Kaggle competition을 통한

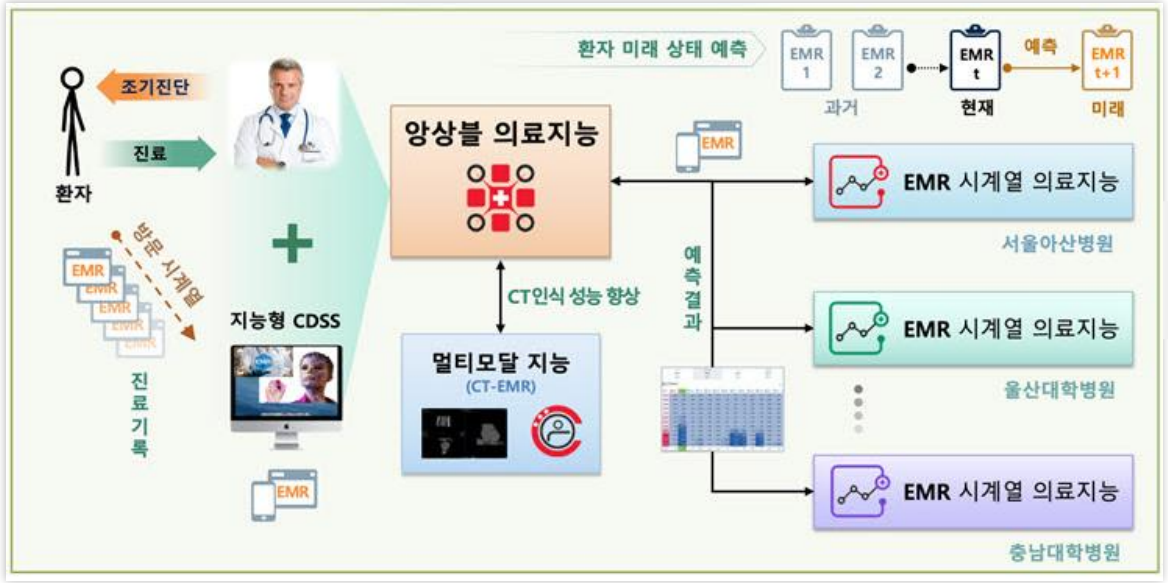
시계열 데이터 분류 모델 연구

Team. 강지원 김영재 오승은 엄태현

CONTENTS

시계열 분석의 중요성

- IoT와 SNS 등으로 인해 다양한 센서로부터 측정된 데이터를 저장하기 위한 기술의 발전으로 데이터의 규모나 복잡성이 커지고 있다
- **IoT와 다양한 센서로부터 산업에서 측정되는 상당수의 데이터가 시계열 특성**을 지니고 있고, 이러한 실시간 데이터를 분석하여 유의미한 미래 예측 분석을 하는 것이 중요해지고 있다.



시계열 EMR 의료지능을 활용하여
'닥터AI ' 의료 인공지능 개발에 성공

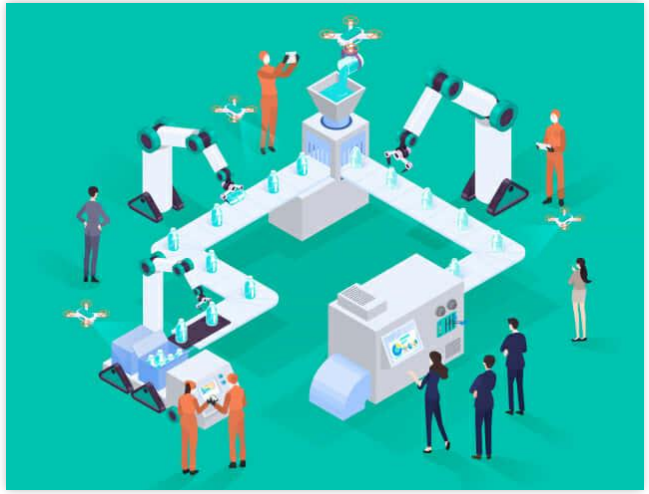
* 닥터 AI : 여러 병원에 구축된 의료지능을 통합해 환자의 현재 상태를 정밀하게 분석하고 미래건강을 예측하는 인공지능 주치의

시계열 분석 활용사례

시계열 데이터는 최근 *금융, 의료, 기상 등 많은 분야*에서 다양하게 활용되고 있다

시계열 데이터를 활용하여
고객자산보호 서비스를 출시한 NH증권

시계열 데이터 분석을 통해
수율을 높이는 스마트 팩토리



시계열 공간정보를 이용한
위성 기반의 재난관리 서비스 출시

긴급 공간정보는 재난 현장을 직접 방문하지 않더라도
재난 및 피해 상황을 확인할 수 있도록 하는 서비스

프로젝트 목표

STEP 1.

시계열 데이터에 대한
전반적인 특징을 이해하고,

STEP 2.

캐글 실습을 통해 시계열 분류에
사용되는 다양한 알고리즘을 비교하여

STEP 3.

시계열 데이터에 가장
적합한 모델을 탐색한다.

시계열 데이터란

: 어떤 현상에 대하여 시간의 흐름에 따라 기록 되어 미래의 변화에 대한 추세를 나타내는 데이터를 총칭
시간의 흐름에 따라 순차적으로 발생한 관측치의 집합 이며, 이때 발생한 연속적인 관측치는 서로 관련 이 있다.

시계열 데이터 분석

시계열 데이터에서 의미 있는 요약과 통계 정보를 추출하기 위한 노력
'과거가 미래에 어떤 영향을 주는가?'와 같은 인과관계를 다루는 질문과 그에 대답하는 과정

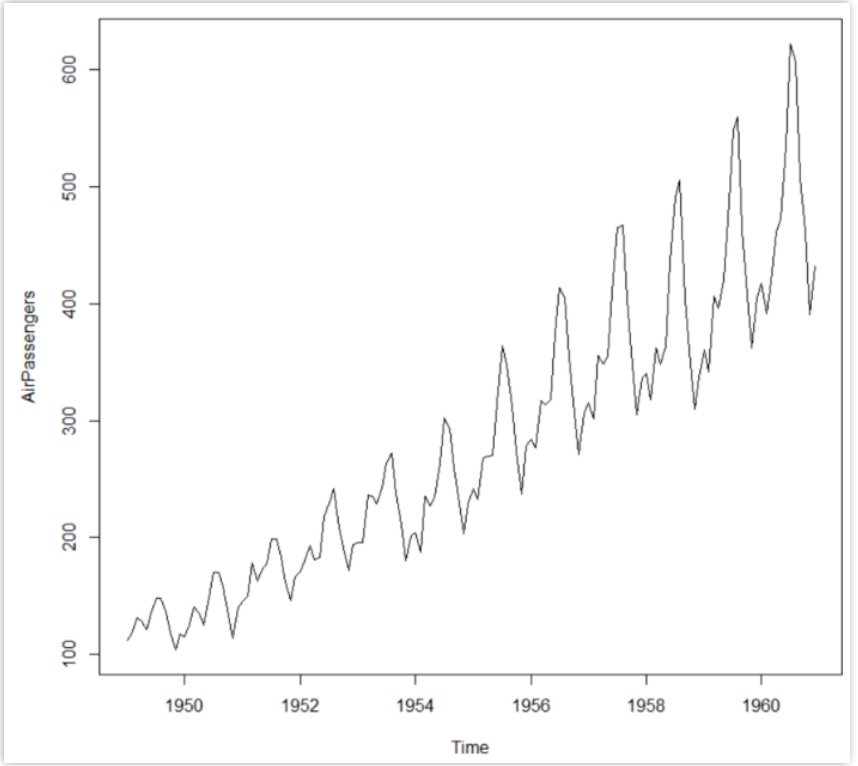
의학 ,기상학, 경제학, 천문학 등 다양한 학문 분야가 시계열 데이터를 분석하는 방법에 영향을 끼쳤다.

시계열 데이터는 여러 측면에서 높은 계산 능력을 요구하며,
오늘날 **웨어러블 컴퓨터, 머신 러닝, GPU** 등의 기술이 데이터의 양과 품질 면에서 혁신을 이끌고 있다.

시계열 데이터의 종류

정상 시계열 (Stationary Time Series)
: **평균을 중심으로 일정한 변동폭**을 갖는 시계열

비정상 시계열 (Non-stationary Time Series)
: 평균 또는 분산이 **시간의 흐름에 따라 증가 혹은 감소하거나**
주기적인 계절성 또는 순환성을 갖는 시계열



R에 내장된 데이터셋 AirPassengers은
전형적인 비정상 시계열의 특징을 가진다

→ 대부분의 시계열 데이터는 비정상 시계열이기 때문에
분석이 쉬운 정상 시계열로 변환 하여 사용하는 것이 좋다.

→ 평균이 일정하지 않은 시계열은 **차분(difference)**을 통해,
분산이 일정하지 않은 시계열은 **로그/제곱근 변환(transformation)**을 통해 정상화할 수 있다.

시계열 데이터의 구성요소

4가지의 변동요인으로 구분

추세변동 (Trend Variation)

: 시간의 흐름에 따른 시계열 자료들의 **상승 또는 하강 경향**의 상태

계절변동 (Seasonal Variation)

: 시계열 자료에서 특정한 패턴을 갖고 반복적으로 나타나는 **주기적인 변동**

순환변동 (Cyclical Variation)

: 수년 간의 간격을 두고 상승과 하락이 주기적으로 나타나는 **장기적인 변동**

불규칙변동 (Irregular Variation)

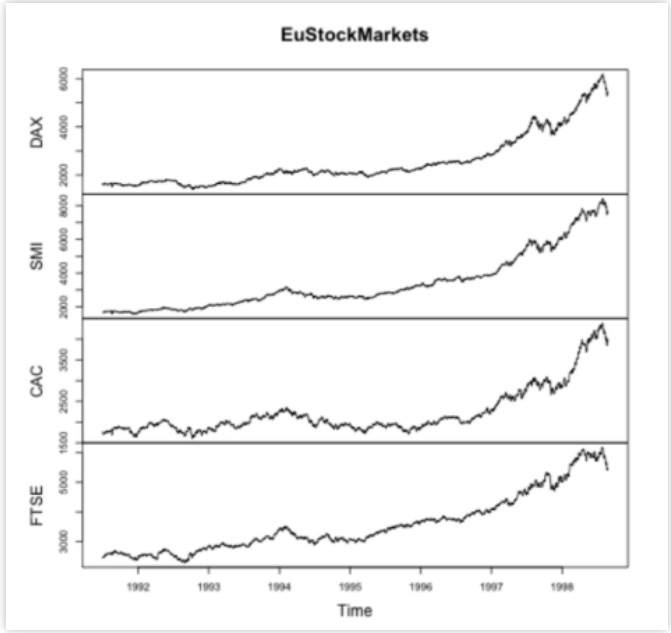
: 명확하게 설명할 수 없는 불규칙적인 요인에 의해 발생되는 **우연적인 변동**

시계열 데이터를 *각 요인으로 분해하여 비교* 함으로써 미래에 대한 예측 및 분류에 활용할 수 있다.

일반적인 EDA 방식

선 그래프

R, Python 등의 plot() 함수를 통해
시간의 흐름에 따른 데이터의 변화 양상을 확인하고
추세변동, 계절변동 등의 변동 요인을 파악할 수 있다.

