



# 운동 동작 분류 AI

2017010715허지혜



# 01. 개요

**운동 동작 분류 AI 경진대회**  
월간 데이콘 11 | 헬스 데이터 | Logloss | 분류

상금 : 100만원  
2021.01.11 ~ 2021.02.22 17:59 [+ Google Calendar](#)  
418팀 D-28

참여중

[주제 및 배경]

운동 동작 인식 알고리즘 개발

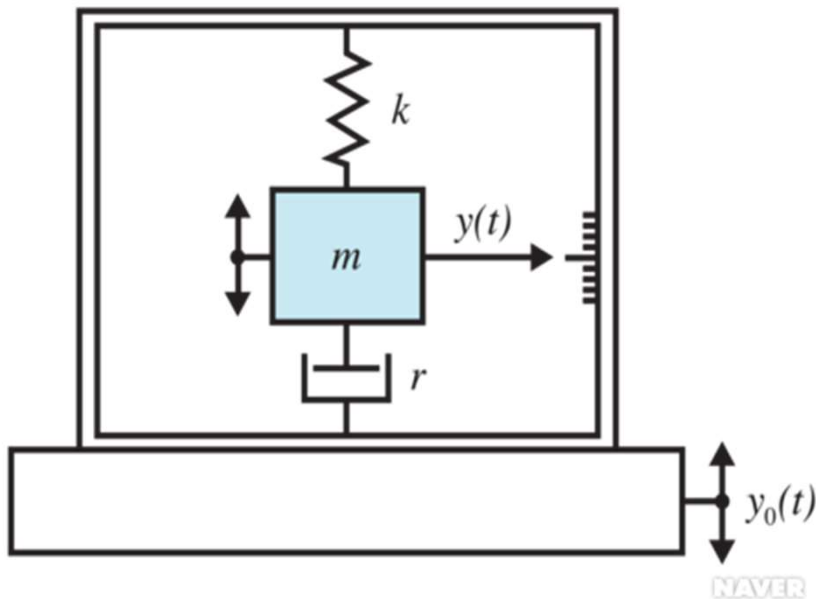
스마트 헬스케어 산업에 적용 가능한 데이터 분석 방법

[데이터]

3축 가속도계와 3축 자이로스코프를 활용한 센서 데이터

# 01. 개요

## 3축 가속도계

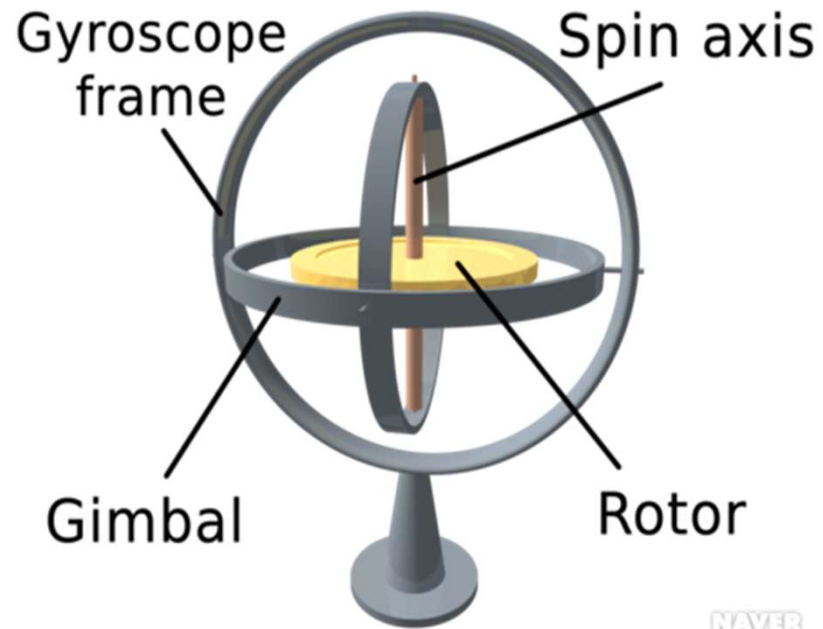


### 가속도계란 ? 가속도 측정 센서

가속도 센서가 3축이라고 함은 센서가 3차원에서 움직일 때 x축, y축, z축 방향의 가속도를 측정할 수 있다는 얘기  
기본적으로 정지 상태에서 중력 가속도를 감지하므로 z축 방향으로 -값이다.

# 01. 개요





## 3축 자이로스코프



## 자이로스코프

회전체의 역학적인 운동을 관찰하는 기구  
가속도 센서와 달리 각속도를 측정함

## 02. 데이터 분석

 sample_submission	2021-01-19 오후 ...	Microsoft Excel...	98KB
 test_features	2021-01-19 오후 ...	Microsoft Excel...	58,021KB
 train_features	2021-01-19 오후 ...	Microsoft Excel...	221,111
 train_labels	2021-01-19 오후 ...	Microso	

Test\_feature

Train\_feature

학습

model

Prediction

Train\_labels

submission



## 02. 데이터 분석

train\_feature ,test\_feature

train\_labels

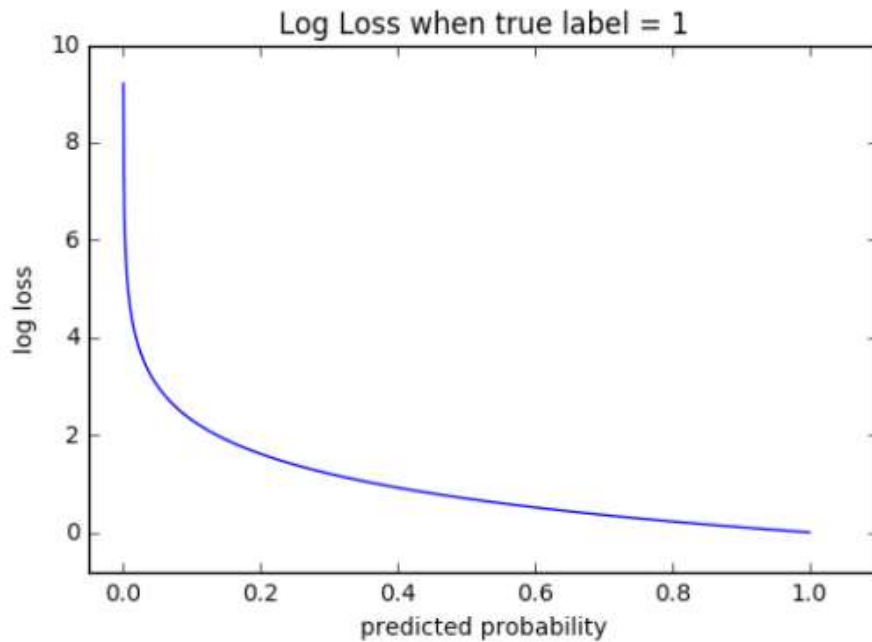
id	time	acc_x	acc_y	acc_z	gy_x	gy_y	gy_z	id	label	label_desc
0	0	0	1.206087	-0.170271	0.148447	0.501608	20.540010	21	0	Press (dumbbell)
1	0	1	1.287696	-	-	-	-	22	0	Non-Exercise
2	0	2	1.304609	-	-	-	-	23	0	Biceps Curl (band)
3	0	3	1.293095	-	-	-	-	24	0	Non-Exercise
4	0	4	1.300887	-0.187757	-0.222523	4.286707	-57.906561	25	0	Non-Exercise

