# 운동 동작 분류

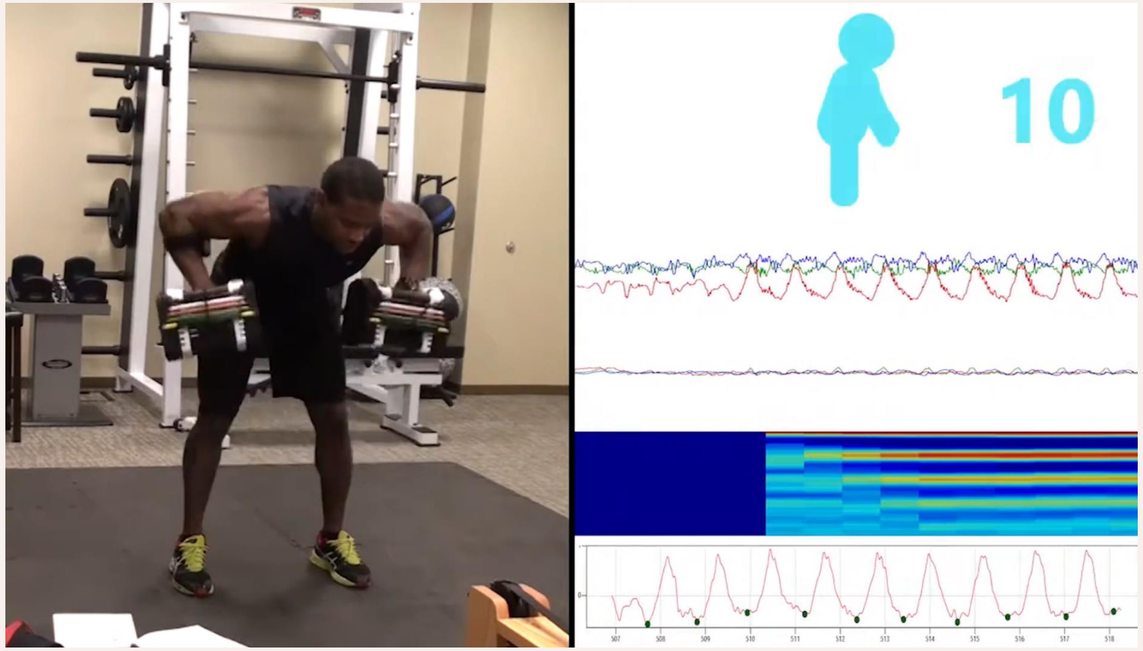
4조 : 이승준 정노아 정성훈 정종인

# CONTENTS

# 01 대회 및 데이터 소개

## 목표

목표 : 측정된 센서 데이터를 이용하여 운동 동작을 분류하는 모델 생성




3축 가속도계(accelerometer)

3축 자이로스코프(gyroscope) (각속도)

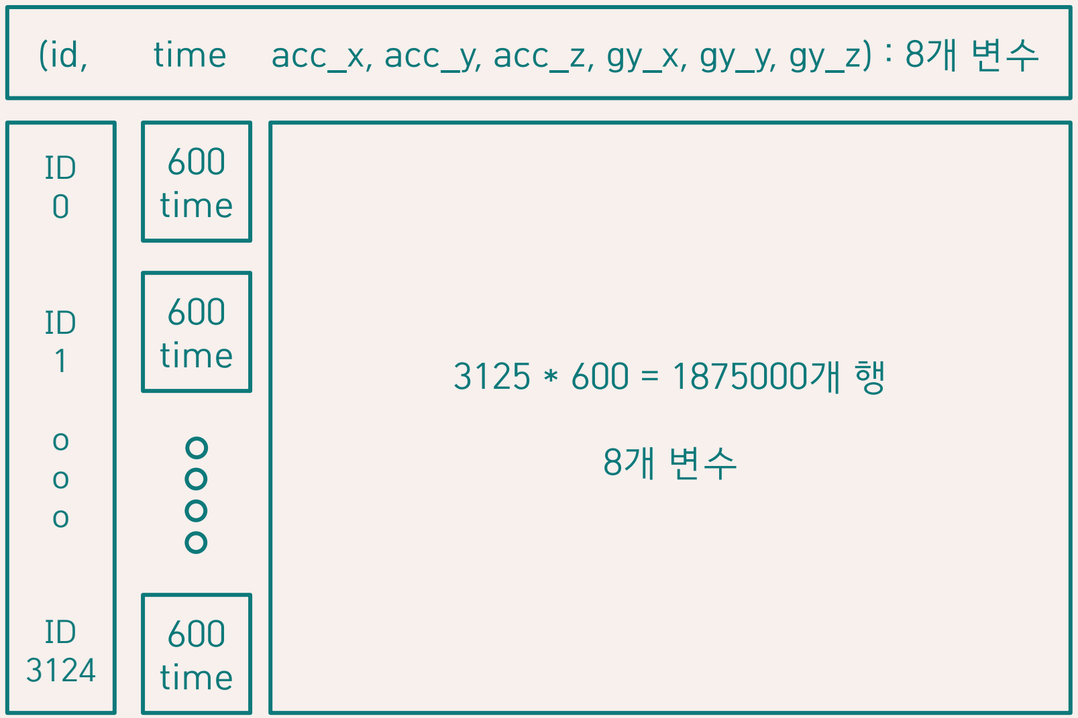
우측 팔뚝에 센서 부착 후 운동 수행

# 01 대회 및 데이터 소개

## 데이터 소개

3125개의 ID별로 0~599까지의 TIME 마다 가속도, 자이로스코프 측정

TIME 의 간격은 0.02초, 한 ID당 12초의 운동




| | id | time | acc_x | acc_y | acc_z | gy_x | gy_y | gy_z |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1.206087 | -0.179371 | -0.148447 | -0.591608 | -30.549010 | -31.676112 |
| **1** | 0 | 1 | 1.287696 | -0.198974 | -0.182444 | 0.303100 | -39.139103 | -24.927216 |
| **2** | 0 | 2 | 1.304609 | -0.195114 | -0.253382 | -3.617278 | -44.122565 | -25.019629 |
| **3** | 0 | 3 | 1.293095 | -0.230366 | -0.215210 | 2.712986 | -53.597843 | -27.454013 |
| **4** | 0 | 4 | 1.300887 | -0.187757 | -0.222523 | 4.286707 | -57.906561 | -27.961234 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **1874995** | 3124 | 595 | -0.712530 | -0.658357 | 0.293707 | -29.367857 | -104.013664 | -76.290437 |
| **1874996** | 3124 | 596 | -0.683037 | -0.658466 | 0.329223 | -30.149089 | -101.796809 | -76.625087 |
| **1874997** | 3124 | 597 | -0.664730 | -0.666625 | 0.364114 | -27.873095 | -98.776072 | -79.365125 |
| **1874998** | 3124 | 598 | -0.630534 | -0.682565 | 0.373696 | -23.636550 | -99.139495 | -80.259478 |
| **1874999** | 3124 | 599 | -0.578351 | -0.700235 | 0.384390 | -17.917626 | -100.181873 | -80.676229 |

1875000 rows × 8 columns

<train_features>

# 01 대회 및 데이터 소개

## 데이터 소개

test 데이터는 train과 id 개수만 다르고 변수, time 동일

| | id | time | acc_x | acc_y | acc_z | gy_x | gy_y | gy_z |
|---|---|---|---|---|---|---|---|---|
| **0** | 3125 | 0 | -0.628100 | -0.160155 | 0.151487 | 49.665357 | 88.435961 | 13.597668 |
| **1** | 3125 | 1 | -0.462548 | 0.012462 | -0.053726 | 56.953059 | 96.185341 | 16.278458 |
| **2** | 3125 | 2 | -0.363481 | -0.091789 | -0.130004 | 29.557396 | 93.836453 | 13.329043 |
| **3** | 3125 | 3 | -0.351750 | -0.239870 | -0.193053 | 23.686172 | 88.608721 | 13.449771 |
| **4** | 3125 | 4 | -0.312934 | -0.123762 | -0.318621 | 20.410071 | 85.327707 | 13.884912 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **469195** | 3906 | 595 | 0.104191 | -0.784979 | 0.639513 | -10.475346 | 14.095361 | -190.358982 |
| **469196** | 3906 | 596 | 0.103297 | -0.758954 | 0.615687 | -25.360272 | -8.523018 | -180.393291 |
| **469197** | 3906 | 597 | 0.128294 | -0.749389 | 0.586184 | -27.917723 | -23.186245 | -162.624160 |
| **469198** | 3906 | 598 | 0.104130 | -0.692731 | 0.573397 | -27.847980 | -30.407555 | -138.761676 |
| **469199** | 3906 | 599 | 0.059299 | -0.613487 | 0.556780 | -29.900725 | -36.586219 | -116.423810 |

469200 rows × 8 columns

<test_features>

운동은 0번부터 60번까지 총 61가지

| | id | label | label_desc |
|---|---|---|---|
| **0** | 0 | 37 | Shoulder Press (dumbbell) |
| **1** | 1 | 26 | Non-Exercise |
| **2** | 2 | 3 | Biceps Curl (band) |
| **3** | 3 | 26 | Non-Exercise |
| **4** | 4 | 26 | Non-Exercise |
| **...** | ... | ... | ... |
| **3120** | 3120 | 26 | Non-Exercise |
| **3121** | 3121 | 26 | Non-Exercise |
| **3122** | 3122 | 15 | Dynamic Stretch (at your own pace) |
| **3123** | 3123 | 26 | Non-Exercise |
| **3124** | 3124 | 2 | Bicep Curl |