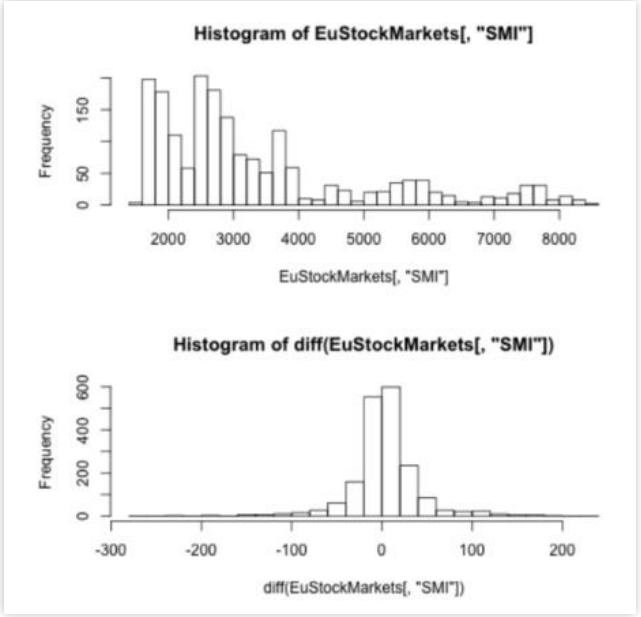


R에 내장된 데이터셋
EuStockMarkets 각 변수의 선 그래프

히스토그램

R, Python 등의 hist() 함수를 통해
데이터의 전체적인 분포 양상을 확인할 수 있다.

이때, 시계열에 내재된 추세를 제거하기 위해
시간상 인접한 데이터의 차이(diff() 함수)를 보면
정규분포에 가까운 형태의 유용한 정보를 얻을 수 있다.



R에 내장된 데이터셋 EuStockMarkets
SMI 변수의 차분 전후 히스토그램

일반적인 EDA 방식

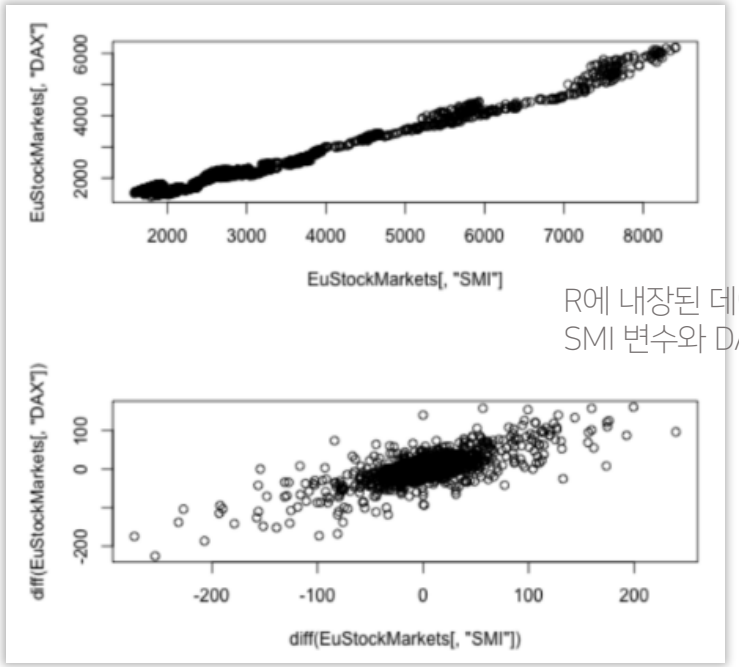
산점도

변수 간의 관계를 알아보는 가장 전통적인 방법으로,
선 그래프나 히스토그램을 통해 추세를 발견한 경우
데이터의 실제 값 그 자체에 대한 산점도보다
차분 값에 대한 산점도가 더 유용한 정보를 제공 한다

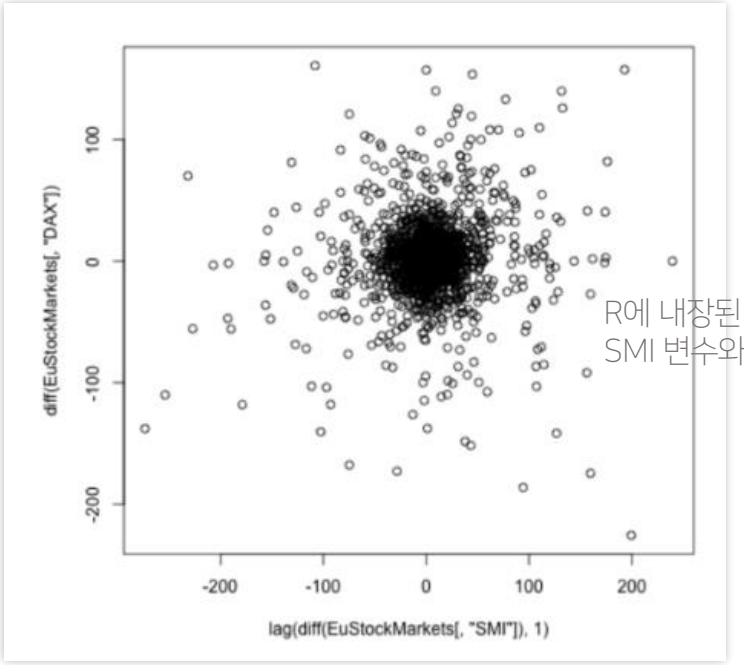
앞서 예시로 제시한 데이터셋의 경우에는
두 변수가 동일한 시점의 값을 취하기 때문에

한 변수에만 R, Python 등의 lag() 함수를 적용하여
동일한 시점에서의 상관관계를 배제 하고자 했다.

그 결과, 두 변수의 상관관계가 사라진 것을 알 수 있다.



R에 내장된 데이터셋 EuStockMarkets
SMI 변수와 DAX 변수의 산점도 (실제 값, 차분 값)



R에 내장된 데이터셋 EuStockMarkets
SMI 변수와 DAX 변수의 산점도 (Lag() 함수 적용)

시계열에 특화된 EDA 방식

정상성 검정 ** 계열의 평균 변화 여부에 중점을 둔 검정*

ADF 검정 : 시계열의 정상성 문제를 가장 보편적으로 평가하는 평가 지표

- 시계열이 **정상성을 가지지 않는다**(시계열에 단위근이 존재한다)는 귀무가설을 상정한다.
- 검정 결과 p-value가 설정한 **신뢰수준보다 작을 때** 해당 데이터는 정상성을 가진다고 할 수 있다.
- Python에서는 statsmodels.tsa.stattools 패키지의 adfuller 모듈을 사용하여 실시한다.

KPSS 검정 : ADF 검정과 함께 정상성을 판단하는 일반적인 평가 지표

- ADF 검정과는 정반대로 시계열이 **정상성을 가진다**는 귀무가설을 상정한다.
- 검정 결과 p-value가 설정한 **신뢰수준보다 클 때** 해당 데이터는 정상성을 가진다고 할 수 있다.
- Python에서는 statsmodels.tsa.stattools 패키지의 kpss 모듈을 사용하여 ADF 검정을 실시한다.

시계열에 특화된 EDA 방식

비정상 시계열을 정상화하는 방법

* 비정상 시계열로 구축한 모델은 시간이 지날수록 모델의 편향과 오차가 달라지므로 신뢰하기 어렵다.
  따라서, 모델을 구축하기 전에 충분한 차분 및 변환 과정을 통해 시계열을 정상화하는 것이 바람직하다.

차분(differencing)

평균값이 일정하지 않은 시계열에서 연속된 값 사이의 차이를 계산함으로써 **평균 변화를 일정하게** 만드는 것

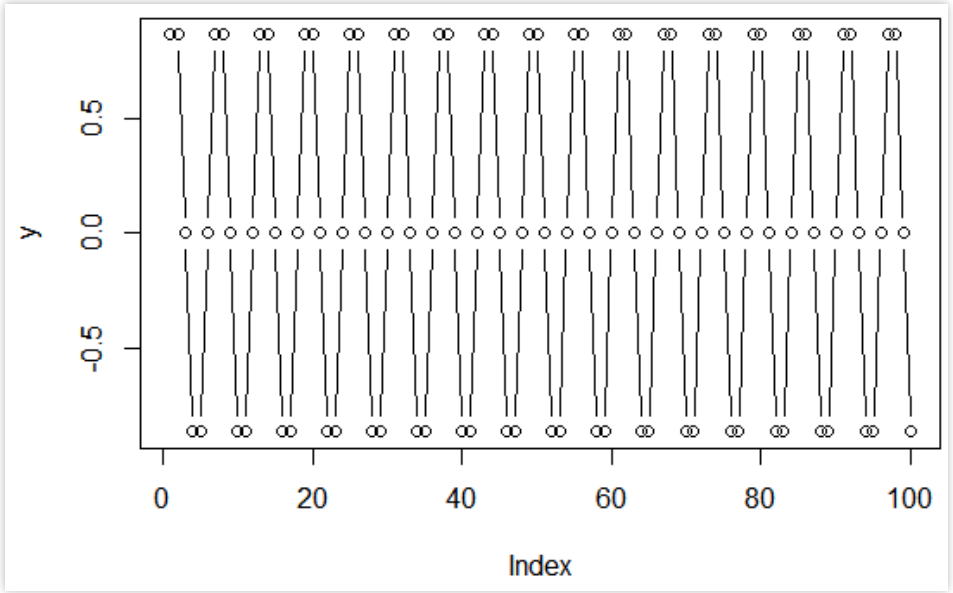
변환(transformation)

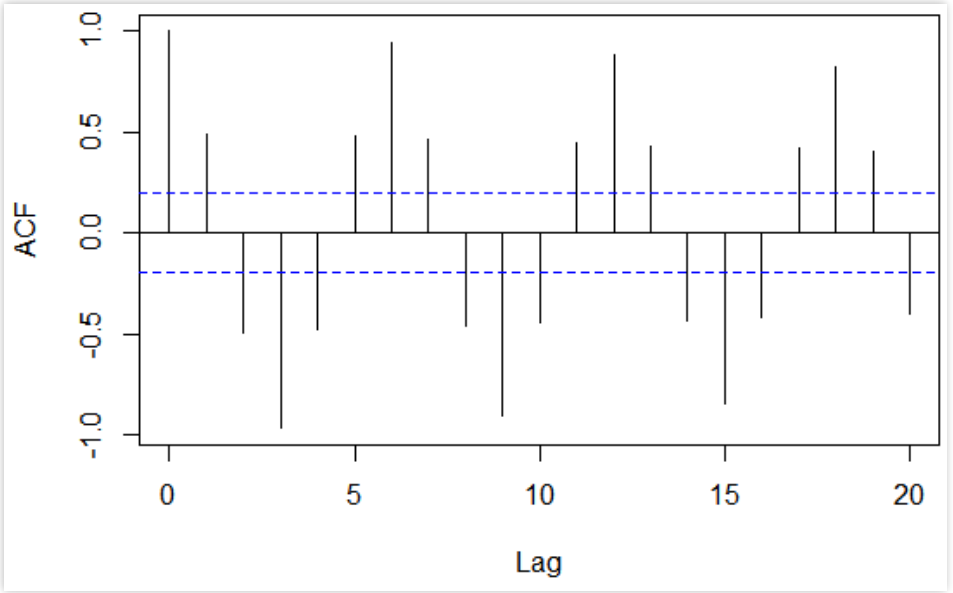
분산이 일정하지 않은 시계열에서 로그 또는 제곱근 변환을 통해 **분산 변화를 일정하게** 만드는 것

시계열에 특화된 EDA 방식

자기상관함수(ACF)

시차 k에서의 Y의 자기상관, 즉 Y와 LAG(Y, k)의 상관관계에 대한 그래프
-> t시점의 관측치가 z_t일 때, z_t와 z_{t+k}의 선형 상관관계를 의미한다.

