지문인식 1차 과제

생체인증보안

사이버보안전공

1871085

주유연

목차

01

Data

- Read Data
- Data Augmentation

02

Split Data

- train_test_split

03

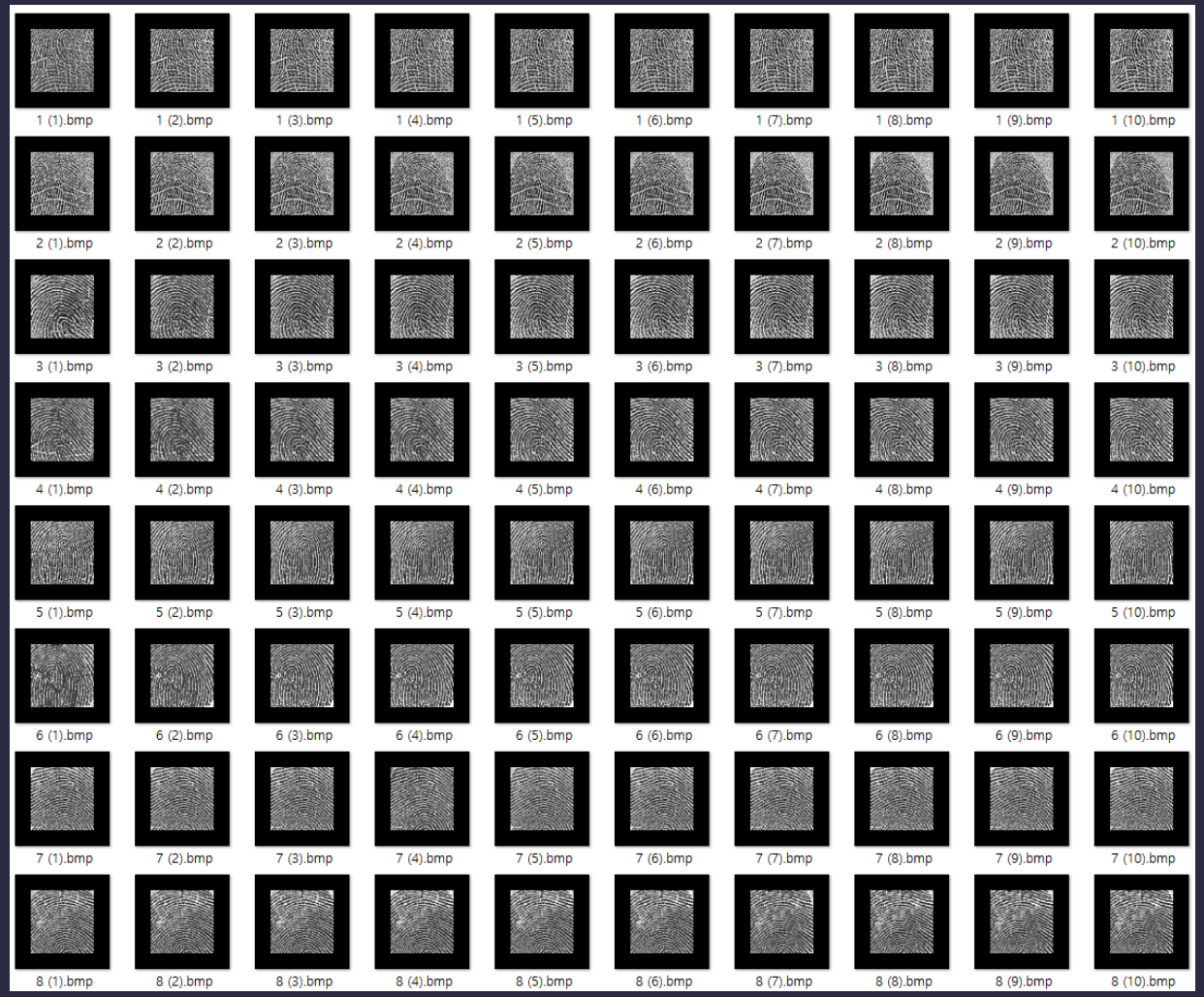
Model

- Xgboost Model
- Hyperparameter tuning
- Training Result

04

Evaluate Model

- Confusion Matrix



Given Data

- 8명의 지문 데이터가 1명당 10개씩 총 80개의 지문 이미지 데이터가 주어짐.
- 각 파일명의 맨 앞 숫자는 label을 의미함.

```
# 이미지 목록
images = glob.glob('./01_finger_training/*.bmp')
len(images)
```

```
80
```

```
r = re.compile('\d+')

img = [] # 이미지
label = [] # 라벨

for fname in images:
    l = r.findall(fname)[1]
    if l == '8': # 처리 편의성을 위해 라벨 범위를 1-8에서 0-7로 변경
        l = '0'
    label.append(l)
    im = pilimg.open(fname)
    pix = np.array(im)[25:120,25:120]/255. # 검은 부분 제거, Normalize
