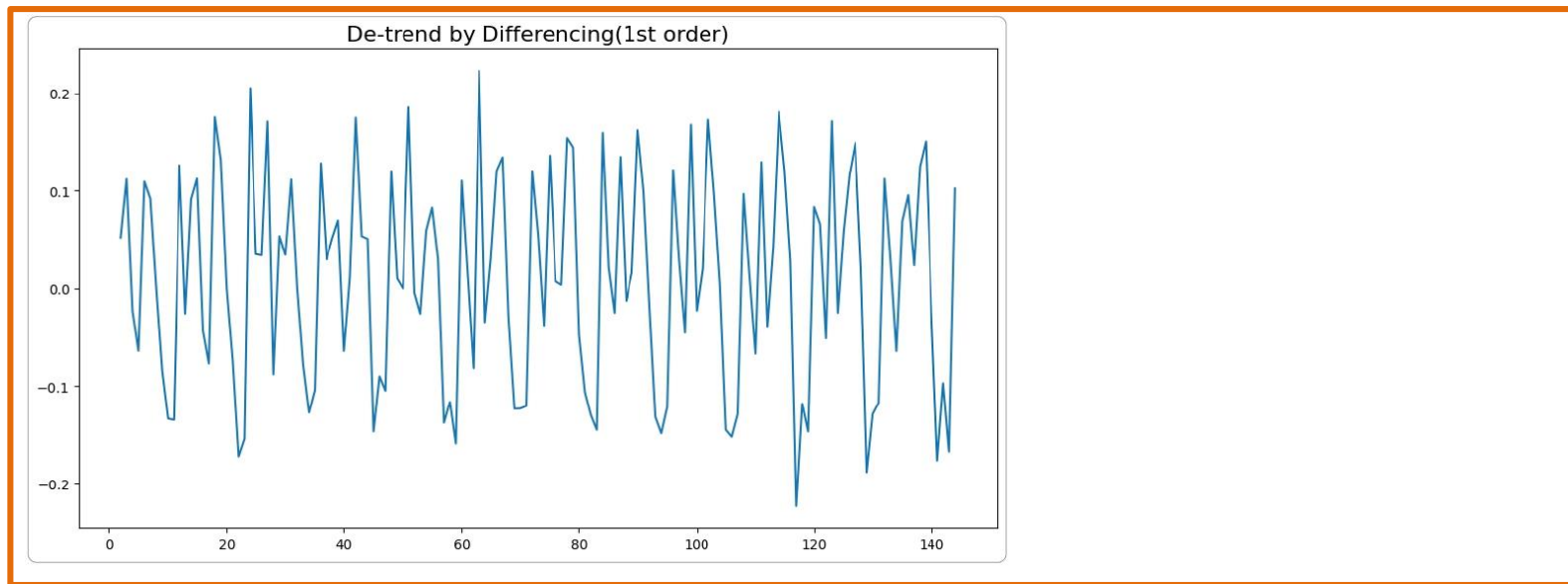
# 시계열 데이터 정상성 확보



- 데이터 정상성 확보

- 차분과 추세 제거

## 실행결과



- 변환 전에는 전체적으로 값이 증가했으나 변환 결과 추세가 제거되었음.



# 시계열 데이터 정상성 확보

- 데이터 정상성 확보

- 차분과 추세 제거

- 차분 변환한 데이터에 대해 ADF 검정을 수행하여 데이터의 정상성이 확보되었는지 판별.

## 차분 결과 ADF 검정

```
print('ADF test with AirPassengers time-series')
ADF_result_diff = adfuller(df_log_air_diff)

print('ADF Stats: %f' % ADF_result_diff[0])
print('p-value: %f' % ADF_result_diff[1])
print('Critical values:' )
for key, value in ADF_result_diff[4].items():
    print('\t%s: %.4f' % (key, value))
```

# 시계열 데이터 정상성 확보



- 데이터 정상성 확보

- 차분과 추세 제거

## 실행결과

```
ADF test with AirPassengers time-series
```

```
ADF Stats: -2.717131
```

```
p-value: 0.071121
```

```
Critical values:
```

```
1%: -3.4825
```

```
5%: -2.8844
```

```
10%: -2.5790
```

- ADF 검정 결과 시계열 데이터의 p-값이 0.07로 유의수준 0.05와 가깝게 되었음.
- ADF 통계량도 임계값과 가까워졌지만 아직 통계적으로 유의한 수준까지 도달하지는 못했으므로, 데이터에 아직 다른 비정상성 패턴이 있음을 추측할 수 있음.

# 시계열 데이터 정상성 확보



## ● 데이터 정상성 확보

### ● 차분과 계절성 제거

- 데이터에 남아있는 계절성을 제거하기 위하여 한 번 더 차분.
- 계절 차분은 1년 중 12개 시점에서 각각 1년 전의 값을 빼는 차분. 따라서 `diff()` 함수의 인자를 차분 계수 12로 설정.