3125 rows × 3 columns

<train_labels>

# 02 변수 소개

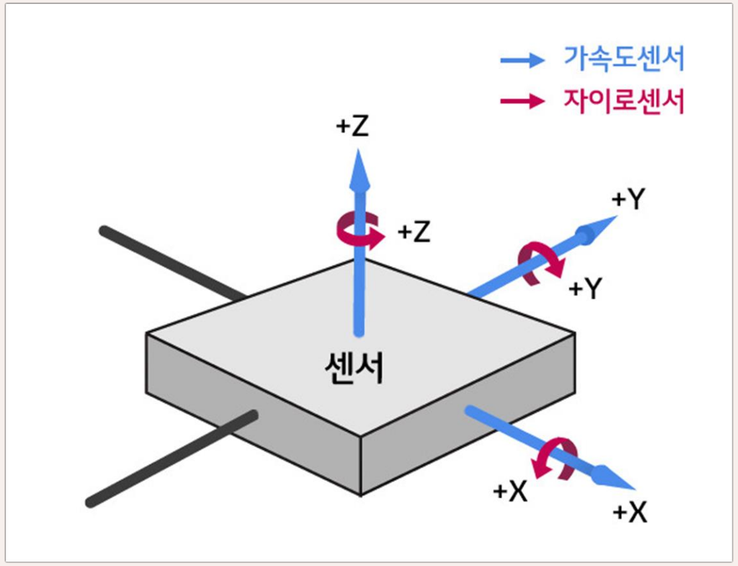
변수소개

## 3축 가속도

3방향 가속도 측정

Time이 길어도 적은 오차

Z축에 대한 회전(각도) 측정 불가
중력가속도 외 가속도에 민감




## 3축 자이로스코프

3축 각도 변화량(각속도) 측정

Z축에 대한 회전 측정가능

Time이 길수록 큰 오차

온도에 따른 오차 발생

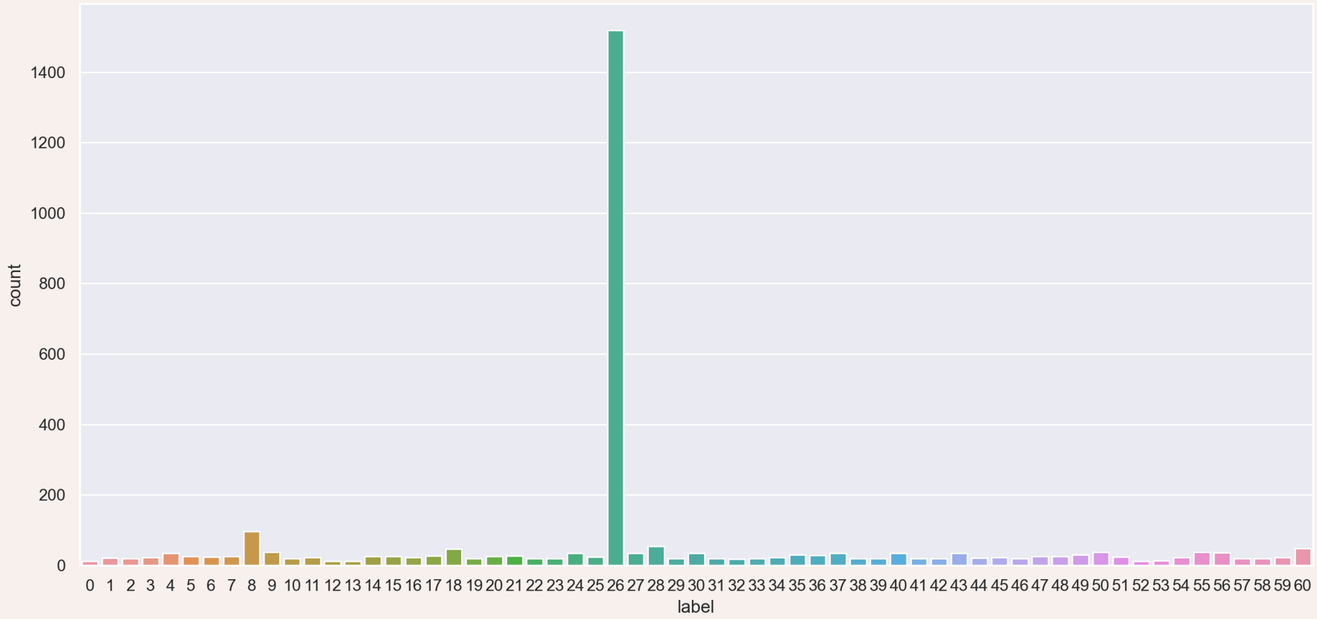
# 02 변수 소개

## 결측치, 불균형

train.isna().sum()

| | |
|---|---|
| id | 0 |
| time | 0 |
| acc_x | 0 |
| acc_y | 0 |
| acc_z | 0 |
| gy_x | 0 |
| gy_y | 0 |
| gy_z | 0 |

**결측치 없음**




3124개 중

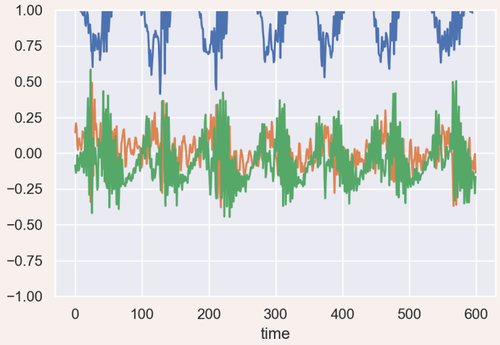
26번 : Non-Exercise -> 1518개

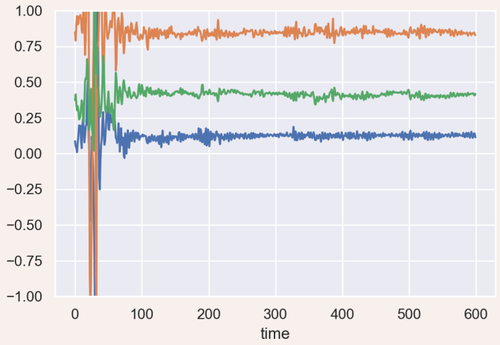
9번 : Device on table -> 97개

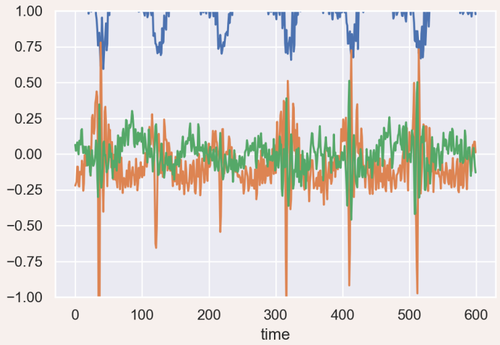
➔**데이터 불균형** 확인

# 02 변수 소개

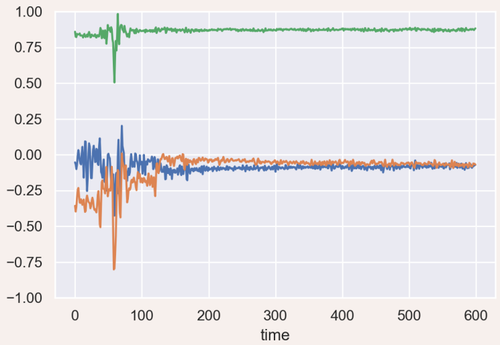
## 시각화



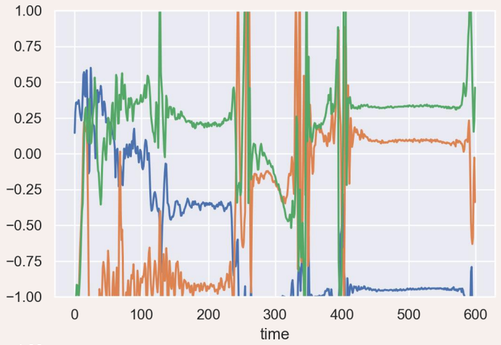



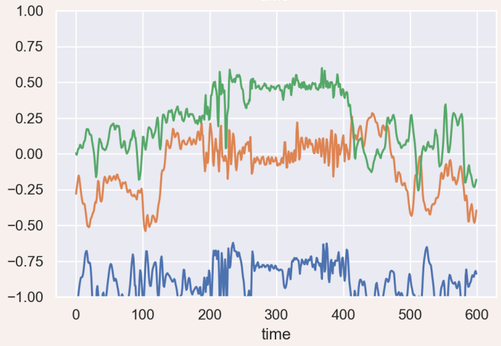




<Chest Press (rack)>

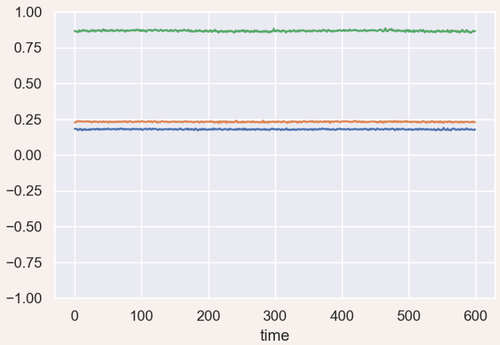


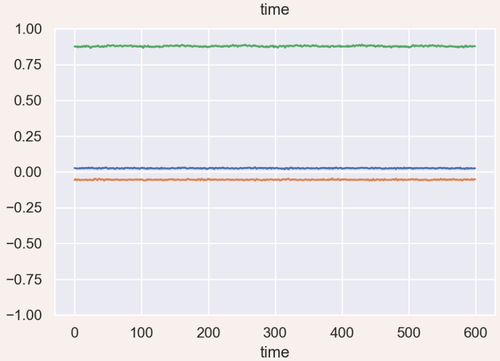

<Plank>




<Non-Exercise>




<Device on Table>

동적, 정적 움직임,

Device on Table, Non_Exercise의 차이점 확인

# 03 전처리

## 변수 생성

**모델의 분류 능력을 향상시키기 위한 변수 생성**

▷ 가속도와 자이로스코프 x, y, z 축의 벡터 변수 생성

가속도 벡터(acc_vector) = $\sqrt{(\text{가속도\_x})^2 + (\text{가속도\_y})^2 + (\text{가속도\_z})^2}$

자이로스코프 벡터(gy_vector) = $\sqrt{(\text{자이로\_x})^2 + (\text{자이로\_y})^2 + (\text{자이로\_z})^2}$

▷ 자이로스코프의 무게중심 변수 생성

자이로스코프 무게중심(gy_Centerofgravity) = (자이로_x + 자이로_y + 자이로_z) / 3

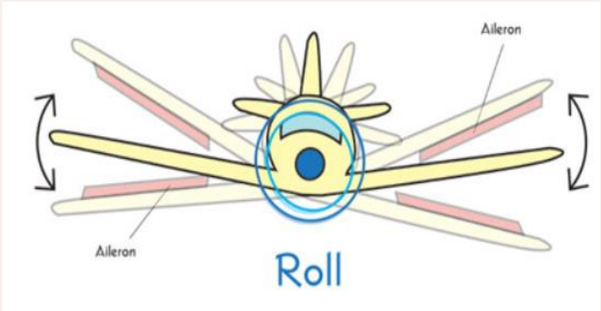
# 03 전처리
## 변수 생성

### ▷ Roll, Pitch (회전반경) 변수 생성

가속도계와 자이로스코프의 x, y, z축 변수에 아래의 roll, pitch의 식을 적용하여 **roll, pitch, gy_roll, gy_pitch** 라는 4개의 변수를 생성

