PE 프로그램 진행과정 (8.12)

01

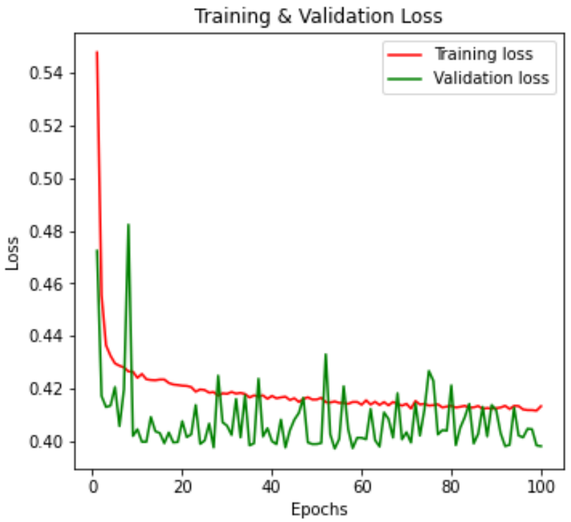
MLP 모델 성능 개선

```
def mlp_model2():
    model = Sequential()

    model.add(Dense(64, input_dim= width, activation = 'relu'))
    model.add(Dense(64, activation = 'relu'))
    model.add(Dense(2, activation = 'sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics=['acc'])

    return model
```

```
results = model2.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] -
Test accuracy:  0.815852165222168
```

- **활성함수(Activation Function)**
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- 배치 정규화(Batch Normalization)
-드랍아웃(Dropout)
-앙상블(Model Ensemble)



활성함수 변경

sigmoid & softmax > ReLU & sigmoid

test accuracy 68%에서 81%로 비약적으로 증가

PE 프로그램 진행과정 (8.12)

02

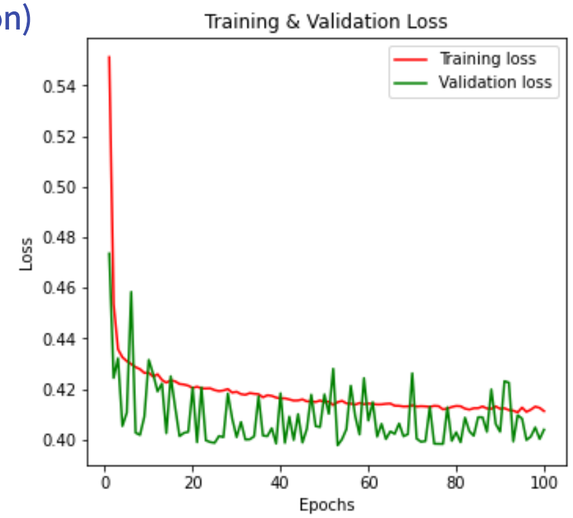
MLP 모델 성능 개선

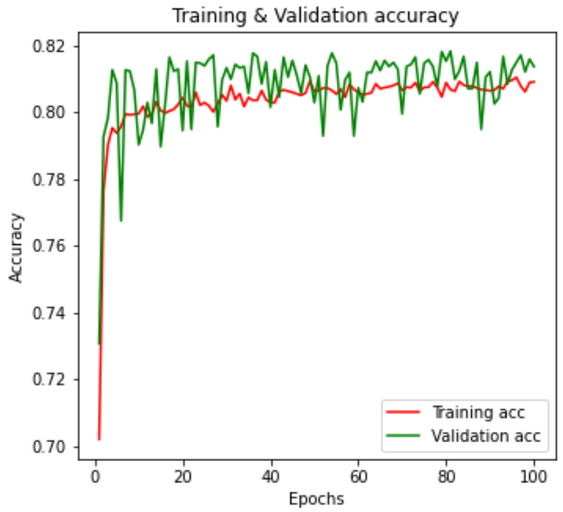
- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- 배치 정규화(Batch Normalization)
-드랍아웃(Dropout)
-앙상블(Model Ensemble)

```
def mlp_model3():
    model = Sequential()

    model.add(Dense(64, input_dim= width, kernel_initializer = 'glorot_uniform'))
    model.add(Activation('relu'))
    model.add(Dense(64, kernel_initializer = 'glorot_uniform'))
    model.add(Activation('relu'))
    model.add(Dense(2, kernel_initializer = 'glorot_uniform'))
    model.add(Activation('sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics=['acc'])

    return model
```

```
results = model3.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] -
Test accuracy:  0.8153985738754272
```





