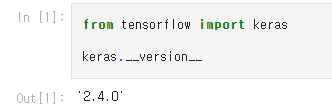
**기계학습[CSE4315-001] Wine Classifier**

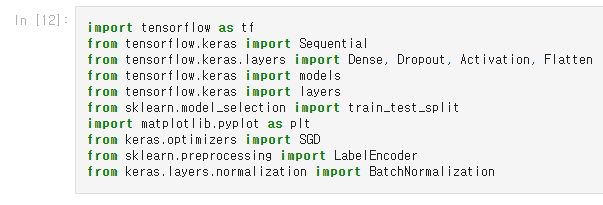
컴퓨터공학과 3학년

12171645 심규범

**1.3 케라스 설치 확인 및 class import**



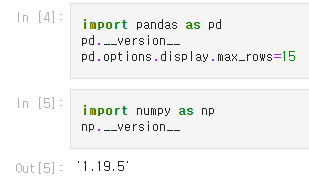
* Tensorflow와 Keras가 정상적으로 설치가 되었는지 확인합니다. Tensorflow 다운로드 중에 pip에 오류가 생겨서 pip update를 해보았으나 해결되지 않아서 Anaconda 3를 삭제 후 재설치 하니 정상적으로 Tensorflow와 Keras가 설치되었습니다.



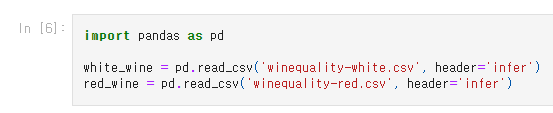
* 필요한 class들을 모두 import 하였습니다.

**2.1 데이터 로드**

**1) Pandas, Numpy Import**



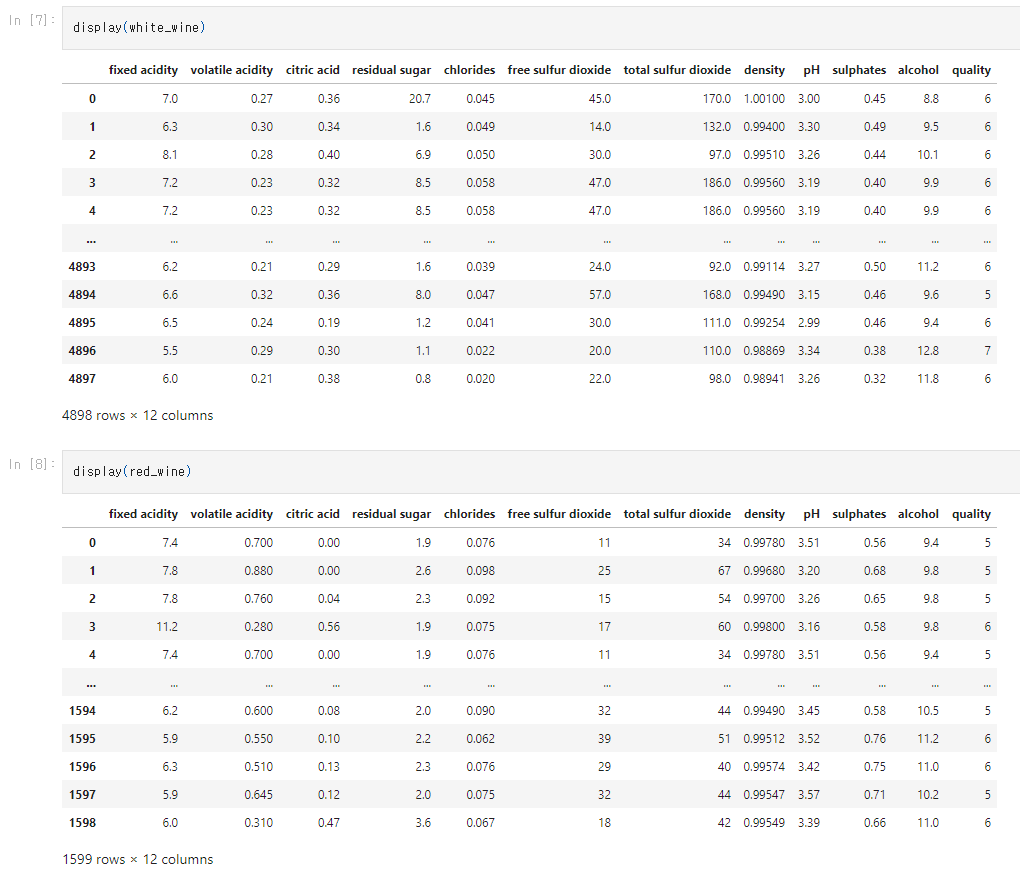
**2) Read CSV files**



= 첫 행을 열 이름으로 사용하여 각 CSV파일을 읽어들입니다.