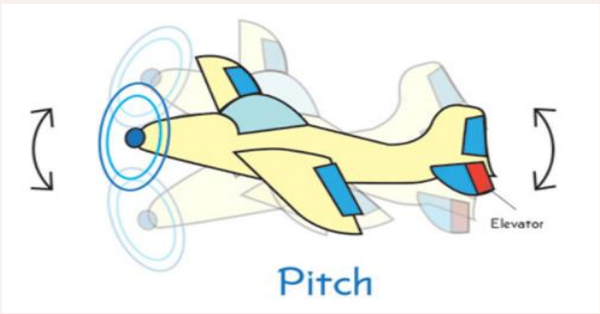
$$\text{roll} = \text{atan}\left(\frac{A_Y}{\sqrt{A_x^2 + A_Y^2}}\right)$$

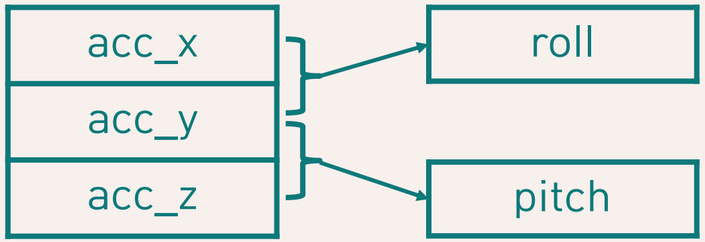
Y축에 대한 회전을 나타내는 roll

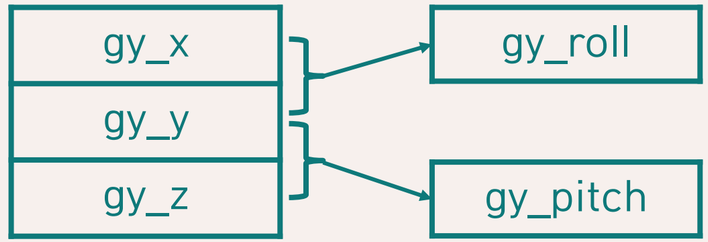



$$\text{pitch} = \text{atan}\left(\frac{A_X}{\sqrt{A_Y^2 + A_z^2}}\right)$$

X축에 대한 회전을 나타내는 pitch



# 03 전처리

## 변수 생성 - Groupby

acc_x
acc_y
acc_z
gy_x
gy_y
gy_z
acc_vector
gy_vector
gy Centerofgravity
roll
pitch
gy_roll
gy_pitch

×

max
min
mean
std
median
range_func
iqr_func2
iqr_func3
iqr_func4
preskew
prekurt

= 13 x 11
143개

ID별 Groupby를 통해 13개의 변수에 대한
11개의 통계량을 적용하여 600개의 time을 요약
-> 총 **143개의 변수** 생성

range_func : 최대값 - 최소값
iqr_func2 : 백분위 20%인 수 - 백분위 80%인 수
iqr_func3 : 백분위 40%인 수 - 백분위 60%인 수
Iqr_func4 : 백분위 15%인 수 - 백분위 95%인 수
preskew : 왜도
prekurt : 첨도

# 03 전처리

## 변수 생성 - Groupby

ID별로 groupby를 통해 3125개의 행(ID)와 143개의 변수 생성

(id, acc_x_max, acc_x_min, · · · · · · · gy_pitch_prekurt) : 143개 변수

ID 0

ID 1

o
o
o

ID 3124

3125 * 1 = 3125 행

143개 변수

| id | acc_x_max | acc_x_min | acc_x_mean | acc_x_std | acc_x_median | acc_x_range_func | acc_x_iqr_func2 |
|---|---|---|---|---|---|---|---|
| **0** | 1.344268 | 0.591940 | 0.931329 | 0.191479 | 0.956149 | 0.752327 | -0.369662 |
| **1** | 1.234020 | -2.156208 | -0.766580 | 0.495528 | -0.805767 | 3.390228 | -0.892320 |
| **2** | 1.219836 | -1.142847 | 0.039836 | 0.711972 | 0.140667 | 2.362683 | -1.561197 |
| **3** | -0.622250 | -1.417751 | -0.887702 | 0.130899 | -0.880343 | 0.795502 | -0.227442 |
| **4** | 0.599720 | -2.429109 | -0.659018 | 0.495170 | -0.941146 | 3.028829 | -0.724182 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **3120** | 0.390798 | -1.624711 | -0.300454 | 0.403175 | -0.105704 | 2.015509 | -0.734687 |
| **3121** | -0.446650 | -1.575455 | -0.974298 | 0.169963 | -0.980053 | 1.128804 | -0.202045 |
| **3122** | 0.744666 | -2.578974 | -1.114246 | 0.683789 | -1.057063 | 3.323641 | -1.396306 |
| **3123** | 0.915846 | -0.929133 | -0.111333 | 0.432722 | -0.178023 | 1.844979 | -0.792673 |
| **3124** | 0.538809 | -1.013813 | -0.434048 | 0.522107 | -0.613512 | 1.552622 | -1.138679 |

# 03 전처리

## 변수 생성 - Groupby

first_acc_x
first_acc_y
first_acc_z

운동 처음 시작부분의 가속도계를 확인하여
각 운동의 시작위치를 구분

1 2 3 4 5 6 7 8 9 10 11 · · · · 150 · · · · · · · · · · · · · · · · · · · · 595 596 597 598 599 600

mean

head_acc_x
head_acc_y
head_acc_z

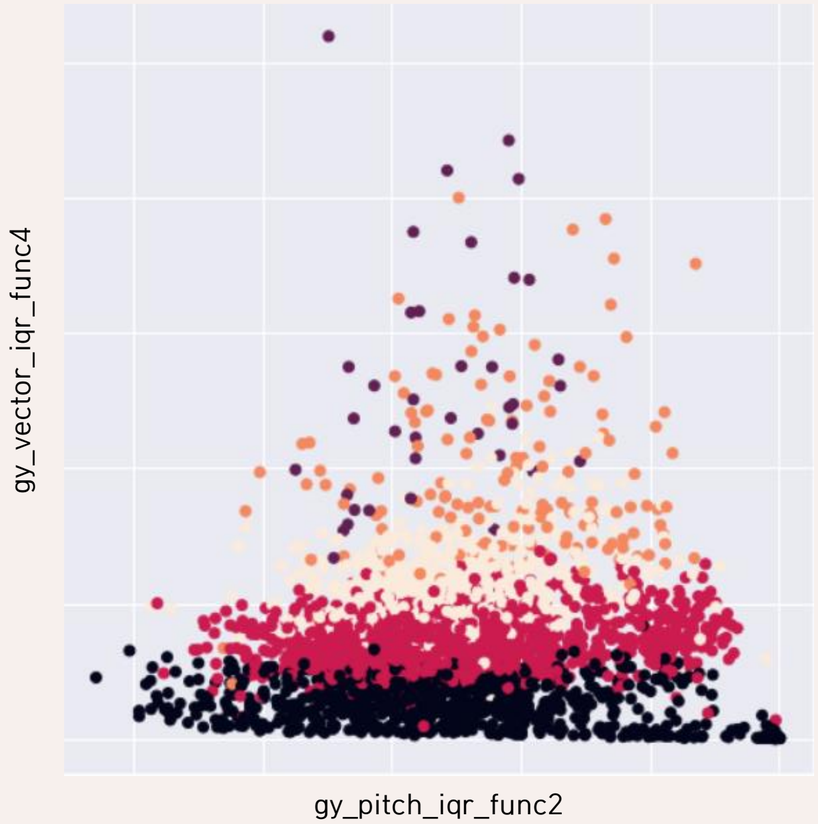
head_gy_x
head_gy_y
head_gy_z

운동시작 3초의 평균으로 Plank와 같은 정적인 운동과
Shoulder Press 같은 동적인 운동을 구분

# 03 전처리

## 변수 생성 - Groupby

### K-means clustering




모든 변수를 기준으로 군집을 5개로 나누어 cluster 변수를 생성하였다.

acc_x_max
acc_x_min
acc_x_mean
acc_x_std
·
·
·
·
·
·
·
gy_pitch_iqr_func4
gy_pitch_preskew
gy_pitch_prekurt

K-means clustering

| id | Cluster |
|---|---|
| 1 | 1 |
| 2 | 0 |
| 3 | 3 |
| · | · |
| · | · |
| · | · |
| · | · |
| · | · |
| 589 | 2 |
| 599 | 0 |
| 600 | 3 |

최종 모델에서 변수 중요도가 가장 높은 두개의 변수를 가지고 만든 scatter plot.
cluster변수에 따라 어느정도 분류됨을 확인할 수 있다.

# 03 전처리

## 최종 변수 - Groupby

acc_x
acc_y
acc_z
gy_x
gy_y
gy_z
acc_vector
gy_vector
gy Centerofgravity
roll
pitch
gy_roll
gy_pitch

