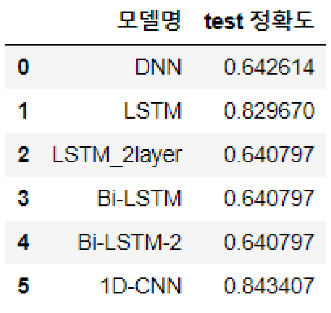
0430 진행상황

1. 악플 리뷰

- **이진분류 성능 개선에 집중하려고함**
- baseline



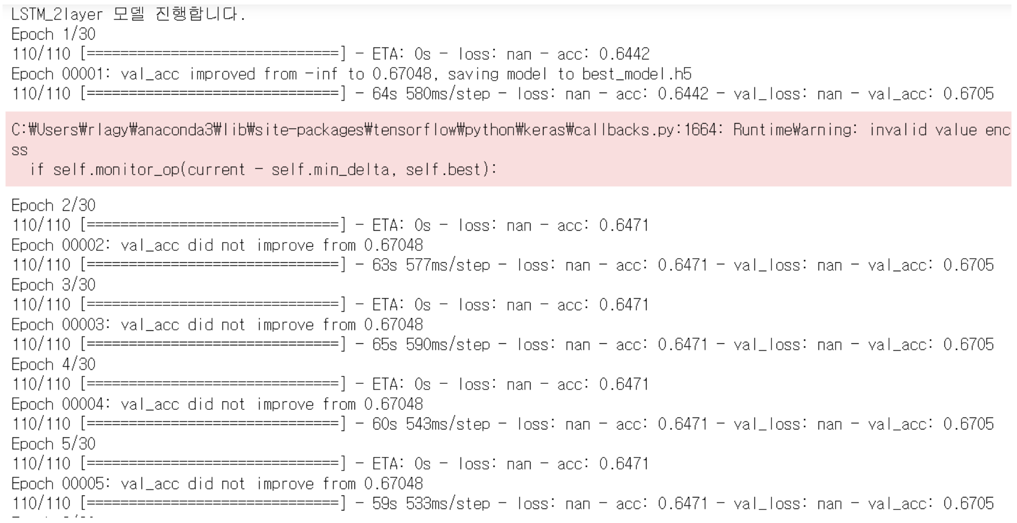
	모델명	test 정확도
0	DNN	0.642614
1	LSTM	0.829670
2	LSTM_2layer	0.640797
3	Bi-LSTM	0.640797
4	Bi-LSTM-2	0.640797
5	1D-CNN	0.843407

LSTM_2layer 성능 높이기 위한 시도

- 모델이 개선안되는 것은 배치 정규화가 필요할 것으로 판단
- 기존 모델

```
model = Sequential()
model.add(Embedding(vocab_size, 100,input_length = max_len))
model.add(LSTM(128, return_sequences=True,activation='relu'))
model.add(LSTM(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

  - loss가 nan 으로 개선되지 않음

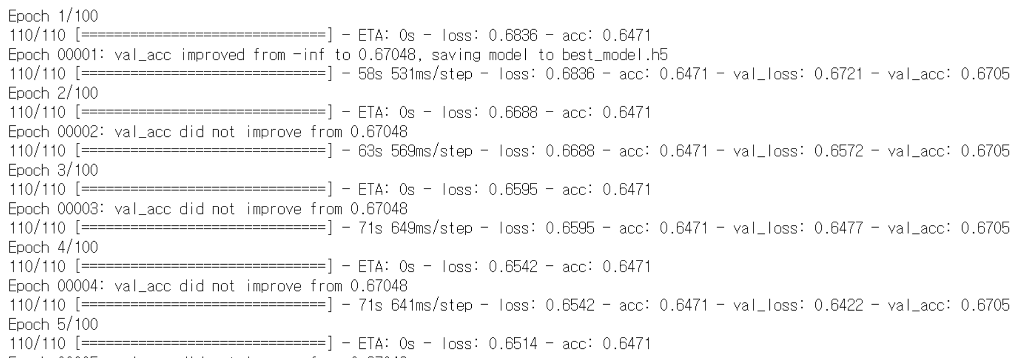


