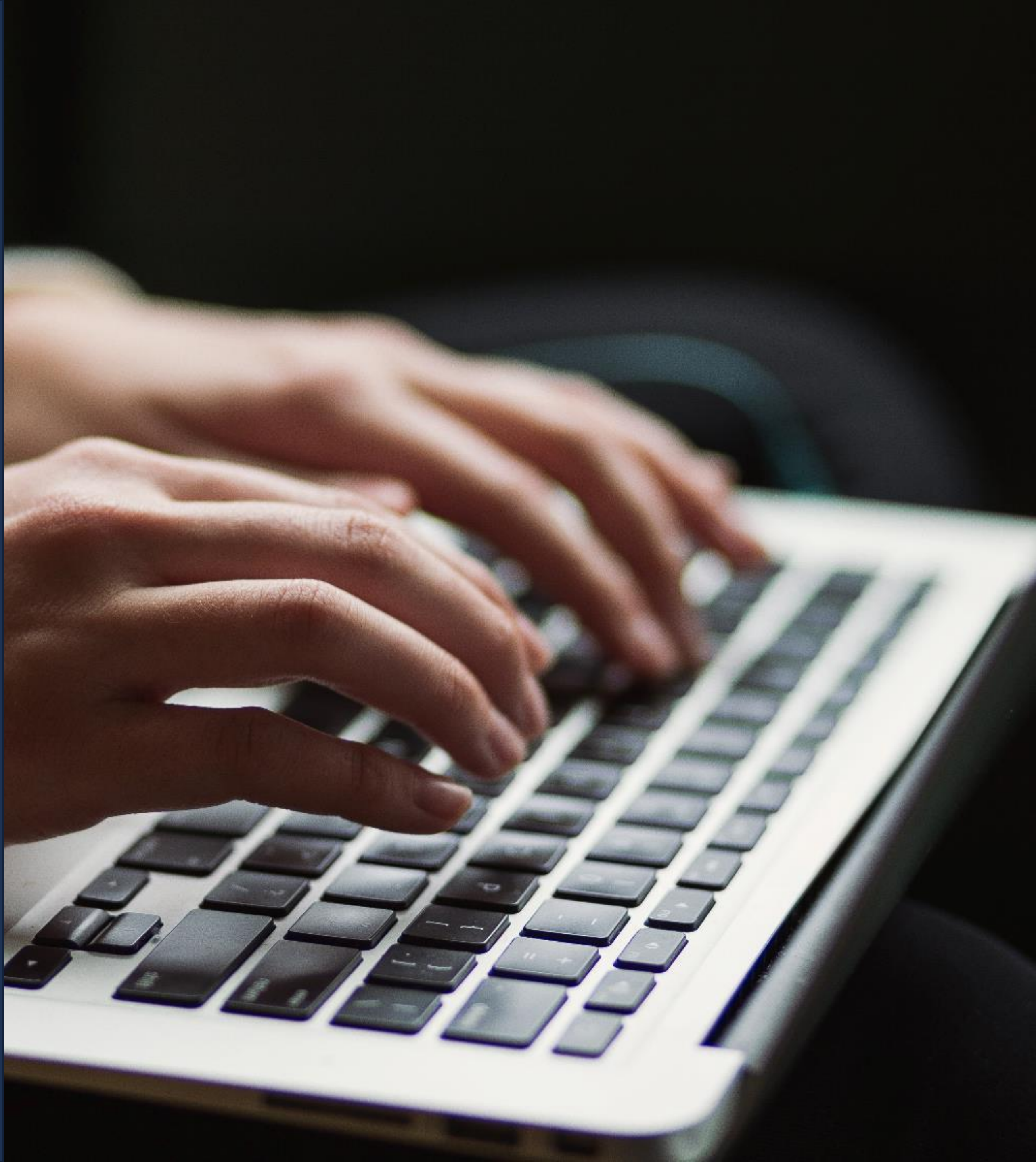
폐암 분석

목차

Part 1, 머신러닝

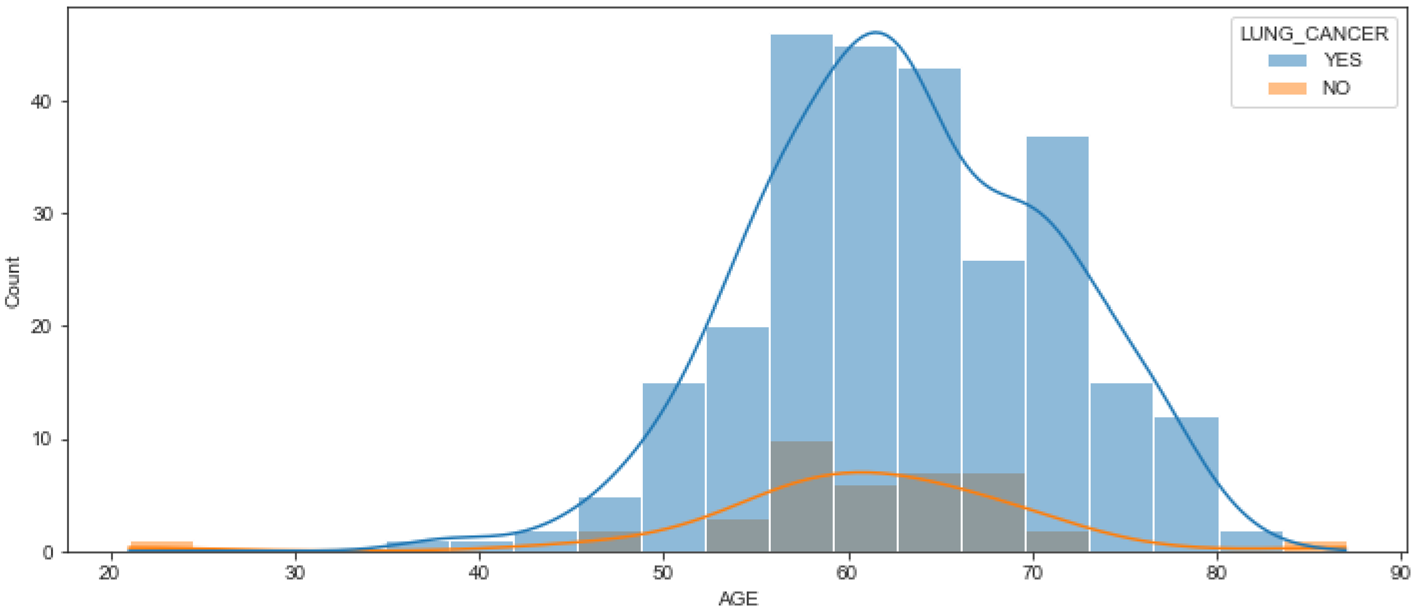


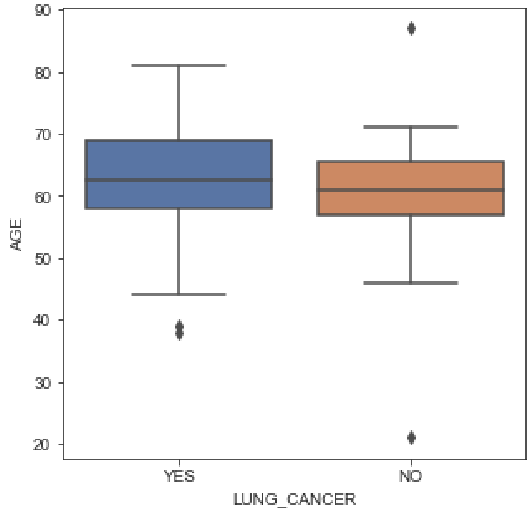
데이터 소개

- Gender: M(male), F(female)
- Age: Age of the patient (나이)
- Smoking: YES=2 , NO=1. (흡연)
- Yellow fingers: YES=2 , NO=1. (노란 손가락)
- Anxiety: YES=2 , NO=1. (불안)
- Peer_pressure: YES=2 , NO=1. (부담감)
- Chronic Disease: YES=2 , NO=1. (만성질환)
- Fatigue: YES=2 , NO=1. (피로)
- Allergy: YES=2 , NO=1. (알레르기)