    img.append(pix)
```

```
X = np.array(img)
X = X.reshape(X.shape[0],95,95,1)
X.shape # img shape
```

```
(80, 95, 95, 1)
```

```
y = np.array(label, dtype='int32')
y # label
```

```
array([7, 4, 6, 4, 6, 3, 7, 1, 7, 3, 3, 0, 1, 4, 4, 7, 6, 7, 1, 6, 5, 4,
       1, 1, 2, 1, 5, 3, 0, 4, 6, 4, 5, 5, 2, 3, 7, 1, 2, 3, 2, 1, 5, 3,
       7, 2, 6, 0, 4, 5, 0, 3, 1, 0, 0, 3, 6, 0, 5, 6, 5, 2, 4, 0, 0, 2,
       2, 0, 6, 3, 7, 1, 7, 5, 5, 2, 4, 2, 7, 6], dtype=int32)
```

데이터 불러오기

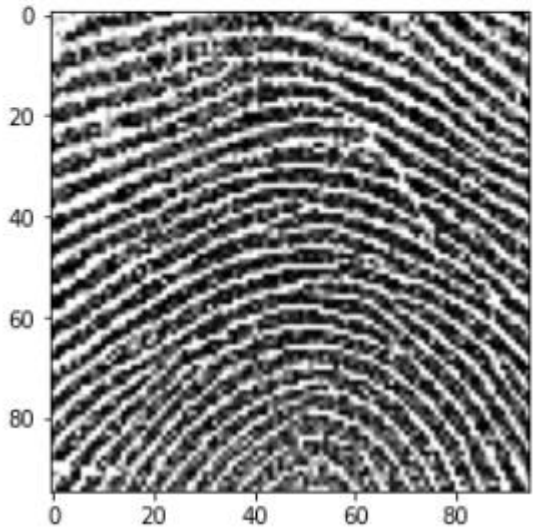
- glob 함수를 이용해 파일명을 불러오고, 파일명에서 label을 읽어 별도의 리스트에 저장함.
- Confusion matrix 등의 처리의 편의성을 위해 8번 label은 0으로 변경함.
- 불러온 이미지를 numpy array로 변환하고, 이미지 자체의 테두리를 제거함.
- 이후 픽셀을 255로 나누어 Normalize함.