```
LSTM_2layer 모델 진행합니다.
Epoch 1/30
110/110 [==============================] - ETA: 0s - loss: nan - acc: 0.6442
Epoch 00001: val_acc improved from -inf to 0.67048, saving model to best_model.h5
110/110 [==============================] - 64s 580ms/step - loss: nan - acc: 0.6442 - val_loss: nan - val_acc: 0.6705

C:₩Users₩rlagy₩anaconda3₩lib₩site-packages₩tensorflow₩python₩keras₩callbacks.py:1664: RuntimeWarning: invalid value enc
ss
  if self.monitor_op(current - self.min_delta, self.best):

Epoch 2/30
110/110 [==============================] - ETA: 0s - loss: nan - acc: 0.6471
Epoch 00002: val_acc did not improve from 0.67048
110/110 [==============================] - 63s 577ms/step - loss: nan - acc: 0.6471 - val_loss: nan - val_acc: 0.6705
Epoch 3/30
110/110 [==============================] - ETA: 0s - loss: nan - acc: 0.6471
Epoch 00003: val_acc did not improve from 0.67048
110/110 [==============================] - 65s 590ms/step - loss: nan - acc: 0.6471 - val_loss: nan - val_acc: 0.6705
Epoch 4/30
110/110 [==============================] - ETA: 0s - loss: nan - acc: 0.6471
Epoch 00004: val_acc did not improve from 0.67048
110/110 [==============================] - 60s 543ms/step - loss: nan - acc: 0.6471 - val_loss: nan - val_acc: 0.6705
Epoch 5/30
110/110 [==============================] - ETA: 0s - loss: nan - acc: 0.6471
Epoch 00005: val_acc did not improve from 0.67048
110/110 [==============================] - 59s 533ms/step - loss: nan - acc: 0.6471 - val_loss: nan - val_acc: 0.6705
```

- 활성화 함수로 ReLU 함수에서 음의 영역은 미분값이 0 라서 gradient vanishing으로 판단됨

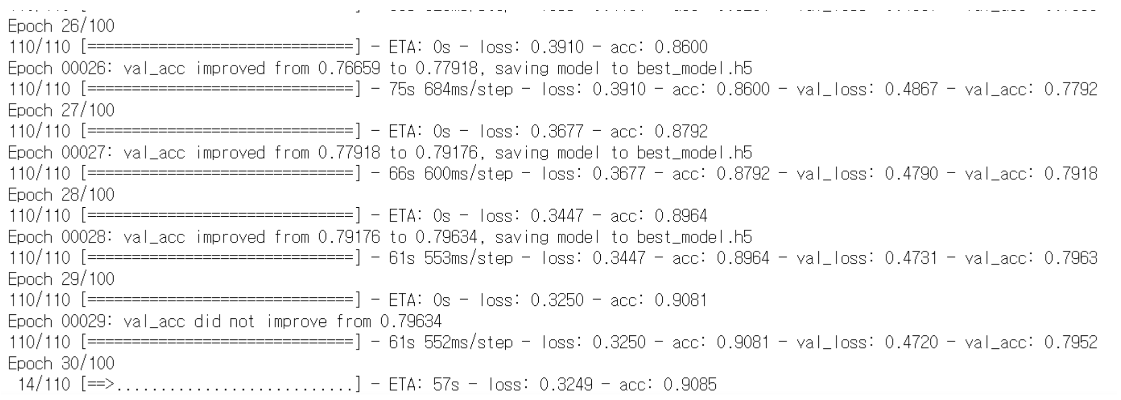
1> ReLU → SoftMax

- 23epoch까지 개선 X



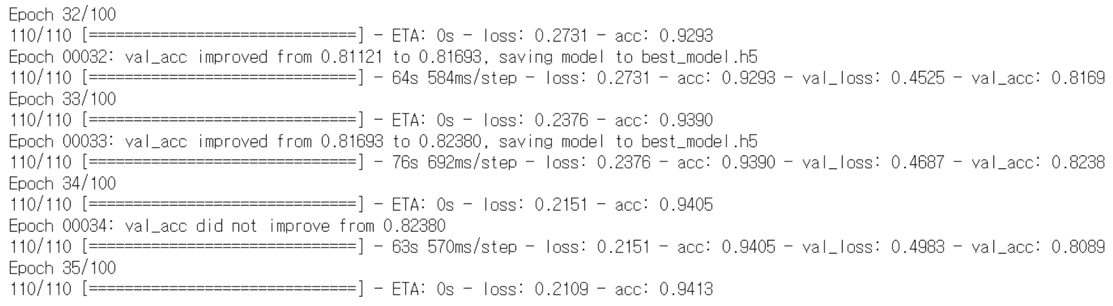
```
Epoch 1/100
110/110 [==============================] - ETA: 0s - loss: 0.6836 - acc: 0.6471
Epoch 00001: val_acc improved from -inf to 0.67048, saving model to best_model.h5
110/110 [==============================] - 58s 531ms/step - loss: 0.6836 - acc: 0.6471 - val_loss: 0.6721 - val_acc: 0.6705
Epoch 2/100
110/110 [==============================] - ETA: 0s - loss: 0.6688 - acc: 0.6471
Epoch 00002: val_acc did not improve from 0.67048
110/110 [==============================] - 63s 569ms/step - loss: 0.6688 - acc: 0.6471 - val_loss: 0.6572 - val_acc: 0.6705
Epoch 3/100
110/110 [==============================] - ETA: 0s - loss: 0.6595 - acc: 0.6471
Epoch 00003: val_acc did not improve from 0.67048
110/110 [==============================] - 71s 649ms/step - loss: 0.6595 - acc: 0.6471 - val_loss: 0.6477 - val_acc: 0.6705
Epoch 4/100
110/110 [==============================] - ETA: 0s - loss: 0.6542 - acc: 0.6471
Epoch 00004: val_acc did not improve from 0.67048
110/110 [==============================] - 71s 641ms/step - loss: 0.6542 - acc: 0.6471 - val_loss: 0.6422 - val_acc: 0.6705
Epoch 5/100
110/110 [==============================] - ETA: 0s - loss: 0.6514 - acc: 0.6471
```

- 24epoch : 0.67 → 0.75