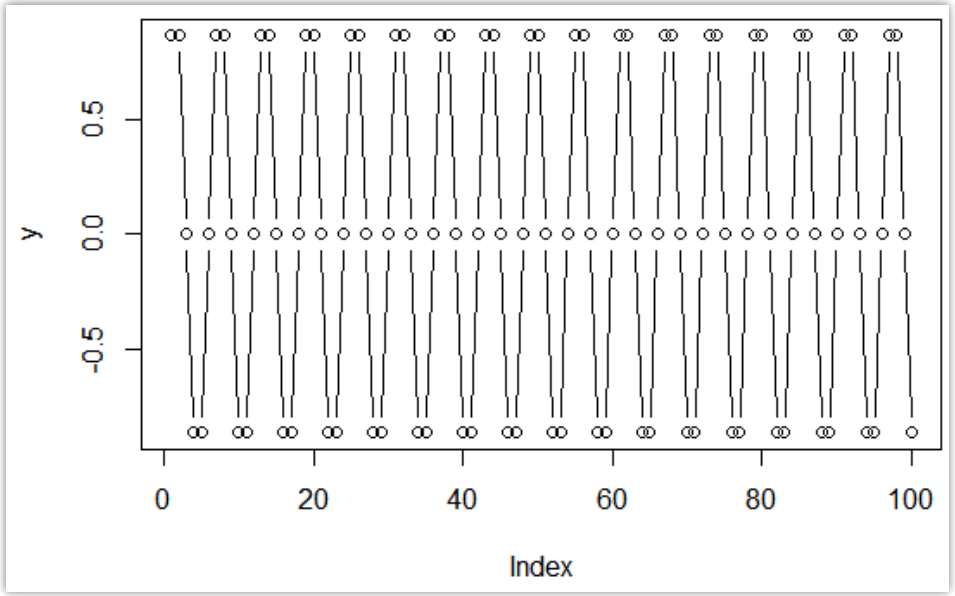


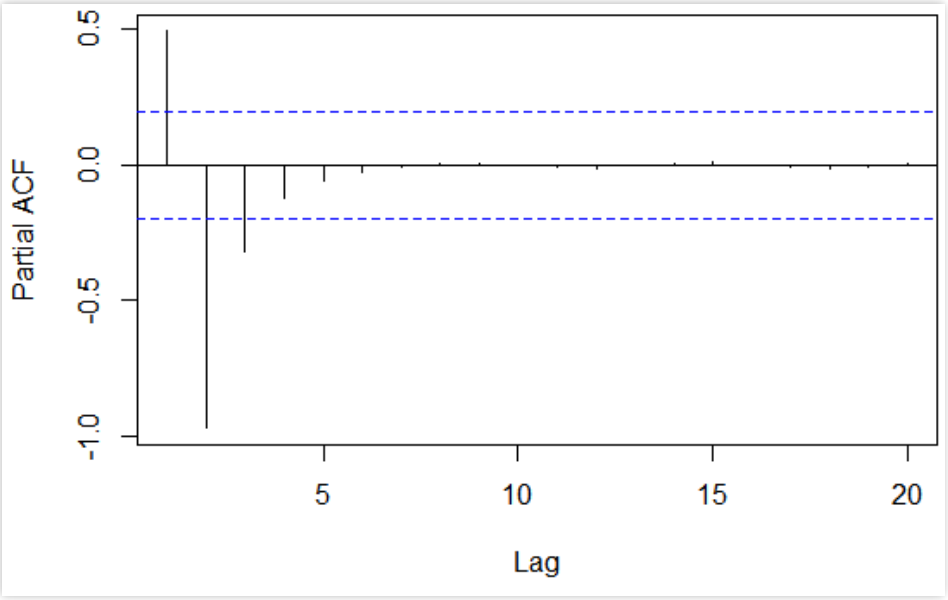


시계열에 특화된 EDA 방식

부분자기상관함수(PACF)

두 시점 사이에 영향을 주는 다른 요인을 제외한 시차 k에서의 자기상관계수에 대한 그래프
-> t시점의 관측치가 z_t일 때, z_{t+1} ~ z_{t+k-1}를 제거하고 구한 z_t와 z_{t+k}의 선형 상관관계를 의미한다.





시계열을 위한 특징 생성

: 시계열 데이터의 **가장 중요한 특성을 정량화** 하여 수치 및 범주형 레이블로 압축하는 방법을 찾는 과정
전체 자료에 대해 가능한 한 많은 정보를 **적은 수의 지표로 압축**하여 데이터의 중요한 정보를 식별하기 위함

시계열의 많은 특징이 정상성을 가정하기 때문에
시계열의 정상성을 먼저 확보하는 것이 중요하다.

데이터의 EDA를 통해 유용한 특징을 발굴하거나
도메인에 대한 배경지식을 활용하여 가설을 세우고
그에 맞는 특징을 생성하는 것이 바람직하다.

새로 생성한 특징이 분석의 타당성을 훼손하지 않는지,
유의미한 통찰을 이끌어낼 수 있는지,
과하게 생성한 특징으로 인해 분석 모델이 학습 데이터에
과대적합 되지 않는지 고려해야 한다.

일반적으로 사용되는 요약 통계

- 평균, 중앙값 등의 중심 경향성
- 최대값, 최소값, 범위, 분산, 표준편차 등의 산포도
- 시계열의 평활(Smoothing) 정도, 주기성 및 시계열 내부의 자기상관 정도

→ 대다수의 머신 러닝은 시계열 그 자체보다는 특징들의 집합으로 표현된 데이터에 적합하기 때문에, 적절하게 데이터를 압축한 특징을 사용함으로써 머신 러닝 모델의 성능을 향상시킬 수 있다.

→ 다양한 조건에서 측정된 데이터를 설명하고 유사성을 식별하기 위한 공통 지표를 제공할 수 있으며, 데이터를 보다 광범위하게 요약할수록 비교하기 어려운 데이터를 보다 쉽게 비교할 수 있다.

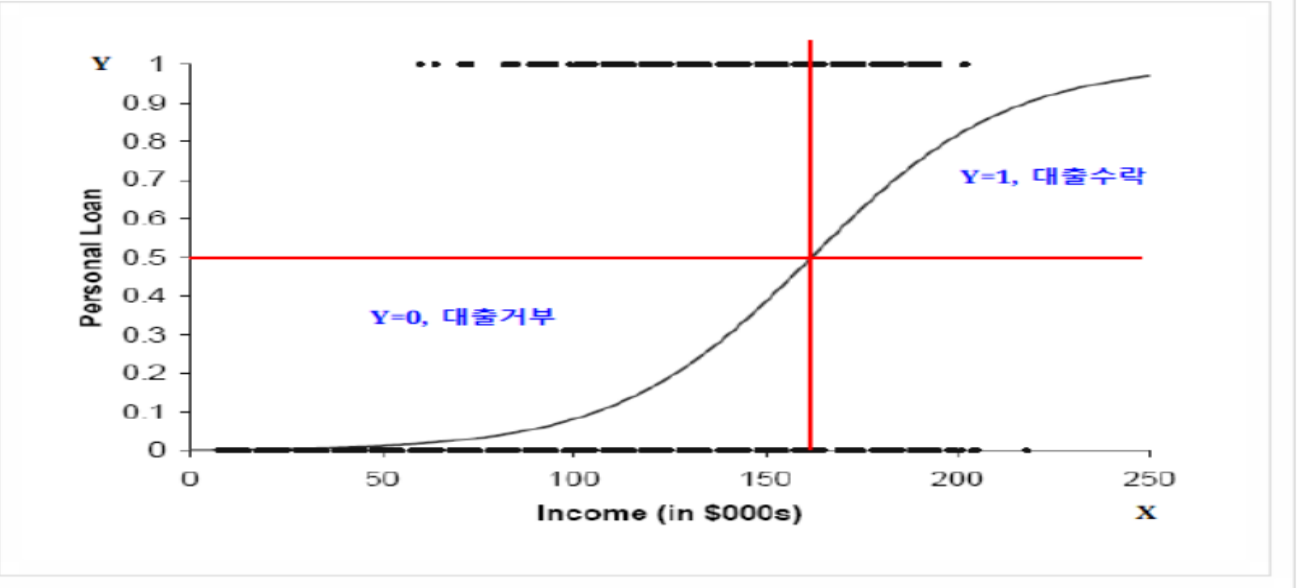
선형모델

Logistic Regression

선형 모델 : F(x)= W transpose X + b (선형조합, 각각의 항들이 더하기로 이루어진 조합)

Logistic Regression : 종속변수 (Y)의 값을 sigmoid 함수를 사용 하여 0과 1로 나누는 모델

<선형모델의 예>

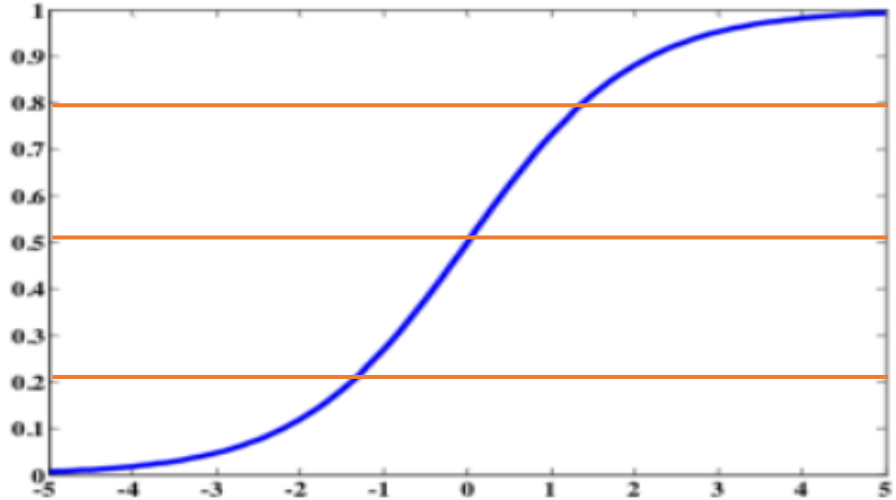


선형모델

Sigmoid 함수 : $y = \frac{1}{1 + e^{(-ax+b)}}$

기본 선형모델에 Sigmoid 함수를 활성화 시켜 줌으로써 , **결과값을 0에서 1사이로 출력**해준다.

이 함수를 통해서 나온 0과 1사이의 결과값에 *Cut off* 를 적용해주어, 어떤 수치를 기준으로 0과 1을 분리 할 것 인지 정해준다.



(1) 일반적으로 0.5를 기준으로 0.5 보다 크면 1, 0.5보다 작으면 0 으로 분리해준다.

(2) 불량 예측 등, 엄격하게 수치를 보아야 하는 경우 0.2 로 cut-off 를 지정 하여 준다.

(3) 성공 범주의 비중을 높게 잡을 때는 0.8로 잡아 주기도 한다.