#### 계절 차분 변환

```
df_log_air_diff_season = df_log_air_diff.diff(12).dropna()

df_log_air_diff_season.plot(figsize=(12.2, 6.4))
plt.title('De-seasonality by Differencing(12 month)', fontsize=16)
plt.show()
```

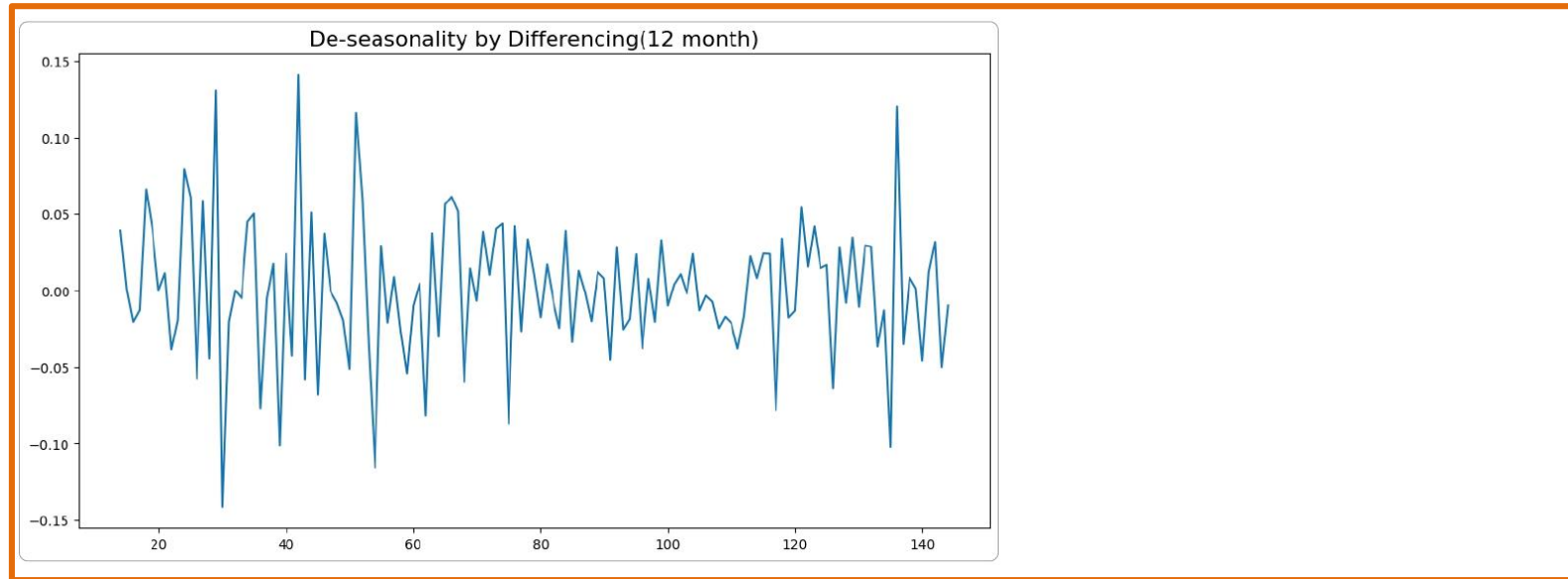
# 시계열 데이터 정상성 확보



- 데이터 정상성 확보

- 차분과 추세 제거

실행결과



- 이전 로그 변환, 추세 차분과 동일하게 `diff( )` 함수에 차분 계수 12를 입력하여 계절 차분을 수행.
- 추세 제거 결과와 비교해 보면 일정 주기로 반복되던 패턴이 사라져 있음.

# 시계열 데이터 정상성 확보



- 데이터 정상성 확보

- 차분과 계절성 제거

최종 변환된 데이터 ADF, KPSS 검정

#ADF 검정

```
print('ADF test with AirPassengers time-series')
ADF_result_diff_season = adfuller(df_log_air_diff_season)
print('ADF Stats: %f' % ADF_result_diff_season[0])
print('p-value: %f' % ADF_result_diff_season[1])
print('Critical values: ' )
for key, value in ADF_result_diff_season[4].items( ):
    print('\t%s: %.4f' % (key, value))
```

#KPSS 검정

```
print('KPSS test with AirPassengers time-series')
KPSS_result_diff_season = kpss(df_log_air_diff_season)
print('KPSS Stats: %f' % KPSS_result_diff_season[0])
print('p-value: %f' % KPSS_result_diff_season[1])
print('Critical values: ' )
for key, value in KPSS_result_diff_season[3].items( ):
    print('\t%s: %.4f' % (key, value))
```



# 시계열 데이터 정상성 확보



## ● 데이터 정상성 확보

### ● 차분과 추세 제거

#### 실행결과

```
ADF test with AirPassengers time-series
ADF Stats: -4.443325
p-value: 0.000249
Critical values:
    1%: -3.4870
    5%: -2.8864
   10%: -2.5800
KPSS test with AirPassengers time-series
KPSS Stats: 0.073191
p-value: 0.100000
Critical values:
   10%: 0.3470
    5%: 0.4630
   2.5%: 0.5740
    1%: 0.7390
```

- ADF 검정 결과 p-값이 0.000249이고, KPSS 검정 결과는 p-값이 0.1.
- AirPassengers 데이터셋은 이제 시계열 예측에 적절한 데이터로 변환되었음.

# 시계열 데이터 정상성 확보



- 데이터 정상성 확보

- 차분과 추세 제거

- ✓ 하나 더 알기: 비정상성 시계열 데이터 변환 순서

AirPassengers처럼 분산이 고정적이지 않으면서 추세나 계절성 패턴이 모두 나타나는 비정상성 데이터는 분산 안정화를 가장 먼저 수행.

추세나 계절성을 제거하는 차분 과정에서 0 또는 음수(-)가 생기면 로그 변환에 제약이 생기기 때문.