61개의 운동을 맞추는 분류 문제

Id 별 600초간 3축 가속도계와 3축 자이로스코프 데이터

Id 별 운동 종류 데이터

## 02. 데이터 분석

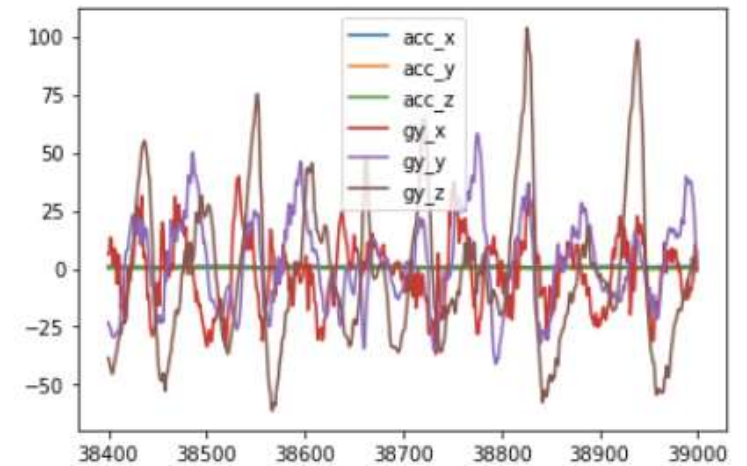
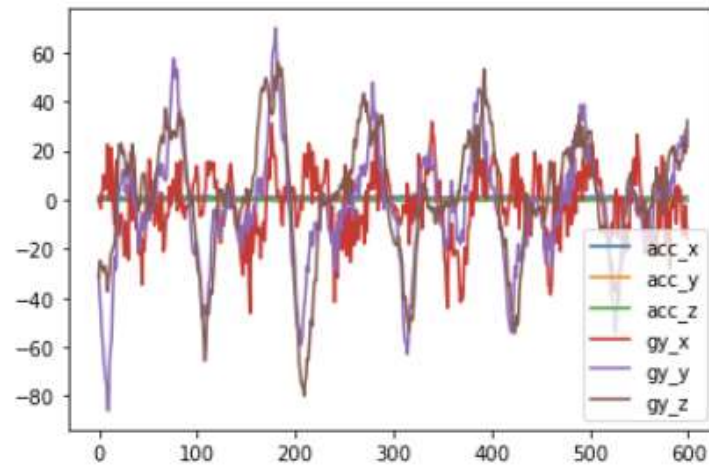
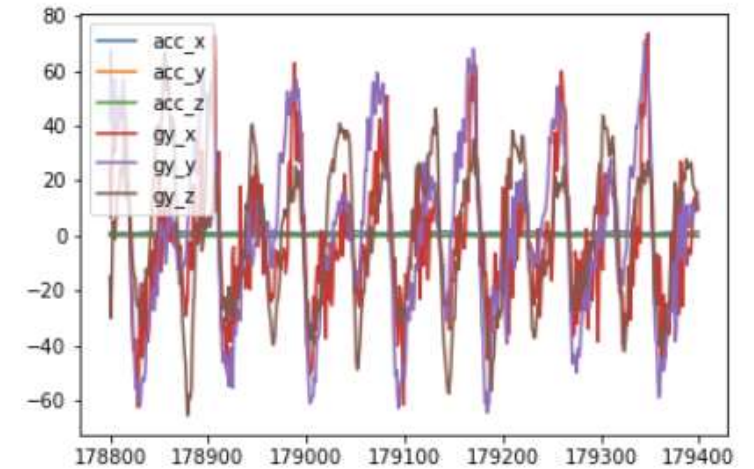
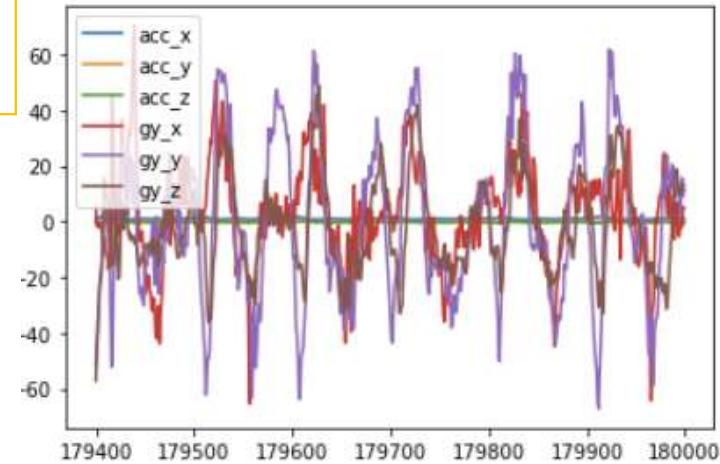


평가 지표 : Log Loss

모델 성능 평가 시 사용 가능한 지표로 분류 모델 평가에 사용한다.  
확률을 기반으로 하는 분류 지표이다.  
로그 손실 값이 낮을수록 더 나은 예측을 의미한다.

## 02. 데이터 분석

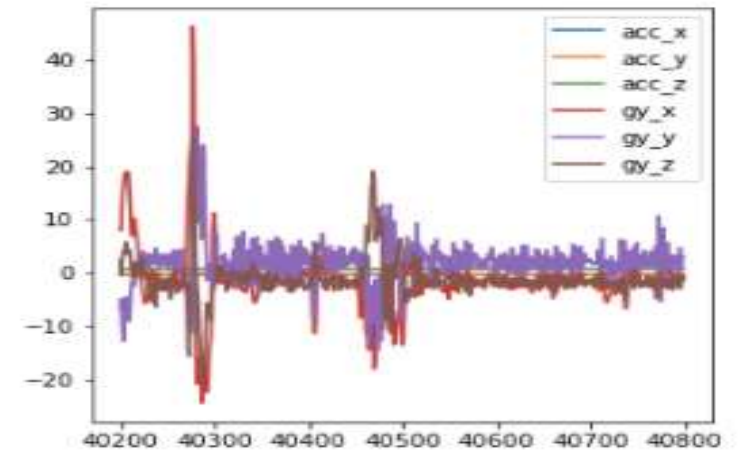
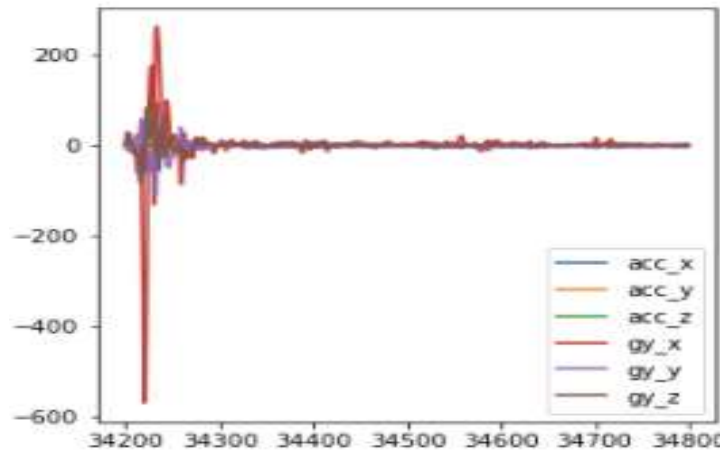
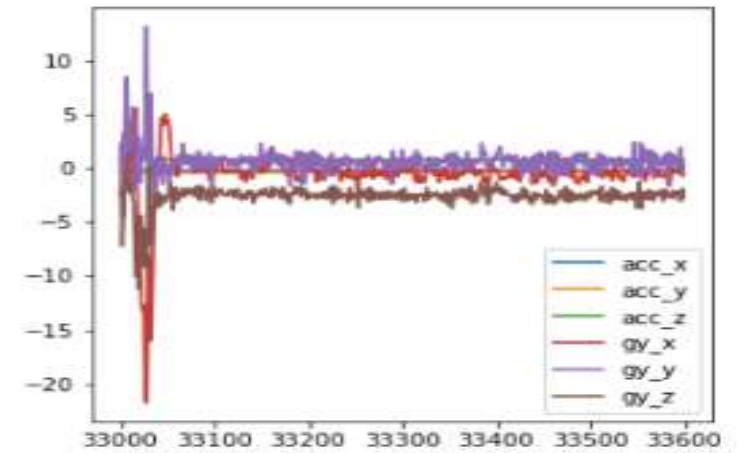
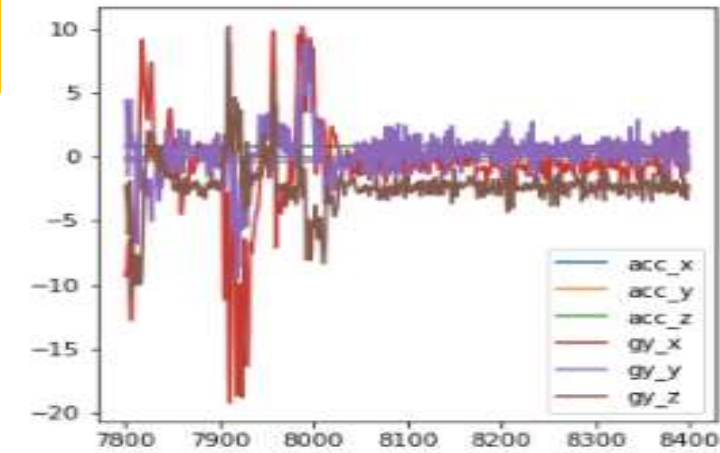
Shoulder Press(dumbbell)