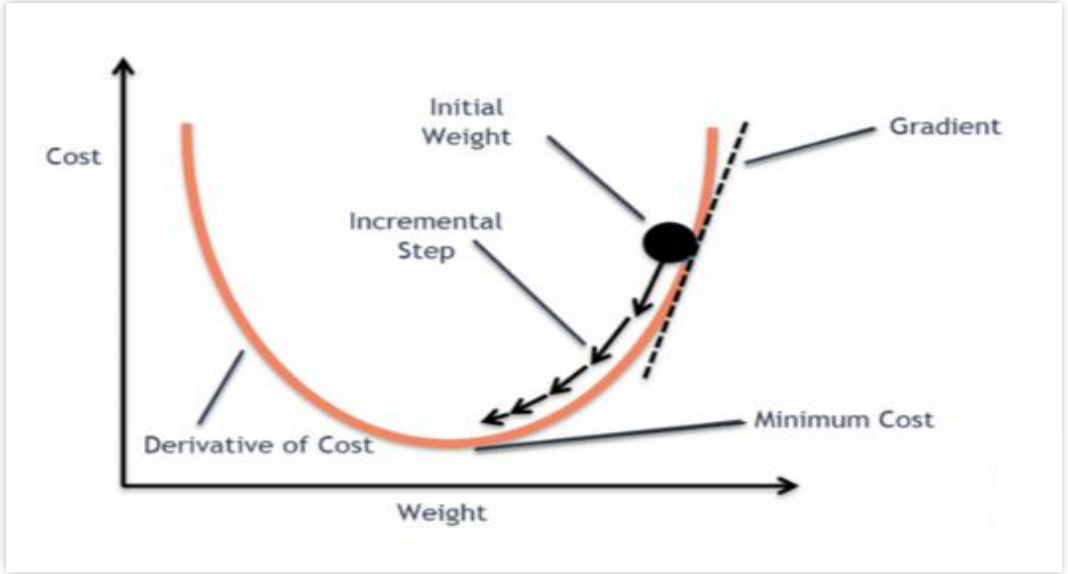
비선형모델

LGBM (Light Gradient Boosting Machine)

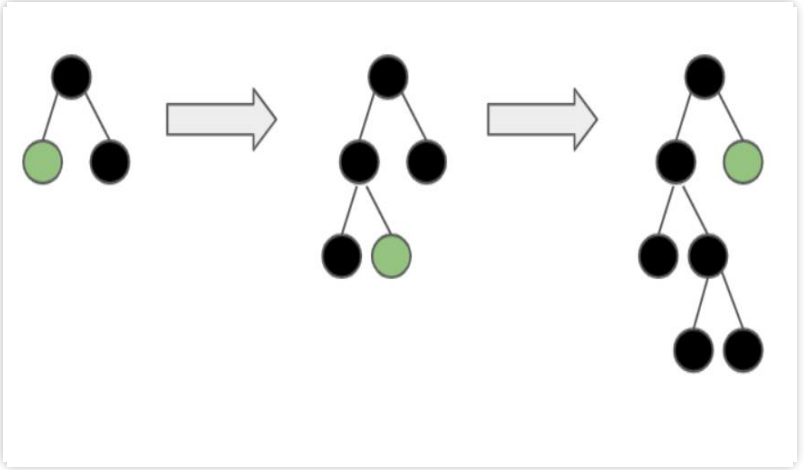
비선형 모델 : 데이터를 변형하더라도 파라미터를 선형 결합식으로 표현할 수 없는 모델을 말한다.

LGBM : 트리모형의 앙상블 기법으로써, 잔차를 기반으로 모델을 형성함으로써,
손실함수의 기울기가 0인 곳을 탐색하는 GBM 컨셉을 따르는 모델

<Gradient Boosting>



<LGBM의 트리 구조>



비선형모델

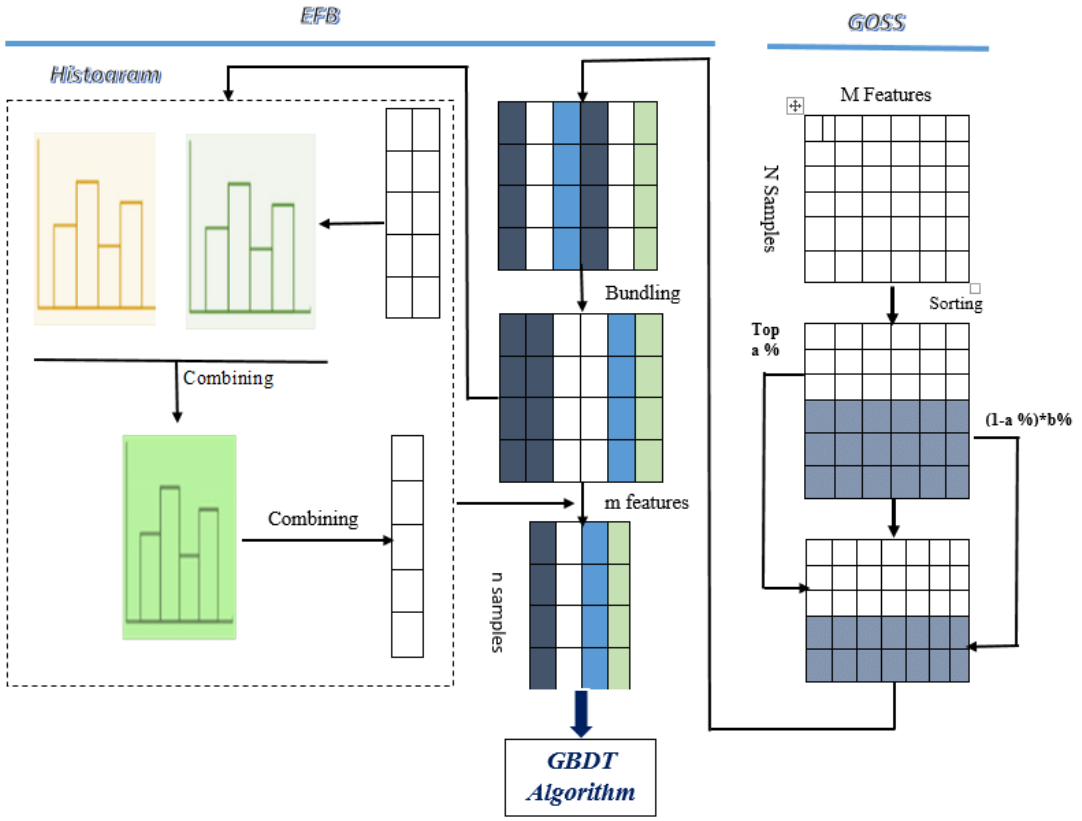
LGBM 의 알고리즘

GOSS(Gradient Based One side Sampling)

: 모델에 도움이 되는 기울기가 높은 부분을 유지하고,
모델이 크게 도움이 되지 않는 기울기가 작은 부분은 Drop 시킴으로써,
탐색 횟수, 탐색시간을 줄여주는 LGBM 의 알고리즘.

EFB(Exclusive Feature Bundling)

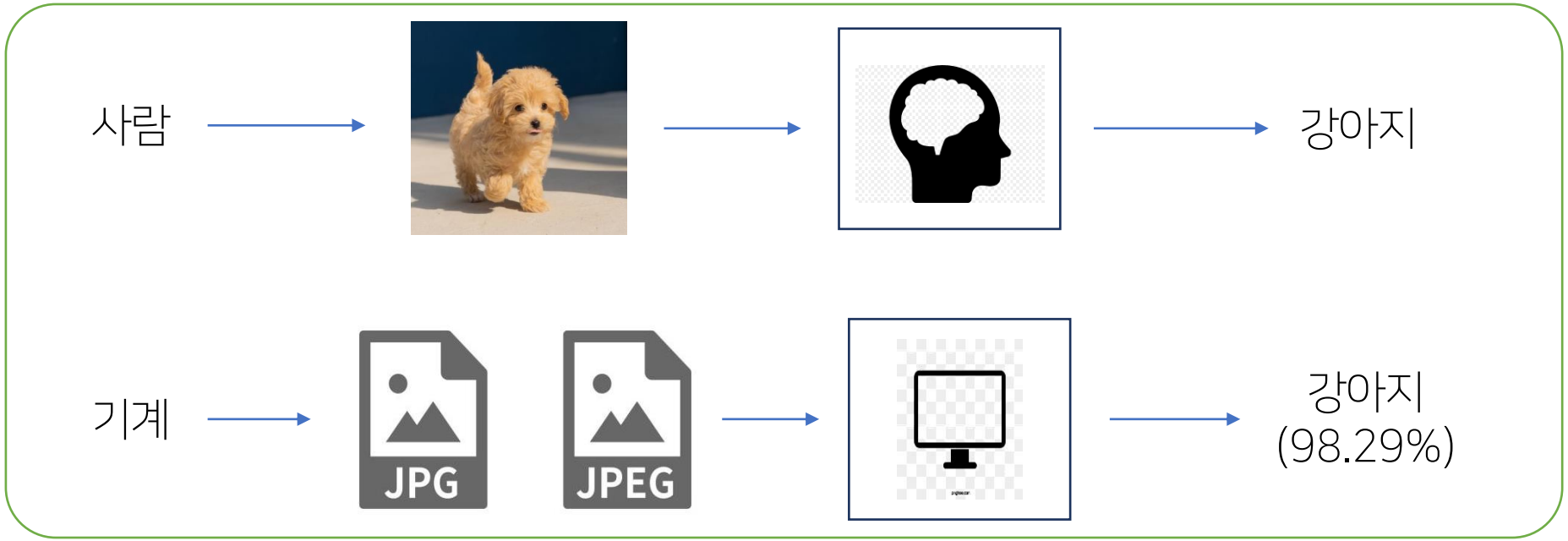
: 상호 배타적인 피쳐들 끼리 번들링을 하여,
피쳐를 줄여주어 모든 피쳐를 검색할 필요가 없어 시간이 단축된다.



딥러닝

Deep Learning :

- 머신러닝의 특정한 분야로서 인공 신경망의 층을 연속적으로 깊게 쌓아 올려 데이터를 학습하는 방식
- 1943년 Warren McCulloch의 "기계에도 인간(생물)이 학습하는 방법을 동일하게 적용해보자" 라는 아이디어에서 시작되어 최근 빅데이터와 결합하여 많은 분야에서 활발하게 사용되고 있다.
- 대표적인 알고리즘 : CNN(합성곱 신경망), RNN(순환 신경망), LSTM(장단기메모리)

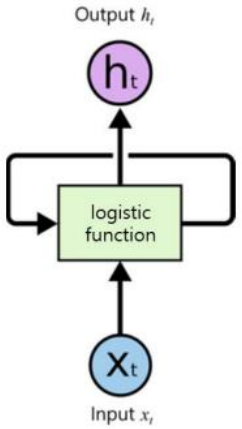


순환신경망 (Recurrent Neural Network: RNN)

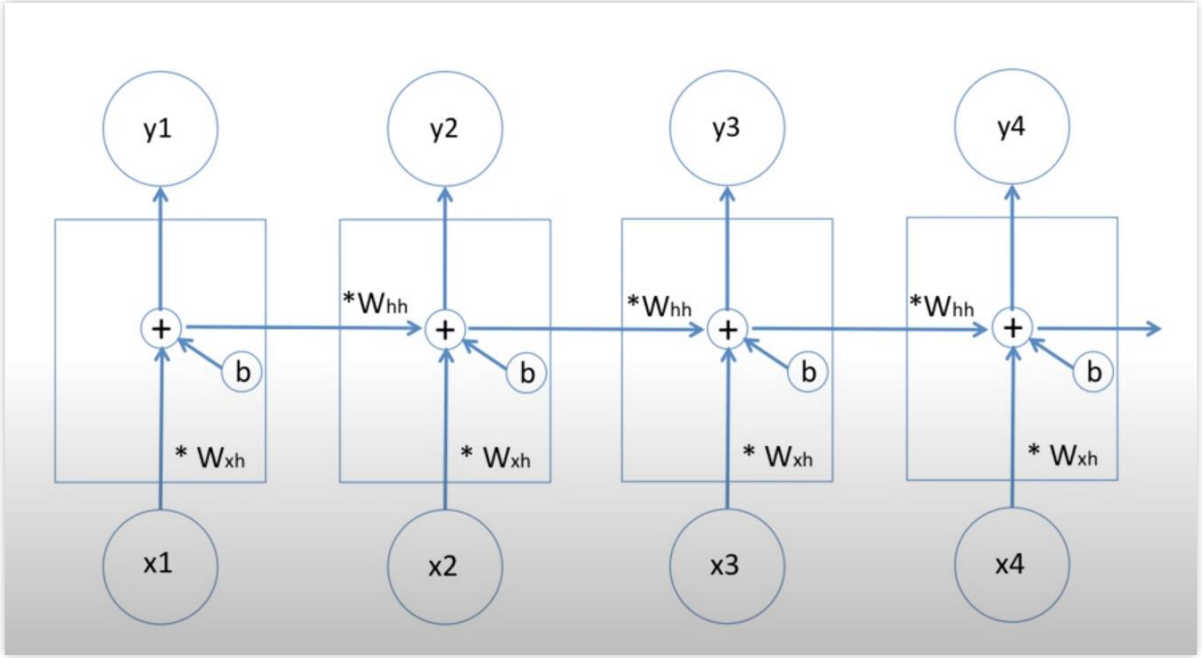
RNN :