✖

max
min
mean
std
median
range_func
iqr_func2
iqr_func3
iqr_func4
preskew
prekurt

head_acc_x
head_acc_y
head_acc_z
head_gy_x
head_gy_x
head_gy_x

first_acc_x
first_acc_y
first_acc_z

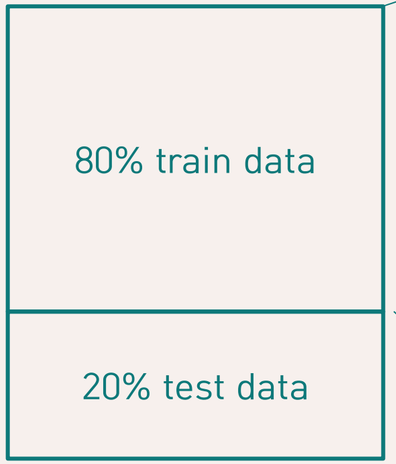
cluster

ID : 3125개

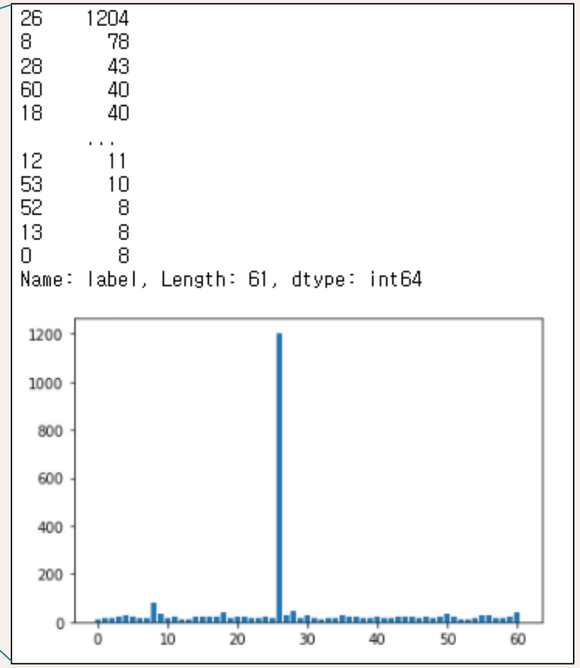
= **153개 변수**

# 03 전처리

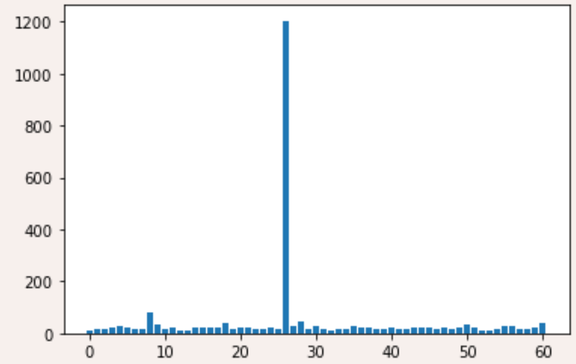
## Augmentation -Groupby




train set, test set 분리




Class Imbalance



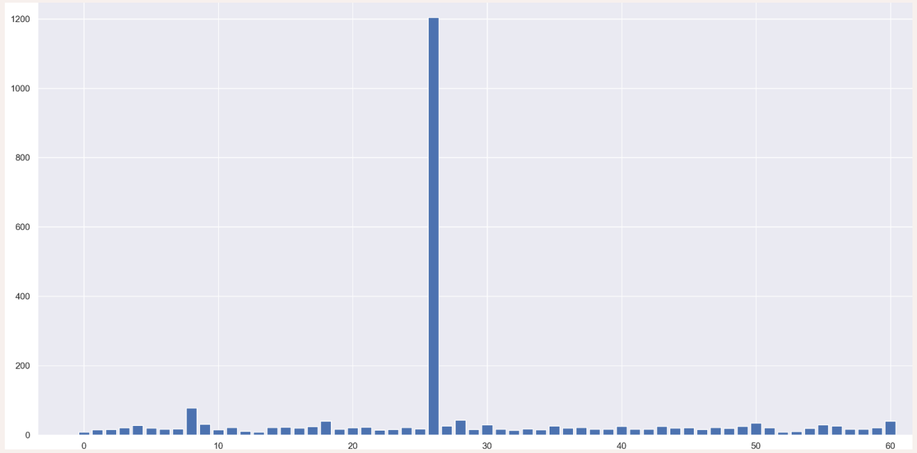

1. Class Imbalanced Data

2. Sampling Data

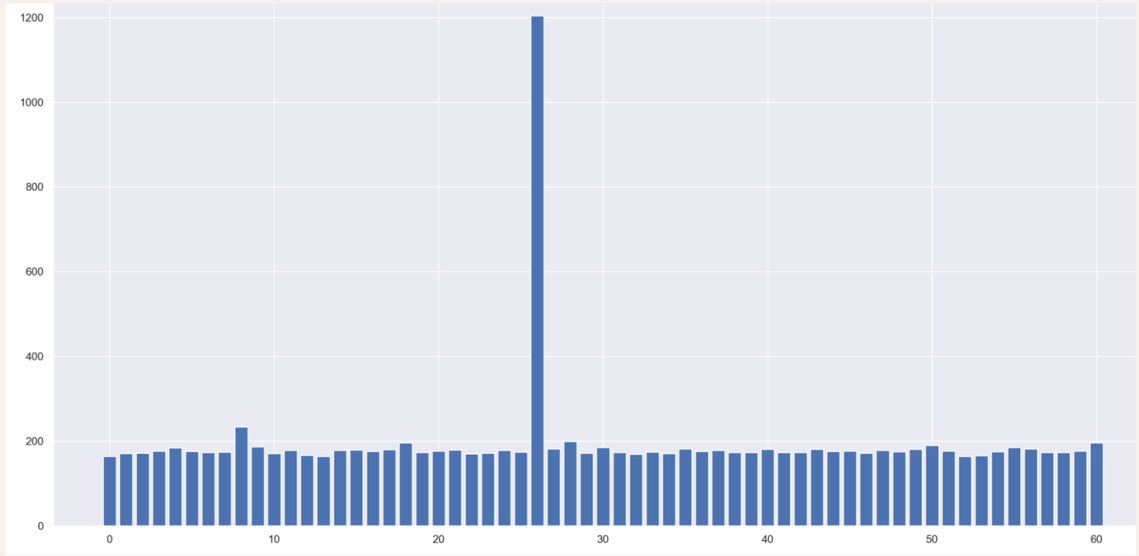
제일 많은 Label : 1204
제일 적은 Label : 8
차이가 심해서 Over Sampling과
Under Sampling을 둘 다 진행