# 시계열 예측 모델링 기법





# 시계열 예측 모델링 기법

- 시계열 데이터를 분석하고 예측하는 모델
- 통계 기반 예측
  - ARIMA 모델
    - AutoRegressive Integrated Moving Average
    - 시계열 데이터를 AR(자기회귀), I(차분), MA(이동 평균)로 분해하여 분석
  - SARIMA 모델
    - 계절성을 반영한 ARIMA 모델
  - 지수 평활법(Exponential Smoothing)
    - 과거 데이터의 중요도를 지수적으로 감소시키는 방식
- 인공지능 시계열 예측
  - RNN (Recurrent Neural Network, 순환신경망) 딥러닝 알고리즘



# 시계열 예측 모델링 기법

- 통계 기반 예측

- 통계 기반 예측

- ARIMA(AutoRegressive Integrated Moving Average)는 자기회귀 모형과 이동평균 모형을 결합한 모형.
  - 자기회귀(AR) 모형: 데이터의 과거 값들을 선형으로 조합하여 미래 값을 예측.
  - 이동평균(MA) 모형: 데이터가 어떤 방향성을 가지고 증가하거나 감소할 때, 즉 데이터가 이동할 때마다 평균을 구하여 예측에 활용.
  - ARMA 모형: AR 모형과 MA모형을 결합한 모형.
- AR, MA, ARMA 모형은 모두 정상성 시계열 데이터를 전제로 사용 가능.
- ARIMA는 AR과 MA가 결합한 점은 ARMA와 동일하지만 비정상성 데이터를 차분하여 예측하도록 지원.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 테슬라 일별 종가 데이터로 주가를 예측하고 결과를 분석.

## 테슬라 주가 데이터

```
import pandas as pd
import yfinance as yf
tsla = yf.download('TSLA', start='2021-11-01', end='2023-03-31')
df_tsla = pd.DataFrame(tsla['Close'])
df_tsla.head( )
```

Close	
Date	
2021-11-01	402.863342
2021-11-02	390.666656
2021-11-03	404.619995
2021-11-04	409.970001
2021-11-05	407.363342



# 시계열 예측 모델링 기법

- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 현재 인덱스인 Date 열의 값을 변경하려면 인덱스 지정을 해제한 다음 변경해야 함.

## 인덱스 가공

```
df_tsla = df_tsla.reset_index()
df_tsla.columns = ['date', 'value']
df_tsla['date'] = pd.to_datetime(df_tsla['date'])
```

- reset\_index( ) 함수로 인덱스를 초기화.
- 데이터프레임의 각 열에 열 이름 date와 value를 부여한 다음 to\_datetime( ) 함수를 사용하여 date 열을 파이썬의 날짜 및 시간 형식인 datetime으로 변환.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

학습 데이터와 테스트 데이터 분할

```
import matplotlib.pyplot as plt

#데이터 분할하기
df_tsla_train = pd.DataFrame(df_tsla['value'][:int(0.8*len(df_tsla))])
df_tsla_test = pd.DataFrame(df_tsla['value'][int(0.8*len(df_tsla)):])
df_tsla_train['date'] = df_tsla['date'][:int(0.8*len(df_tsla))]
df_tsla_test['date'] = df_tsla['date'][int(0.8*len(df_tsla)):]
df_tsla_train.set_index('date', inplace=True)
df_tsla_test.set_index('date', inplace=True)

df_tsla_train['value'].plot(figsize=(12.2, 6.4), color='blue')
df_tsla_test['value'].plot(color='green')
plt.show()
```



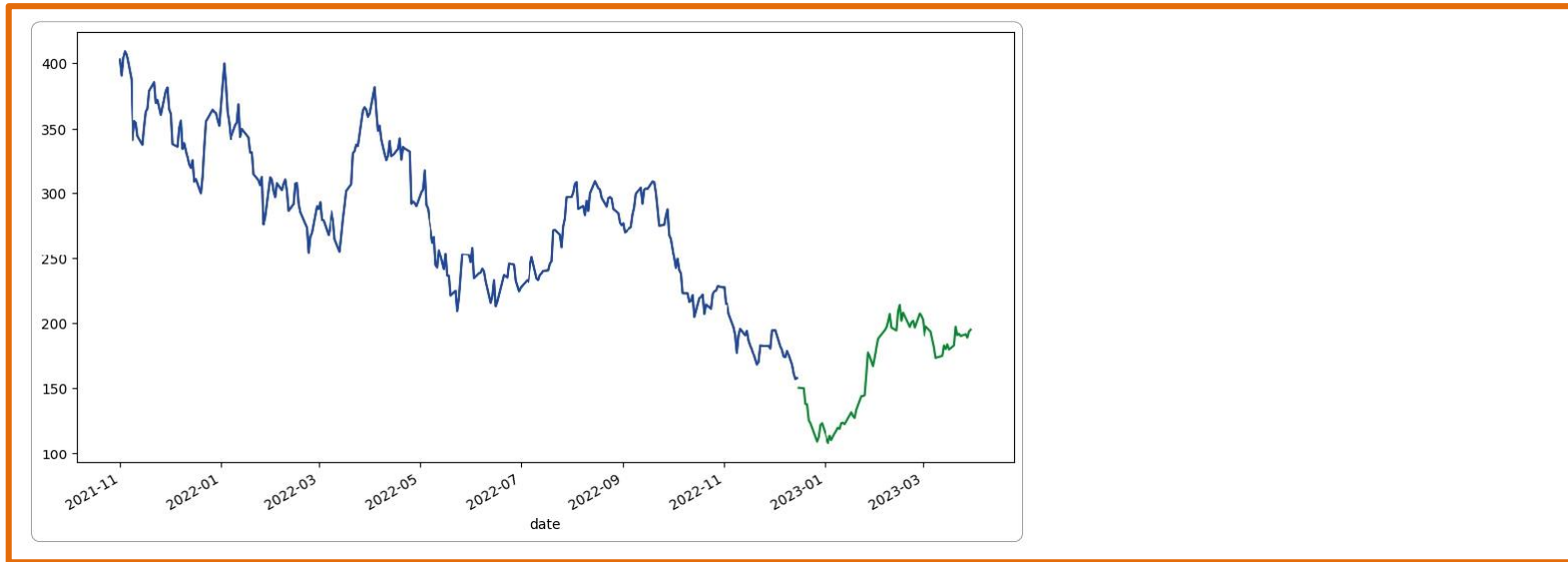
# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

## 실행결과



- 시간 순서대로 나열한 데이터에서 오래된 순으로 80%를 학습 데이터로 정함.
- 학습 데이터 및 테스트 데이터의 date 열을 인덱스로 지정.
- 2021년 11월 1일~ 2022년 12월 15일의 데이터를 학습하고 2022년 12월 16일~ 2023년 3월 30일 주가를 예측.