- 텍스트 데이터, 시계열 데이터과 같은 순차적인 데이터 학습에 특화된 알고리즘
- 이전 은닉층의 결과와 현재 파라미터를 고려하여 값을 계산함으로써 과거의 학습을 현재의 학습에 반영한다.

$$h_t = logistic\ (W_h x_t + U_t h_{t-1} + b_h)$$



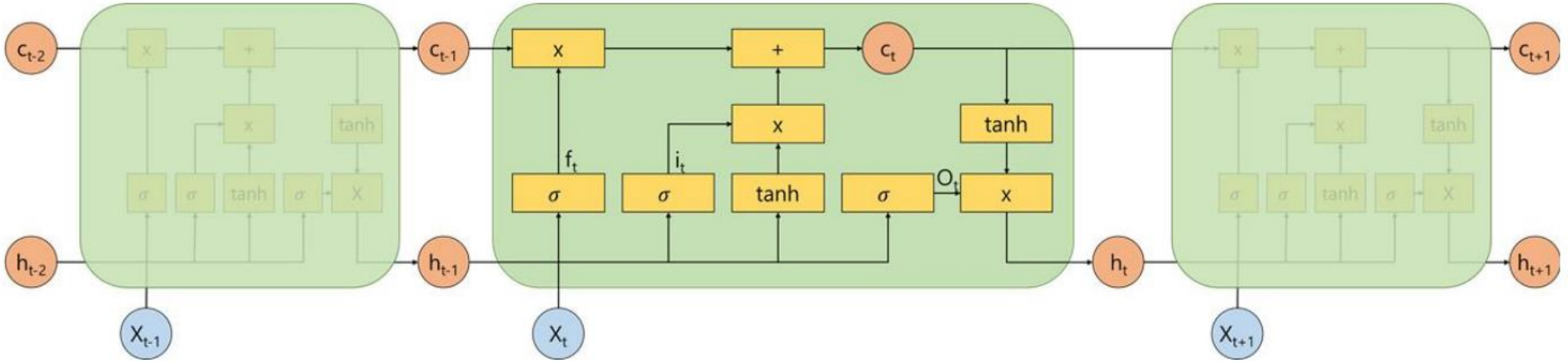
<그림 II-3-3> RNN 모델 구조



LSTM (Long short-term memory)

LSTM :

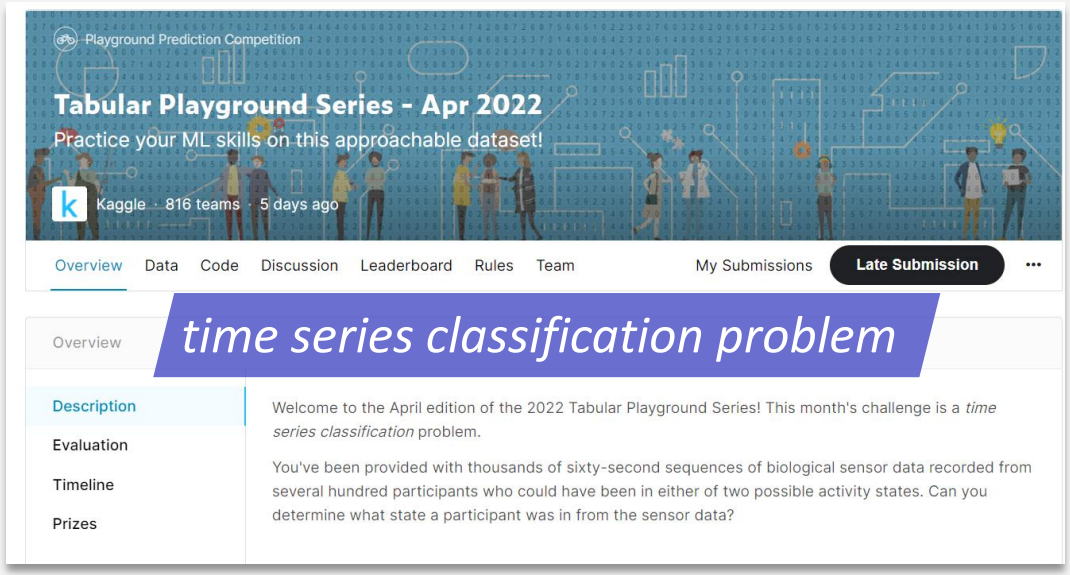
- RNN 알고리즘은 갈수록 먼 시점의 정보가 점점 희미해지는 **장기의존성** 문제를 가지고 있다.
- LSTM은 RNN의 장기의존성을 해결하기 위해 개발된 알고리즘으로,
  순환구조에서 불필요한 정보를 삭제하거나 정보의 중요도에 따라 자체적으로 가중치를 조절한다.



<그림 Ⅱ-3-4> LSTM 모델 구조

대회 Introduction

<Tabular Playground Series – Apr 2022>



수백명의 참가자들로부터 기록된 생물학적 센서 데이터로부터 60초로 구성된 수천개의 시퀀스를 통해 두 가지의 활성 상태로 분류하는 모델을 개발하는 대회이다.

<Data description>

train.csv

- *sequence* : a unique id for each sequence
- *subject* : a unique id for the subject in the experiment
- *step* : time step of the recording, in one second intervals
- *sensor_00~sensor_12* : the value for each of the thirteen sensors at that time step

train_labels.csv

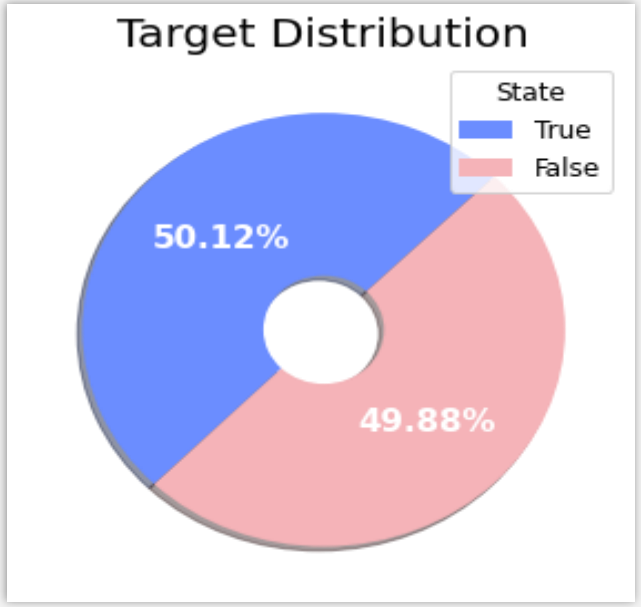
- *sequence* : a unique id for each sequence
- *state* : the state associated to each sequence.
  -> This is the target which you are trying to predict.

test.csv

: the test set. For each of the ~12,000 sequences, you should predict a value for that sequence's state.

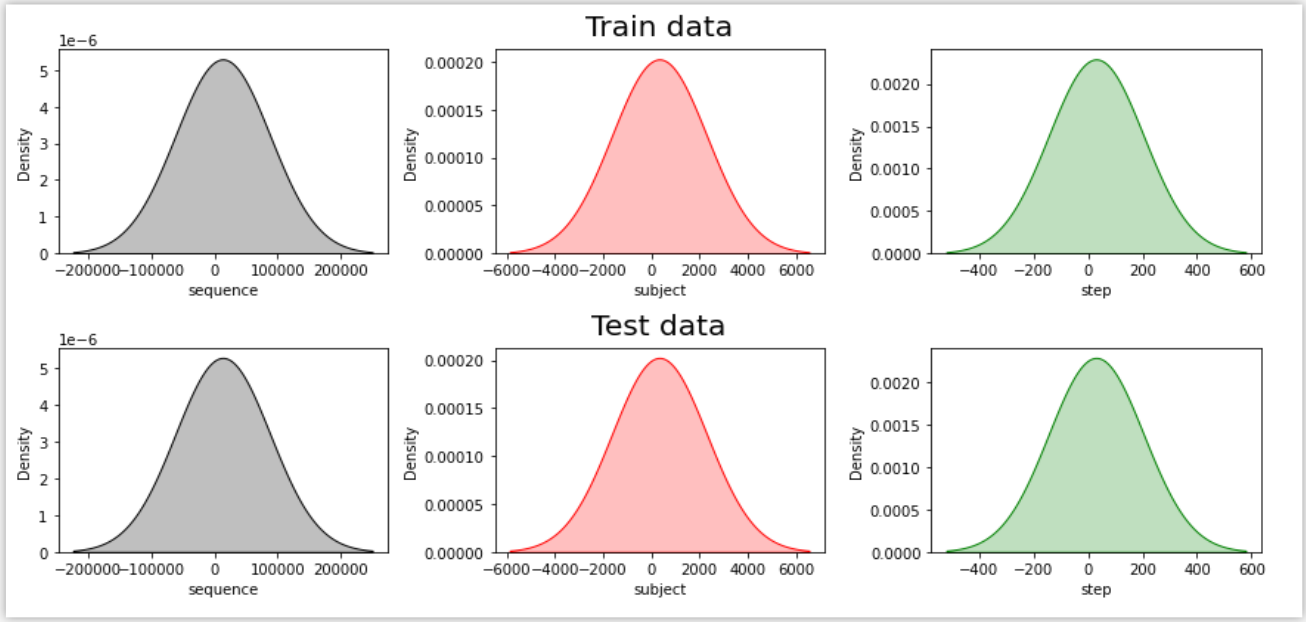
변수들의 분포 확인

<종속변수인 'state'의 분포 확인>



0과 1의 분포가 거의 비슷하므로
과소 표집 혹은 과대 표집이 필요없다.

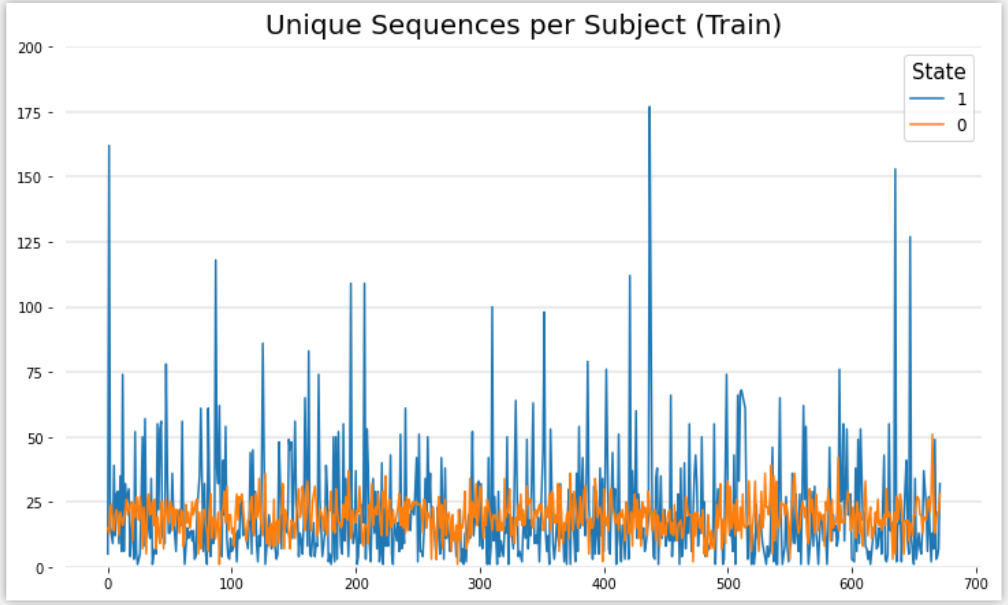
<'sequence', 'subject', 'step'의 분포 확인>



train 데이터와 test 데이터 모두
sequence, subject, step은 정규분포의 형태를 보인다.