# 03 전처리

## Augmentation -Groupby

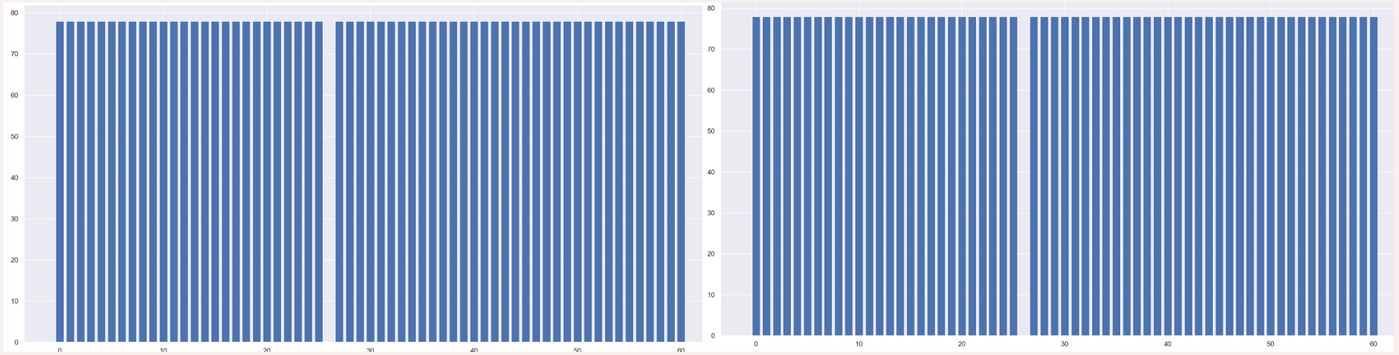



Raw




Over Sampling Data

3개의 데이터를 병합

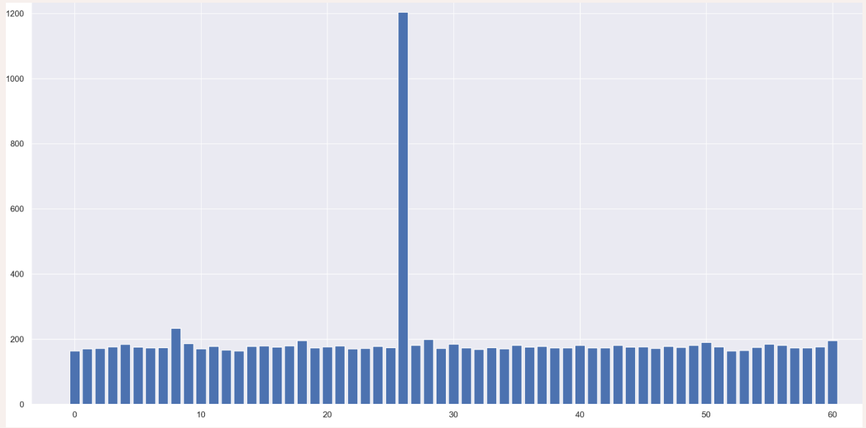



Smote

Random Over Sampling

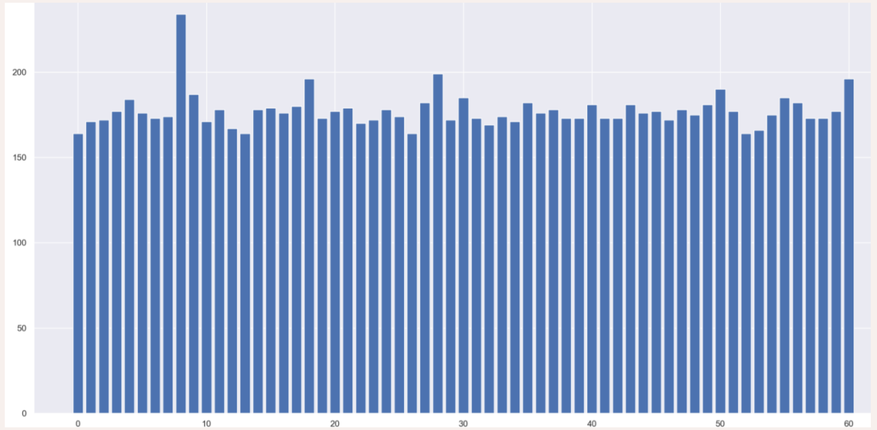
# 03 전처리

## Augmentation -Groupby




Over Sampling Data

Label : 26에 대하여
Random
Under
Sampling




2. Sampling Data

1. Class Imbalanced Data, 2. Sampling Data

2개의 데이터 모두 모델에 적용

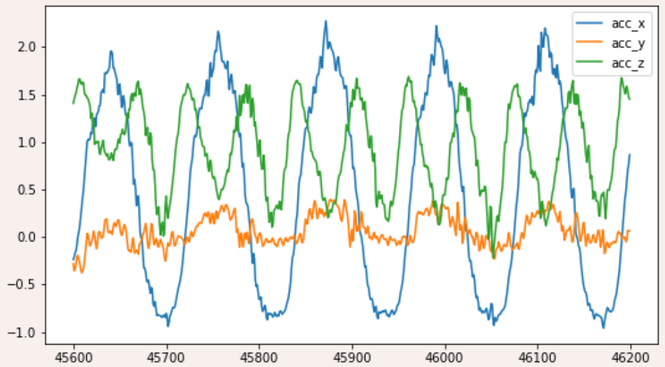
# 03 전처리

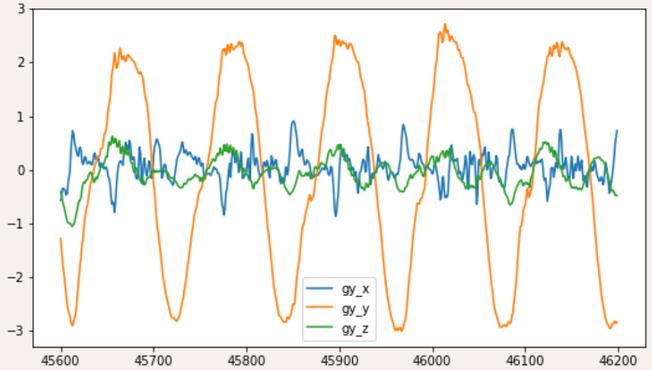
## Reshape - CNN

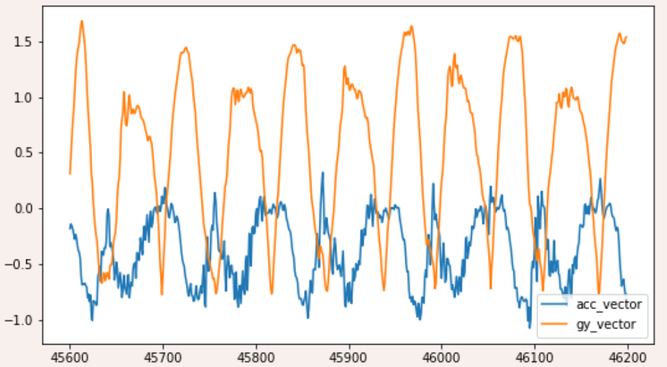
### ▷ Standard Scaling

단위가 다른 변수들이 존재하기 때문에, scaling 진행

| acc_x | acc_y | acc_z | gy_x | gy_y | gy_z | acc_vector | gy_vector | gy Centerofgravity | roll | pitch | gy_roll | gy_pitch |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.206087 | -0.179371 | -0.148447 | -0.591608 | -30.549010 | -31.676112 | 1.228355 | 44.010999 | -20.938910 | -0.146550 | 1.380095 | -0.767200 | -0.013443 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |



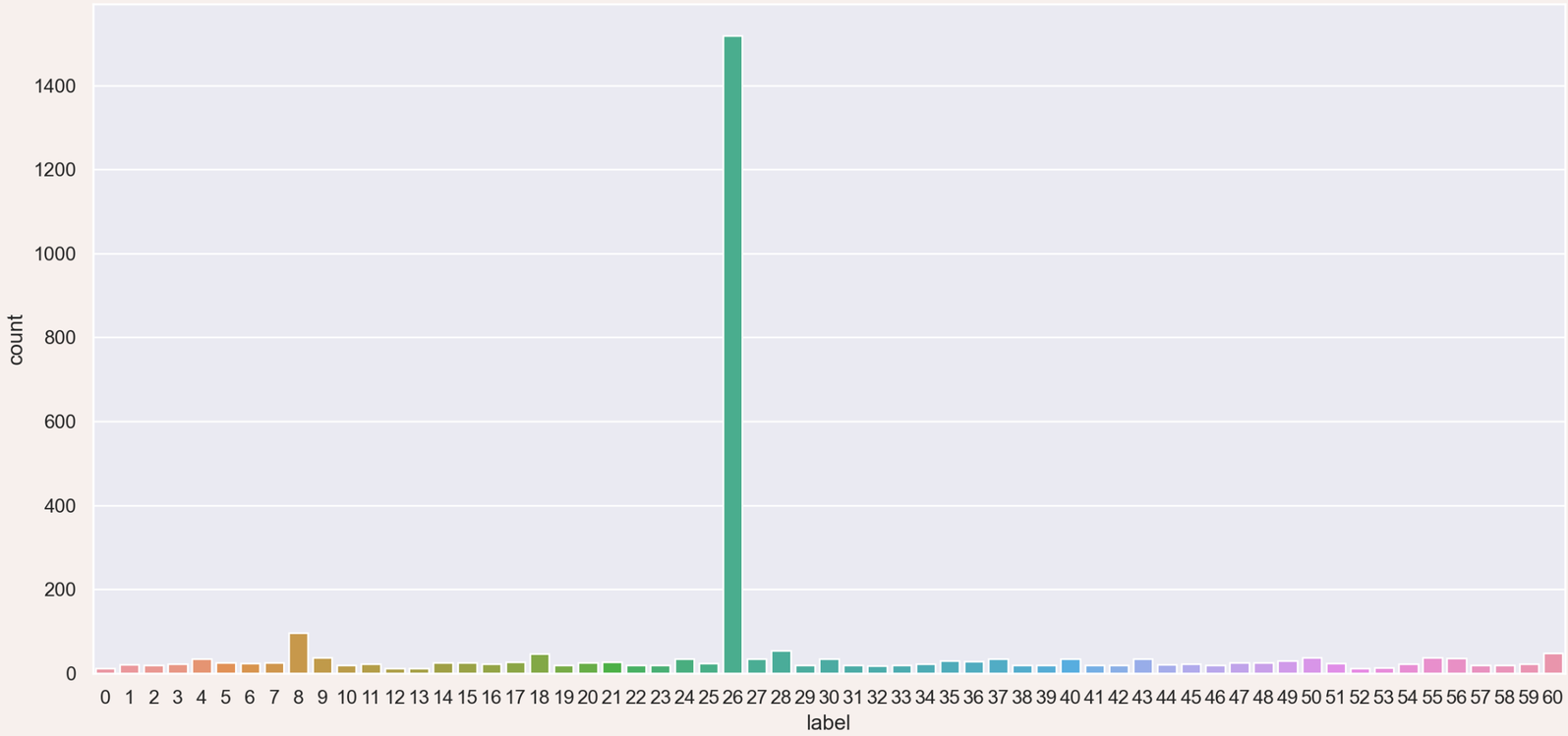




시계열 데이터 -> **1D Convolution**

# 03 전처리

## 불균형 - CNN




**Class Imbalance**

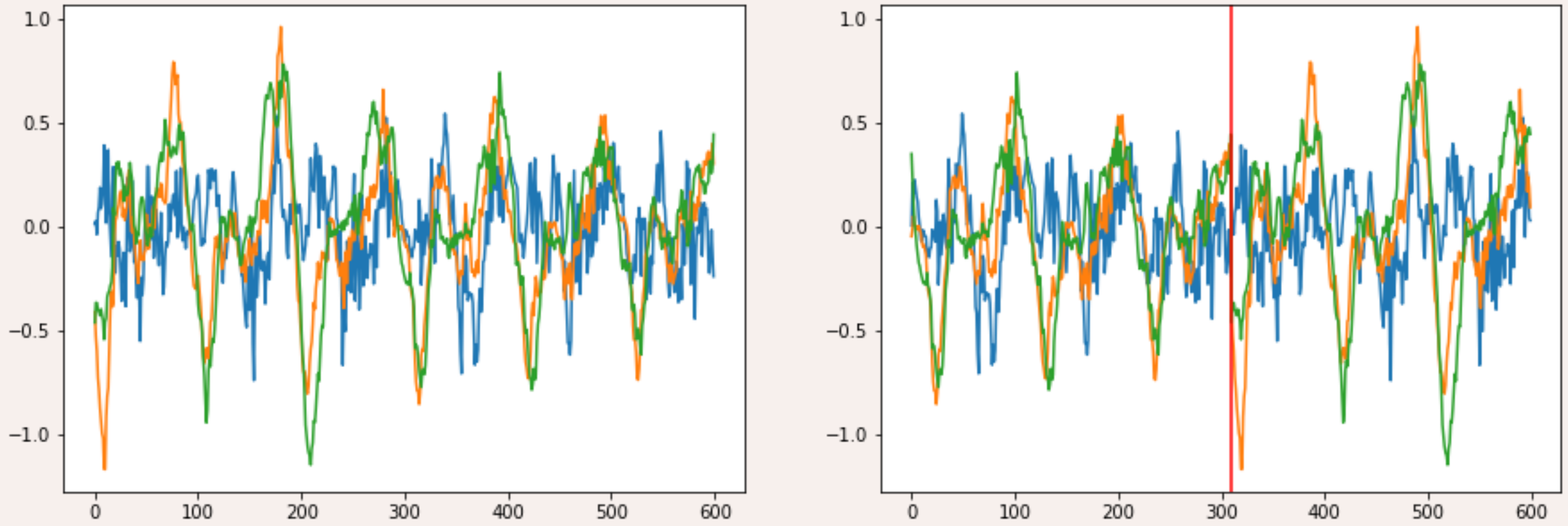
26 ( Non-Exercise ) 전체 데이터의 48.6%

# 03 전처리

## Augmentation

### Rolling

데이터 좌우를 무작위로 이동시켜 데이터 증강




3125개 id -> 6250개 id

# 03 전처리

## Reshape - CNN

### X reshape

2차원 Array ( 3750000, 13 )

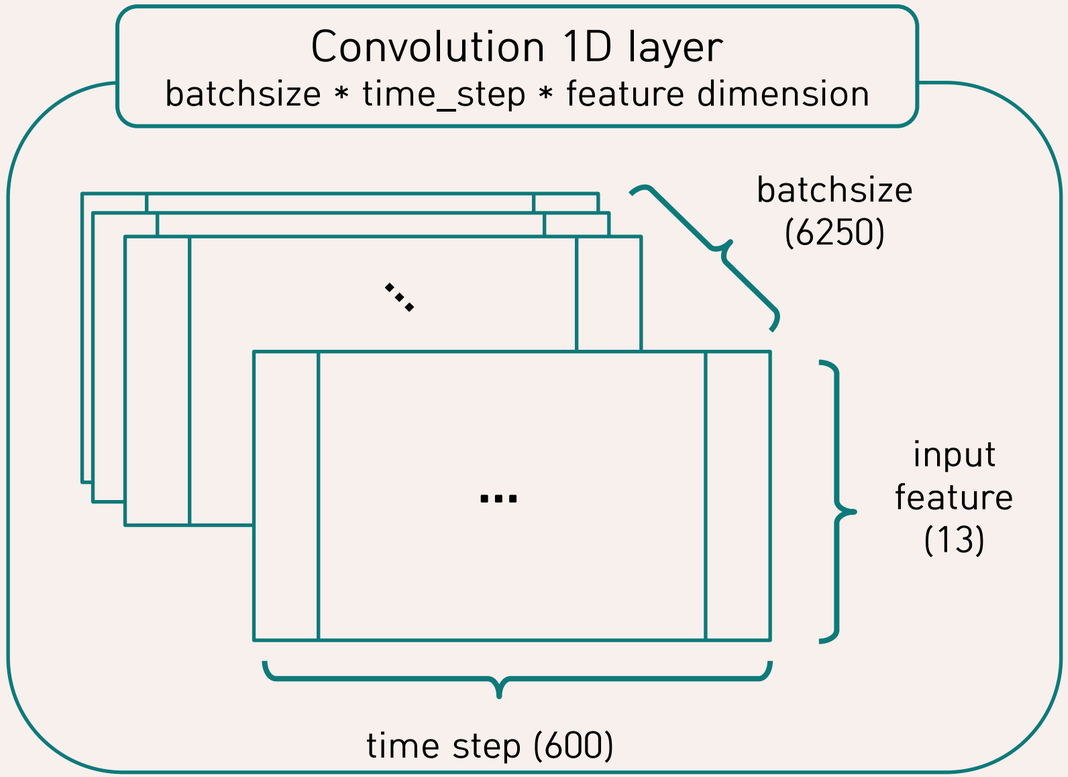
-> 3차원 Array ( 6250, 600, 13 )