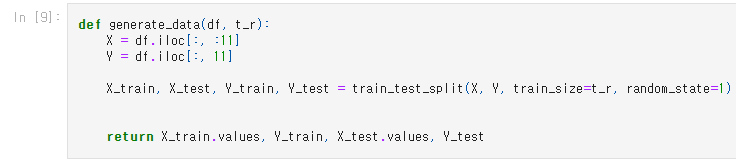
**2.2 데이터 전처리**

**1) White\_wine, Red\_wine display**



* 각각의 데이터를 display합니다.

**2) Generate\_data()**



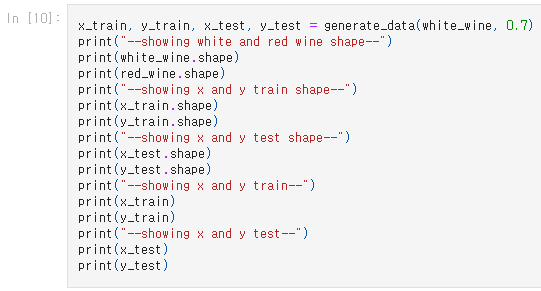
* 입력되는 DataFrame을 받아 row와 column의 이름이 아닌 각각의 index값으로 받습니다
* X는 DataFrame의 row 전체와 0~10번 column을 받습니다.
* Y는 DataFrame의 row 전체와 11번 column을 받습니다.
* sklearn.model\_selection에서 train\_test\_split을 import합니다.
* ‘train\_size=t\_r’ 은 ‘test\_size=(1-t\_r)’로 대체 가능합니다.
* Random\_state에 1을 대입하여 숫자를 random하게 생성할 때 seed로 사용합니다.
* t\_r의 비율에 의해 training\_set과 test\_set이 나눠집니다.

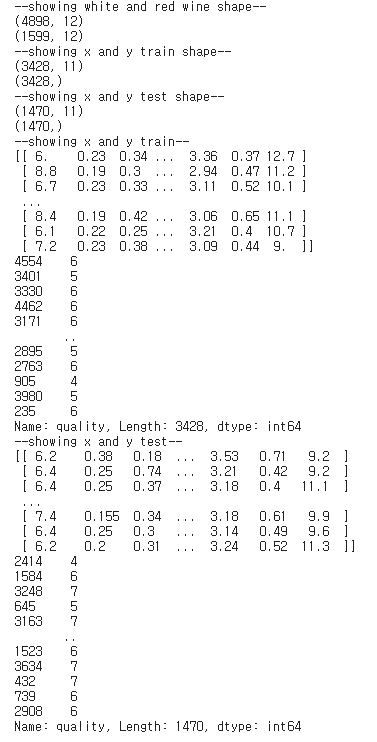
**3) x\_train, y\_train, x\_test, y\_test, white\_wine, red\_wine의 형태 출력**

- white\_wine과 red\_wine의 DataFrame을 행과 열의 개수의 tuple로 변환하여 출력합니다.

- x와 y training set, test set을 행과 열의 개수의 tuple로 변환하여 출력합니다.

- x와 y training set, test set을 출력합니다.