- Wheezing: YES=2 , NO=1. (천식 호릅)
- Alcohol: YES=2 , NO=1. (음주)
- Coughing: YES=2 , NO=1. (기침)
- Shortness of Breath: YES=2 , NO=1. (숨가쁨)
- Swallowing Difficulty: YES=2 , NO=1. (삼키기 어려움)
- Chest pain: YES=2 , NO=1. (흉통)
- Lung Cancer: YES , NO. (폐암)

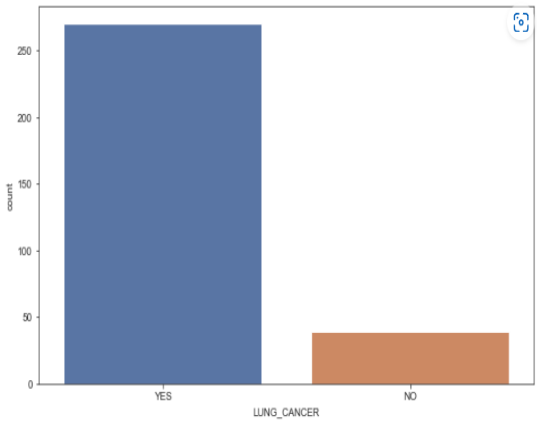
Part 1, 변수 비교

EDA : 암환자와 아닌 건강한 사람들의 나이 비교.





1. 50~60대에서 가장 암이 많이 보임,
2. 암환자데이터가 건강한 사람데이터 보다 훨씬 많음
암환자와 건강한 사람 간의 데이터 불균형이 보임 **(오버 샘플링 필요)**