```
Epoch 26/100
110/110 [==============================] - ETA: 0s - loss: 0.3910 - acc: 0.8600
Epoch 00026: val_acc improved from 0.76659 to 0.77918, saving model to best_model.h5
110/110 [==============================] - 75s 684ms/step - loss: 0.3910 - acc: 0.8600 - val_loss: 0.4867 - val_acc: 0.7792
Epoch 27/100
110/110 [==============================] - ETA: 0s - loss: 0.3677 - acc: 0.8792
Epoch 00027: val_acc improved from 0.77918 to 0.79176, saving model to best_model.h5
110/110 [==============================] - 66s 600ms/step - loss: 0.3677 - acc: 0.8792 - val_loss: 0.4790 - val_acc: 0.7918
Epoch 28/100
110/110 [==============================] - ETA: 0s - loss: 0.3447 - acc: 0.8964
Epoch 00028: val_acc improved from 0.79176 to 0.79634, saving model to best_model.h5
110/110 [==============================] - 61s 553ms/step - loss: 0.3447 - acc: 0.8964 - val_loss: 0.4731 - val_acc: 0.7963
Epoch 29/100
110/110 [==============================] - ETA: 0s - loss: 0.3250 - acc: 0.9081
Epoch 00029: val_acc did not improve from 0.79634
110/110 [==============================] - 61s 552ms/step - loss: 0.3250 - acc: 0.9081 - val_loss: 0.4720 - val_acc: 0.7952
Epoch 30/100
 14/110 [==>...........................] - ETA: 57s - loss: 0.3249 - acc: 0.9085
```

- 33epoch : 0.82380