지문 이미지 확인

- 테두리 제거
- normalization

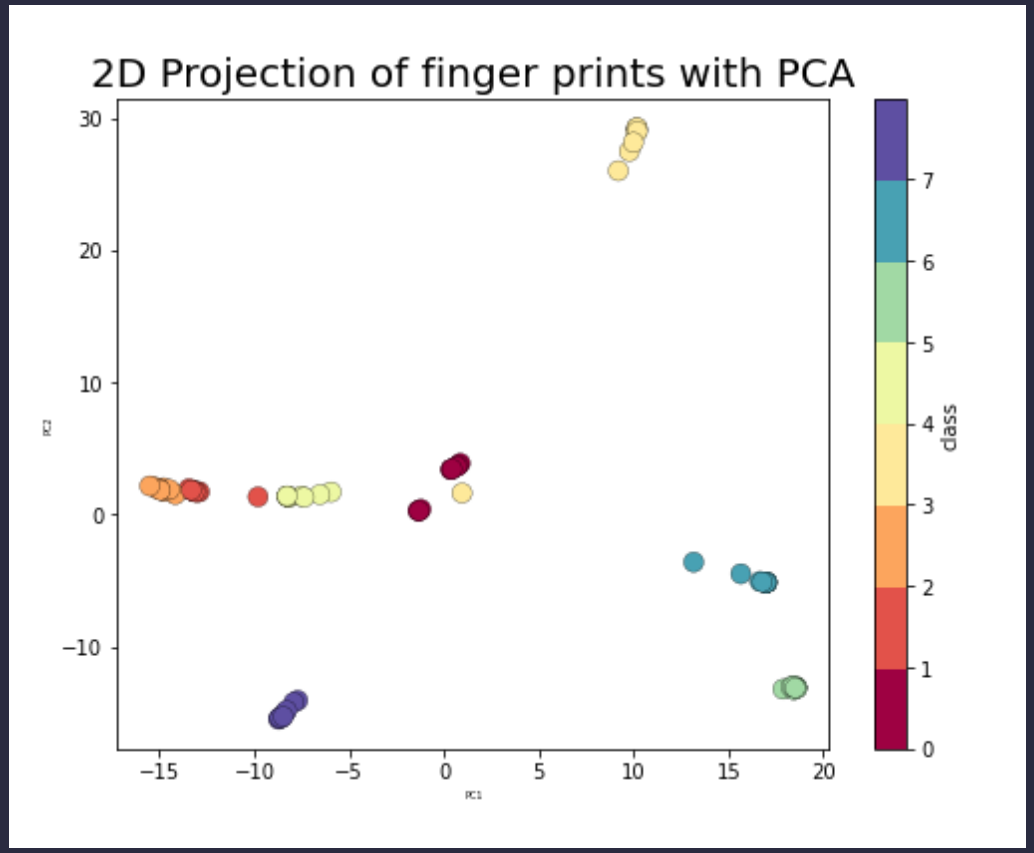
```
# 지문 이미지
plt.imshow(X[0], cmap='gray')
print(X[0].shape)
print(y[0])
```

```
(95, 95, 1)
7
```



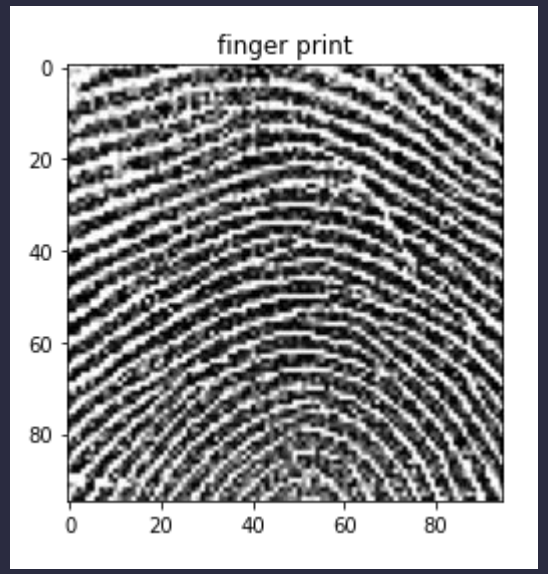
PCA로 데이터 분포 확인

- 데이터의 분포 양상을 확인하기 위해 PCA로 2차원 그래프에 데이터를 표시.