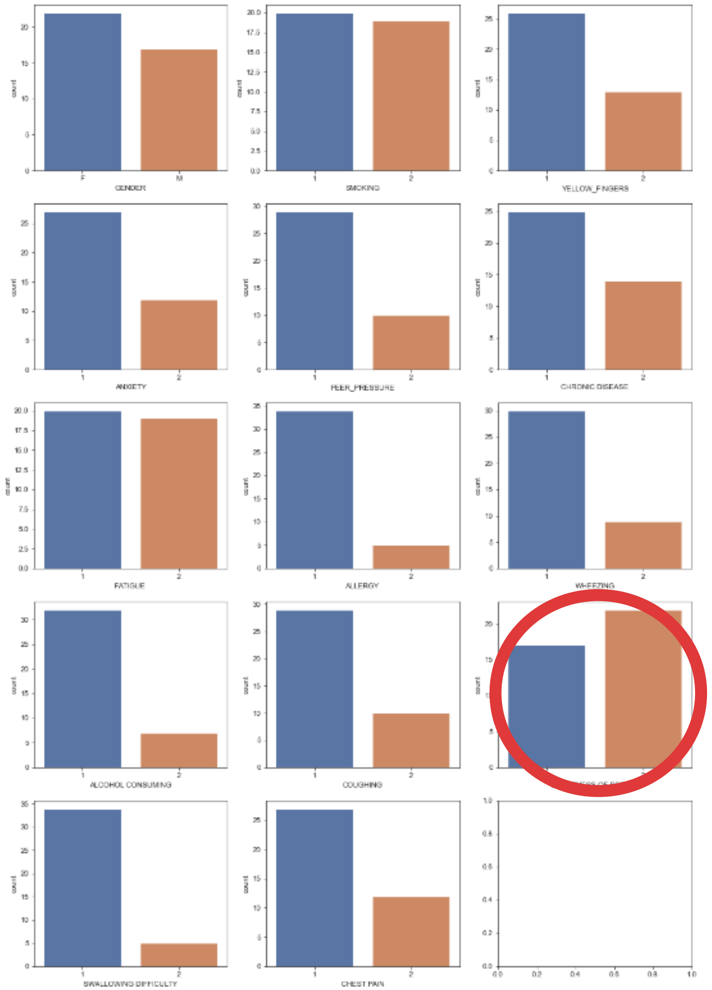


변수 비교

EDA

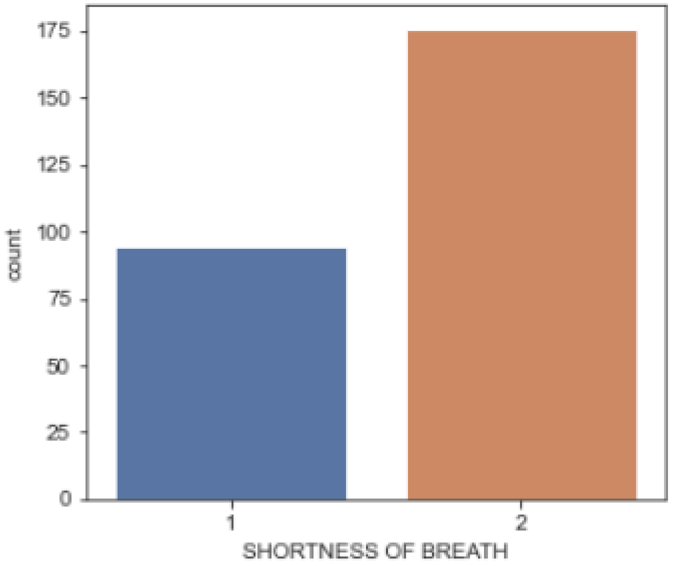
암환자 그래프

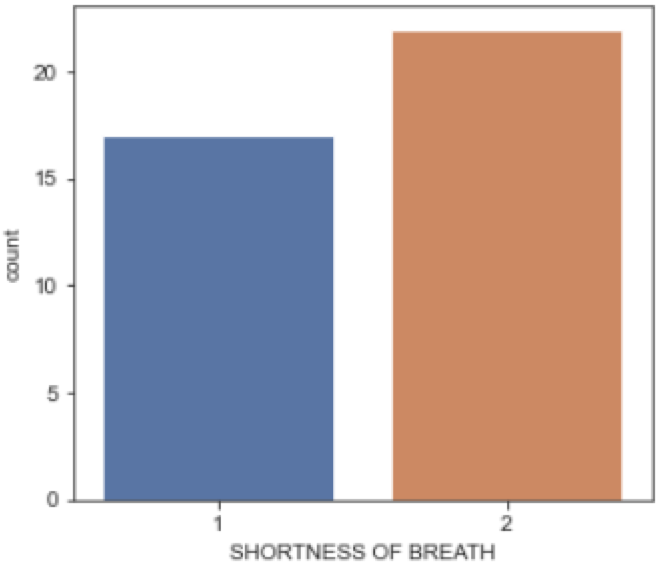
건강한사람 그래프



두개가 비슷함

EDA SHORTNESS OF BREATH(숨가쁨)





SHORTNESS OF BREATH(숨가쁨) 은 암환자와 건강한 사람 둘 다 no보다 yes가 높으므로 변별력이 없다 생각해서 분석 시 제외

데이터 전처리

```
le=preprocessing.LabelEncoder()
data['GENDER']=le.fit_transform(data['GENDER']) #남자1 여자0
data['LUNG_CANCER']=le.fit_transform(data['LUNG_CANCER']) #폐암1 아니면0
```

- 성별 M,F = 1,0
- 암 YES,NO = 1,0 으로 변환

```
X['AGE']=StandardScaler().fit_transform(X[['AGE']])
```

- AGE 표준화

```
X_train, X_test, y_train, y_test= train_test_split(X,y,test_size= 0.3,random_state=77)
```

- Train, test 비율 7:3

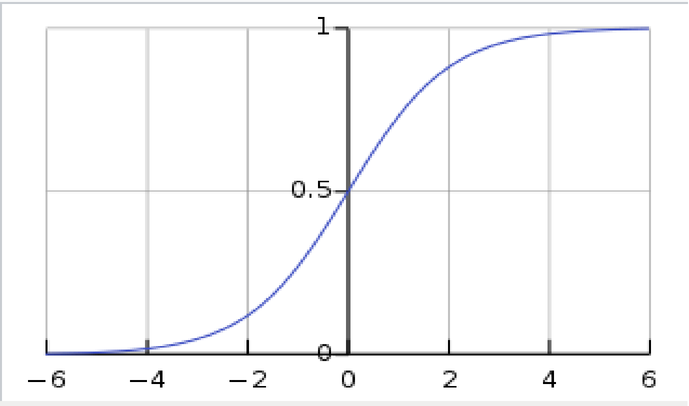
```
from imblearn.over_sampling import RandomOverSampler
X_over,y_over=RandomOverSampler().fit_resample(X,y)
```