# 시계열 예측 모델링 기법

## ● 통계 기반 예측

### ● ARIMA 모델을 이용하는 예측

- 이미 테슬라 데이터가 비정상성 데이터임을 알아냄.
- 따라서 ARIMA 모델을 사용할 때 차분 계수를 입력하고 데이터가 정상성을 띠게 하여 예측을 진행.

#### 차분 횟수 확인

```
!pip install pmdarima
from pmdarima.arima import ndiffs, nsdiffs
print(f"최적의 차분 횟수 (ADF): {ndiffs(df_tsla_train, test='adf')}")
print(f"최적의 차분 횟수 (KPSS): {ndiffs(df_tsla_train, test='kpss')}")
print(f"최적의 차분 횟수 (PP): {ndiffs(df_tsla_train, test='pp')}")
```

최적의 차분 횟수 (ADF): 1

최적의 차분 횟수 (KPSS): 1

최적의 차분 횟수 (PP): 1



# 시계열 예측 모델링 기법

## ● 통계 기반 예측

### ● ARIMA 모델을 이용하는 예측

- 데이터에 계절성이 존재하는지 간단히 확인해보기.

#### 계절성 확인

```
print(f"최적의 차분 계수 (OSCB): {nsdiffs(df_tsla_train, m=12, test='ocsb')}")  
print(f"최적의 차분 계수 (CH): {nsdiffs(df_tsla_train, m=12, test='ch')}")
```

최적의 차분 계수 (OSCB): 0

최적의 차분 계수 (CH): 0

- m=12로 지정하여 OCSB 알고리즘으로 검정한 결과 최적의 차분 계수가 0이므로 계절성이 없음.
- CH 알고리즘으로 검정한 결과도 마찬가지로 계절성이 없으므로 테슬라 데이터에서는 계절성이 검출되지 않음.



# 시계열 예측 모델링 기법

- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 데이터의 세 가지 패턴이 제거되었으니 ARIMA로 최적 모델을 탐색할 수 있음.

## ARIMA 모형 탐색

```
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm

n_diffs = 1
model_fit = pm.auto_arima(
    y=df_tsla_train['value'],
    d=n_diffs,
    start_p=0, max_p=2,
    start_q=0, max_q=2,
    m=1, seasonal=False, #데이터에 계절성이 없음
    stepwise=True,
    trace=True)
print(model_fit.summary( ))
```

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

실행결과

```
SARIMAX Results
=====
Dep. Variable:                y                No. Observations: 284
Model:                SARIMAX(0, 1, 0)        Log Likelihood -1097.679
Date:                Tue, 02 May 2023        AIC 2197.357
Time:                09:10:30                BIC 2201.003
Sample:                0                    HQIC 2198.819
                                - 284
Covariance Type:                opg
=====
              coef      std err      z      P>|z|      [0.025 0.975]
-----
sigma2          136.9497      8.602    15.920    0.000      120.089 153.810
=====
Ljung-Box (L1) (Q):                0.61      Jarque-Bera (JB): 29.69
Prob(Q):                0.43      Prob(JB): 0.00
Heteroskedasticity (H):            0.34      Skew: -0.17
Prob(H) (two-sided): 0            .00      Kurtosis: 4.55
=====
```

- ARIMA(0, 1, 0) 모델이 주어진 데이터와 가장 적합한 모델로 판별되었음.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

ARIMA 모형 탐색

```
tsla_pred= model_fit.predict(n_periods=len(df_tsla_test))
df_tsla_pred = pd.DataFrame(tsla_pred)

result = pd.DataFrame(df_tsla_test['value'].values,\
                      index=df_tsla_test.index, columns=['value'])
result
```

- 최적으로 생성된 ARIMA모형으로 2022년 12월 16일부터 마지막 날짜까지의 주가를 예측.
- 인덱스를 초기화하고 가공하여 재지정.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

실행결과

value	
date	
2022-12-16	157.669998
2022-12-19	157.669998
2022-12-20	157.669998
2022-12-21	157.669998
2022-12-22	157.669998
...	...
2023-03-24	157.669998
2023-03-27	157.669998
2023-03-28	157.669998
2023-03-29	157.669998
2023-03-30	157.669998

- 모든 날짜에 같은 주가가 예측되었음.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 예측한 주가를 그래프로도 확인.

## 예측 주가 시각화

```
fig, axes = plt.subplots(1, 1, figsize=(12, 4))
plt.plot(df_tsla_train, label='Train')           # 훈련 데이터
plt.plot(df_tsla_test, label='Test')            # 테스트 데이터
plt.plot(result, label='Prediction')            # 예측 데이터

plt.legend( )
plt.show( )
```