```
Epoch 32/100
110/110 [==============================] - ETA: 0s - loss: 0.2731 - acc: 0.9293
Epoch 00032: val_acc improved from 0.81121 to 0.81693, saving model to best_model.h5
110/110 [==============================] - 64s 584ms/step - loss: 0.2731 - acc: 0.9293 - val_loss: 0.4525 - val_acc: 0.8169
Epoch 33/100
110/110 [==============================] - ETA: 0s - loss: 0.2376 - acc: 0.9390
Epoch 00033: val_acc improved from 0.81693 to 0.82380, saving model to best_model.h5
110/110 [==============================] - 76s 692ms/step - loss: 0.2376 - acc: 0.9390 - val_loss: 0.4687 - val_acc: 0.8238
Epoch 34/100
110/110 [==============================] - ETA: 0s - loss: 0.2151 - acc: 0.9405
Epoch 00034: val_acc did not improve from 0.82380
110/110 [==============================] - 63s 570ms/step - loss: 0.2151 - acc: 0.9405 - val_loss: 0.4983 - val_acc: 0.8089
Epoch 35/100
110/110 [==============================] - ETA: 0s - loss: 0.2109 - acc: 0.9413
```

Word2Vec 임베딩 벡터

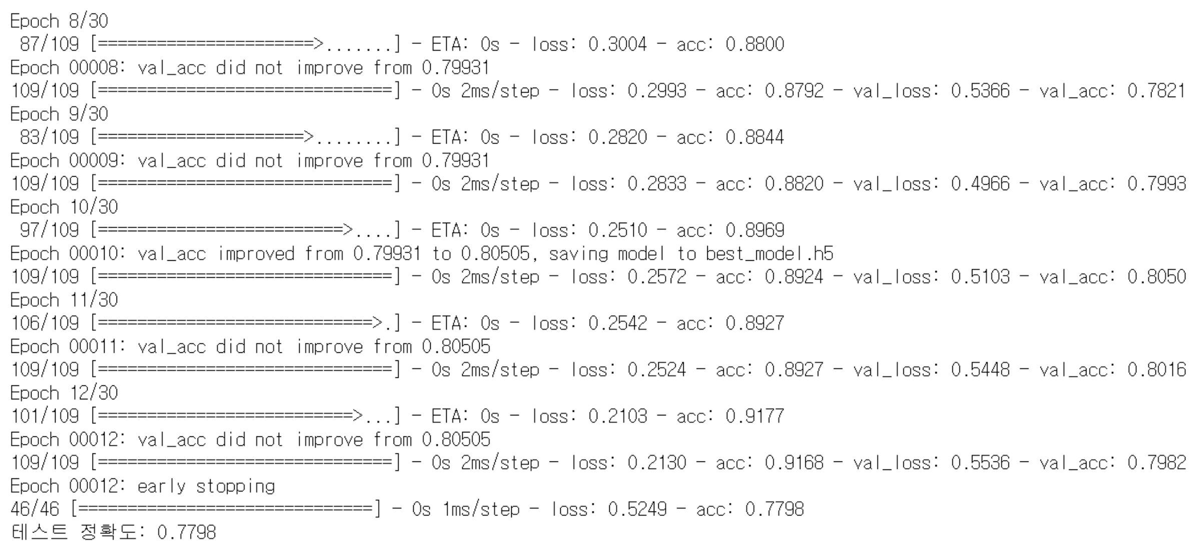
- 인풋 형태 (4367, 100)

```
array([-0.02549689,  0.24735869,  0.3063997 , -0.14861175,  0.22034401,
       -0.05685987, -0.18278548,  0.02033075,  0.02566293,  0.04855688,
       -0.21327452, -0.00349124,  0.07556671,  0.24129547, -0.03331769,
        0.05811284, -0.0717685 ,  0.026247  , -0.08210494,  0.16270405,
       -0.1350276 ,  0.02370879, -0.17512017, -0.03608694,  0.13943648,
       -0.21575195,  0.22488207,  0.32437962, -0.03021993,  0.10286586,
       -0.01721002,  0.08643222, -0.18955639, -0.09883802, -0.03502379,
        0.28606951,  0.01635696,  0.15862948, -0.03655412,  0.17878653,
       -0.12862492, -0.04100036, -0.08516292, -0.17757446,  0.09372382,
        0.01253376, -0.06607044,  0.2771281 ,  0.08628678, -0.05863844,
        0.21659534, -0.00168831,  0.00553003,  0.11483034, -0.08004195,
        0.08319696,  0.24648687,  0.0088496 , -0.0580439 , -0.01402395,
        0.17426305, -0.11382988, -0.02616155,  0.05249323, -0.1843337 ,
        0.12482968,  0.01941842, -0.02086451, -0.16731793,  0.05321756,
       -0.20937675,  0.08012627,  0.01338697, -0.01999496, -0.13521172,
        0.08114912, -0.0306044 ,  0.21554352, -0.04108846,  0.07340673,
        0.18891294, -0.04754576,  0.07606068, -0.00824437, -0.12784453,
        0.19512208,  0.24537489, -0.07527512, -0.12111431, -0.19499163,
       -0.13916944,  0.00882497,  0.17978239, -0.012806  ,  0.17308277,
        0.10000542, -0.00247007, -0.10192502,  0.07840657, -0.08825628])
```

1.Baseline에서 Embedding layer만 제거한 DNN