- 오버 샘플링

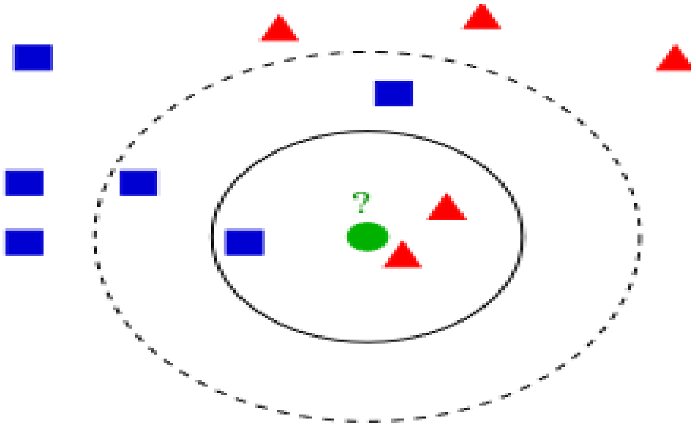
Part 1

분류 모델

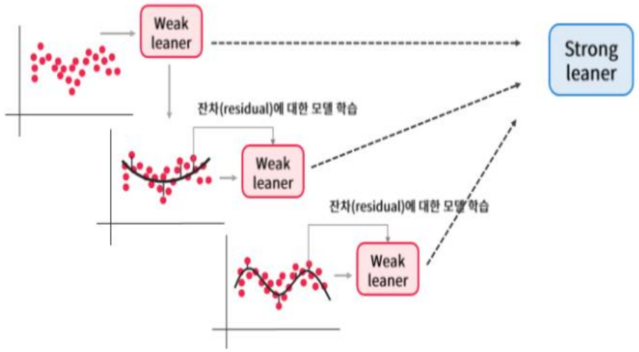
분류모델소개



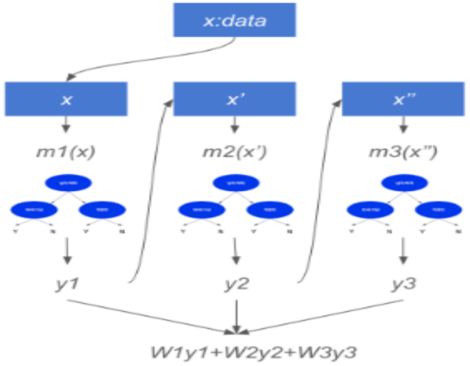
Logistic Regression



Knn



GradientBoosting



Xgboost

출처: 위키피디아

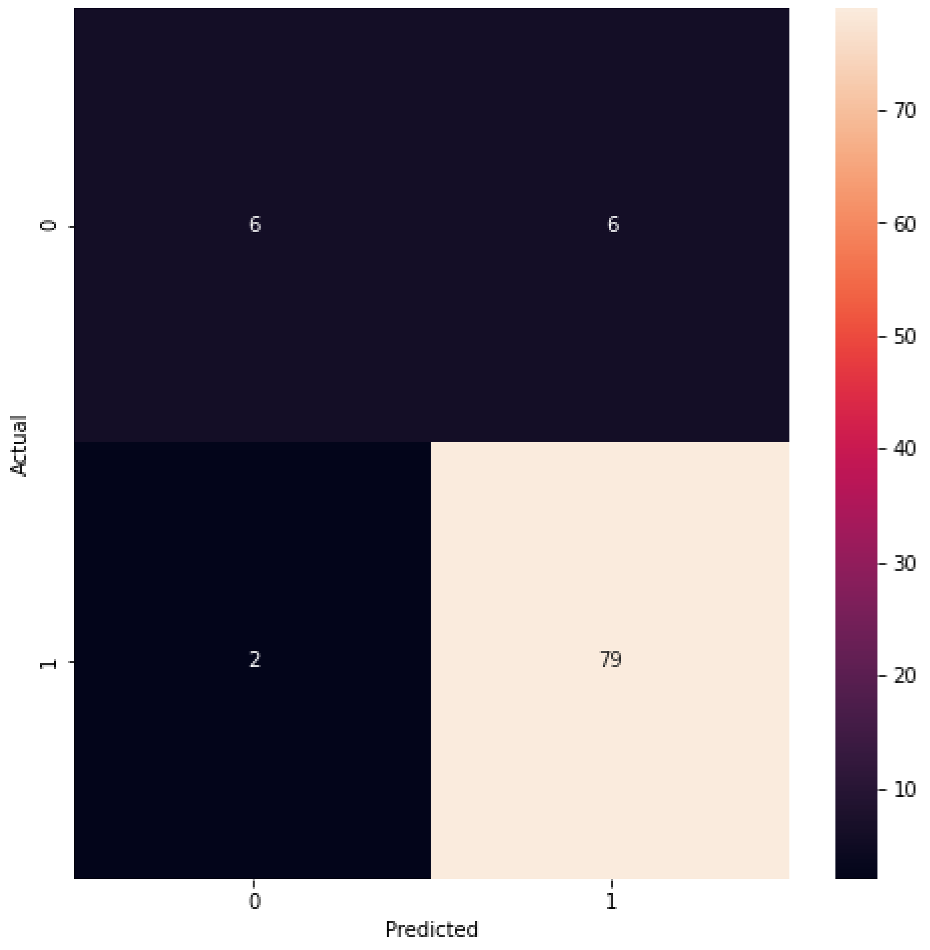
Logistic Regression

결과값

```
print(f"train:",lr.score(X_train,y_train))
print(f'test:', lr.score(X_test,y_test))
```

```
train: 0.9398148148148148
test: 0.9139784946236559
```

	precision	recall	f1-score	support
0	0.75	0.50	0.60	12
1	0.93	0.98	0.95	81
accuracy			0.91	93
macro avg	0.84	0.74	0.78	93
weighted avg	0.91	0.91	0.91	93



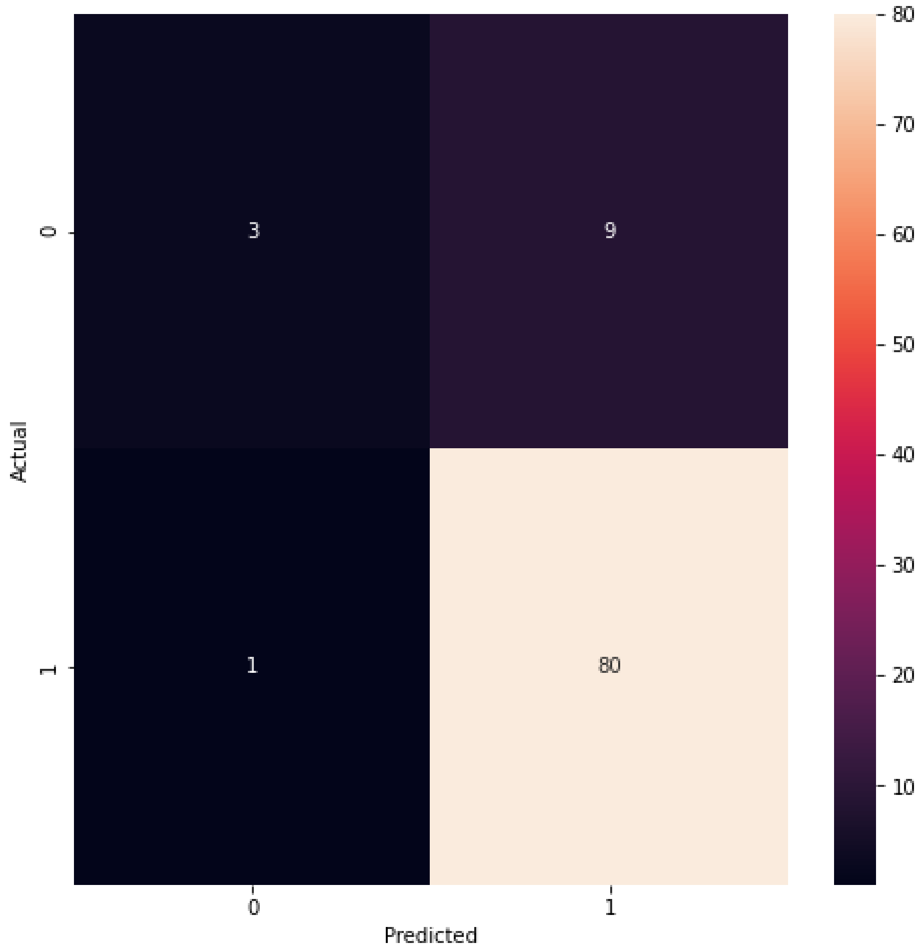
KNN

결과값

	precision	recall	f1-score	support
0	0.75	0.25	0.38	12
1	0.90	0.99	0.94	81
accuracy			0.89	93
macro avg	0.82	0.62	0.66	93
weighted avg	0.88	0.89	0.87	93

```
print(f"train:",knn.score(X_train,y_train))
print(f'test:', knn.score(X_test,y_test))

train: 0.9027777777777778
test: 0.8924731182795699
```