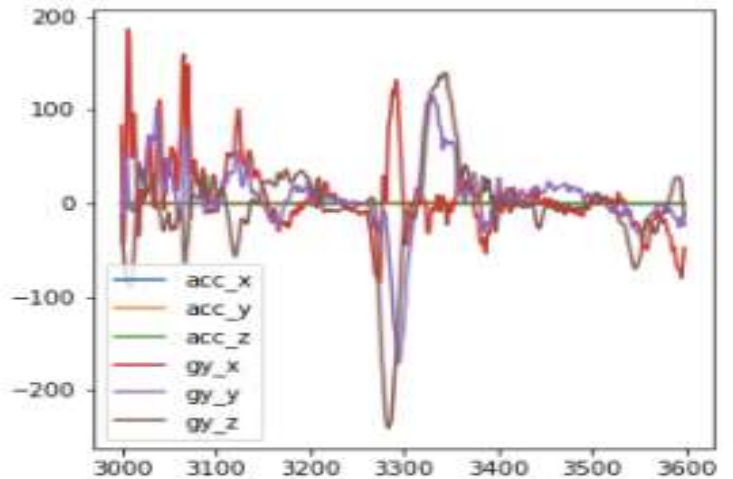
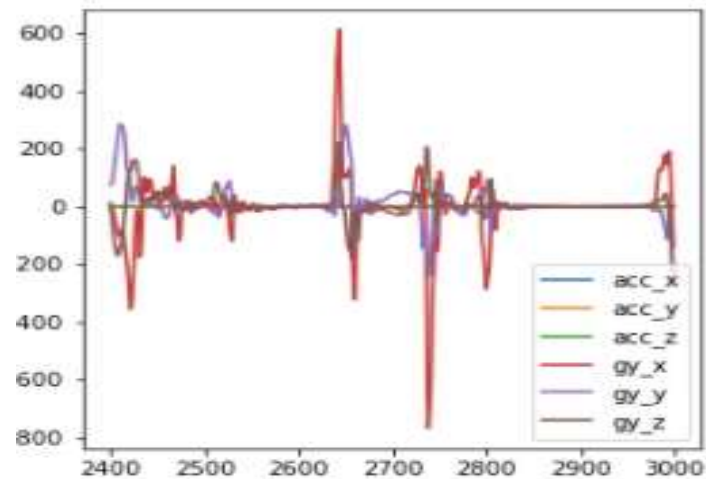
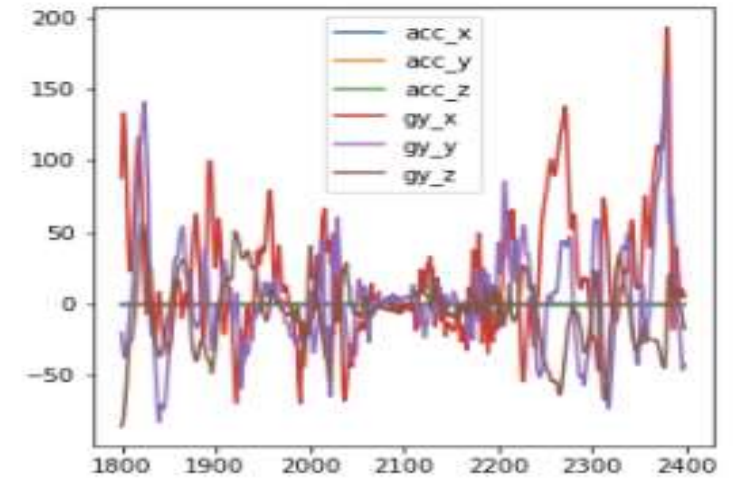
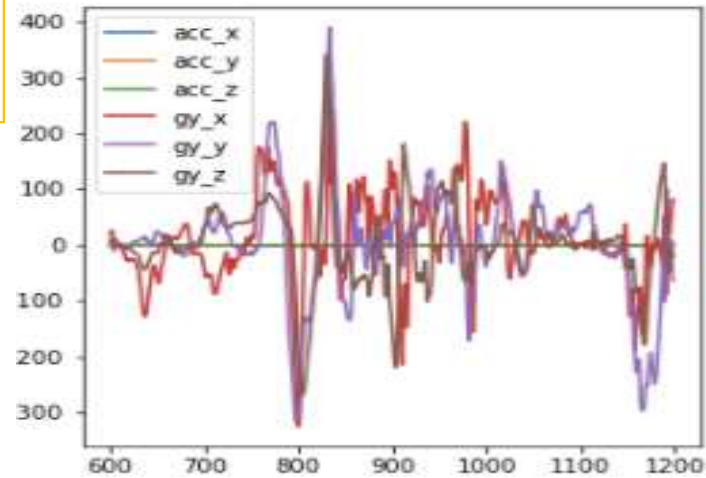
## 02. 데이터 분석