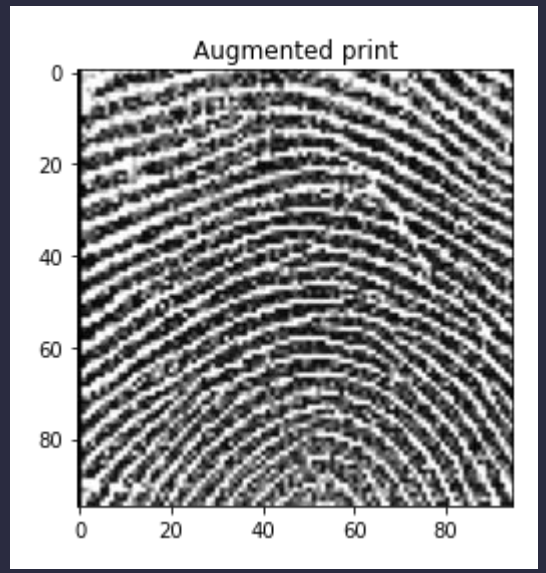


Data Augmentation

- 1명당 10개의 데이터는 지문을 구별하기에 부족하다고 판단.
- 지문 이미지를 blur처리하고 이미지의 각도를 비틀어 새 데이터를 생성함. (크게 변형되지 않도록)
- 지문패턴 분류 구분에 loop의 방향을 고려하여 이미지를 flip하지 않도록 함.
- 각 이미지당 10개의 augmented image를 생성하여 총 800개의 데이터를 확보함.

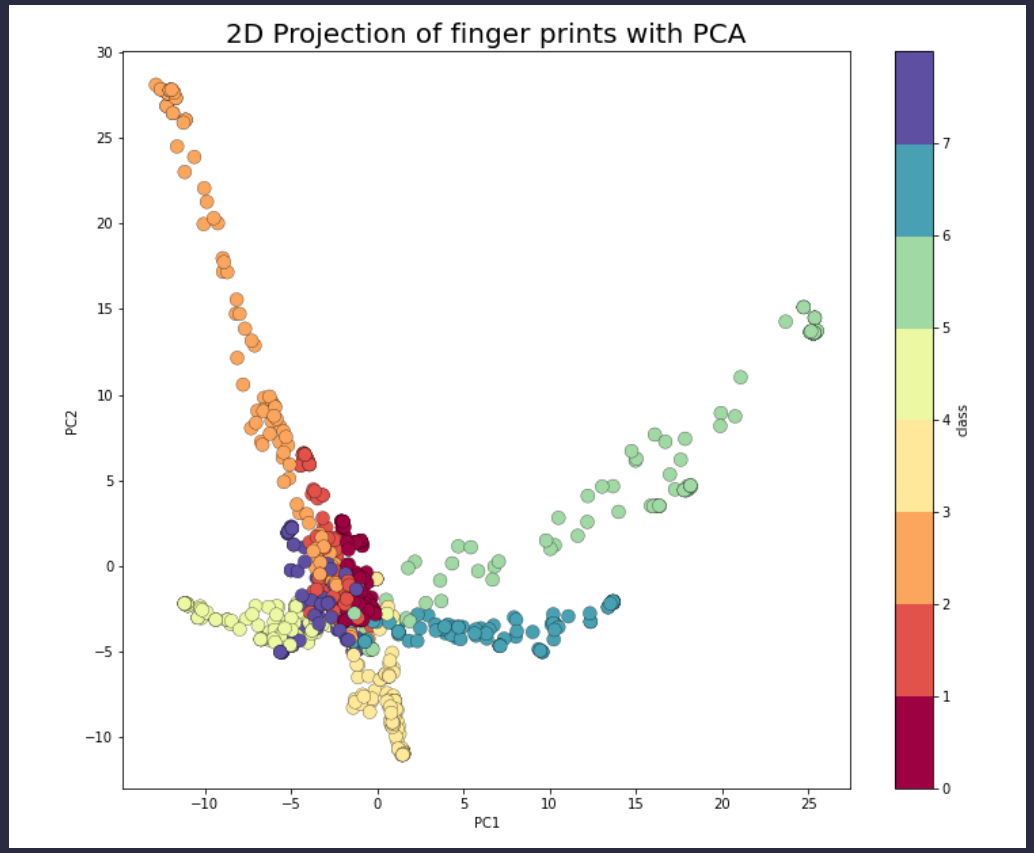


- GaussianBlur
- Affine – translate, rotate



PCA로 데이터 분포 확인

- Augmentation 이후 데이터 분포 PCA로 시각화.