Part 1

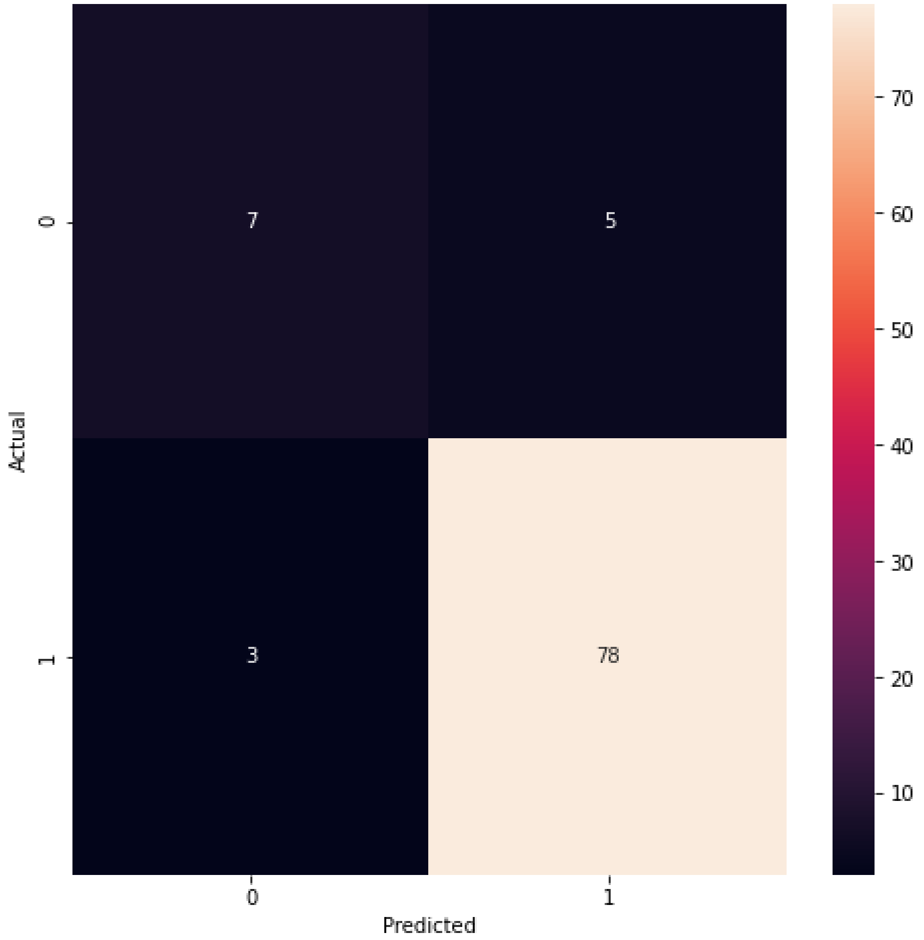
GraduebtBoosting

결과값

	precision	recall	f1-score	support
0	0.70	0.58	0.64	12
1	0.94	0.96	0.95	81
accuracy			0.91	93
macro avg	0.82	0.77	0.79	93
weighted avg	0.91	0.91	0.91	93

```
print(f"train:",gb.score(X_train,y_train))
print(f'test:', gb.score(X_test,y_test))
```

```
train: 1.0
test: 0.9139784946236559
```