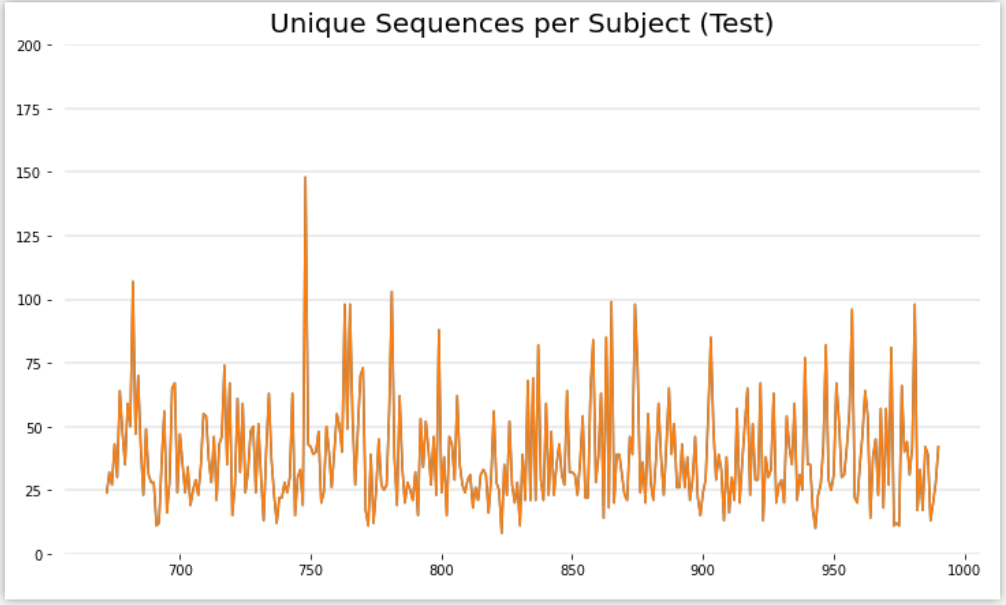
변수들의 분포 확인

subject 별 sequence의 개수 확인 - <Train>



각 subject에 대한 고유 sequence의 수는
state 값이 1인 경우에 대체로 큰 값을 보이며,
state 값이 0인 경우에는 50 이내에서 비교적 고른 분포를 보인다.

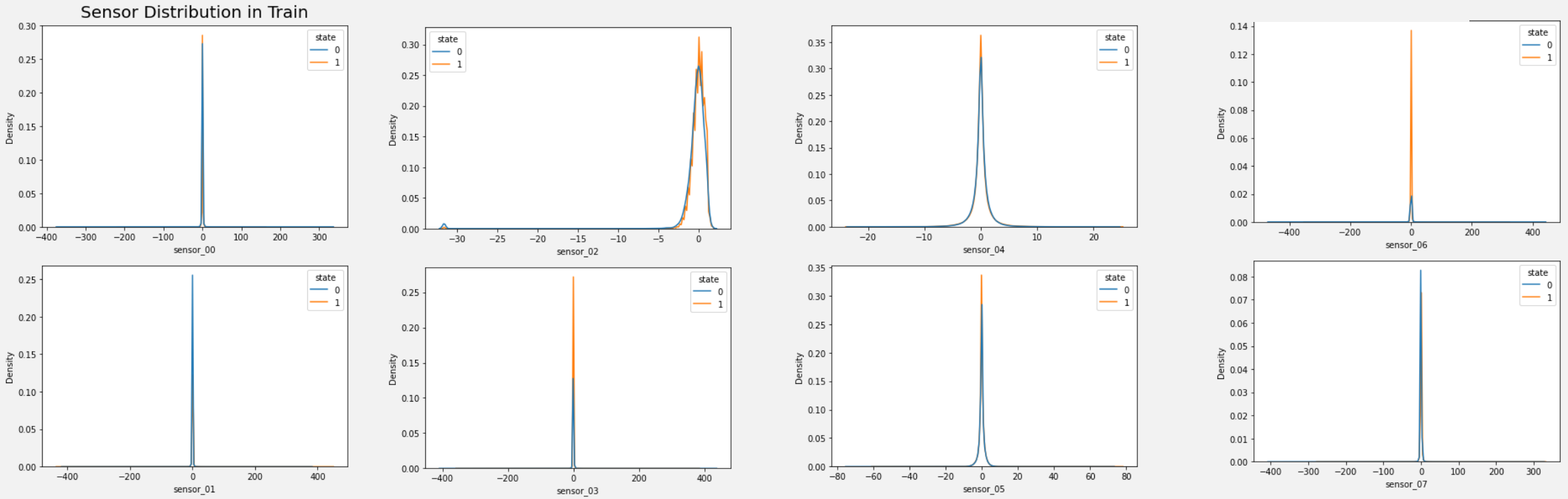
subject 별 sequence의 개수 확인 - <Test>



같은 맥락으로, Test 데이터에서도 subject에 대한 고유
sequence의 수가 state 값에 영향을 미칠 것으로 보인다.

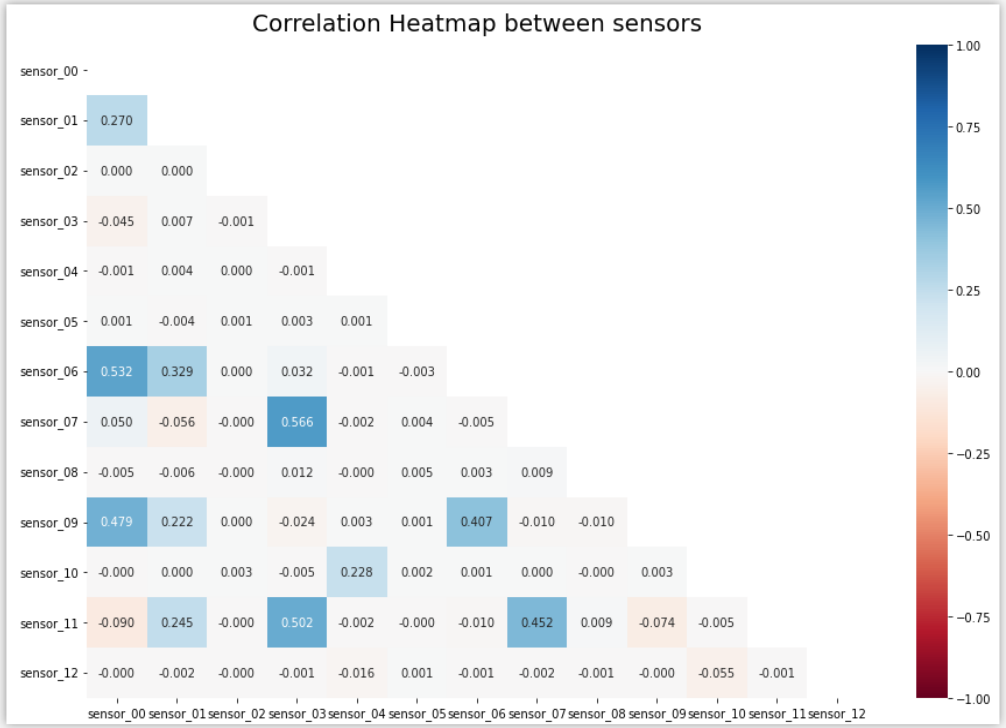
센서들의 분포 확인

< train 데이터셋의 sensor 데이터의 일부 >



대부분의 sensor는 0을 기준으로 좌우 대칭의 형태를 띄고 있으나, sensor_02의 경우 왼쪽 꼬리가 긴 분포를 보인다.
각 sensor의 이상치로 인해 데이터의 분포를 제대로 확인하기 어려우므로 이상치를 제외한 데이터의 분포를 확인할 필요가 있다.

센서들 간의 상관관계 확인



대부분의 변수들 사이에는 상관관계가 약하거나 거의 없는 것으로 나타나지만,
(sensor_00, sensor_06, sensor_09),
(sensor_03, sensor_07, sensor_11)은 뚜렷한 양의 상관관계를 가진다.

그러나,

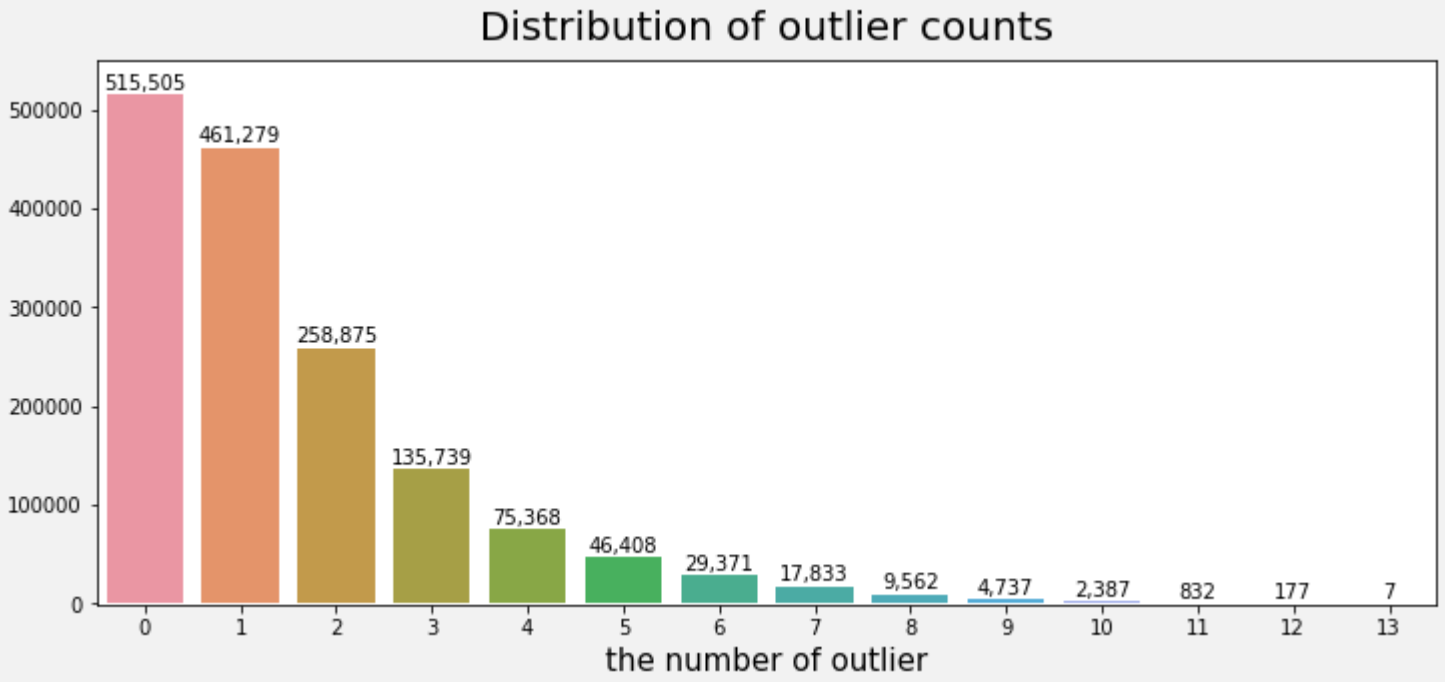
상관관계가 있는 변수들을 제거했을 때, 모델의 성능이 근소하게 낮아지는 것을 확인하였기 때문에 최종적으로는 제거하지 않고 진행하였다.