가중치 초기화 방식 변경

random_uniform > Xavier initialization

test accuracy 81%

여전히 불안정한 상태

PE 프로그램 진행과정 (8.12)

03

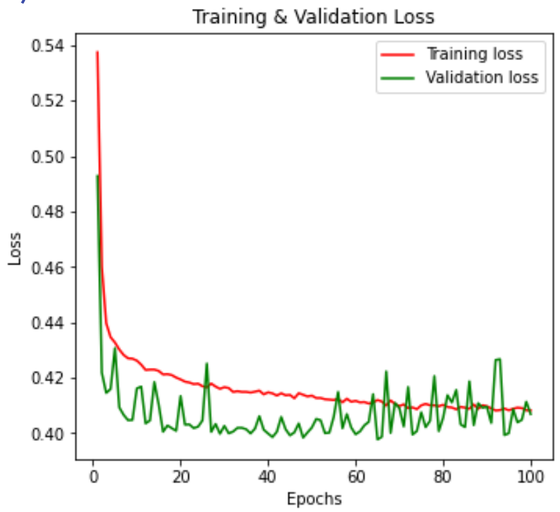
MLP 모델 성능 개선

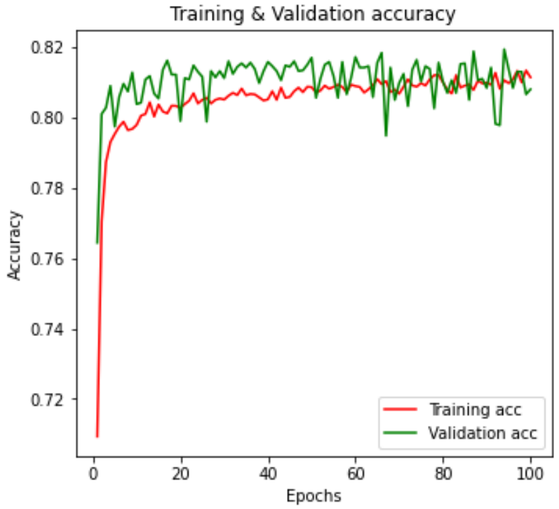
- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- 배치 정규화(Batch Normalization)
-드랍아웃(Dropout)
-앙상블(Model Ensemble)

```
def mlp_model4():
    model = Sequential()

    model.add(Dense(64, input_dim= width, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dense(64, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dense(2, kernel_initializer = 'HeNormal'))
    model.add(Activation('sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics=['acc'])

    return model
```

```
results = model4.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] - [
Test accuracy:  0.8156253695487976
```





가중치 초기화 방식 변경

Xavier initialization > He initialization

test accuracy 81%

검증 정확도의 증가가 미세히 안정적으로 이루어짐

다층퍼셉트론과 순환신경망 기반한국 프로야구 결과 예측 모델

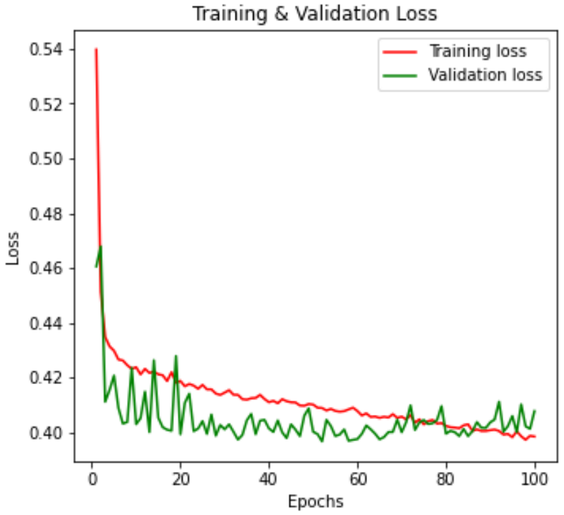
PE 프로그램 진행과정 (8.12)