```
def DNN():
    # 모델 구조 정의하기
    model = models.Sequential()
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(128, activation='relu')) #ReLU 활성화함수 채택
    model.add(layers.Dense(1, activation='sigmoid'))
    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
    mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
    history = model.fit(X_train, y_train, epochs=30, callbacks=[es, mc], batch_size=32, validation_split=0.2)
    loaded_model = load_model('best_model.h5')
    score = loaded_model.evaluate(X_test, y_test)[1]
    print("테스트 정확도: %.4f" % (score))
#     test_result.append(('DNN',score))
```

```
Epoch 1/30
109/109 [==============================] - ETA: 0s - loss: 0.5372 - acc: 0.7181
Epoch 00001: val_acc improved from -inf to 0.77867, saving model to best_model.h5
109/109 [==============================] - 1s 9ms/step - loss: 0.5372 - acc: 0.7181 - val_loss: 0.4818 - val_acc: 0.7787
Epoch 2/30
 97/109 [=========================>....] - ETA: 0s - loss: 0.4416 - acc: 0.7951
Epoch 00002: val_acc improved from 0.77867 to 0.79128, saving model to best_model.h5
109/109 [==============================] - 1s 5ms/step - loss: 0.4403 - acc: 0.7959 - val_loss: 0.4527 - val_acc: 0.7913
Epoch 3/30
104/109 [===========================>..] - ETA: 0s - loss: 0.4056 - acc: 0.8188
Epoch 00003: val_acc did not improve from 0.79128
109/109 [==============================] - 0s 4ms/step - loss: 0.4031 - acc: 0.8206 - val_loss: 0.4693 - val_acc: 0.7878
Epoch 4/30
 98/109 [=========================>....] - ETA: 0s - loss: 0.3827 - acc: 0.8243
Epoch 00004: val_acc did not improve from 0.79128
109/109 [==============================] - 0s 4ms/step - loss: 0.3839 - acc: 0.8246 - val_loss: 0.4579 - val_acc: 0.7821
Epoch 5/30
 85/109 [======================>.......] - ETA: 0s - loss: 0.3628 - acc: 0.8393
Epoch 00005: val_acc improved from 0.79128 to 0.79817, saving model to best_model.h5
109/109 [==============================] - 0s 2ms/step - loss: 0.3657 - acc: 0.8387 - val_loss: 0.4663 - val_acc: 0.7982
Epoch 6/30
 93/109 [========================>.....] - ETA: 0s - loss: 0.3334 - acc: 0.8599
Epoch 00006: val_acc improved from 0.79817 to 0.79931, saving model to best_model.h5
109/109 [==============================] - 0s 3ms/step - loss: 0.3360 - acc: 0.8588 - val_loss: 0.4678 - val_acc: 0.7993
```