이상치 확인

IQR 방식 활용

하한 값 : 1분위수 - IQR * 1.5
상한 값 : 3분위수 + IQR * 1.5

*하한 값보다 작거나 상한 값보다 큰 값*을 이상치로 간주



train 데이터의 시퀀스별 이상치 개수를 분석한 결과,
이상치가 없는 시퀀스가 전체의 약 3분의 1 정도이고,
반대로 모든 값이 정상 범위를 벗어나는 경우도 존재한다.

PCA (주성분 분석)

: 차원을 축소하기 위해 사용하며 말 그대로 주성분을 찾아내는 것이다.
첫 번째 주성분이 원 데이터의 분포를 가장 많이 보존하고, 두 번째 주성분이 그 다음으로 원 데이터의 분포를 많이 보존하는 식이다.

< 주성분 2개가 전체 분산의 99.97%를 설명 >

싸이킷런의 내장 라이브러리를 사용하여 계산한 결과

```
total_var = pca.explained_variance_ratio_.sum() * 100
print(f'Total Explained Variance : {total_var:.2f} %')
```

```
Total Explained Variance : 99.97 %
```

<로지스틱 회귀에서 PCA 전후 교차검증도 비교>

```
print("Exsisting Data :", np.mean(scores['train_score']), np.mean(scores['test_score']))
print("PCA Data :", np.mean(scores_pca['train_score']), np.mean(scores_pca['test_score']))
```

```
Exsisting Data : 0.6164844827333275 0.6164766892585747
PCA Data : 0.5051464439750023 0.50513544142241
```

<LGBM에서 PCA 전후 교차검증도 비교>

```
print("Exsisting Data :", np.mean(scores['train_score']), np.mean(scores['test_score']))
print("PCA Data :", np.mean(scores_pca['train_score']), np.mean(scores_pca['test_score']))
```

```
Exsisting Data : 0.8767379448698764 0.8751577032538216
PCA Data : 0.6547178945515358 0.652623742041487
```

로지스틱 회귀와 LGBM 모델링 모두 PCA 후에 교차 검증 정확도가 더 낮아졌다.

따라, 해당 데이터에서는 *주성분이 모델에 불필요* 하므로 PCA는 사용하지 않기로 하였다.

정상성 확인

KPSS test 사용

< train 과 test 데이터셋을 합쳐서 검증한 결과 >

```
sensor_00
Results of KPSS Test:
Test Statistic           0.052708
p-value                  0.100000
Lags Used             1239.000000
dtype: float64
--------------------

sensor_01
Results of KPSS Test:
Test Statistic           0.027774
p-value                  0.100000
Lags Used             1068.000000
dtype: float64
--------------------

sensor_02
Results of KPSS Test:
Test Statistic           0.090332
p-value                  0.100000
Lags Used              752.000000
dtype: float64
--------------------

sensor_03
Results of KPSS Test:
Test Statistic           0.110382
p-value                  0.100000
Lags Used              757.000000
dtype: float64
--------------------
```

```
sensor_04
Results of KPSS Test:
Test Statistic           0.390871
p-value                  0.081090
Lags Used              964.000000
dtype: float64
--------------------

sensor_05
Results of KPSS Test:
Test Statistic           0.075491
p-value                  0.100000
Lags Used             1338.000000
dtype: float64
--------------------

sensor_06
Results of KPSS Test:
Test Statistic           0.144015
p-value                  0.100000
Lags Used             1116.000000
dtype: float64
--------------------

sensor_07
Results of KPSS Test:
Test Statistic           0.294329
p-value                  0.100000
Lags Used             1193.000000
dtype: float64
--------------------
```

```
sensor_08
Results of KPSS Test:
Test Statistic           0.20255
p-value                  0.10000
Lags Used             1229.00000
dtype: float64
--------------------

sensor_09
Results of KPSS Test:
Test Statistic           0.190786
p-value                  0.100000
Lags Used             1214.000000
dtype: float64
--------------------

sensor_10
Results of KPSS Test:
Test Statistic           0.160319
p-value                  0.100000
Lags Used              806.000000
dtype: float64
--------------------

sensor_11
Results of KPSS Test:
Test Statistic           0.118962
p-value                  0.100000
Lags Used             1081.000000
dtype: float64
--------------------
```

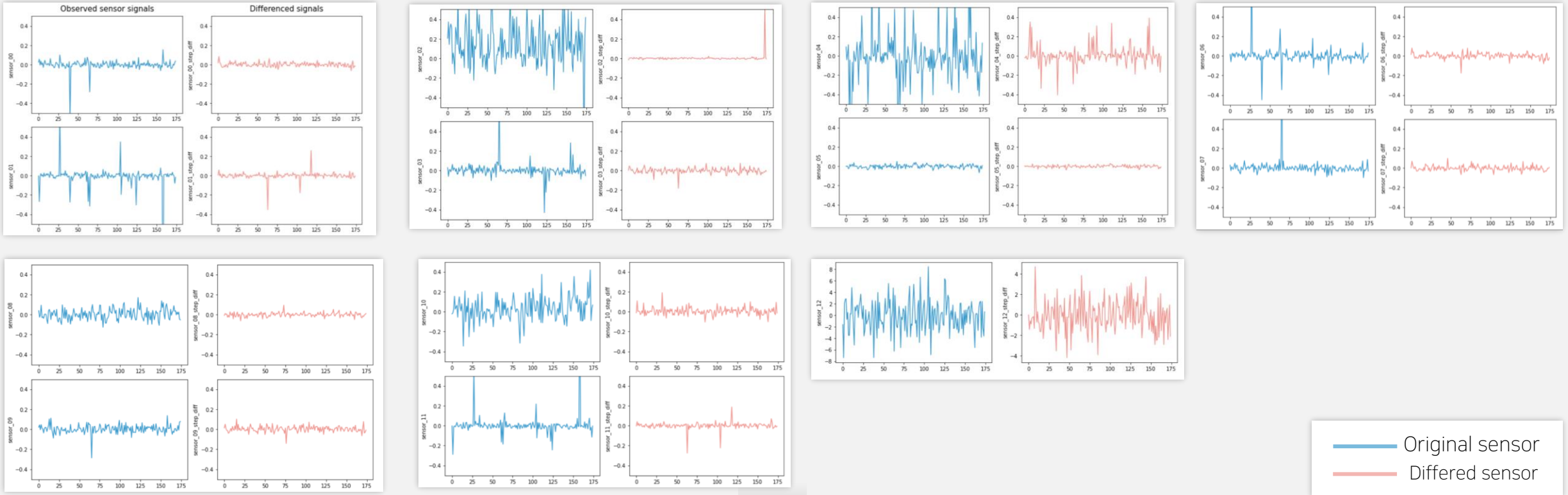
```
sensor_12
Results of KPSS Test:
Test Statistic           0.079625
p-value                  0.100000
Lags Used              914.000000
dtype: float64
--------------------
```

sensor들의 p-value값이 모두 0.05 이상이므로 귀무가설을 기각할 수 없다.

따라서 sensor 변수들이 비정상적인 시계열이라고 할 수 없다.

정상성 확인

이는 센서 데이터와 한 번 차분한 센서 데이터의 시각화를 통해서도 확인할 수 있었다.



이상치를 제외하고는 대부분 평균이나 분산의 변동폭이 비교적 일정한 양상을 보인다.

추가적인 변수 생성

- 각 센서들에 대한 통계적 변수 추가

Sensor_00_step_mean ~ Sensor_12_step_mean
: 각 센서의 60 step 데이터 **평균**

Sensor_00_step_median ~ Sensor_12_step_median
: 각 센서의 60 step 데이터 **중앙값**

Sensor_00_step_std ~ Sensor_12_step_std
: 각 센서의 60 step 데이터 **표준편차**

Sensor_00_step_min ~ Sensor_12_step_min
: 각 센서의 60 step 데이터 **최소값**

Sensor_00_step_max ~ Sensor_12_step_max
: 각 센서의 60 step 데이터 **최댓값**

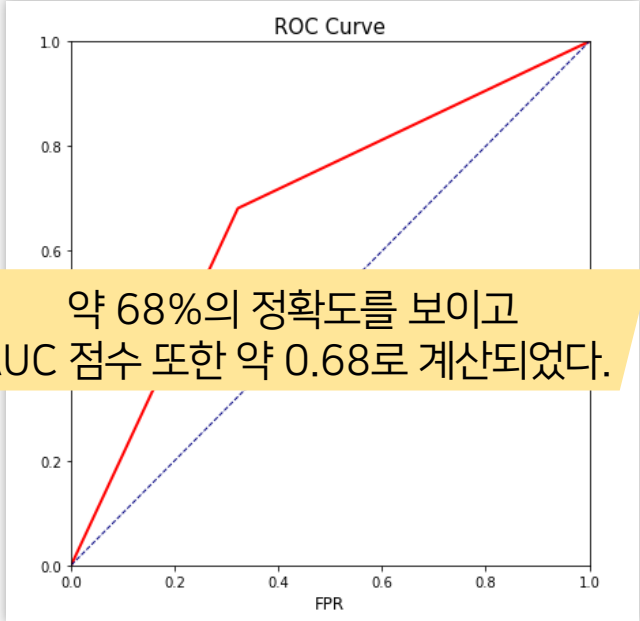
Sensor_00_step_diff ~ Sensor_12_step_diff
: 센서 간 60 step 데이터 **차이값**

- 각 subject별 sequence의 개수 변수 추가

Num_sequence
: sequence 개수에 따라, state 1의 비율이 다르기 때문에 sequence Feature 생성

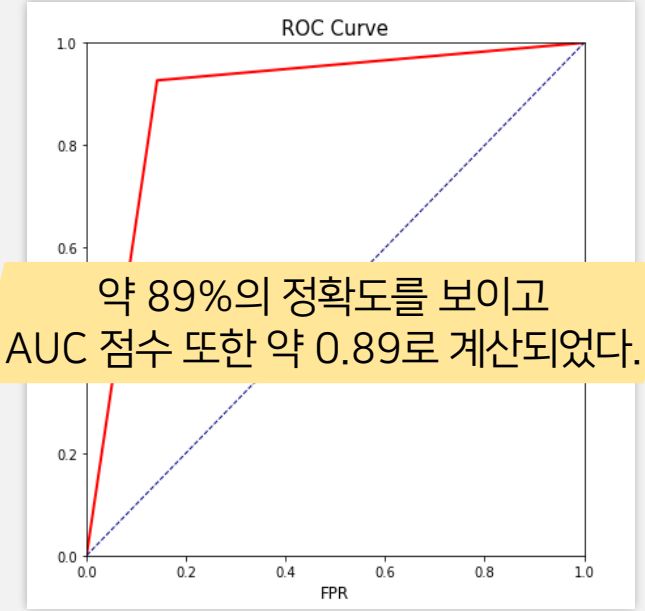
선형 | 비선형 | 딥러닝(신경망) 비교

< 로지스틱 회귀 >



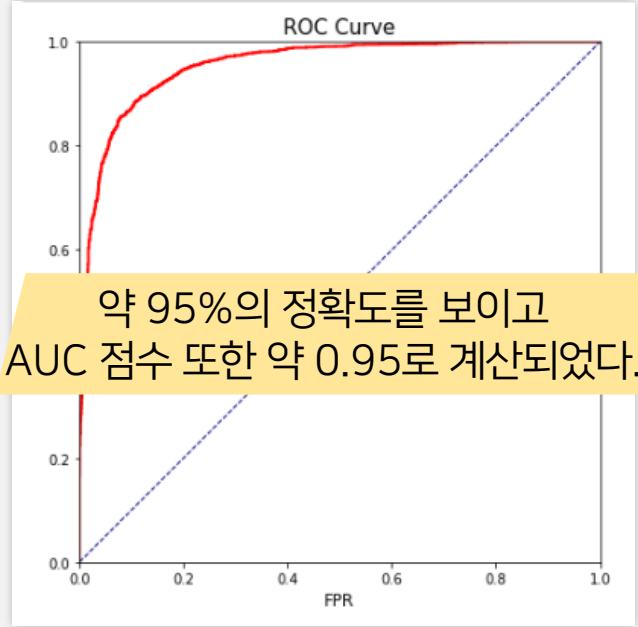
약 68%의 정확도를 보이고
AUC 점수 또한 약 0.68로 계산되었다.