Plank

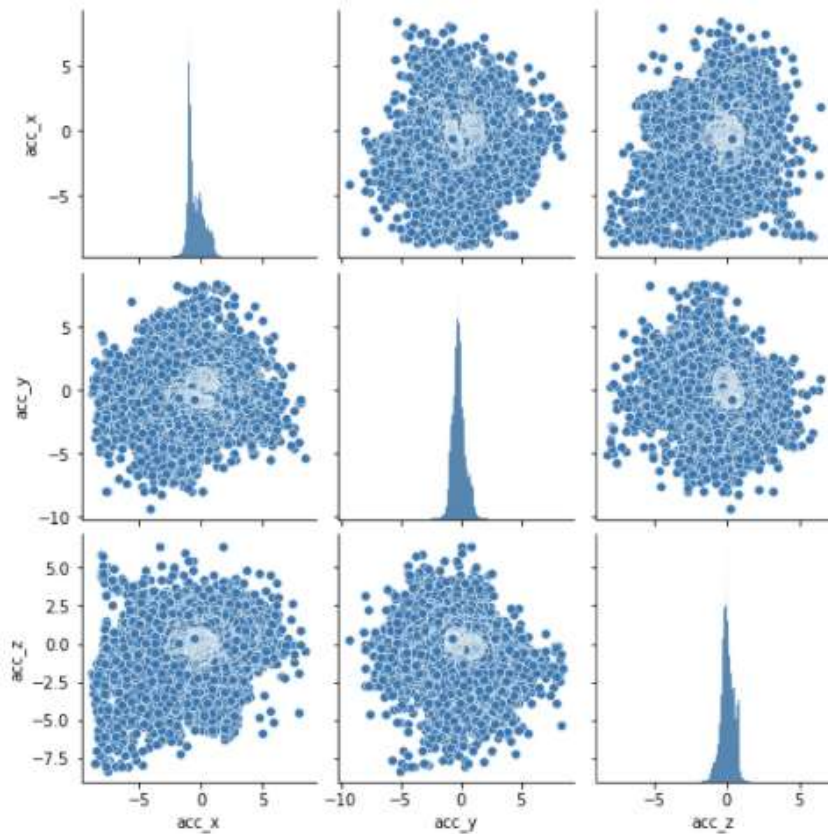


## 02. 데이터 분석

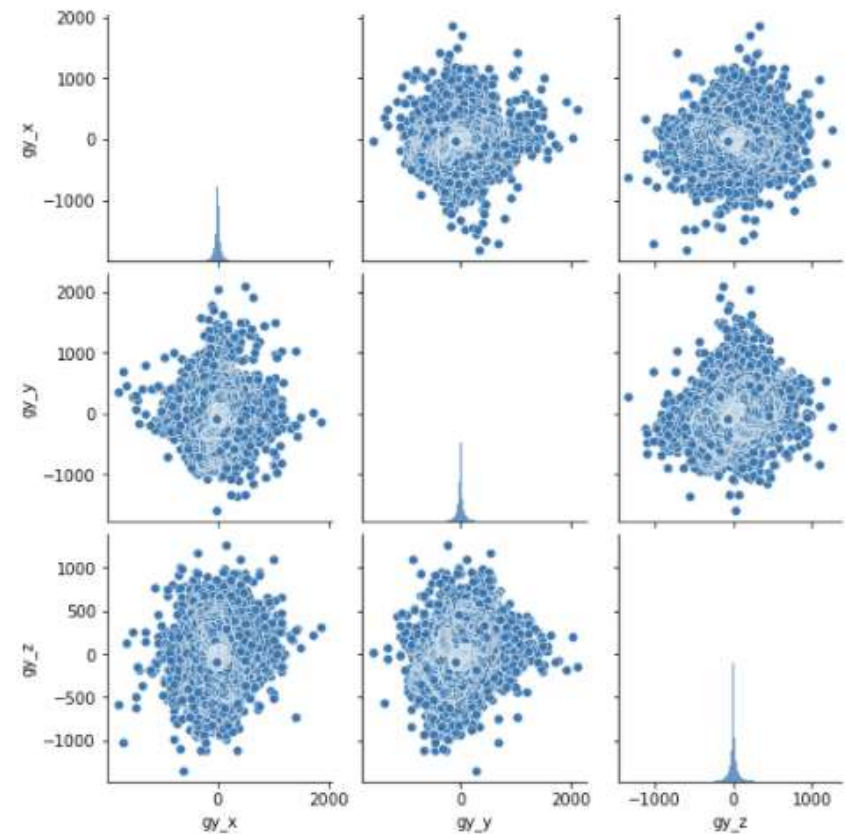
Non-Exercise



## 02. 데이터 분석



가속도계(ACC) 분포

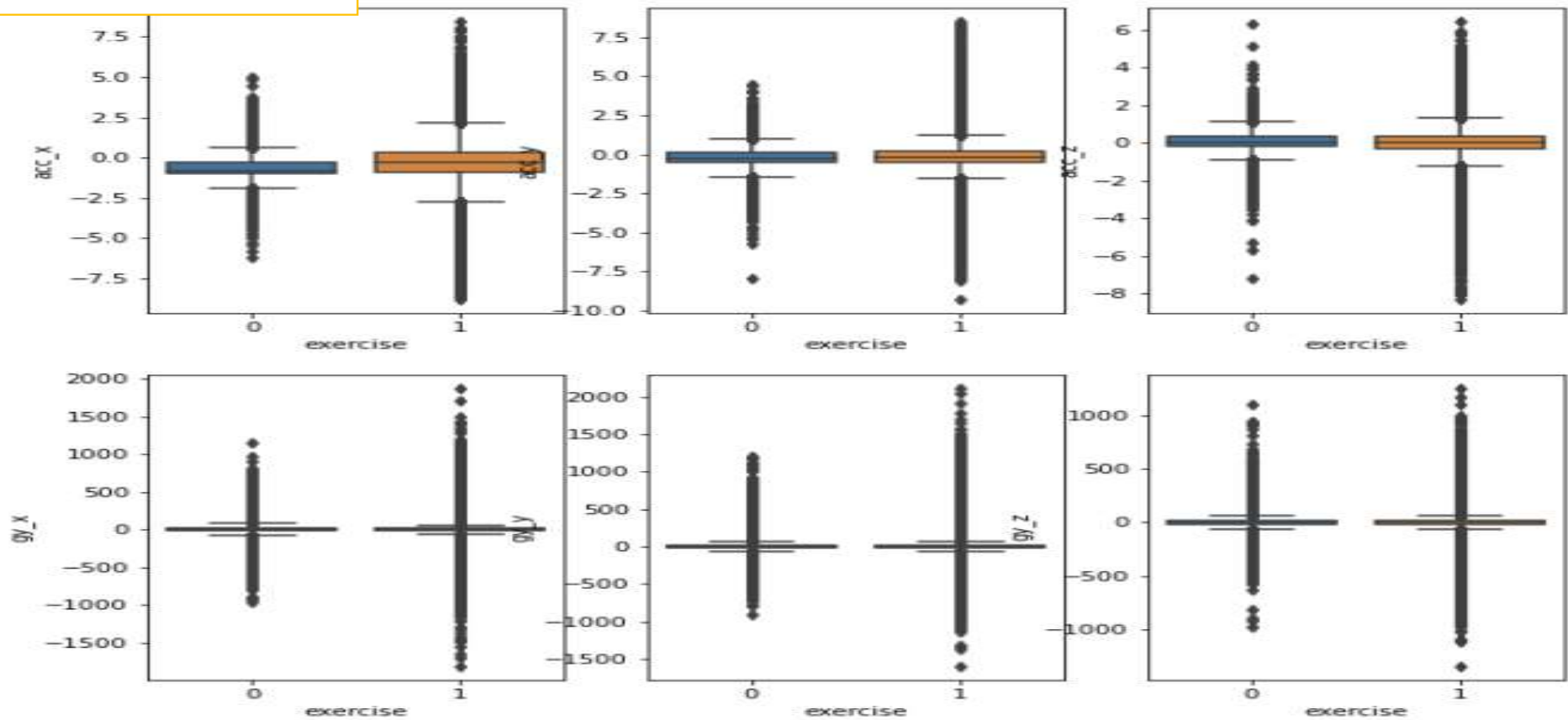


중력(GY) 분포

## 02. 데이터 분석

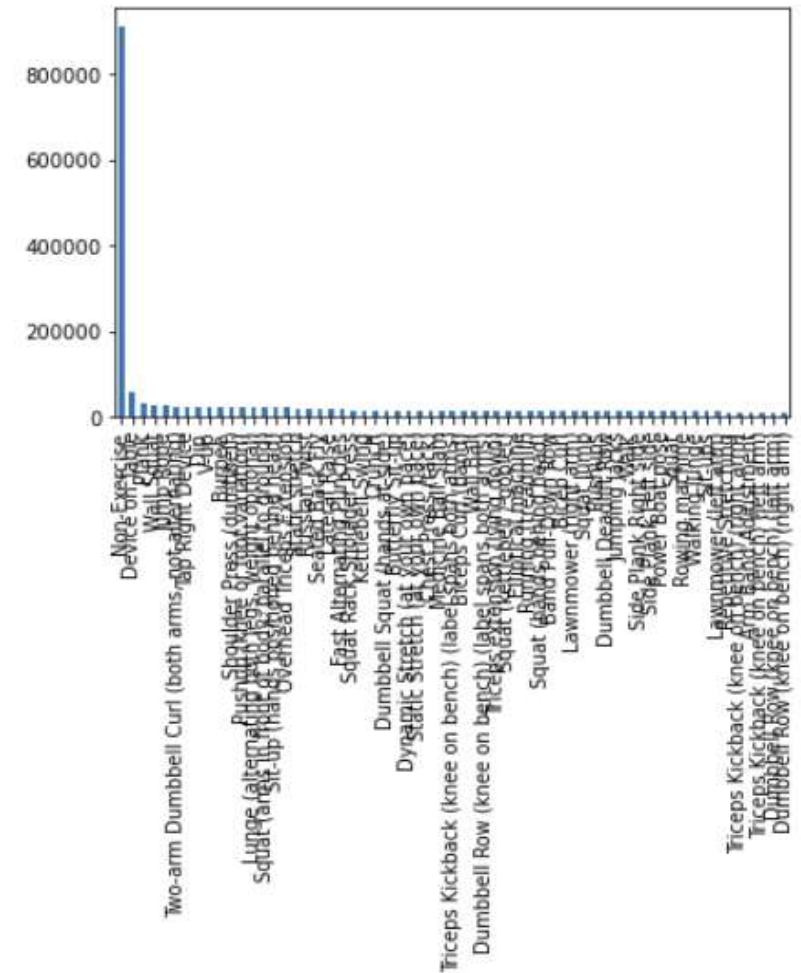
운동상태에 따른 분포

0 = 운동 아님, 1 = 운동임



## 02. 데이터 분석

```
all_train= pd.merge(train, train_labels, on='id',how='outer')
all_train.head(5)
```

[illegible]



# 03. 모델링-DNN(Deep Neural Network)

## 1. train, test 나누기

```
import tensorflow as tf
X=tf.reshape(np.array(train.iloc[:,2:]),[-1, 600, 6])
X.shape
```

TensorShape([3125, 600, 6])

```
y = tf.keras.utils.to_categorical(train_labels['label'])
y.shape
```

(3125, 61)

```
test_X=tf.reshape(np.array(test.iloc[:,2:]),[-1, 600, 6])
test_X.shape
```

TensorShape([782, 600, 6])

Model: "sequential\_16"

Layer (type)	Output Shape	Param #
dense_44 (Dense)	(None, 600, 32)	224
dense_45 (Dense)	(None, 600, 128)	4224
flatten_3 (Flatten)	(None, 76800)	0
dense_46 (Dense)	(None, 61)	4684861
Total params: 4,689,309		
Trainable params: 4,689,309		
Non-trainable params: 0		

metrics=['accuracy']

Log Loss : 11.965815

```
pred=model.predict(test_X)
```

```
submission.iloc[:,1:]=prediction
```

```
submission.to_csv('ann_1.csv', index=False)
```

# 03. 모델링-DNN(Deep Neural Network)

## 2. train,test 나누기

```