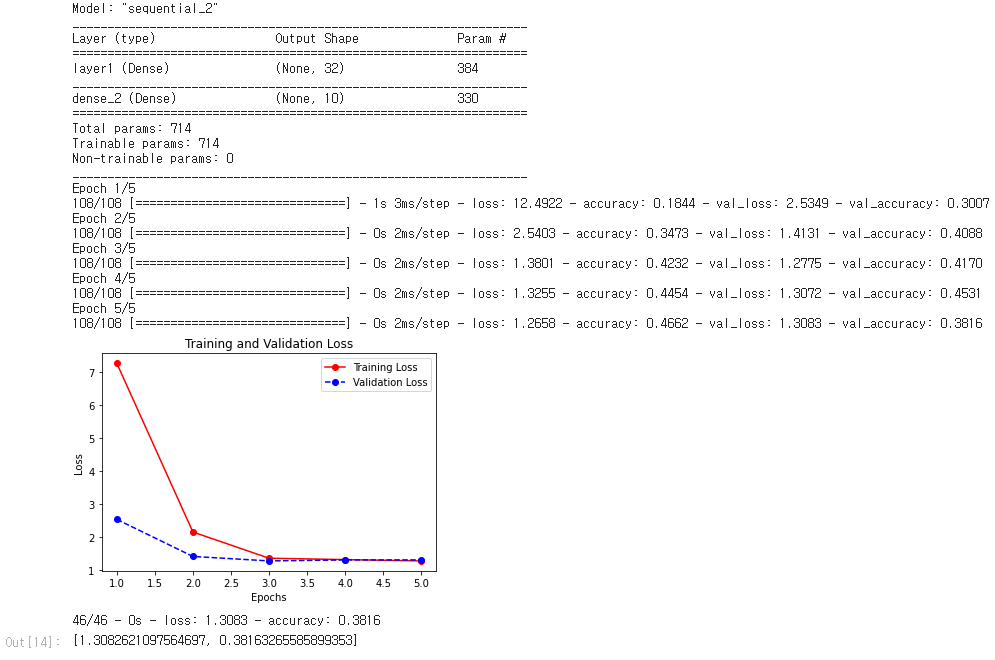




**3.1 화이트, 레드 와인 분류 모델 설계 및 학습**

**1) white\_wine 분류 모델 설계 및 학습**



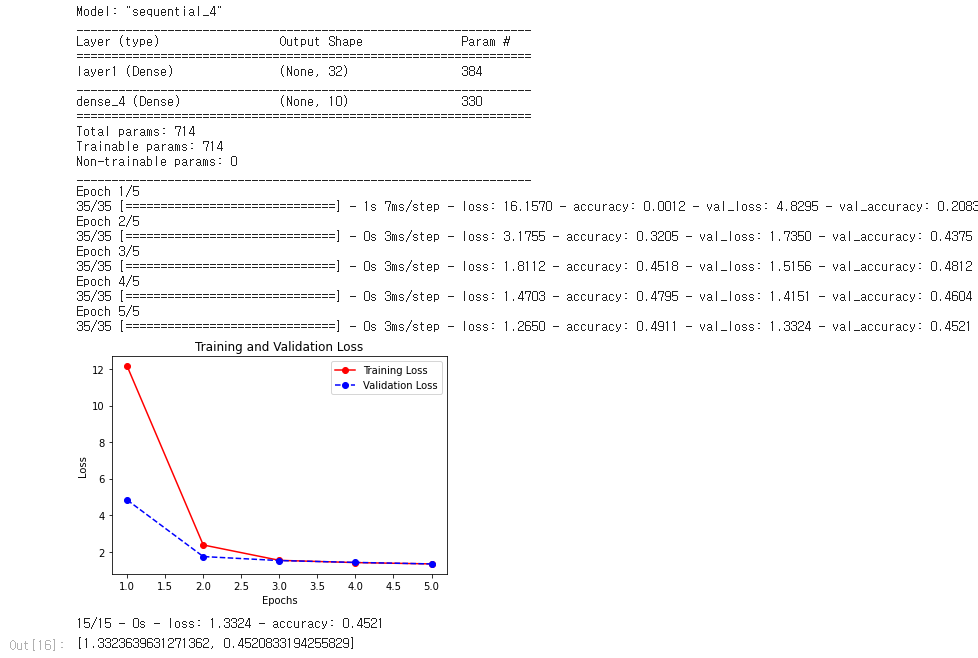


- Node가 32개인 Hidden Layer 한 개가 있는 white\_wine 분류 모델을 설계 및 학습하고, 시각화하여 출력하여 대조군을 만듭니다.