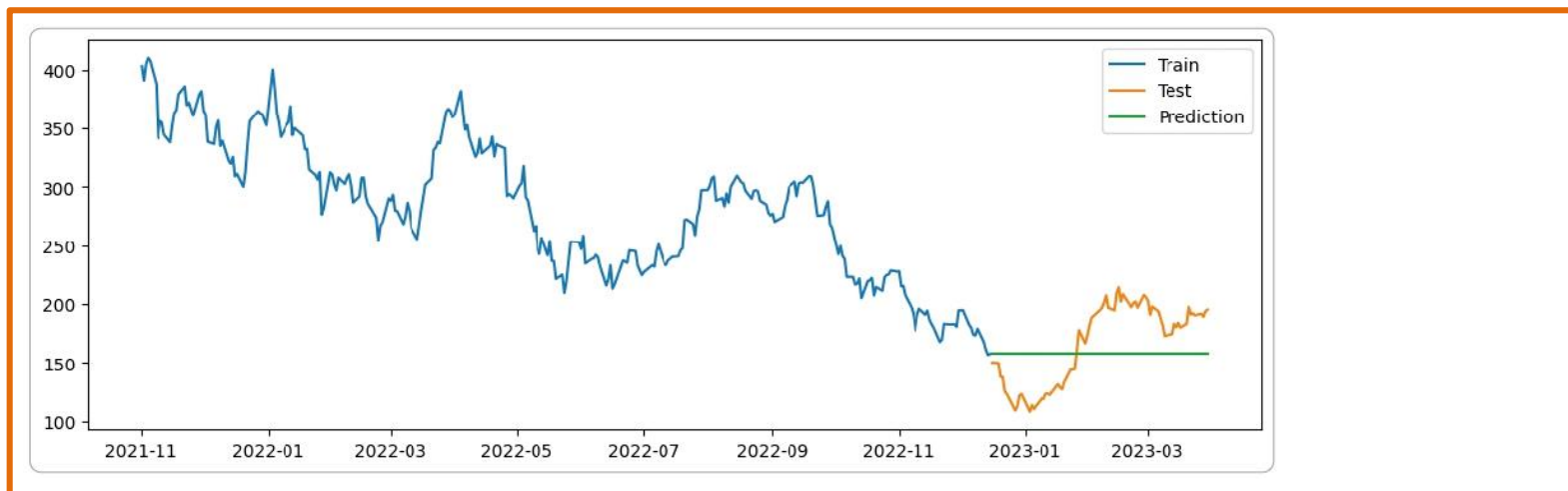
# 시계열 예측 모델링 기법



## ● 통계 기반 예측

### ● ARIMA 모델을 이용하는 예측

#### 실행결과



- result 데이터의 녹색 선이 예측 기간 내내 같은 값을 나타냄.
- 이렇게 예측한 이유는  $ARIMA(0, 1, 0)$  모형의 특성 때문.
  - 일정 기간의 주가를 한꺼번에 예측할 수 없다면 한 번에 단 하루씩 주가를 예측하면 됨.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 다음 날짜를 예측할 때 전날의 예측된 주가를 반영하는 방법.

주가를 하루씩 예측

```
def each_step_prediction():  
    pred_next = model_fit.predict(n_periods=1)  
    return pred_next.tolist()[0] # 리스트 형태로 반환하기  
  
pred_steps = [ ]  
for new_inst in df_tsla_test['value']:  
    pred = each_step_prediction()  
    pred_steps.append(pred)  
    model_fit.update(new_inst)
```

- 다음 날의 주가를 예측하는 each\_step\_prediction( ) 함수를 정의.
      - 예측이 이루어질 때마다 update( ) 함수로 모델을 업데이트.

# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

- 주가를 하루씩 예측하는 방법이 효과가 있는지 예측 주가를 다시 시각화하여 확인.

하루씩 예측한 주가 시각화

```
fig, axes = plt.subplots(1, 1, figsize=(12, 4))
plt.plot(df_tsla_train.index, df_tsla_train['value'], label='Train')
plt.plot(df_tsla_test.index, df_tsla_test['value'], label='Test')
plt.plot(df_tsla_test.index, pred_steps, label='Prediction')

plt.legend( )
plt.show( )
```