Part 1

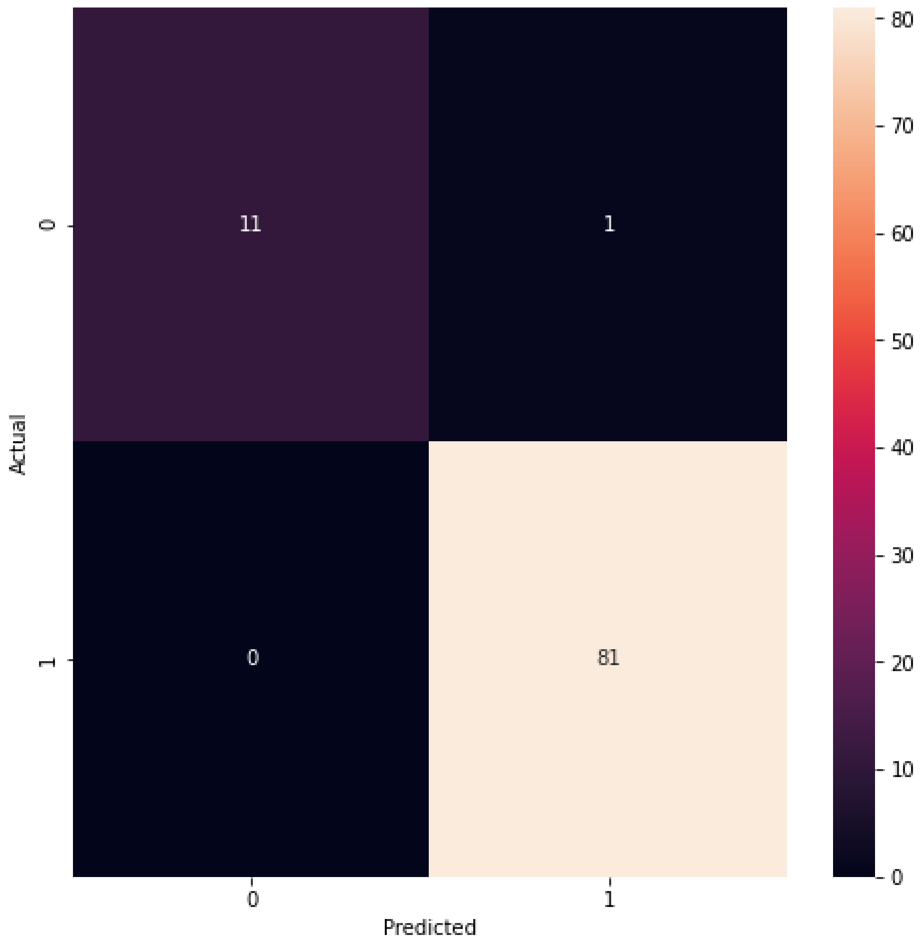
Xgboost

결과값

	precision	recall	f1-score	support
0	1.00	0.92	0.96	12
1	0.99	1.00	0.99	81
accuracy			0.99	93
macro avg	0.99	0.96	0.98	93
weighted avg	0.99	0.99	0.99	93

```
print(f"train:",xgb.score(X_train,y_train))
print(f'test:', xgb.score(X_test,y_test))
```

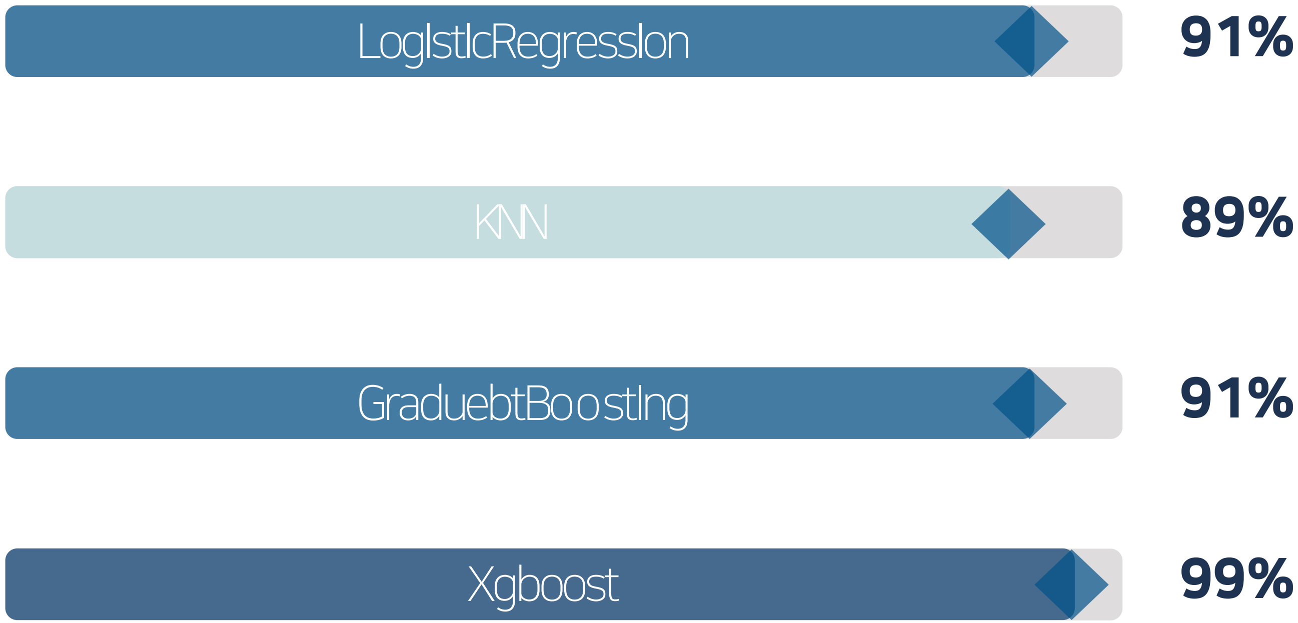
```
train: 1.0
test: 0.989247311827957
```