```
Epoch 8/30
 87/109 [======================>.......] - ETA: 0s - loss: 0.3004 - acc: 0.8800
Epoch 00008: val_acc did not improve from 0.79931
109/109 [==============================] - 0s 2ms/step - loss: 0.2993 - acc: 0.8792 - val_loss: 0.5366 - val_acc: 0.7821
Epoch 9/30
 83/109 [=====================>........] - ETA: 0s - loss: 0.2820 - acc: 0.8844
Epoch 00009: val_acc did not improve from 0.79931
109/109 [==============================] - 0s 2ms/step - loss: 0.2833 - acc: 0.8820 - val_loss: 0.4966 - val_acc: 0.7993
Epoch 10/30
 97/109 [=========================>....] - ETA: 0s - loss: 0.2510 - acc: 0.8969
Epoch 00010: val_acc improved from 0.79931 to 0.80505, saving model to best_model.h5
109/109 [==============================] - 0s 2ms/step - loss: 0.2572 - acc: 0.8924 - val_loss: 0.5103 - val_acc: 0.8050
Epoch 11/30
106/109 [============================>.] - ETA: 0s - loss: 0.2542 - acc: 0.8927
Epoch 00011: val_acc did not improve from 0.80505
109/109 [==============================] - 0s 2ms/step - loss: 0.2524 - acc: 0.8927 - val_loss: 0.5448 - val_acc: 0.8016
Epoch 12/30
101/109 [==========================>...] - ETA: 0s - loss: 0.2103 - acc: 0.9177
Epoch 00012: val_acc did not improve from 0.80505
109/109 [==============================] - 0s 2ms/step - loss: 0.2130 - acc: 0.9168 - val_loss: 0.5536 - val_acc: 0.7982
Epoch 00012: early stopping
46/46 [==============================] - 0s 1ms/step - loss: 0.5249 - acc: 0.7798
테스트 정확도: 0.7798
```

테스트 정확도 : 0.7798

- embedding layer로 3차원으로 나타내지 못하기 때문에 앞에서 사용한 다양한 모델을 사용할 수 있을까

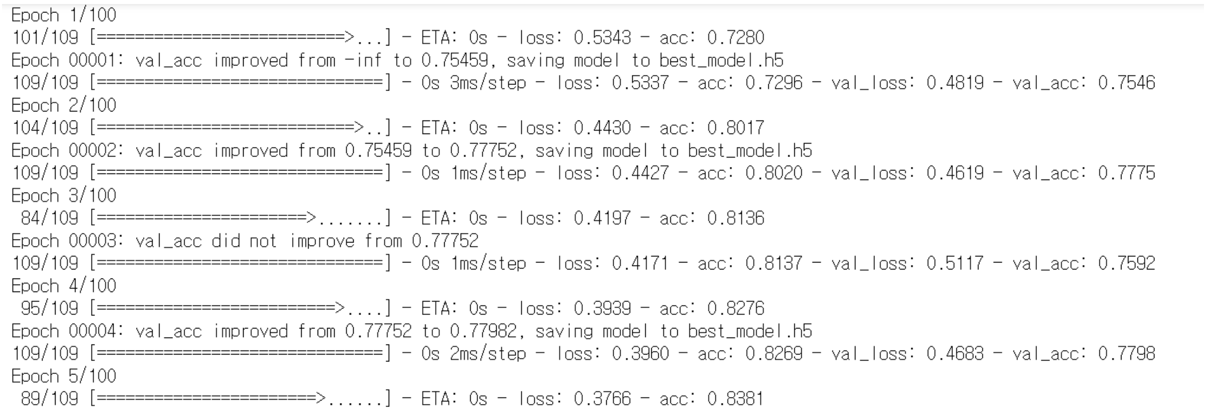
개선 Task

1> 배치 정규화 : 효과 없었음

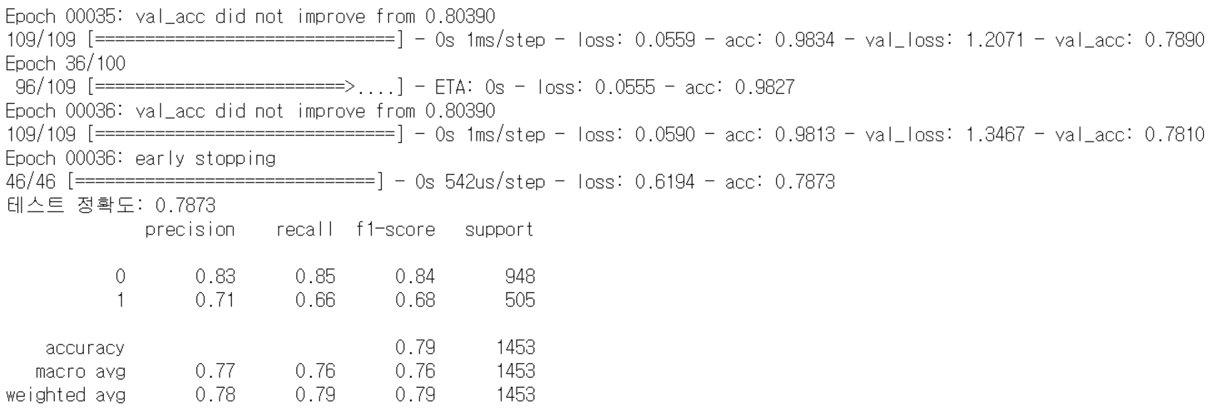
2> ReLU → Softmax : 테스트 정확도 0.7522

3> **옵티마이저 : Adam → RMSprop (테스트 정확도 : 0.7818)**