3125*2 ids, 600 times, 13 features

test set

2차원 Array ( 469200, 15 )

-> 3차원 Array ( 782, 600, 13 )



# 03 전처리

## Reshape - CNN

### Y reshape

One-hot encoding

정수를 이진 클래스 행렬로 변환 (6250,61)

* 61(0~60) type 운동 분류

ex)

[ [ 1, 0, 0, 0, 0,  ··· 0, 0 ], -> label 0 ( Arm Band Adjustment )

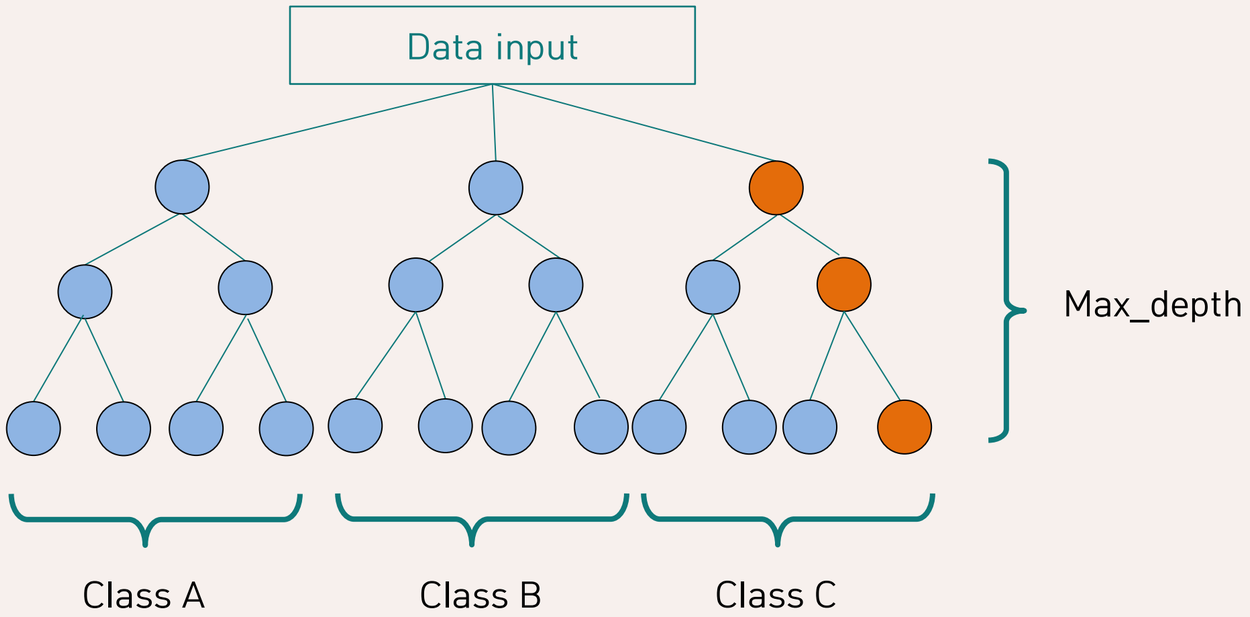
[ 0, 0, 1, 0, 0, ···. 0, 0 ], -> label 2 ( Bicep Curl )

··· ]

**X - (6250, 600, 13) , Y - (6250, 61)**

# 04 모델링

## Random Forest



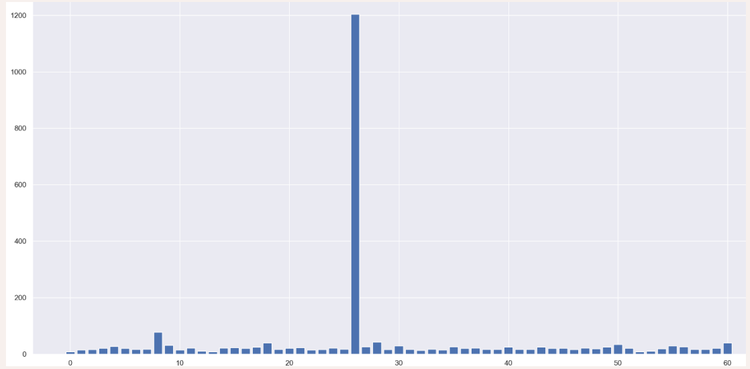

**<Random Forest>**

여러 개의 의사결정나무를 생성

파라미터를 조정함으로써 over-fitting(과적합) 방지.

# 04 모델링

## Random Forest




### 1. Class Imbalanced data

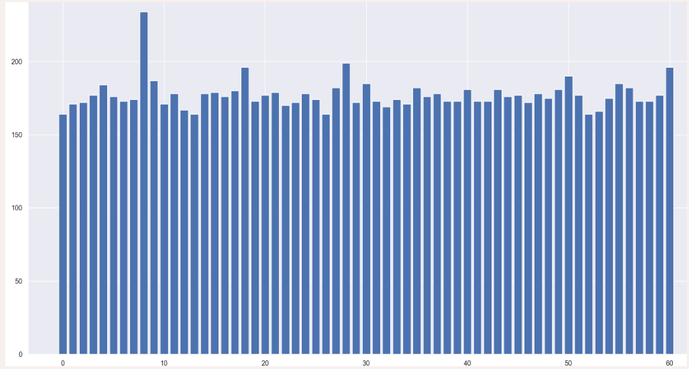
n_estimator = 50
max_depth = 11
min_samples_split = 8
min_samples_leaf =8