import tensorflow as tf
X=tf.reshape(np.array(train.iloc[:,2:]),[-1, 600, 6])
X.shape
TensorShape([2500, 600, 6])
y_train_encoding.shape
TensorShape([2500, 61])
y.shape
TensorShape([2500, 61])
y_test_encoding.shape
TensorShape([625, 61])
```

625

```
test_X=tf.reshape(np.array(test.iloc[:,2:]),[-1, 600, 6])
test_X.shape
```

TensorShape([782, 600, 6])

Model: "sequential\_15"

Layer (type)	Output Shape	Param #
dense_41 (Dense)	(None, 600, 32)	224
dense_42 (Dense)	(None, 600, 128)	4224
flatten_2 (Flatten)	(None, 76800)	0
dense_43 (Dense)	(None, 61)	4684861
Total params: 4,689,309		
Trainable params: 4,689,309		
Non-trainable params: 0		

```
print("테스트 세트의 정확도: {:.2f}".format(np.mean(y_pred == y_test_encoding)))
```

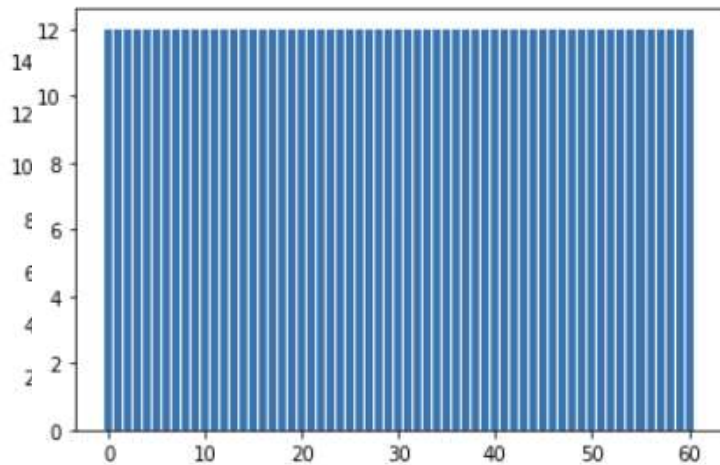
테스트 세트의 정확도: 0.64

```
y_pred = model.predict(X_test)
```

Log Loss : 11.96581

## 03. 모델링-DNN(Deep Neural Network)

### 3. 다운 샘플링



```
from imblearn.under_sampling import RandomUnderSampler  
  
X_resampled, y_resampled = RandomUnderSampler(random_state=0).fit_resample(X_1,y)  
count_and_plot(y_resampled)
```

Model: "sequential\_21"

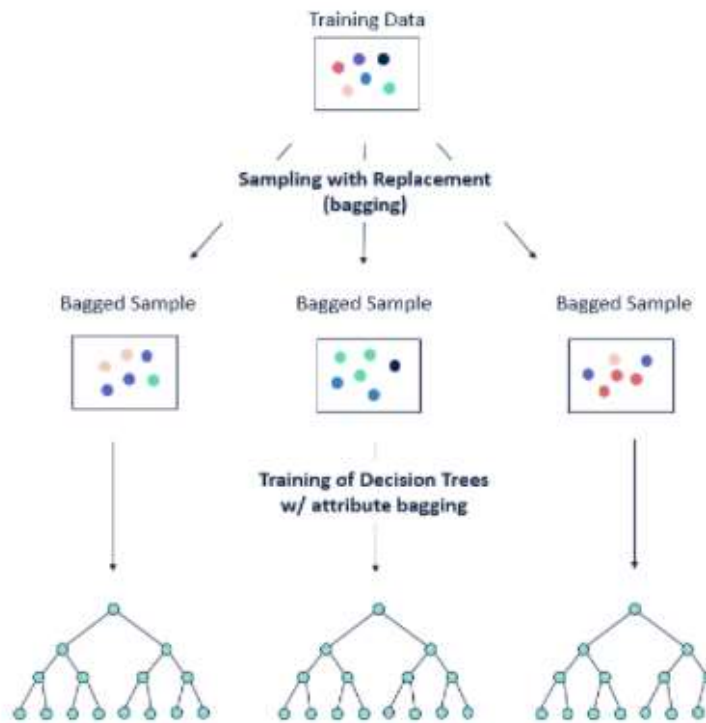
Layer (type)	Output Shape	Param #
dense_60 (Dense)	(None, 32)	115232
dense_61 (Dense)	(None, 128)	4224
flatten_8 (Flatten)	(None, 128)	0
dense_62 (Dense)	(None, 61)	7869