**-** white\_wine 대조군의 정확도: 0.3816

**2) red\_wine 분류 모델 설계 및 학습**



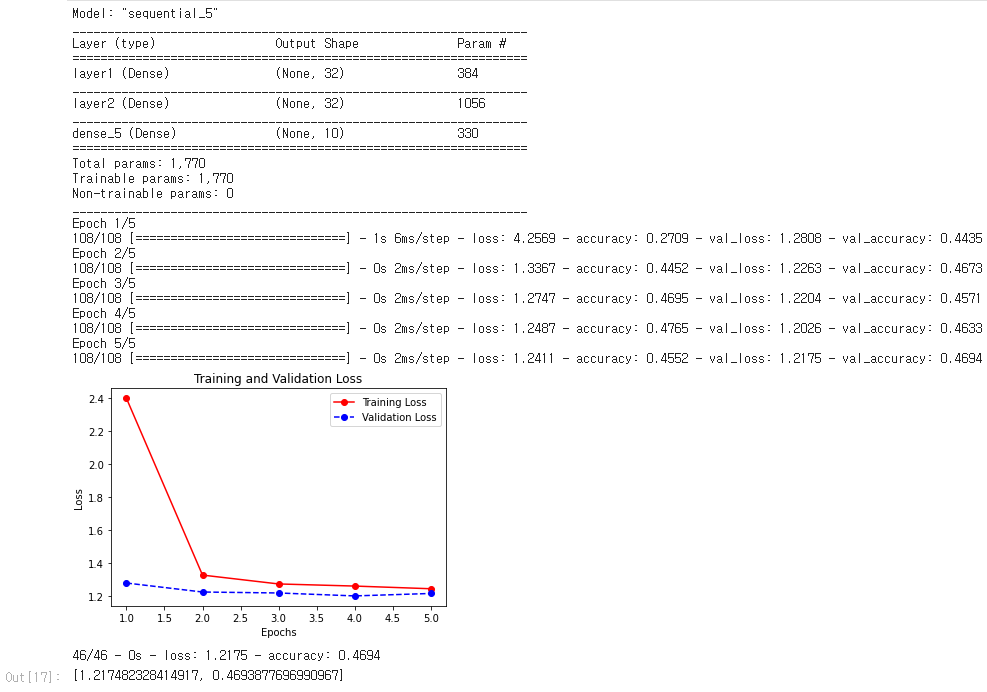
  
- Node가 32개인 Hidden Layer 한 개가 있는 red\_wine 분류 모델을 설계 및 학습하고, 시각화 하여 출력하여 대조군을 만듭니다.

- red\_wine 대조군의 정확도: 0.4521

**3.2 각 모델의 성능을 향상시킬 수 있는 방법 적용 (white\_wine 모델로 테스트 진행)**

**1) 실험군-1: Hidden Layer 개수 증가(2개)**





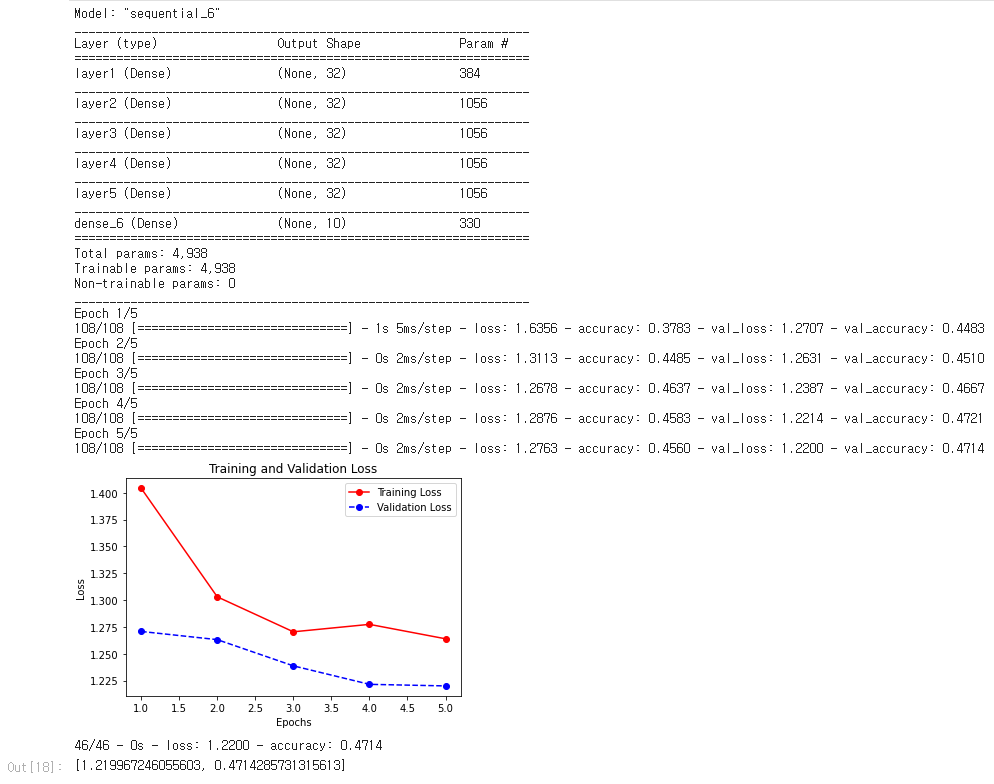
- 32개 node를 가진 Hidden Layer을 1개에서 2개로 증가시켜 정확도의 변화를 기록합니다.