```
def DNN():
    # 모델 구조 정의하기
    model = models.Sequential()
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(128, activation='relu')) #ReLU 활성화함수 채택
    model.add(layers.Dense(1, activation='sigmoid'))
    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=30)
    mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
    model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
    history = model.fit(X_train, y_train, epochs=100, callbacks=[es, mc], batch_size=32, validation_split=0.2)
    loaded_model = load_model('best_model.h5')
    score = loaded_model.evaluate(X_test, y_test)[1]
    print("테스트 정확도: %.4f" % (score))
#     test_result.append(('DNN',score))
```



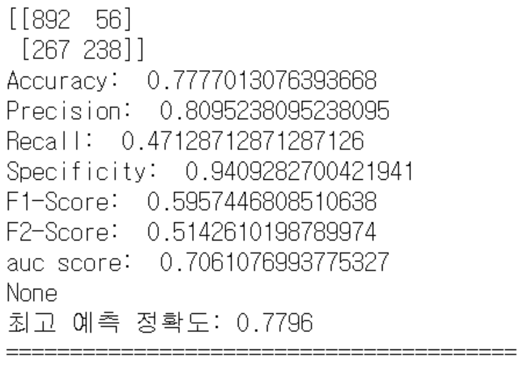
```
Epoch 1/100
101/109 [==========================>...] - ETA: 0s - loss: 0.5343 - acc: 0.7280
Epoch 00001: val_acc improved from -inf to 0.75459, saving model to best_model.h5
109/109 [==============================] - 0s 3ms/step - loss: 0.5337 - acc: 0.7296 - val_loss: 0.4819 - val_acc: 0.7546
Epoch 2/100
104/109 [===========================>..] - ETA: 0s - loss: 0.4430 - acc: 0.8017
Epoch 00002: val_acc improved from 0.75459 to 0.77752, saving model to best_model.h5
109/109 [==============================] - 0s 1ms/step - loss: 0.4427 - acc: 0.8020 - val_loss: 0.4619 - val_acc: 0.7775
Epoch 3/100
 84/109 [======================>.......] - ETA: 0s - loss: 0.4197 - acc: 0.8136
Epoch 00003: val_acc did not improve from 0.77752
109/109 [==============================] - 0s 1ms/step - loss: 0.4171 - acc: 0.8137 - val_loss: 0.5117 - val_acc: 0.7592
Epoch 4/100
 95/109 [=========================>....] - ETA: 0s - loss: 0.3939 - acc: 0.8276
Epoch 00004: val_acc improved from 0.77752 to 0.77982, saving model to best_model.h5
109/109 [==============================] - 0s 2ms/step - loss: 0.3960 - acc: 0.8269 - val_loss: 0.4683 - val_acc: 0.7798
Epoch 5/100
 89/109 [=======================>......] - ETA: 0s - loss: 0.3766 - acc: 0.8381
```