최종 Data

```
# 생성한 이미지
x_d = np.array(x_d)
y_d = np.array(y_d)
print(x_d.shape)
print(y_d.shape)
```

```
(720, 95, 95, 1)
(720,)
```

```
# 기존 이미지
print(X.shape)
print(y.shape)
```

```
(80, 95, 95, 1)
(80,)
```

```
X_data = np.concatenate([X, x_d], axis=0)
y_data = np.concatenate([y, y_d], axis=0)
print(X_data.shape)
print(y_data.shape)
```

```
(800, 95, 95, 1)
(800,)
```

Split Data

- sklearn.model_selection의 train_test_split 함수 이용.
- Train : Test의 비율은 8:2로 하고, stratify 옵션을 주어 label이 균등하게 나뉘도록 함.
- X_test, y_test는 모델 훈련 후 evaluate으로 이용함.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data,y_data, test_size=0.2, shuffle=True, stratify=y_
```

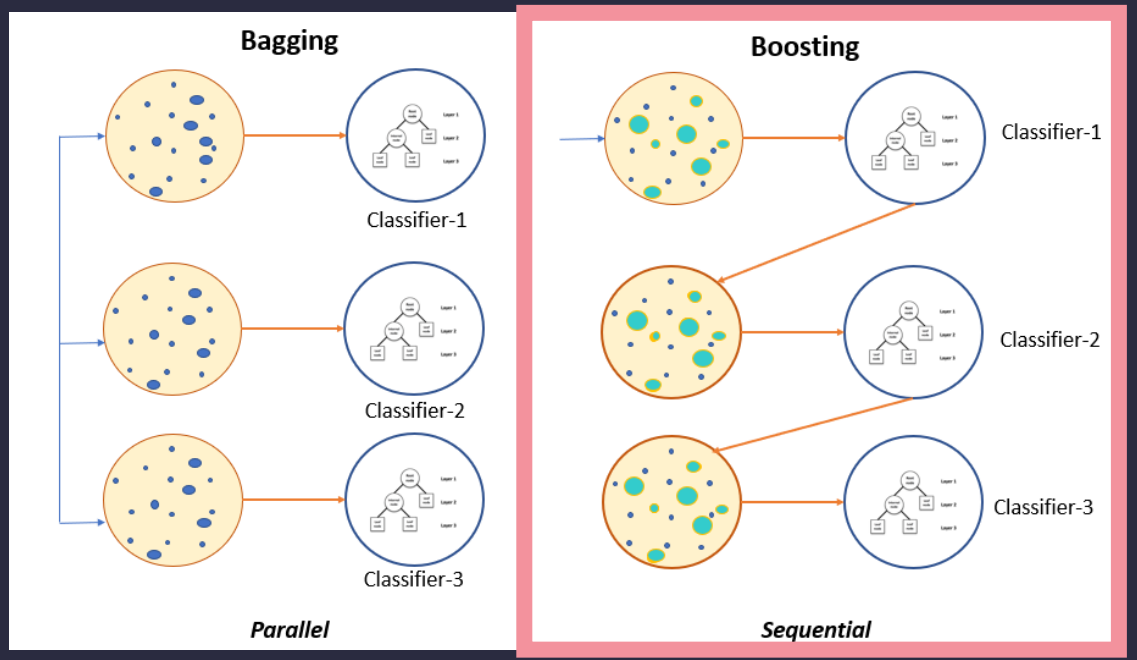
```
X_train=X_train.astype('float32')
y_train=y_train.astype('float32')
X_test=X_test.astype('float32')
y_test=y_test.astype('float32')
```