Total params: 127,325  
Trainable params: 127,325  
Non-trainable params: 0

Log Loss : 12.3396



## 03. 모델링-RandomForest



여러 개의 결정 트리를 활용한 배깅의 대표적인 알고리즘

장) 결정 트리의 쉽고 직관적인 장점을 그대로 가지고 있음  
동일한 알고리즘 중 비교적 빠른 수행 속도를 가짐  
다양한 분야에서 좋은 성능을 나타냄

단) 파라미터가 많아 튜닝을 위한 시간이 많이 소요됨

```
model = RandomForestClassifier()  
model
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',  
                        max_depth=None, max_features='auto', max_leaf_nodes=None,  
                        min_impurity_decrease=0.0, min_impurity_split=None,  
                        min_samples_leaf=1, min_samples_split=2,  
                        min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,  
                        oob_score=False, random_state=None, verbose=0,  
                        warm_start=False)
```

## 03. 모델링-RandomForest

```
print(X_train.shape)
print(y_train_encoding.shape)
print(X_test.shape)
print(y_test_encoding.shape)
```

```
(2500, 600, 6)
(2500, 61)
(625, 600, 6)
(625, 61)
```

Tensor 3d to 2d

```
X_train = X_train.numpy()
```

```
X_train = X_train.reshape((2500, -1))
```

```
X_train.shape
```

```
(2500, 3600)
```

```
y_train_encoding.shape
```

```
print(X_train.shape, y_train_encoding.shape)
print(X_test.shape, y_test_encoding.shape)
```

```
(2500, 3600) (2500, 61)
(625, 3600) (625, 61)
```

```
X_test.shape
(625, 600, 6)
```

```
X_test = X_test.reshape((625, -1))
```

```
X_test.shape
```

```
(625, 3600)
```

## 03. 모델링-RandomForest

```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
forest2 = RandomForestClassifier(random_state=0)
```

```
from sklearn.model_selection import GridSearchCV

params = { 'n_estimators' : [10, 100],
           'max_depth' : [6, 8, 10, 12],
           'min_samples_leaf' : [8, 12, 18],
           'min_samples_split' : [8, 16, 20] }
grid = GridSearchCV(forest2, param_grid=params, cv=5, n_jobs=-1)
grid.fit(X_train, y_train_encoding)
```

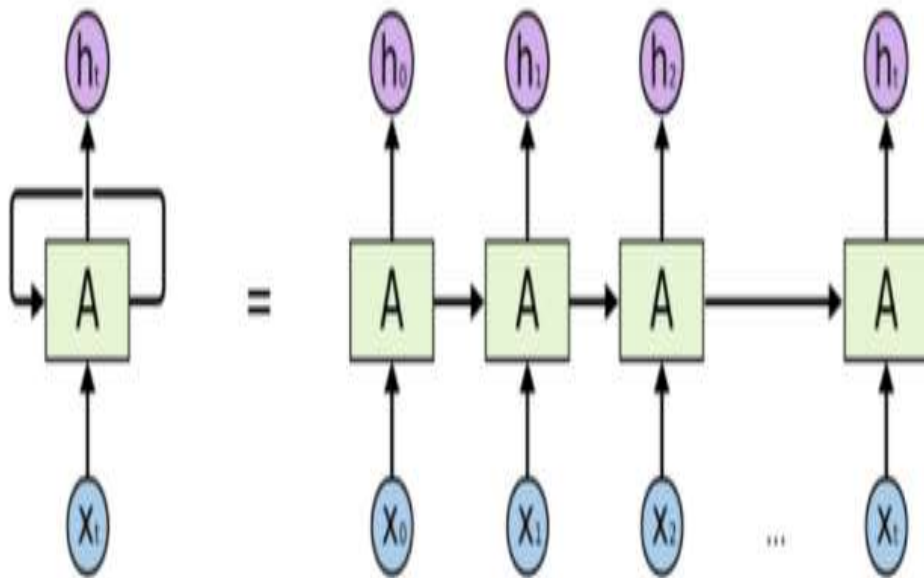
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=0), n_jobs=-1,
             param_grid={'max_depth': [6, 8, 10, 12],
                          'min_samples_leaf': [8, 12, 18],
                          'min_samples_split': [8, 16, 20],
                          'n_estimators': [10, 100]})
```