```
Epoch 00035: val_acc did not improve from 0.80390
109/109 [==============================] - 0s 1ms/step - loss: 0.0559 - acc: 0.9834 - val_loss: 1.2071 - val_acc: 0.7890
Epoch 36/100
 96/109 [=========================>....] - ETA: 0s - loss: 0.0555 - acc: 0.9827
Epoch 00036: val_acc did not improve from 0.80390
109/109 [==============================] - 0s 1ms/step - loss: 0.0590 - acc: 0.9813 - val_loss: 1.3467 - val_acc: 0.7810
Epoch 00036: early stopping
46/46 [==============================] - 0s 542us/step - loss: 0.6194 - acc: 0.7873
테스트 정확도: 0.7873
              precision    recall  f1-score   support

           0       0.83      0.85      0.84       948
           1       0.71      0.66      0.68       505

    accuracy                           0.79      1453
   macro avg       0.77      0.76      0.76      1453
weighted avg       0.78      0.79      0.79      1453
```

2. 랜덤포레스트 모델 (Word2Vec 임베딩 벡터)

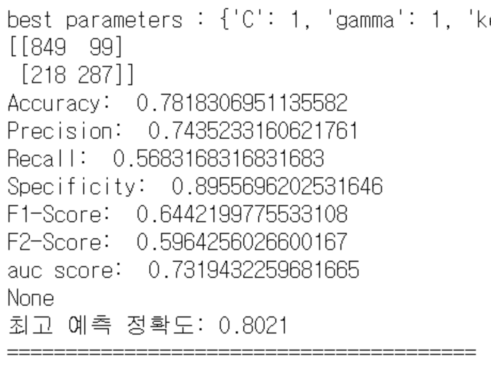
```
#최적 파라미터 값 찾기
params = { 'n_estimators' : [10, 100,200],
           'max_depth' : [6, 8, 10],
           'min_samples_leaf' : [3,5 ,7],
           'min_samples_split' : [2,3,6]}
rf_clf = RandomForestClassifier(random_state = 0, n_jobs = -1)
kfold = KFold(n_splits=5, shuffle=True, random_state=0) #업샘플링할때
# skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0) #업샘플링 하지 않았을때
grid_cv = GridSearchCV(rf_clf, param_grid = params, cv = kfold, n_jobs = -1)
grid_cv.fit(X_train, y_train)
print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
```



```
[[892  56]
 [267 238]]
Accuracy:  0.7777013076393668
Precision:  0.8095238095238095
Recall:  0.47128712871287126
Specificity:  0.9409282700421941
F1-Score:  0.5957446808510638
F2-Score:  0.5142610198789974
auc score:  0.7061076993775327
None
최고 예측 정확도: 0.7796
========================================
```

3. SVM 모델 (Word2Vec 임베딩 벡터)

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
#최적 파라미터 값 찾기
param_grid = [ { 'C' : [0.1, 1, 10 ], 'kernel': [ 'rbf' ], 'gamma' : [ 1,0.1 ]},
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                     ]

def svm_model(train_x, train_y,test_x, test_y,cv):
    print("데이터셋 : ", train_x ," &  cv: " , cv)
    grid_search = GridSearchCV(SVC(),param_grid, cv=cv, return_train_score = True)
    grid_search.fit(train_x, train_y)
    print("best parameters : {}".format(grid_search.best_params_))
    predicted = grid_search.predict(test_x)
    print(confusion_matrix(test_y, predicted))
    print(model_evaluation(test_y, predicted))
    print('최고 예측 정확도: {:.4f}'.format(grid_search.best_score_))
    print('='*40)
    print()
    print()
```



```
best parameters : {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
[[849  99]
 [218 287]]
Accuracy:  0.7818306951135582
Precision:  0.7435233160621761
Recall:  0.5683168316831683
Specificity:  0.8955696202531646
F1-Score:  0.6442199775533108
F2-Score:  0.5964256026600167
auc score:  0.7319432259681665
None
최고 예측 정확도: 0.8021
========================================
```

정리 및 다음주 진행방향

- Word2Vec 임베딩 벡터를 통한 DNN 모델은 이전의 DNN 모델보다 상당한 성능 개선이 이루어짐
- 그러나, 임베딩 벡터는 이미 Embedding 되어 있는 형태로 LSTM, 1D-CNN 등 다른 모델에 적용하는데 있어서 차원이 안 맞는 실정
    - **(이를 해결하는 방안을 알아볼 것)**
- **텍스트 분야에서 data argumentation 공부**