```
X_train = X_train.reshape(X_train.shape[0],95,95,1)
X_test = X_test.reshape(X_test.shape[0],95,95,1)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(640, 95, 95, 1) (640,) (160, 95, 95, 1) (160,)
```

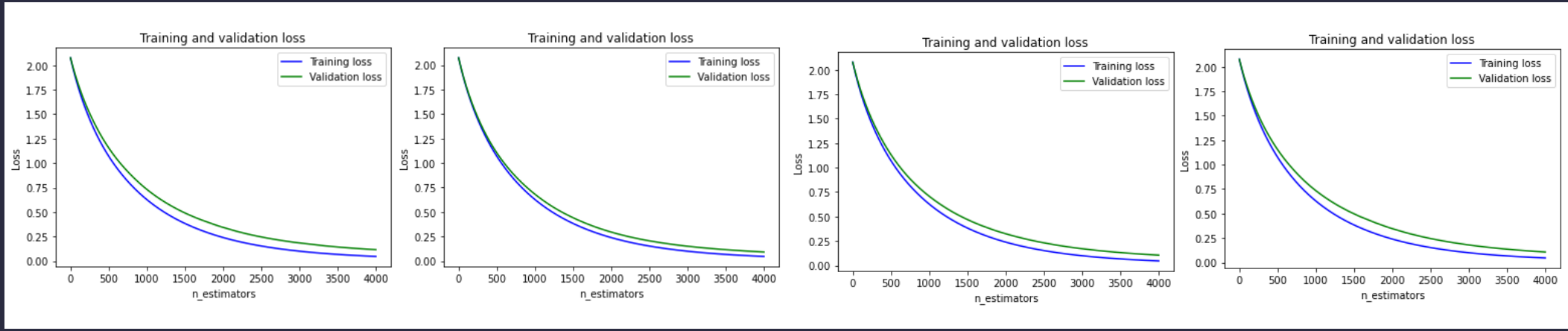
XGBoost

- 여러 개의 Decision Tree를 조합해서 사용하는 Ensemble Boosting 알고리즘.
- Tree 생성 시 CART를 활용해 모든 리프 노드가 최종 스코어에 연관되도록 하여 모델 간의 우위를 확인 할 수 있다.
- 과적합 방지를 위한 규제(Regularization)가 내부에 포함되어 있다.
- Ensemble Boosting의 특징인 가중치 부여를 경사하강법(Gradient Descent)로 계산한다.



Training and Validation Loss

- KFold (k=4) Validation
- Fold 별 Train Loss와 Validation Loss
- 거의 비슷한 양상으로 loss가 줄어들었음을 확인할 수 있음.
- 4번째 fold에서의 마지막 loss: train=0.03754 val=0.22916

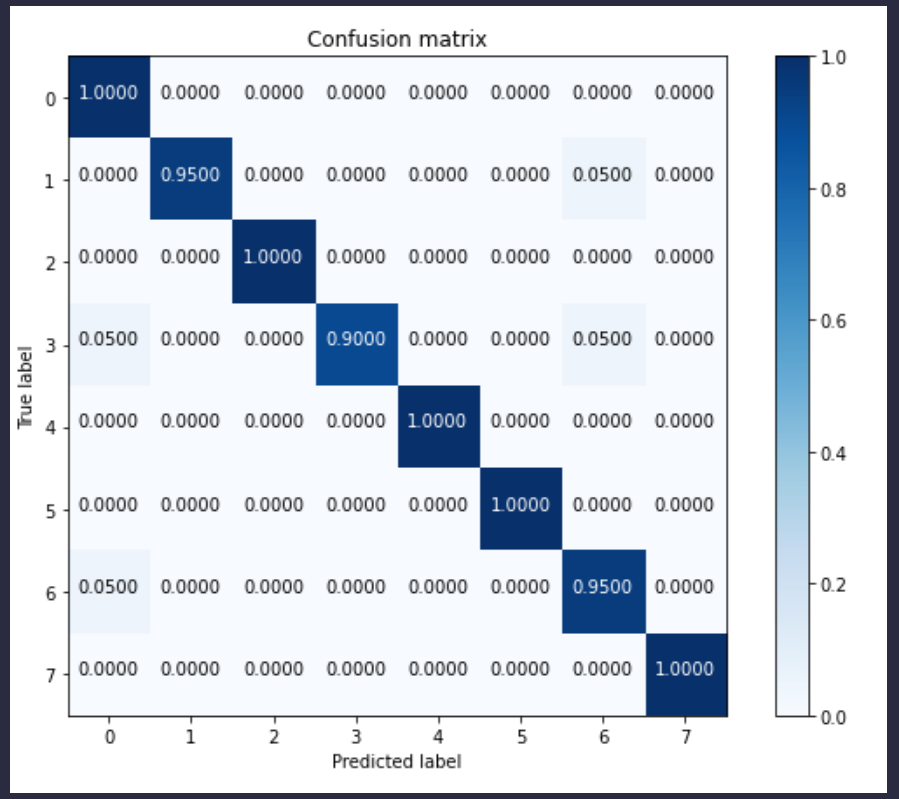


Confusion Matrix

- 앞서 train_test_split으로 분리한 test 데이터를 이용해 모델을 평가함.
- Multi-class model임을 고려하여 confusion matrix를 구함.