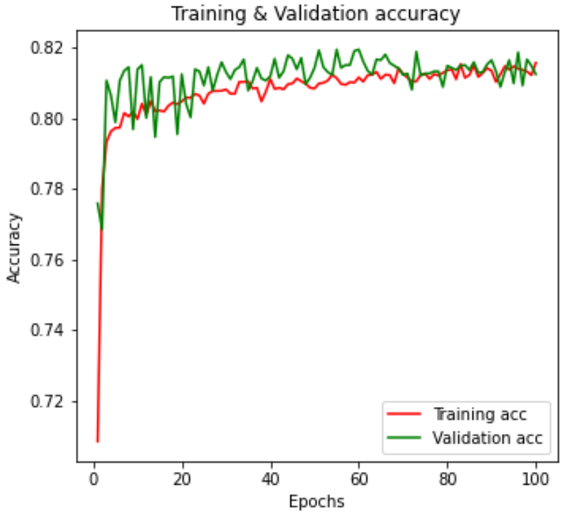
04

MLP 모델 성능 개선

- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- **최적화(Optimization)**
- 배치 정규화(Batch Normalization)
-드랍아웃(Dropout)
-앙상블(Model Ensemble)

```
def mlp_model5():
    model = Sequential()

    model.add(Dense(64, input_dim= width, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dense(64, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dense(2, kernel_initializer = 'HeNormal'))
    model.add(Activation('sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics=['acc'])
```






```
results = model5.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] -
Test accuracy:  0.8104093670845032
```

최적화 방식 변경

RMSprop > Adam

test accuracy 81%

검증 정확도의 증가가 훨씬 안정적으로 이루어짐

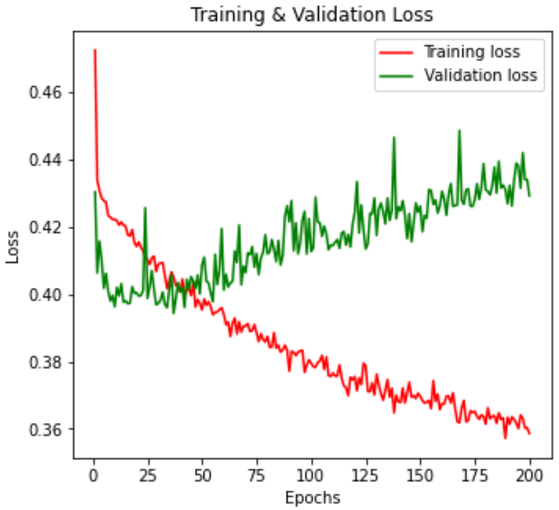
PE 프로그램 진행과정 (8.12)

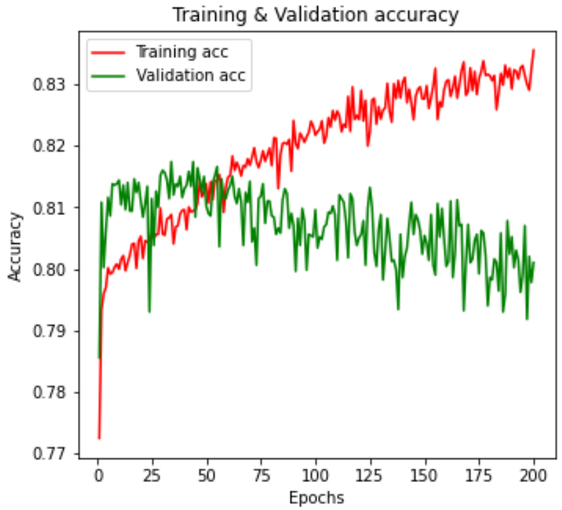
05

MLP 모델 성능 개선

- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- **배치 정규화(Batch Normalization)**
-드랍아웃(Dropout)
-앙상블(Model Ensemble)

```
def mlp_model6():
    model = Sequential()

    model.add(Dense(64, input_dim= width, kernel_initializer = 'HeNormal'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dense(64, kernel_initializer = 'HeNormal'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dense(2, kernel_initializer = 'HeNormal'))
    model.add(Activation('sigmoid'))
```






```
results = model6.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] -
Test accuracy:  0.7043882608413696
```