최고 예측 정확도 : 0.7236

train.score = 0.8564
test.score = 0.7456

n_estimator = 200
max_depth = 7
min_samples_split = 8
min_samples_leaf =8

train.score = 0.7868
test.score = 0.7312




### 2. Sampling Data

n_estimator = 100
max_depth = 11
min_samples_split = 8
min_samples_leaf =8

최고 예측 정확도 : 0.9329

train.score = 0.9593
test.score = 0.8016

n_estimator = 200
max_depth = 8
min_samples_split = 8
min_samples_leaf =8

train.score = 0.8196
test.score = 0.7712

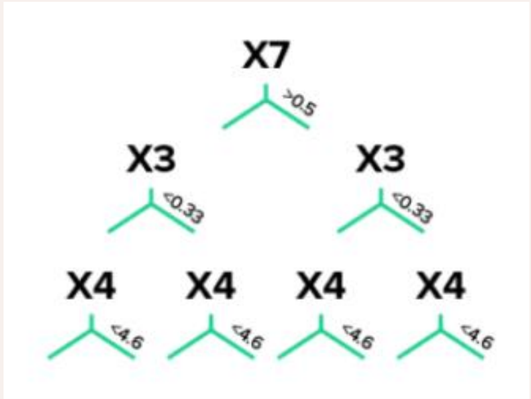
# 04 모델링
## Catboost

### Catboost?

데이터의 일부를 학습시켜 만든 모델로 다른 데이터를 예측한 후 알게되는 잔차를 이용

지속적인 학습으로 잔차를 줄이는데 중점을 둔 알고리즘

Stochastic Gradient Descent(확률적 경사 하강법)와 매우 유사




### Catboost 특징

Boosting 계열 중 GBM기반으로 만들어진 알고리즘

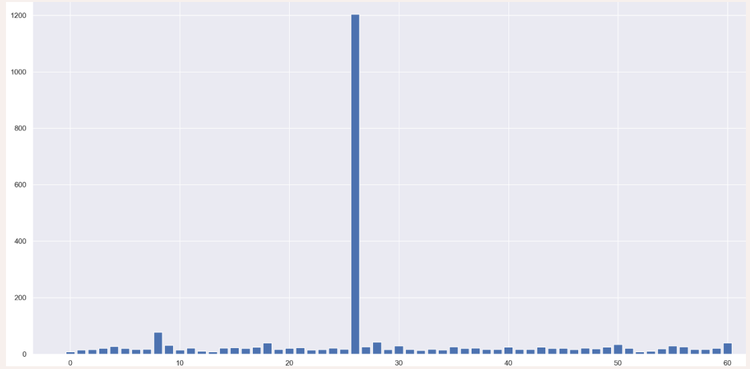
Symmetric Tree 구조

순서가 있는 부스팅 - Ordered Boosting

데이터를 N개의 Fold로 나누어서 각 Fold에 속한 데이터셋들에 Ordered Boosting을 적용 - 과적합 방지

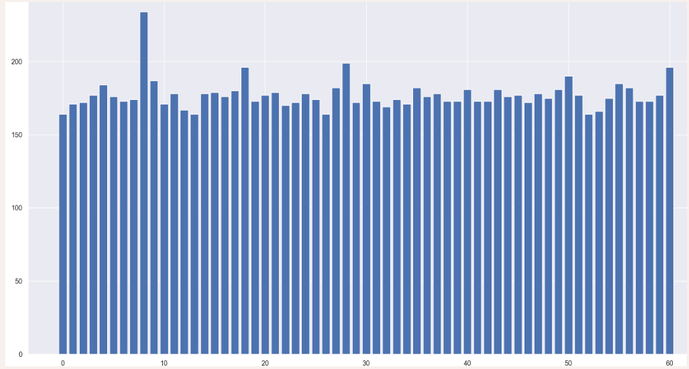
# 04 모델링

## Catboost




1. Class Imbalanced data

train_score= 1.0
test_score= 0.8256




2. Sampling Data

train_score= 1.0
test_score= 0.7952

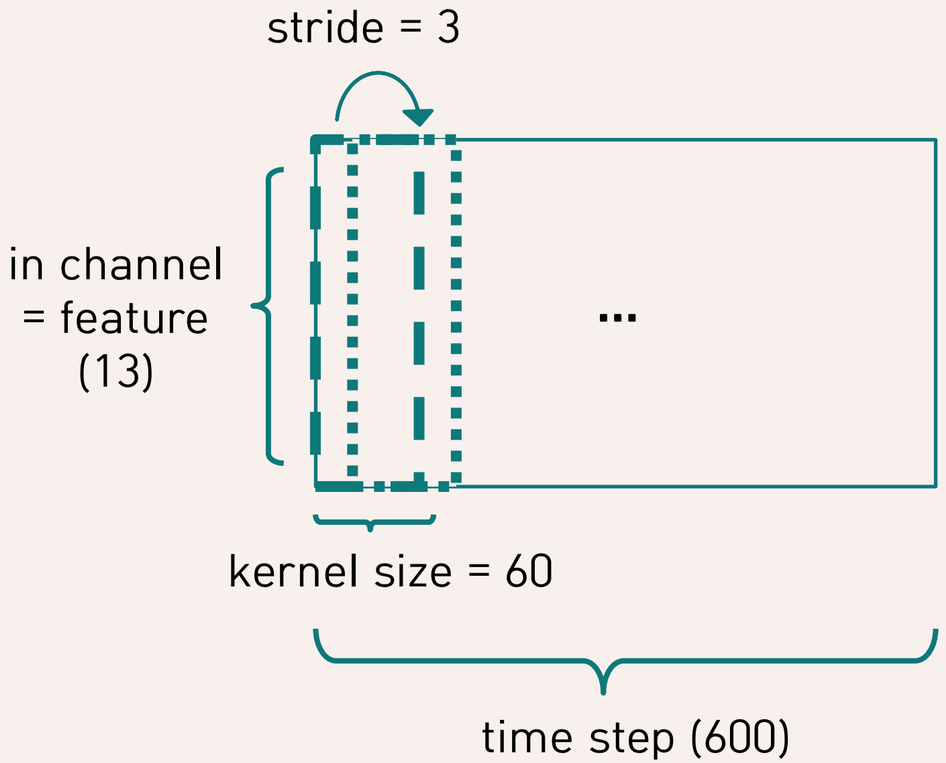
Catboost 모델은 하이퍼파라미터의 최적화가 잘 되어있기 때문에
default 값으로 진행

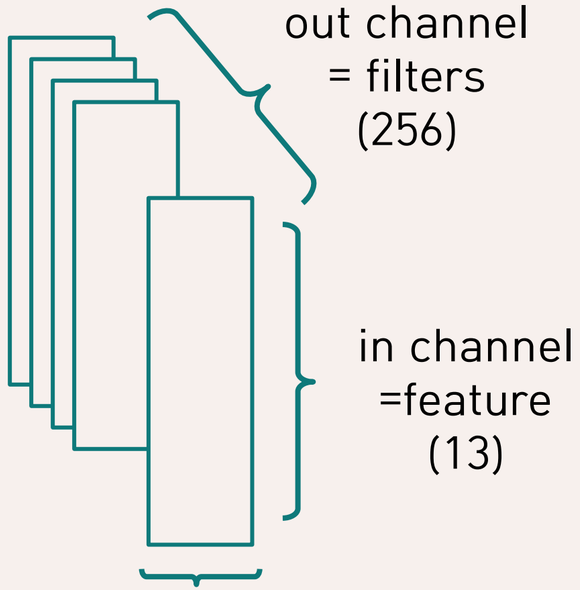
# 04 모델링

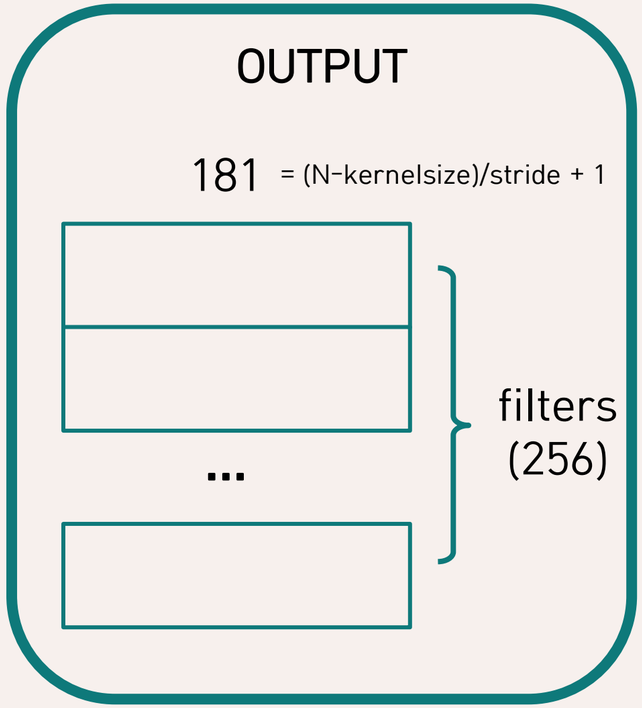
## CNN

### Convolution 1D Layer

- kernel_size = 60
- filters = 256
- strides = 3
- input_shape = [600,13]



# 04 모델링

## CNN

### Convolution 1D Layer

- kernel_initializer = ‘he_uniform’
- activation = ‘relu’

#### he_uniform

[ -limit , limit ] 범위를 가진 균등분포에서 값 선택

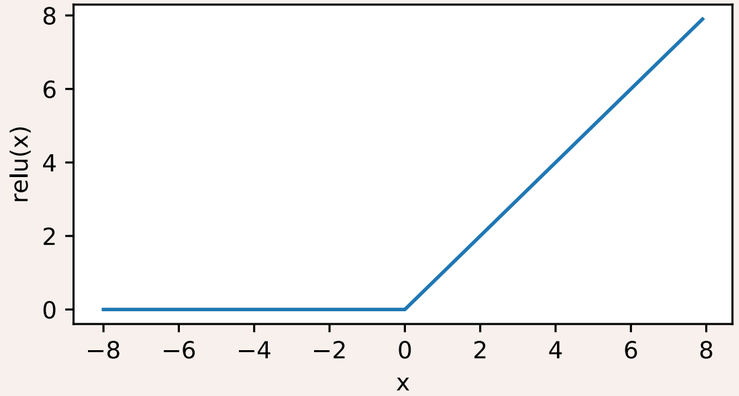
* 가중치 초기화

: Gradient exploding/vanishing, local minimum과 같은 문제 방지

#### relu

-는 0으로 +는 그대로 반환

Gradient vanishing 해결



## Batch Normalization Layer

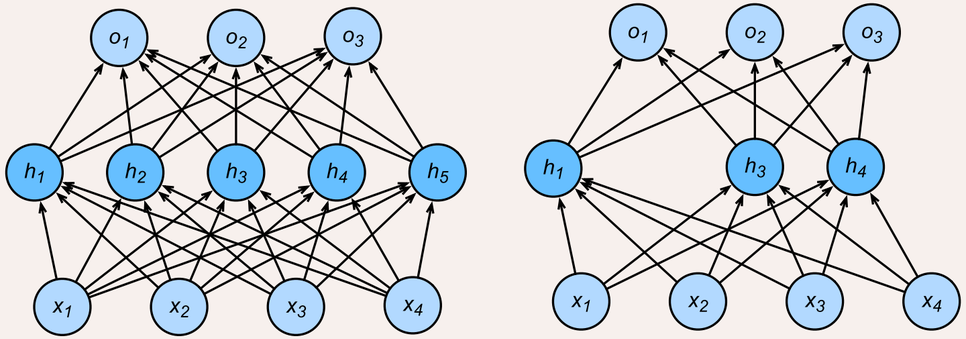
신경망 안에 포함되어 미니배치 평균과 분산을 조정(scale, shift)하면서 출력값을 정규화

Convolution layer에서 채널별로 normalization

Gradient Exploding / Vanishing 문제 해결과 Overfitting 위험 줄임

## Dropout Layer

- drop = 0.2




임의의 노드를 드랍해서 학습시키지 않음

Overfitting 방지

# 04 모델링

## CNN

Conv1D kernel_size=60 filters=256 strides=3
Batch Normalization
Dropout

output shape (None, 181, 256)

Conv1D kernel_size=60 filters=128 strides=1
Batch Normalization
Dropout

output shape (None, 122, 128)

Conv1D kernel_size=60 filters=64 strides=1
Batch Normalization

output shape (None, 63, 64)

# 04 모델링

## CNN

### Global Average Pooling 1D Layer

Pooling : convolution layer resizing - 크기를 줄이거나 특정부분을 강조

Global Average Pooling 1D

: feature map상의 노드들의 평균을 계산해 1차원 벡터로

### Dense Layer ▪ 61

input 과 ouput의 모든 뉴런 결합

### Activation Layer ▪ softmax

다중클래스 분류에서 사용

61차원의 벡터를 입력 받아 각 클래스에 속할 확률 반환

# 04 모델링

## CNN

### Sequential Model

| | |
|---|---|
| Conv1D | |
| Batch Normalization | |
| Dropout | output shape (None, 181, 256) |
| Conv1D | |
| Batch Normalization | |
| Dropout | output shape (None, 122, 128) |
| Conv1D | |
| Batch Normalization | output shape (None, 63, 64) |
| GlobalAveragePooling1D | output shape (None, 64) |
| Dense | |
| Activation | output shape (None, 61) |

## Model Compile

optimizer = adam ( learning rate = 0.001 )