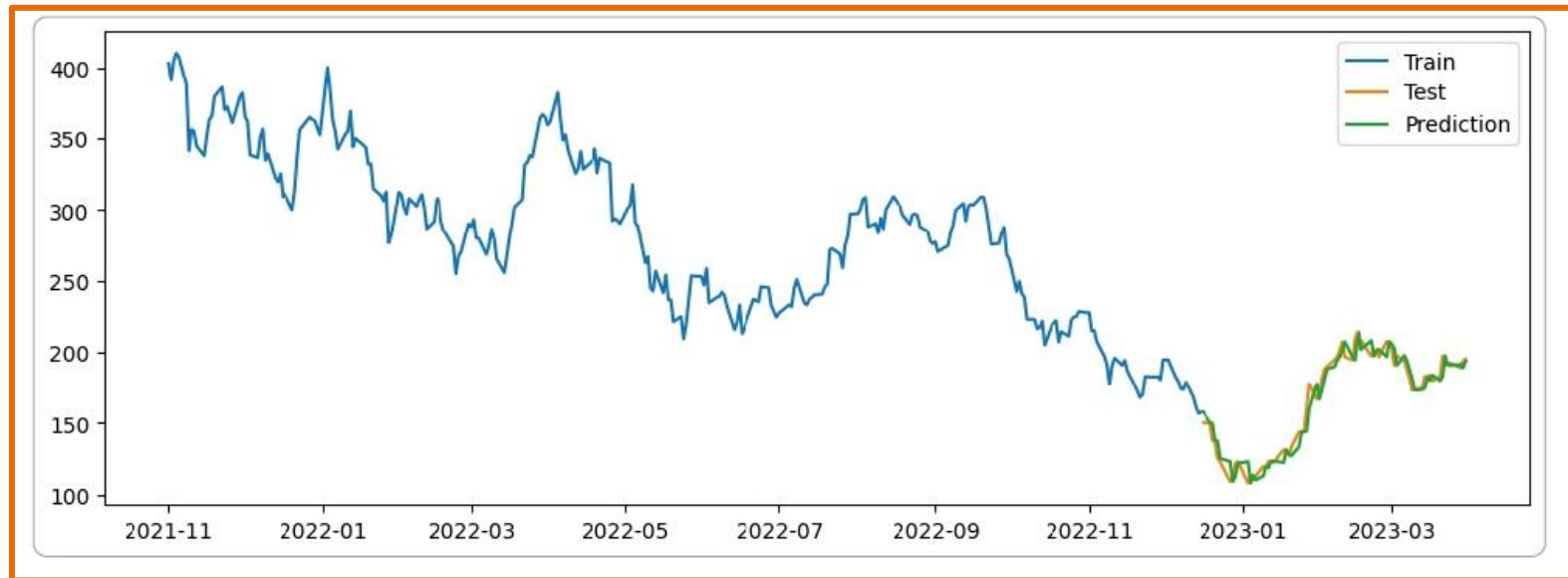
# 시계열 예측 모델링 기법



- 통계 기반 예측

- ARIMA 모델을 이용하는 예측

실행결과



- 하루 주가를 예측하고 모델에 업데이트하여 다음날의 주가를 예측하는 전략이 성공했음.

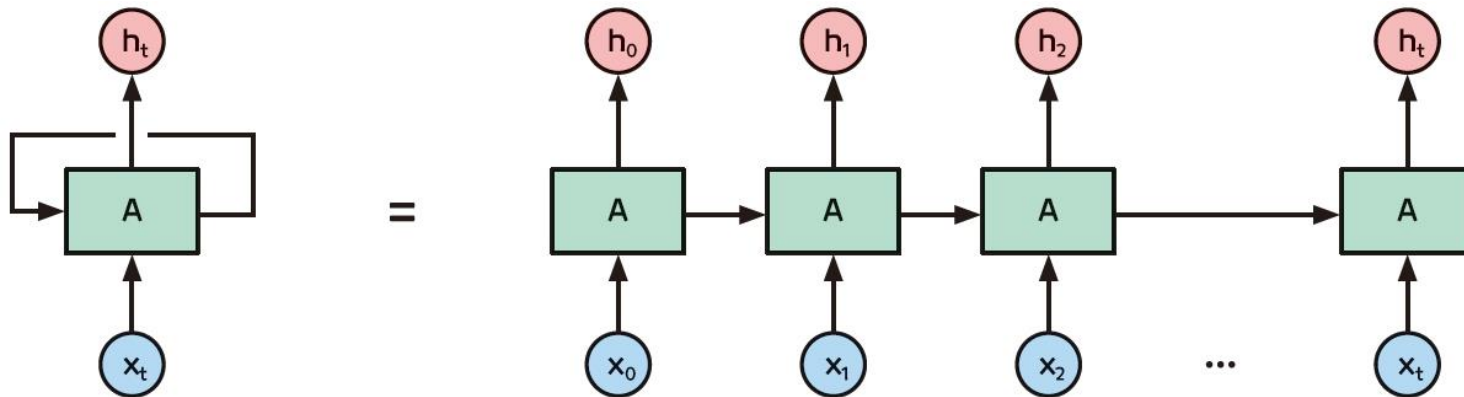
# 인공지능 시계열 예측



- 인공지능 시계열 예측 모델 심화 분석: 인공신경망을 활용한 주가 예측

- RNN의 특징

- RNN(Recurrent Neural Network, 순환신경망)은 가장 대중적인 딥러닝 알고리즘.
- 현재 단계의 결과를 다음 단계의 입력으로 순차적으로(Sequentially) 전달.
- 단어 간의 관계나 문맥의 의미의 이해가 필요한 자연어 처리, 음성인식, 번역기에 폭넓게 사용됨.



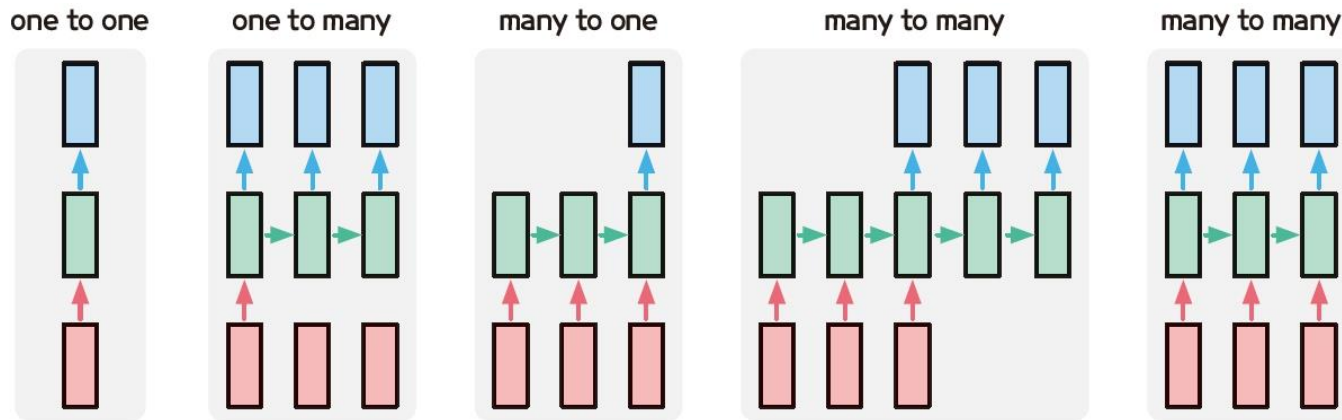


# 인공지능 시계열 예측

## ● 인공지능 시계열 예측 모델 심화 분석 : 인공신경망을 활용한 주가 예측

### ● RNN의 특징

- RNN에서는 일반적으로 LSTM(Long short-term memory) 활성화 함수를 사용함.
- 이전 기간에서의 주가 변동을 바탕으로 미래의 가격을 예측하는 문제에 RNN을 사용해도 됨.
- RNN은 모형의 구성에 따라 다양한 용도로 활용할 수 있음.
  - one to many: 그림을 한 장 입력 받아 그림을 설명하는 문장을 여러 개 생성하는 데 활용.
  - many to many: 여러 단어로 구성된 영어 문장을 한국어로 변환하는 번역기에 활용.