```
print('최적 하이퍼 파라미터: ', grid.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid.best_score_))
```

최적 하이퍼 파라미터: {'max\_depth': 6, 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimators': 100}  
최고 예측 정확도: 0.4444

Log Loss : 6.4417

### 03. 모델링-RNN(Recurrent Neural Network)



RNN(Recurrent Neural Networks) : 순환신경망  
은닉층의 결과가 다시 은닉층의 입력으로 들어  
가도록 연결됨  
⇒ 순서 또는 시간의 측면을 고려할 수 있음

## 03. 모델링-RNN(Recurrent Neural Network)

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense

np.random.seed(0)
model = Sequential()
model.add(SimpleRNN(32, input_shape=(600,6)))
model.add(Dense(128, activation='relu'))
model.add(Dense(61, activation='softmax'))

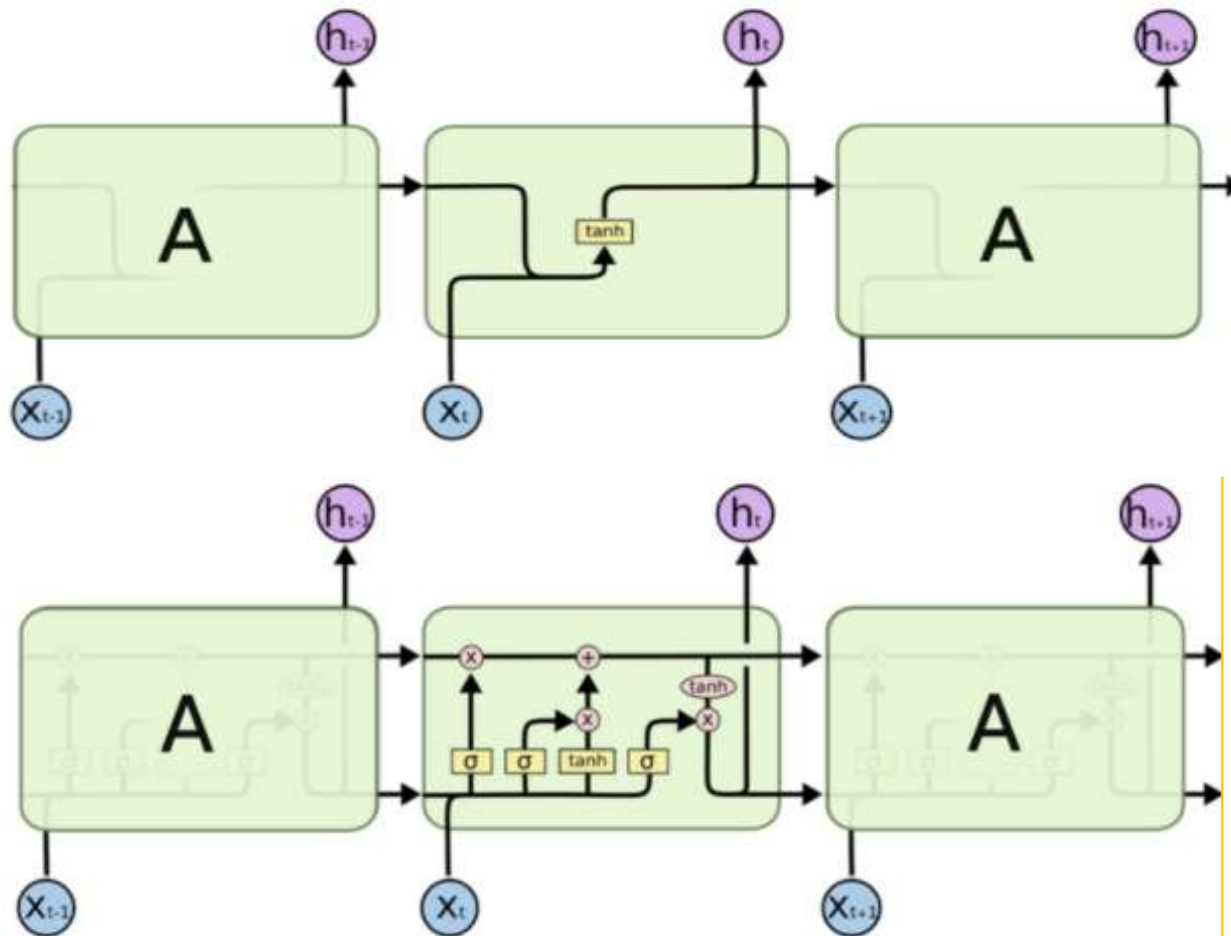
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Model: "sequential\_9"

Layer (type)	Output Shape	Param #
simple_rnn_9 (SimpleRNN)	(None, 32)	1248
dense_14 (Dense)	(None, 128)	4224
dense_15 (Dense)	(None, 61)	7869
Total params: 13,341		
Trainable params: 13,341		
Non-trainable params: 0		

Log Loss : 2.6568

### 03. 모델링-LSTM(Long Short Term Memory)



$$\begin{aligned} f_t &= \sigma(W_{xh_f}x_t + W_{hh_f}h_{t-1} + b_{h_f}) \\ i_t &= \sigma(W_{xh_i}x_t + W_{hh_i}h_{t-1} + b_{h_i}) \\ o_t &= \sigma(W_{xh_o}x_t + W_{hh_o}h_{t-1} + b_{h_o}) \\ \hat{c}_t &= \tanh(W_{xh_c}x_t + W_{hh_c}h_{t-1} + b_{h_c}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

LSTM  
(Long Short Term Memory)

장기 의존성 문제 해결  
RNN의 일종인데  
반복되는 모듈이 다름

## 03. 모델링-LSTM(Long Short Term Memory)

1

```
from keras.models import Sequential
from keras.layers import LSTM, Dense
```

```
np.random.seed(0)
model = Sequential()
model.add(LSTM(32, input_shape=(600,6)))
model.add(Dense(128, activation='relu'))
model.add(Dense(61, activation='softmax'))
```

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
print("테스트 세트의 정확도: {:.2f}".format(np.mean(y_pred == y_test_encoding)))
```

Model: "sequential"

테스트 세트의 정확도: 0.98

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 32)	4992
dense (Dense)	(None, 128)	4224
dense_1 (Dense)	(None, 61)	7869

-----

Total params: 17,085  
Trainable params: 17,085  
Non-trainable params: 0

-----

Log Loss : 2.47035

## 03. 모델링-LSTM(Long Short Term Memory)

2

BiLSTM(양방향 장단기 기억) 계층은 시계열 또는 시퀀스 데이터의 시간 스텝 간의 양방향 장기 종속성을 학습.

Model: "sequential\_17"

Layer (type)	Output Shape	Param #
lstm_18 (LSTM)	(None, 600, 32)	4992
bidirectional_8 (Bidirection	(None, 64)	
dense_26 (Dense)	(None, 128)	
flatten_11 (Flatten)	(None, 128)	
dense_27 (Dense)	(None, 61)	

Total params: 37,821  
Trainable params: 37,821  
Non-trainable params: 0

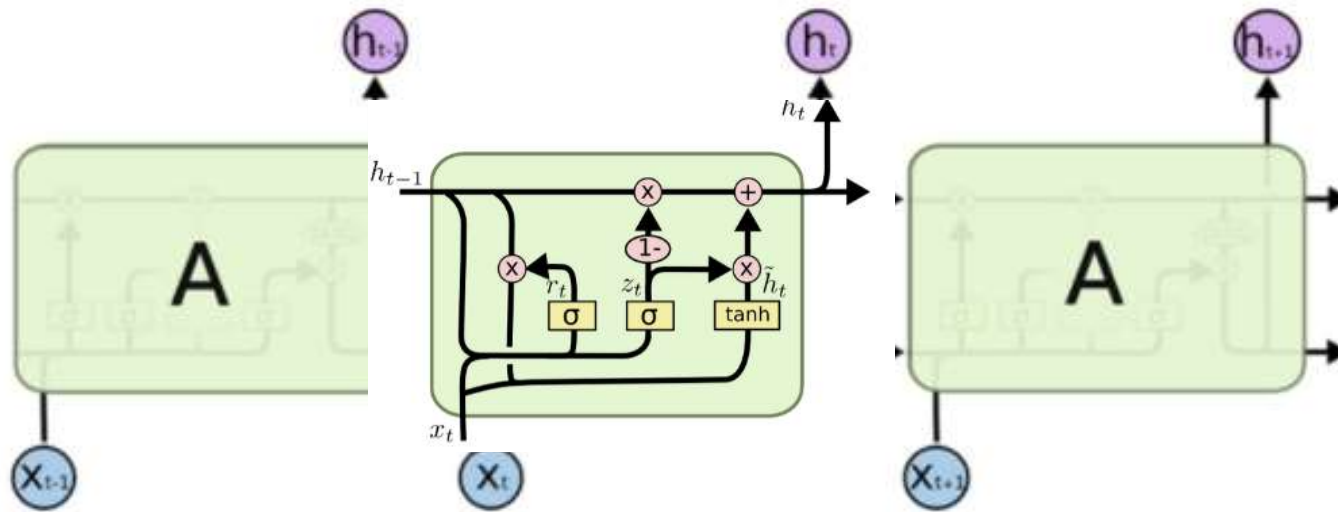
```
from tensorflow.keras.models import load_model
print("테스트 정확도: %.4f" % model.evaluate(X_test, y_test_encoding)[1])
```

20/20 [=====] - 3s 152ms/step - loss: 2.0583 - accuracy: 0.5360  
테스트 정확도: 0.5360

Log Loss : 2.28553



### 03. 모델링-GRU(The Gated Recurrent Unit)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU(The Gated Recurrent Unit)