< LightGBM >



약 89%의 정확도를 보이고
AUC 점수 또한 약 0.89로 계산되었다.

< LSTM >



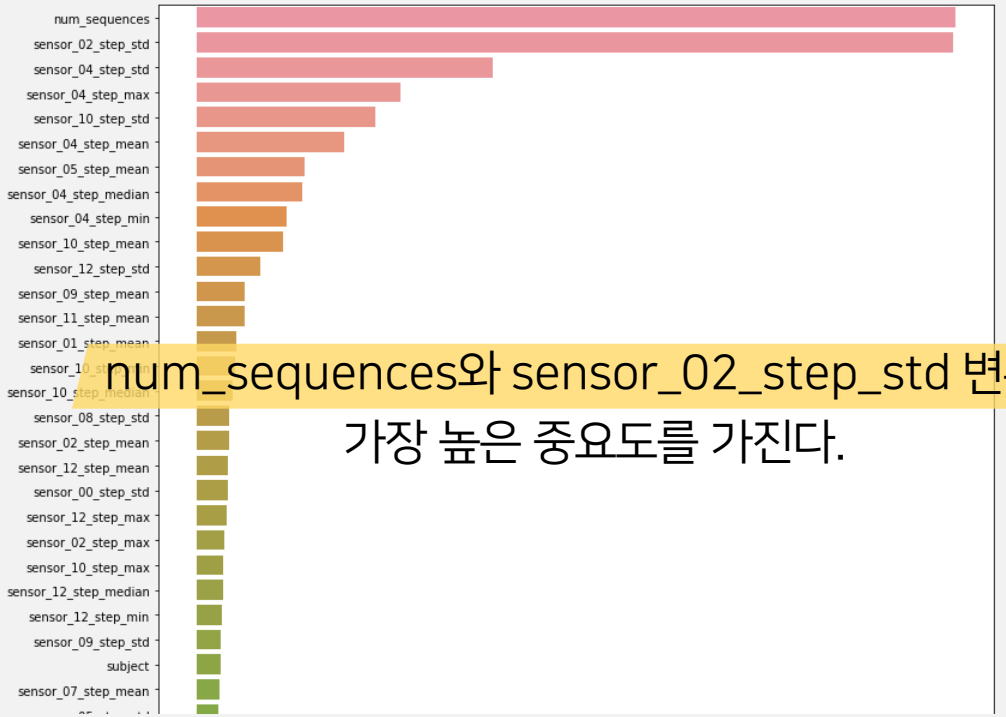
약 95%의 정확도를 보이고
AUC 점수 또한 약 0.95로 계산되었다.

"결과적으로, 이번 대회에서는 LSTM 모델을 사용한 것이 가장 결과가 좋았다."

순열 중요도

: 변수의 중요도를 확인할 수 있는 방법 중 하나로
특성 값을 순열로 만든 후 모델의 *예측 오차 증가량을 계산*하는 방식으로 특성의 중요도를 측정한다.

< LightGBM 모델에서의 순열 중요도 >



num_sequences와 sensor_02_step_std 변수가
가장 높은 중요도를 가진다.

Insight

subject에 대한 고유 sequence의 수를 새로운 변수로 추가했던 것이
모델의 성능 향상에 큰 영향을 미친 것으로 보인다.

불균형한 분포를 보인 sensor_02의 표준편차 값 역시
모델의 성능 향상에 큰 영향을 미친 것으로 보인다.

많은 변수들의 중요도가 0으로 나타나지만,
중요도가 음수로 나오는 변수는 없기 때문에 추가적인 변수 선택은 생략한다.

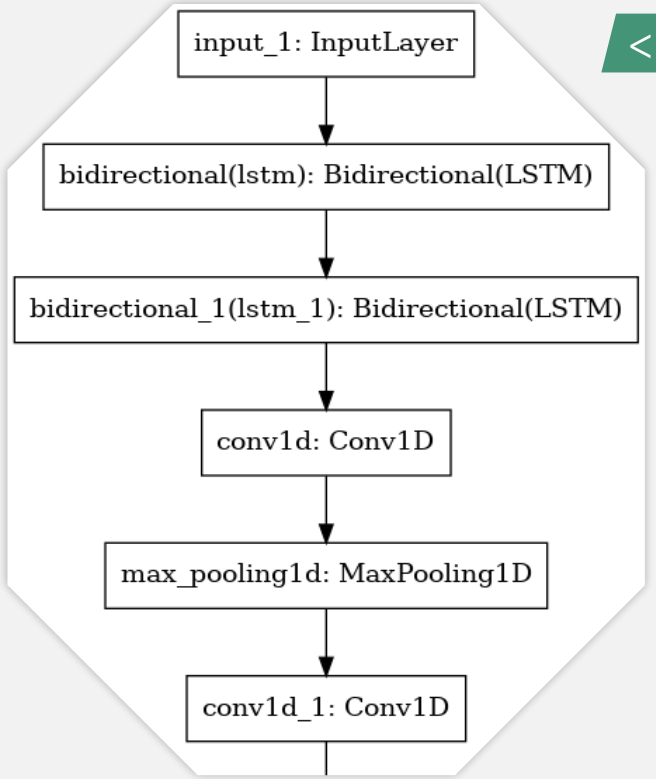
캐글 대회에서 사용한 LSTM 모델

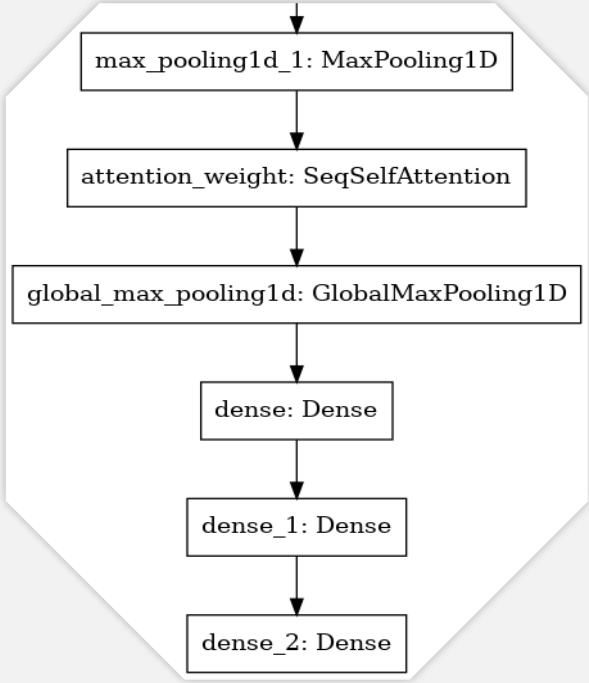
-> 양방향 LSTM 레이어, CNN 레이어, Attention 레이어를 사용하였을 때 모델의 성능 지표가 가장 높게 나타났다.

< Fold를 5개로 지정 >

```
Fold-1 | OOF Score: 0.9593504550878398
-------------- > Fold 2 < --------------
Fold-2 | OOF Score: 0.958282127080081
-------------- > Fold 3 < --------------
Fold-3 | OOF Score: 0.9630203440373502
-------------- > Fold 4 < --------------
Fold-4 | OOF Score: 0.9546956444601095
-------------- > Fold 5 < --------------
Fold-5 | OOF Score: 0.9580090341081677
Mean AUC on 5 folds - 0.9586715209547096
```

< 사용한 LSTM 레이어 구조도 >





시계열에서의 딥러닝

머신러닝보다 딥러닝에서 점수가 더 높게 나온 이유는 ?

- 딥러닝 모델에서는 정상성을 요구하지 않기 때문에,
  계절형 ARIMA 모델의 차수, 계절성에 따른 평가와 같은 파라미터를 고르는 과정을 밟지 않아도 된다.
- 머신러닝 알고리즘은 입력 데이터와 차원성이라는 측면에서 꽤 불안정한 경향이 있는 반면,
  딥러닝 알고리즘은 모델과 입력 데이터의 특성에 관해서 매우 유연하다.
  -> 시계열 문제의 해결에 있어서 딥러닝 알고리즘의 유연성이 안정적으로 적용된다.

"딥러닝 모델 적용의 주의점"

- 딥러닝 구조를 추세에 맞도록 변경해야 학습이 잘 이루어지며, 입력 값이 -1과 1사이일 때 가장 잘 동작한다.
- LSTM과 같이 시계열에 적합한 알고리즘의 경우 간단한 모델일수록 오히려 성능이 더 좋을 수 있다.

Reference

<캐글 노트북>

https://www.kaggle.com/code/kellibelcher/time-series-classification-with-lstms-sensor-eda
https://www.kaggle.com/code/sytuannguyen/tps-april-2022-eda-model#Subjects-with-100%-state-0
https://www.kaggle.com/code/hasanbasriakcay/tpsapr22-eda-fe-baseline#Distribution
https://www.kaggle.com/code/hamzaghanmi/tps-april-tensorflow-bi-lstm
https://www.kaggle.com/code/landfallmotto/tps-apr-lstms-attention
https://www.kaggle.com/code/ryanbarretto/lstm-baseline
https://www.kaggle.com/code/javigallego/tps-apr22-eda-fe-lstm-tutorial
https://www.kaggle.com/code/ninjaac/tps-apr22-basic-eda#1.2-

<이외 참고문헌>

(LSTM 모델)
https://www.youtube.com/watch?v=bX6GLbpw-A4&t=1s
점프 투 파이썬 딥러닝 LSTM, https://wikidocs.net/22888
머신러닝과 딥러닝을 이용한 시계열 빅데이터 예측 연구 . 아산.순천향대학교 대학원 . 2021 :
http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=ac74792c0c061f2effe0bdc3ef48d419

(시계열 이론)
http://www.yes24.com/Product/Goods/98576347
https://sodayeong.tistory.com/19
https://skyeong.net/285
https://direction-f.tistory.com/65

(서론 레퍼런스)
http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=ac74792c0c061f2effe0bdc3ef48d419
https://zdnet.co.kr/view/?no=20211027105111
https://biz.chosun.com/stock/stock_general/2022/05/03/2DPR2IMFGRGVPGDFCOZPLLMWEM/
https://www.korea.kr/news/pressReleaseView.do?newsId=156496713
https://zdnet.co.kr/view/?no=20201020145525

Personal info.

이름 : 오승은
연락처 : 010-4096-3159
이메일 :
dhtmddms4043@naver.com
역할 : 팀장, 기획, ppt 제작, EDA, FE

이름 : 강지원
연락처 : 010-3644-1912
이메일 :
donumm64@gmail.com
역할 : 기획, 개념 정리, EDA, FE

이름 : 김영재
연락처 : 010-4161-9914
이메일 :
kimyoungjae123@naver.com
역할 : 기획, EDA, FE, LSTM

이름 : 엄태현
연락처 : 010-7417-8917
이메일 :
gkdlrhfo@naver.com
역할 : 기획, EDA, 딥러닝, FE