```
cm
```

```
array([[20,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 19,  0,  0,  0,  0,  1,  0],
       [ 0,  0, 20,  0,  0,  0,  0,  0],
       [ 1,  0,  0, 18,  0,  0,  1,  0],
       [ 0,  0,  0,  0, 20,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 20,  0,  0],
       [ 1,  0,  0,  0,  0,  0, 19,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 20]])
```



Accuracy, Precision, Recall, F1 score

- Multi-class model임을 고려하여 평가지표들을 계산함.
- 각 label마다의 confusion matrix TP, FN, FP, TN의 평균인 Macro Average를 계산함.

```python
# label마다의 confusion matrix
from sklearn.metrics import multilabel_confusion_matrix
cm_mat = multilabel_confusion_matrix(y_test, preds)
cm_mat
# TP FN
# FP TN
```

```
array([[[138,   2],
        [  0,  20]],

       [[140,   0],
        [  1,  19]],

       [[140,   0],
        [  0,  20]],

       [[140,   0],
        [  2,  18]],

       [[140,   0],
        [  0,  20]],

       [[140,   0],
        [  0,  20]],

       [[138,   2],
        [  1,  19]],

       [[140,   0],
        [  0,  20]]])
```

```python
# Macro Average로 평가지표 계산
accuracy_avg = 0
precision_avg = 0
recall_avg = 0
F1_avg = 0

for i in range(len(cm_mat)):
    accuracy_avg += accuracy(cm_mat[i])
    precision_avg += precision(cm_mat[i])
    recall_avg += recall(cm_mat[i])
    F1_avg += F1(cm_mat[i])

accuracy_avg /= len(cm_mat)
precision_avg /= len(cm_mat)
recall_avg /= len(cm_mat)
F1_avg /= len(cm_mat)

print('accuracy: ', accuracy_avg)
print('precision: ', precision_avg)
print('recall: ', recall_avg)
print('F1: ', F1_avg)
```

```
accuracy:  0.99375
precision:  0.9964536312214837
recall:  0.9964285714285714
F1:  0.9964252687226088
```

Test 예측 결과

	Image	Answer
2	1	7
5	2	2
42	3	2
33	4	6
38	5	1
...	...	...
64	76	1
61	77	7
19	78	1
31	79	7
60	80	8

80 rows × 2 columns

```
res_df['Answer'].value_counts()
```

```
2    20
6    15
3    10
5     9
7     9
8     9
1     5
4     3
Name: Answer, dtype: int64
```

개선방향

- Data Augmentation 시 생성하는 데이터양을 더 늘리거나, 기존 이미지의 특성을 최대한 보존할 수 있도록 augmentation 옵션을 조절.
- 모델 학습 시 하이퍼파라미터 값 조절 (learning_rate, max_depth, n_estimators)
- Cross Validation Fold 수 조절

감사합니다