2014년 cho, etal에 의해 소개된  
LSTM의 변칙 패턴

Forget gate와 input gate를 하나의 업데이트 gate로 통일하고  
cell state와 hidden state를 하나로 합침

## 03. 모델링-GRU(The Gated Recurrent Unit)

```
from keras.models import Sequential
from keras.layers import GRU, Dense, Dropout

np.random.seed(0)
model = Sequential()
model.add(GRU(500, input_shape=(600,6)))
model.add(Dense(1500, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(350, activation = 'relu'))
model.add(Dense(100, activation = 'relu'))
model.add(Dense(61, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
model.summary()
```

Model: "sequential\_2"

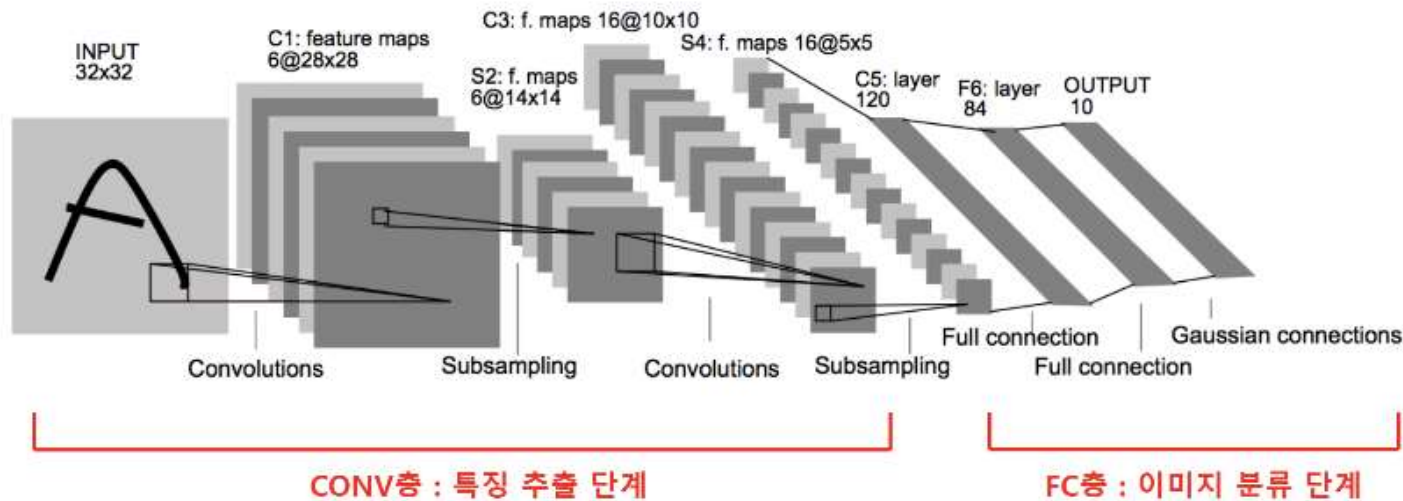
Layer (type)	Output Shape	Param #
gru_2 (GRU)	(None, 500)	762000
dense_4 (Dense)	(None, 1500)	751500
dropout (Dropout)	(None, 1500)	0
dense_5 (Dense)	(None, 350)	525350
dense_6 (Dense)	(None, 100)	35100
dense_7 (Dense)	(None, 61)	6161
Total params: 2,080,111		
Trainable params: 2,080,111		
Non-trainable params: 0		

Log Loss : 5.9345

### 03. 모델링-CNN(Convolution Neural Network)

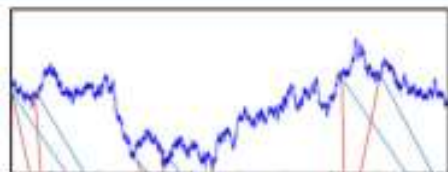
CNN이란?

Convolutional Neural Network  
이미지, 비디오, 텍스트 등을 분류하는 모델

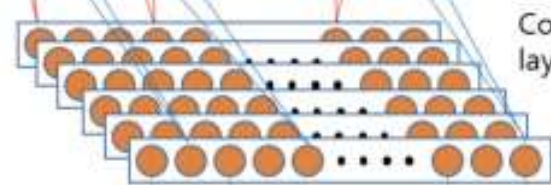


CNN (Convolution Neural Network) : 합성곱 신경망

# 03. 모델링-CNN(Convolution Neural Network)



Input data



Convolutional layer



Pooling layer



Fully-connected layers



Softmax layer

Model: "sequential\_21"

Epoch 1/30

63/63 [=====] - 4s 72ms/step - loss: 2.7196 - acc: 0.4935  
- val\_loss: 2.3754 - val\_acc: 0.5200

Epoch 00001: val\_acc improved from 0.51680 to 0.52000, saving model to best\_model.h5

Epoch 2/30

63/63 [=====] - 4s 69ms/step - loss: 2.6109 - acc: 0.4980  
- val\_loss: 2.4347 - val\_acc: 0.5160

Epoch 00002: val\_acc did not improve from 0.52000

Epoch 3/30

63/63 [=====] - 4s 70ms/step - loss: 2.4978 - acc: 0.5035  
- val\_loss: 2.9648 - val\_acc: 0.5100

Epoch 00003: val\_acc did not improve from 0.52000

Epoch 4/30

63/63 [=====] - 4s 68ms/step - loss: 2.4481 - acc: 0.5035  
- val\_loss: 2.4591 - val\_acc: 0.5120

Epoch 00004: val\_acc did not improve from 0.52000

Epoch 00004: early stopping

non-trainable params: 0

Log Loss :2.5198

## 04. 결론

Model: "sequential\_17"

Layer (type)	Output Shape	Param #
lstm_18 (LSTM)	(None, 600, 32)	4992
bidirectional_8 (Bidirectional)	(None, 64)	16640
dense_26 (Dense)	(None, 128)	8320
flatten_11 (Flatten)	(None, 128)	0
dense_27 (Dense)	(None, 61)	7869
Total params: 37,821		
Trainable params: 37,821		
Non-trainable params: 0		

BILSTM 일 때  
가장 logloss가 낮음

## 04. 결론

ANN  
cache  
CNN  
GRU  
KNN  
LSTM  
RANDOMFOREST  
RNN

2021-01-24 오후 ... 파일 폴더  
2021-01-26 오전 ... 파일 폴더  
2021-01-26 오전 ... 파일 폴더  
2021-01-26 오후 ... 파일 폴더  
2021-01-26 오전 ... 파일 폴더  
2021-01-25 오후 ... 파일 폴더  
2021-01-26 오전 ... 파일 폴더  
2021-01-24 오전 ... 파일 폴더

- 1) 다양한 모델링을 시도하면서 공부를 했다.
- 2) 늘 2차원을 다루다가 3차원 데이터를 다루서 배울점이 많았다.

### 아쉬운점

- 1) 파라미터를 많이 조정 못해서 조정을 하면 더 좋은 결과가 나올 수 있을 것 같다.
- 2) 모델 간 LOSS값이 유의미한 차이를 보이지는 않는다.
- 3) 데이터 전처리 과정을 추가하면 더 좋은 결과를 기대할 수 있을 것이다.

# 출처

<https://dacon.io/competitions/official/235689/overview/>

<https://wikidocs.net/94748>

<https://wegonnamakeit.tistory.com/25>

<https://brunch.co.kr/@chris-song/9>

<https://m.blog.naver.com/PostView.nhn?blogId=magnking&logNo=221311273459&proxyReferer=https:%2F%2Fwww.google.com%2F>

[https://ml-cheatsheet.readthedocs.io/en/latest/loss\\_functions.html](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html)

<https://injo.tistory.com/30>

END