Part 1

결과

ACC비교

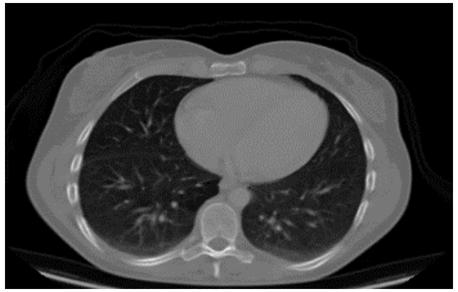


Part 2, 딥러닝

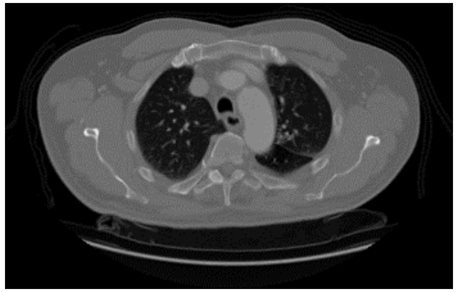
Part 2,

데이터 소개

흉부CT사진

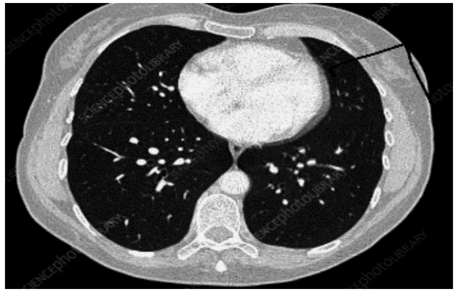


선암종



편평 세포 암종

대세포암



건강한 폐

데이터 소개

Train, Test, Valid

Train 데이터
Test 데이터
Valid 데이터

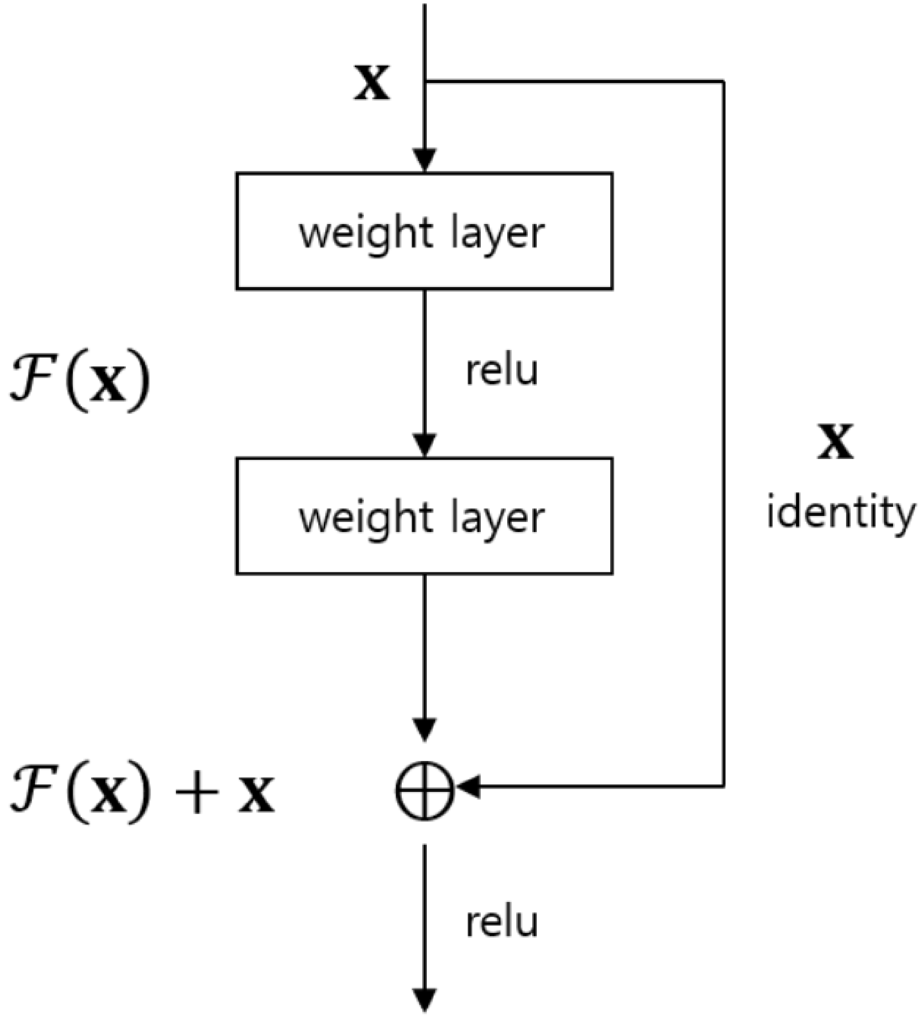
```
Found 613 images belonging to 4 classes.
Found 315 images belonging to 4 classes.
Found 72 images belonging to 4 classes.
```

이미지 증식(image Data Generator)

```
train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                   horizontal_flip = True,
                                   fill_mode = 'nearest',
                                   zoom_range=0.2,
                                   shear_range = 0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   rotation_range=0.4)
```

Base 모델

Resnet 모델



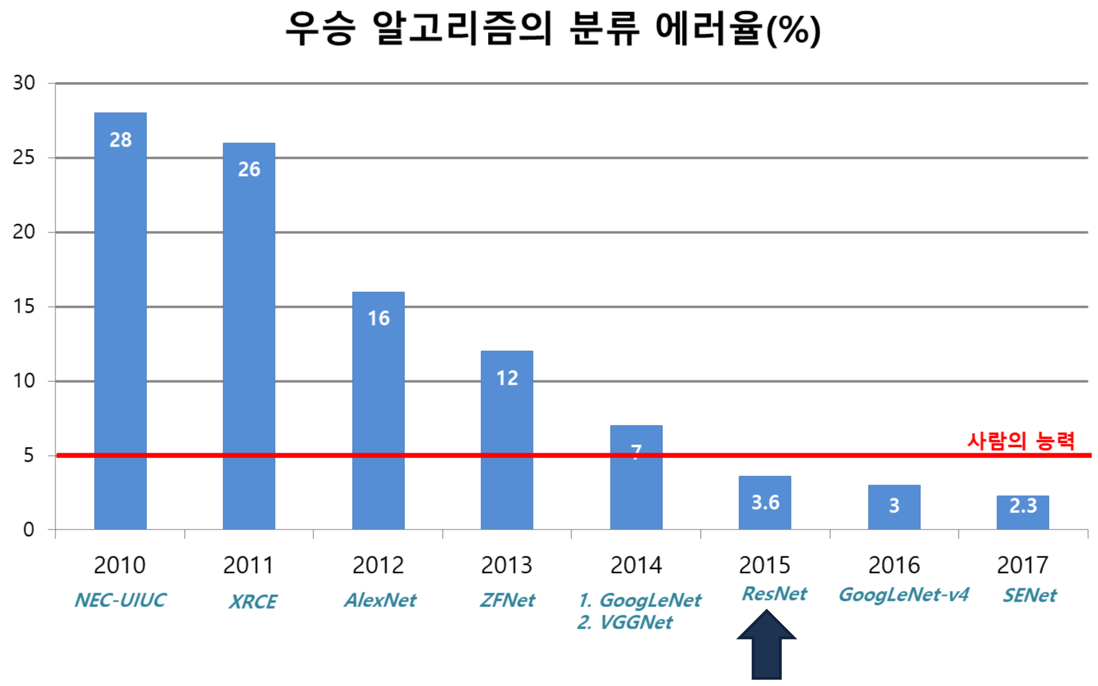
Residual neural network의 준말로 신경망의 층을 깊게 쌓을수 있는 있는 모델.