네트워크의 학습이 잘 일어나도록 배치 정규화

Dense layer와 Activation layer사이에 배치

test accuracy 70%

오히려 불안정해질뿐더러 정확도 역시 대폭 감소

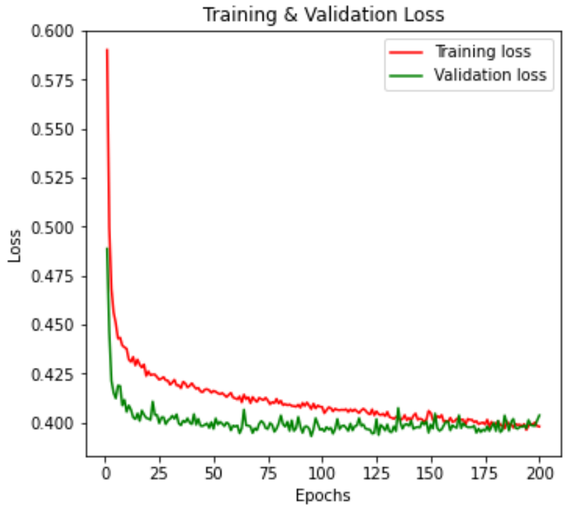
PE 프로그램 진행과정 (8.12)

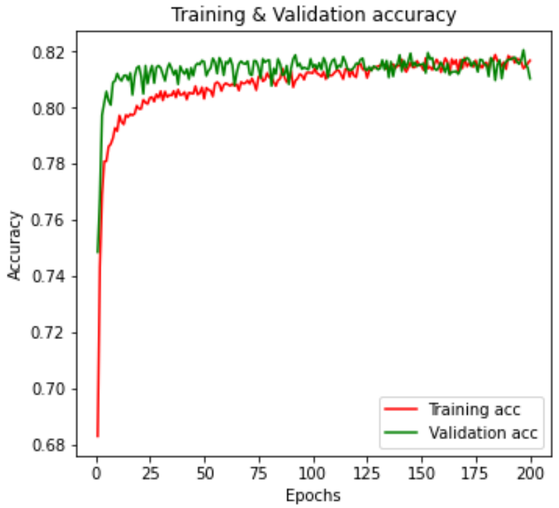
06

MLP 모델 성능 개선

- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- 배치 정규화(Batch Normalization)
- **-드랍아웃(Dropout)**
- -앙상블(Model Ensemble)

```
def mlp_model7():
    model = Sequential()

    model.add(Dense(64, input_dim= width, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(64, kernel_initializer = 'HeNormal'))
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(2, kernel_initializer = 'HeNormal'))
    model.add(Activation('sigmoid'))
```






```
results = model7.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
276/276 [==============================] - 0s
Test accuracy:  0.8033790588378906
```

Overfitting 줄이기 위해 드랍아웃
Activation layer뒤에 배치

test accuracy 80%
Overfitting 줄이고 및 검증 정확도 증가

PE 프로그램 진행과정 (8.12)

07

MLP 모델 성능 개선

- 활성함수(Activation Function)
- 가중치 초기화(Weight Initialization)
- 최적화(Optimization)
- 배치 정규화(Batch Normalization)
-드랍아웃(Dropout)
-앙상블(Model Ensemble)

```
en_model1 = KerasClassifier(build_fn = mlp_model_en, epochs = 200, verbose = 0)
en_model1._estimator_type="classifier"
en_model2 = KerasClassifier(build_fn = mlp_model_en, epochs = 200, verbose = 0)
en_model2._estimator_type="classifier"
en_model3 = KerasClassifier(build_fn = mlp_model_en, epochs = 200, verbose = 0)
en_model3._estimator_type="classifier"
```

```
ensemble_clf = VotingClassifier(estimators = [('en_model1', en_model1), ('en_model2', en_model2), ('en_model3', en_model3)], voting = 'soft')
ensemble_clf.fit(X_train, y_train)
```

```
y_pred = ensemble_clf.predict(X_test)
print('Acc: ', accuracy_score(y_pred, y_test))
```

```
Acc:  0.8081415126431568
```

서로 다른 모델 3개를 만들어 앙상블

test accuracy 81%

검증 정확도면에서 큰 차이는 없음

MLP model 최종 결과 81%의 정확도

추후 네트워크 구조를 바꾼다면 성능 개선의 여지가 있음