# 인공지능 시계열 예측

- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측
  - RNN을 활용하는 예측

## 테슬라 주가 데이터 로드

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout

tsla = yf.download('TSLA', start='2021-11-01', end='2023-03-31')
df_tsla = pd.DataFrame(tsla['Close'])

df_tsla = df_tsla.reset_index( )
df_tsla.columns = ['date', 'value']
df_tsla['date'] = pd.to_datetime(df_tsla['date'])
df_tsla.set_index('date', inplace=True)
```

- 주가정보의 Close 열과 date 열만 데이터프레임으로 저장.



# 인공지능 시계열 예측

- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측
  - RNN을 활용하는 예측
    - 데이터를 정규화하면 RNN 학습의 성능이 좋아짐. 주가를 0에서 1사이로 스케일링.

테스트 데이터 분할 및 데이터 스케일링

```
df_tsla.reset_index( )
dataset_tsla = df_tsla.values

#데이터 분할하기
df_tsla_train = dataset_tsla[:int(0.8*len(dataset_tsla)), :]
df_tsla_test = dataset_tsla[int(0.8*len(dataset_tsla)):, :]

#데이터 스케일링하기
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(dataset_tsla)
```

- 사이킷런 라이브러리의 MinMaxScaler( ) 함수로 스케일링.





- RNN을 활용하는 예측

```
x_train_data,y_train_data=[ ],[ ]
```

```
for i in range(28,len(df_tsla_train)):
    x_train_data.append(scaled_data[i-28:i,0])
    y_train_data.append(scaled_data[i,0])
```

[illegible]



# 인공지능 시계열 예측

- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측
  - RNN을 활용하는 예측

## RNN 모형 설계 및 파라미터 정의

```
lstm_tsla = Sequential( )

lstm_tsla.add(LSTM(units=28, return_sequences=True,\
                    input_shape=(x_train_data.shape[1],1)))
lstm_tsla.add(LSTM(units=28))
lstm_tsla.add(Dense(1))

#데이터 재가공하기
inputs_data = df_tsla[len(df_tsla) - len(df_tsla_test)-28:].values
inputs_data = inputs_data.reshape(-1,1)
inputs_data = scaler.transform(inputs_data)

#모형의 학습 방법 설정하여 학습 진행하기
lstm_tsla.compile(loss='mean_squared_error', optimizer='adam')
lstm_tsla.fit(x_train_data, y_train_data, epochs=100, batch_size=1, verbose=2)
```



# 인공지능 시계열 예측

- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측
  - RNN을 활용하는 예측

## 실행결과

```
Epoch 1/100
256/256 - 9s - loss: 0.0141 - 9s/epoch - 37ms/step
.....
(중략)
.....
Epoch 97/100
256/256 - 3s - loss: 0.0015 - 3/epoch - 13ms/step
Epoch 98/100
256/256 - 3s - loss: 0.0018 - 4/epoch - 15ms/step
Epoch 99/100
256/256 - 4s - loss: 0.0016 - 4/epoch - 14ms/step
Epoch 100/100
256/256 - 3s - loss: 0.0015 - 3/epoch - 13ms/step
```

- 학습을 진행할수록 loss 값이 작아지므로 학습이 잘 이루어지고 있다는 의미.



# 인공지능 시계열 예측

- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측

- RNN을 활용하는 예측

학습이 완료된 RNN 모형으로 주가 예측

```
X_test = [ ]
for i in range(28, inputs_data.shape[0]):
    X_test.append(inputs_data[i-28:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
predicted_value = lstm_tsla.predict(X_test)
predicted_value = scaler.inverse_transform(predicted_value)
```

- 28일씩 묶어낸 데이터를 predict( ) 함수에 입력하여 주가를 예측.
- 변수 predicted\_value에 일자별로 예측한 결과를 연속하여 담음.
- scaler.inverse\_transform( ) 함수를 사용하여 결과를 스케일링 이전의 범위로 되돌림.

# 인공지능 시계열 예측



- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측
  - RNN을 활용하는 예측

주가 예측 결과 시각화

```
df_tsla_train_vis = tsla[:284]
df_tsla_test_vis = tsla[284:]