신경망의 깊은 층을 만들면 생기는 신경망의 레이어가 깊어지면 생기는 기울기 소실 문제, 성능이 떨어지는 문제를 해결하기 위해 만들어진 모델

Part 2,

Base 모델

Resnet 선택 이유



출처: https://bskyvision.com/425

설정 값

Hyperparameters

Optimizer: adam
loss: categoricalcrossentropy
Metrics: acc

layers

```
x = baseModel.output
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1024, activation = 'relu')(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(4, activation = "softmax")(x)
```

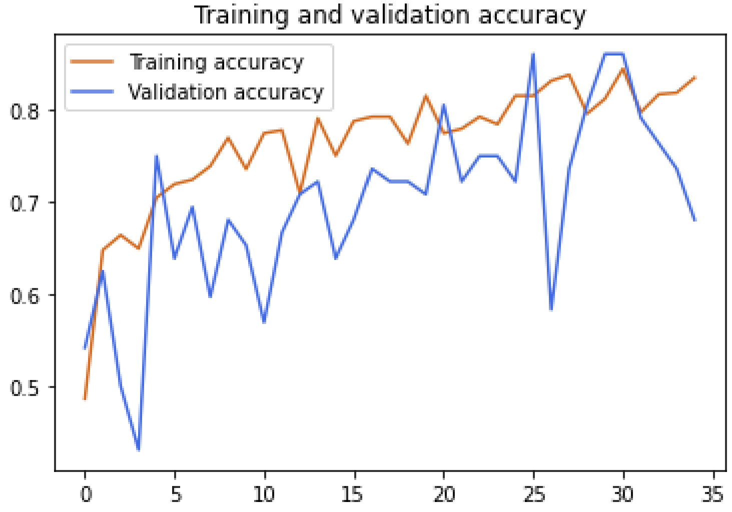
Earlystop(과대적합 방지)

```
earlystop = EarlyStopping(patience=5, restore_best_weights = True)
```

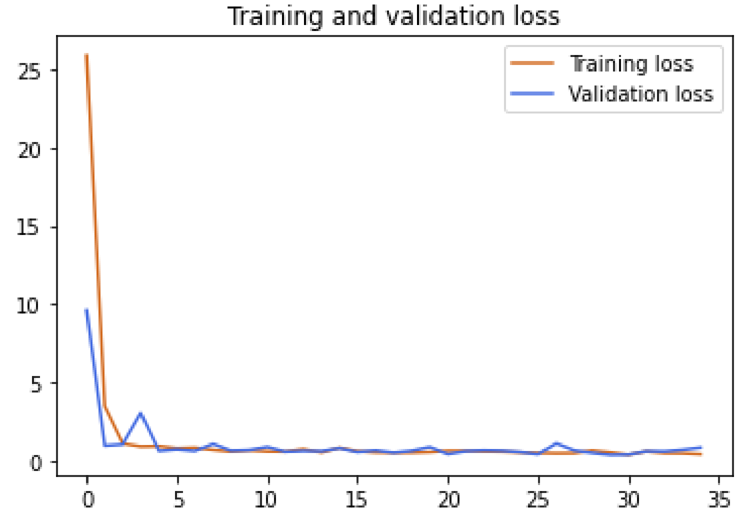
Part 2, 학습

```
Epoch 1/1000
123/123 [==============================] - 141s 1s/step - loss: 25.8661 - acc: 0.4861 - val_loss: 9.6044 - val_acc: 0.5417
Epoch 2/1000
123/123 [==============================] - 139s 1s/step - loss: 3.4898 - acc: 0.6476 - val_loss: 0.9593 - val_acc: 0.6250
Epoch 3/1000
123/123 [==============================] - 135s 1s/step - loss: 1.0986 - acc: 0.6639 - val_loss: 1.0391 - val_acc: 0.5000
Epoch 4/1000
123/123 [==============================] - 126s 1s/step - loss: 0.9104 - acc: 0.6493 - val_loss: 3.0401 - val_acc: 0.4306
Epoch 5/1000
123/123 [==============================] - 127s 1s/step - loss: 0.9287 - acc: 0.7047 - val_loss: 0.6317 - val_acc: 0.7500
Epoch 6/1000
123/123 [==============================] - 126s 1s/step - loss: 0.7943 - acc: 0.7194 - val_loss: 0.7321 - val_acc: 0.6389
Epoch 7/1000
123/123 [==============================] - 131s 1s/step - loss: 0.8320 - acc: 0.7243 - val_loss: 0.6283 - val_acc: 0.6944
Epoch 8/1000
123/123 [==============================] - 129s 1s/step - loss: 0.6992 - acc: 0.7390 - val_loss: 1.0734 - val_acc: 0.5972
Epoch 9/1000
123/123 [==============================] - 129s 1s/step - loss: 0.5929 - acc: 0.7700 - val_loss: 0.6460 - val_acc: 0.6806
Epoch 10/1000
123/123 [==============================] - 129s 1s/step - loss: 0.6682 - acc: 0.7357 - val_loss: 0.6862 - val_acc: 0.6528
Epoch 11/1000
123/123 [==============================] - 129s 1s/step - loss: 0.6040 - acc: 0.7749 - val_loss: 0.8815 - val_acc: 0.5694
Epoch 12/1000
123/123 [==============================] - 129s 1s/step - loss: 0.5863 - acc: 0.7781 - val_loss: 0.5838 - val_acc: 0.6667
Epoch 13/1000
123/123 [==============================] - 130s 1s/step - loss: 0.7475 - acc: 0.7096 - val_loss: 0.6291 - val_acc: 0.7083
Epoch 14/1000
123/123 [==============================] - 129s 1s/step - loss: 0.5386 - acc: 0.7912 - val_loss: 0.6192 - val_acc: 0.7222
Epoch 15/1000
123/123 [==============================] - 133s 1s/step - loss: 0.8304 - acc: 0.7504 - val_loss: 0.8036 - val_acc: 0.6389
Epoch 16/1000
123/123 [==============================] - 132s 1s/step - loss: 0.6153 - acc: 0.7879 - val_loss: 0.5626 - val_acc: 0.6806
Epoch 17/1000
123/123 [==============================] - 137s 1s/step - loss: 0.5386 - acc: 0.7928 - val_loss: 0.6628 - val_acc: 0.7361
Epoch 18/1000
123/123 [==============================] - 130s 1s/step - loss: 0.5251 - acc: 0.7928 - val_loss: 0.5211 - val_acc: 0.7222
Epoch 19/1000
```

최대 Acc: 0.8611



최소 Loss: 0.3981



예측 평가

```
model.evaluate(test_dataset)

63/63 [==============================] - 39s 618ms/step - loss: 0.5993 - acc: 0.7873

[0.5993189811706543, 0.7873015999794006]
```

결과

Loss: 0.5993
Acc: 0.7873

예측 평가

	Training data	Test data
Loss	0.3981	0.5993
Accuracy	0.8611	0.7873

결론

- 머신러닝 결과: acc가 98.92% 로 결과가 훌륭하다

- 하지만 딥러닝이
- 머신러닝에 비해 아쉬운 기록

자료 출처: kaggle

“ 감사합니다. ”