Adam : momentum과 gradient 히스토리 모두 고려

loss = categorical cross entropy

one-hot 형태로 제공된 다중 클래스 분류

metrics = accuracy

# 04 모델링

## CNN

### Stratified K-fold

n_splits = 5, shuffle = True

레이블 데이터 분포에 따라 분류

### Model fit

epoch = 50

early stopping 사용 : monitor = val_loss, patience = 8

batch_size = 64

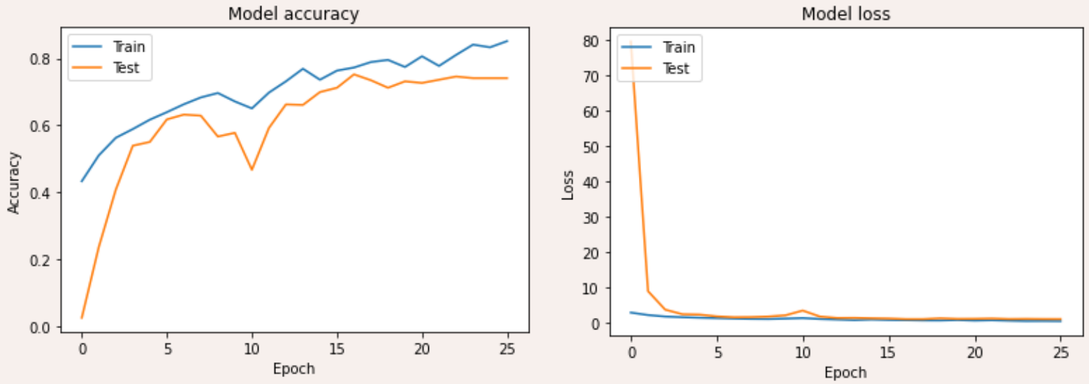
# 04 모델링

## CNN

### No Augmentation

5 fold 평균 accuracy = 0.74624

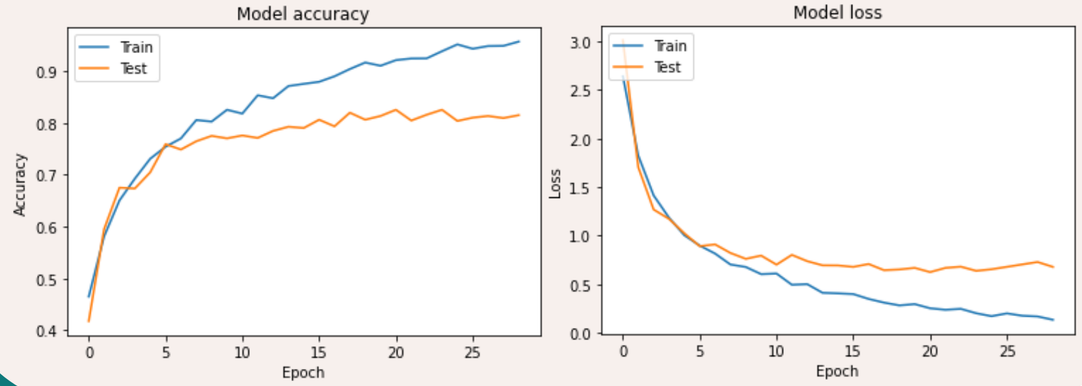
5 fold 평균 loss = 1.05870




### Augmentation

5 fold 평균 accuracy = 0.81312

5 fold 평균 loss = 0.72797



# 04 모델링

## CNN

Augmentation

5 fold 평균 accuracy = 0.81312

5 fold 평균 loss = 0.72797

acc_x, acc_y, acc_z,
gy_x, gy_y, gy_z,
Acc_vector, gy_vector

**No Augmentation** - 변수 8개 모델(증강없음)

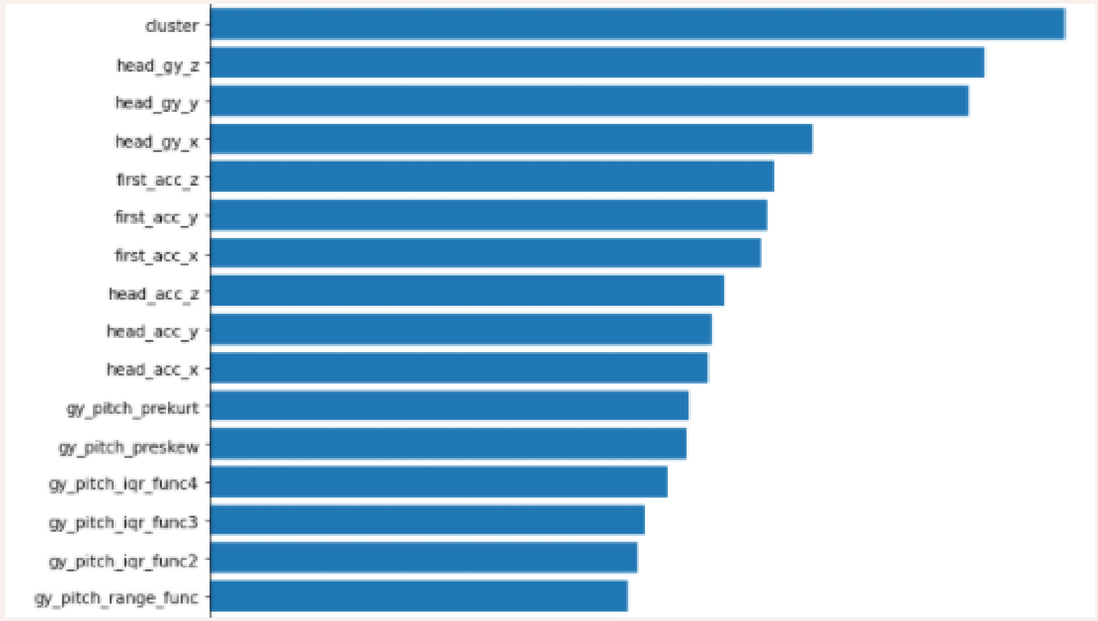
5 fold 평균 accuracy = **0.78496**

5 fold 평균 loss = **0.83028**

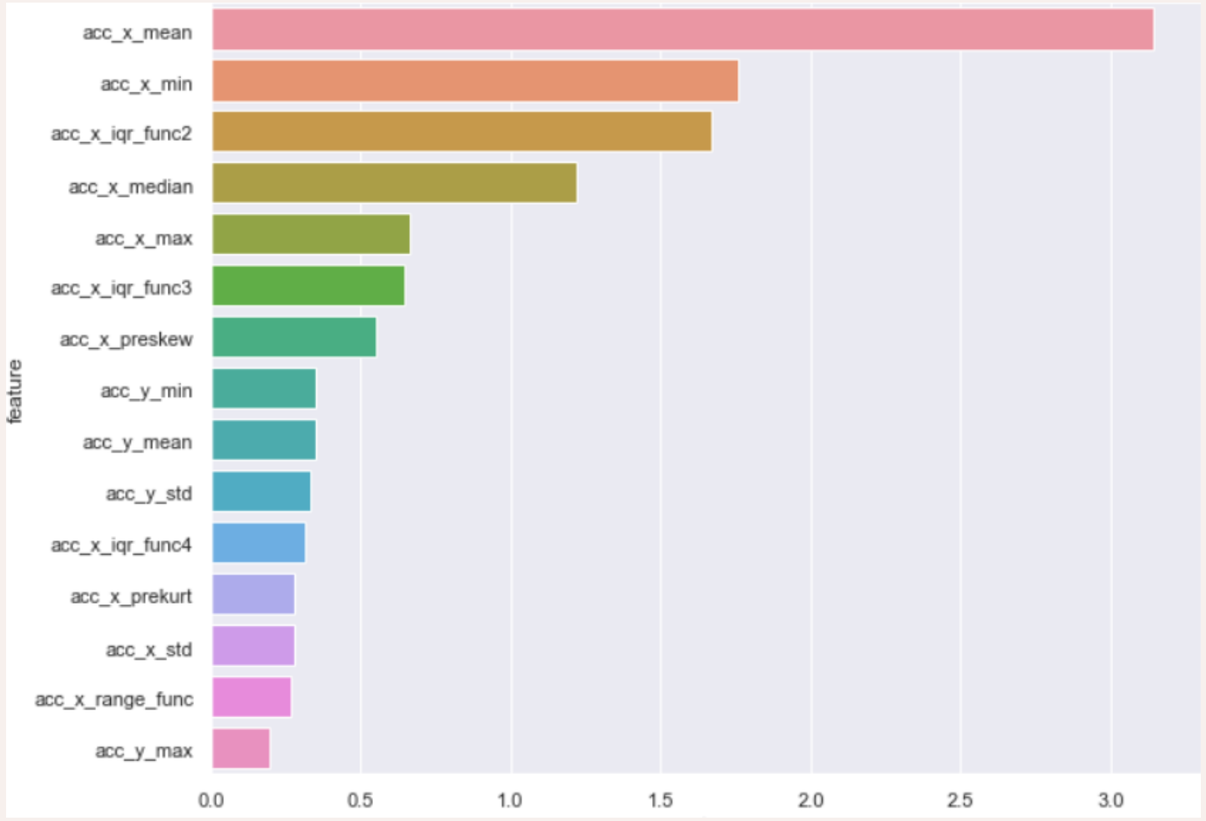
Raw Data 임에도 좋은 성능을 보여줌

# 05 결론

## 변수 중요도




Random Forest




Catboost

모델별 변수 중요도가 다르다.

Random Forest는 자이로 센서와 cluster 변수의 중요도가 높고

Catboost는 가속도 센서  변수의 중요도가 높았다.

# 05 결론

## 변수 중요도

CNN

Feature : 8,  NO Augmentation ->  0.59/0.63

Feature : 13,  NO Augmentation->  1.16/1.10

Feature : 13,  Augmentation-> 1.49/1.46

(Pri/Pub)

CNN 모델 성격상 모델이 데이터의 특성을 알아서 찾기 때문에,

추가한 변수들이 분류 모델에 중요한 역할을 하지 않았다고 생각

# 05 결론

## 모델 장단점

| | 장점 | 단점 |
|---|---|---|
| Random Forest | -빠른 학습시간<br>-간단한 알고리즘<br>-과적합 방지 | -차원이 큰 데이터는 성능 미흡<br>-트리가 증가할수록 메모리 소모 |
| Catboost | -Random Forest에 비해 대규모 데이터에서 좋은 결과(정확도)<br>-매개변수에 대한 최적화가 잘되어있음 | -연속형 변수가 많을 때, 느린 학습 속도<br>-데이터가 클수록 딥러닝의 성능이 향상되기 때문에 전처리에 따라 CNN보다 낮은 성능 가능성 |
| CNN | -Raw Data 와 최소한의 feature로 훈련 가능<br>-Conv1D는 고정된 시간에 따라 기록된 데이터 처리에 용이<br>-1차원 이미지로 취급하고 중요한 특징 추출 | -큰 연산량, 3개의 모델 중 가장 느림<br>-인접한 데이터 간의 정보가 중요한 경우 RNN(LSTM)이 더 효과적 |

DACON test 점수 1등

# 05 결론

## 한계점

Random Forest - 과적합은 방지되었지만, 모델 정확성에서 한계가 있음

Cat Boost - 잔차를 기반으로 학습이 이루어짐, 오버 샘플링을 통해 같거나 비슷한 데이터가 많이 생성되었기에 비슷한 잔차에 대해 더 집중하였을 것

CNN의 경우 모델 학습시간이 오래 걸려 Rolling(shifting)외 crop, magnitude warping 등 다양한 증강기법을 적용해보지 못함.

CNN - LSTM등, CNN - Ensemble을 사용하지 못함

세개의 모델을 사용했기에, 하나의 모델 전처리에 집중하지 못함.

데이터 불균형 해소에 따른 점진적인 결과를 확인하지 못했음

# 감사합니다