df_tsla_test_vis['Predictions']=predicted_value
plt.plot(df_tsla_train_vis["Close"])
plt.plot(df_tsla_test_vis[['Close','Predictions']])
```

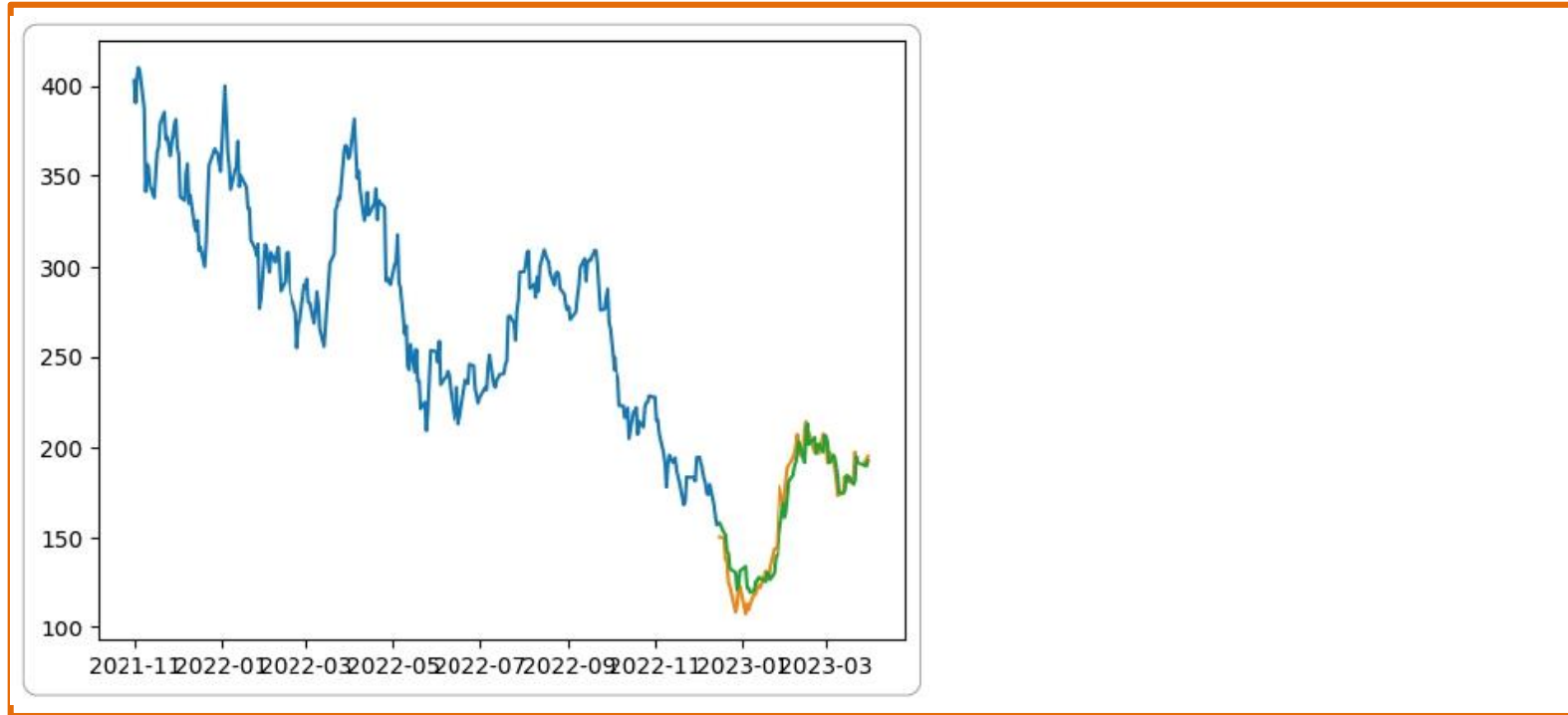
# 인공지능 시계열 예측



- 인공지능 시계열 예측 모델 심화분석: 인공신경망을 활용한 주가 예측

- RNN을 활용하는 예측

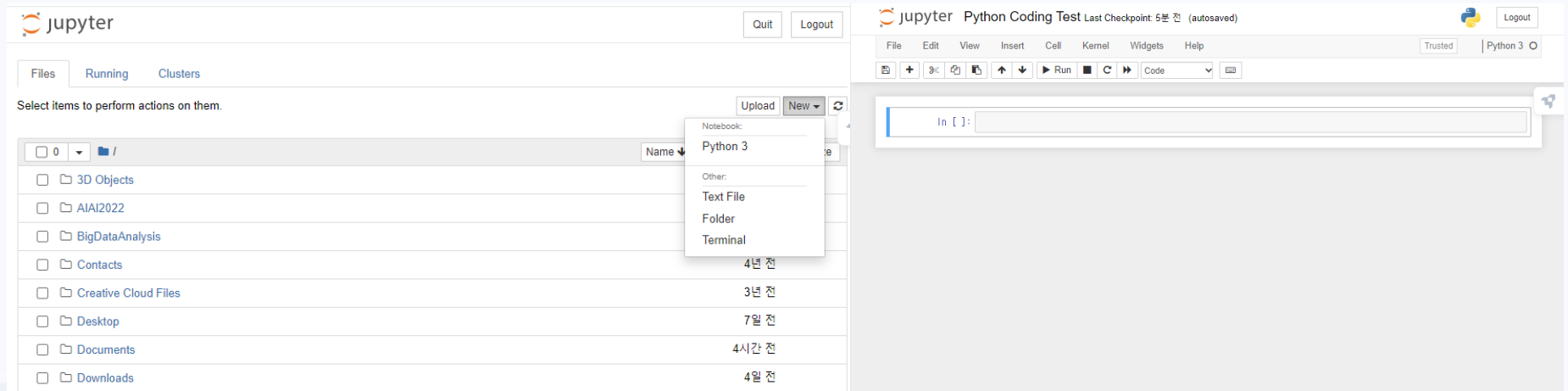
실행결과



- 학습 데이터로 얻은 예측 주가가 테스트 데이터와 매우 유사함.

# 학습활동: Python 코딩 실습

## Python 코드 소스 파일 작성 실습



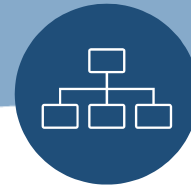
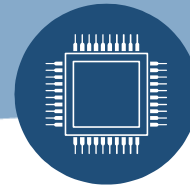
The screenshot displays the JupyterLab interface. On the left, the 'Files' tab is active, showing a file browser with a list of folders and files. A context menu is open over the 'Python 3' notebook, offering options: 'Notebook: Python 3', 'Other: Text File', 'Folder', and 'Terminal'. The main area on the right shows a Jupyter notebook titled 'Python Coding Test Last Checkpoint: 5분 전 (autosaved)' with a single code cell containing the prompt 'In [ ]: '.

Name	Time
Python 3	4년 전
3D Objects	3년 전
AIAI2022	7일 전
BigDataAnalysis	4시간 전
Contacts	4일 전
Creative Cloud Files	
Desktop	
Documents	
Downloads	



# Thank you!

See you next time.



담당교수 : 유 현 주  
comjoo@uok.ac.kr