- Hidden Layer이 2개일 때의 정확도: 0.4694

- 대조군과의 정확도 차이: 0.0878

**2) 실험군-2: Hidden Layer 개수 증가(5개)**



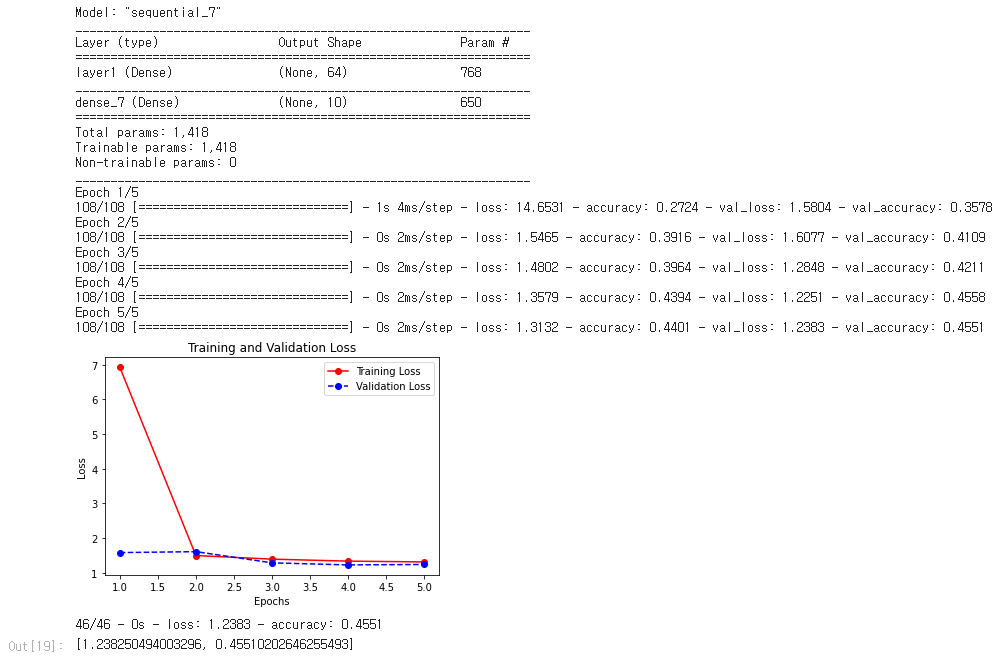


- 32개 node를 가진 Hidden Layer을 1개에서 5개로 증가시켜 정확도의 변화를 기록합니다.

- Hidden Layer이 5개일 때의 정확도: 0.4714

- 대조군과의 정확도 차이: 0.0898

**3) 실험군-3: Node 개수를 32개에서 64개로 변경**



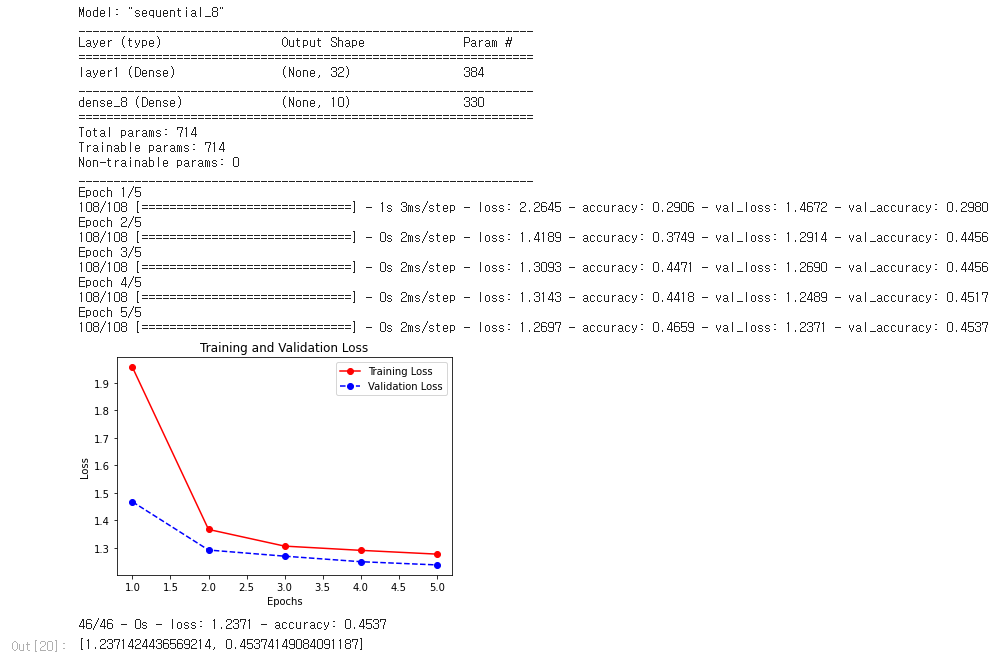
- Hidden Layer의 node 개수를 32개에서 64개로 증가시켜 정확도의 변화를 기록합니다.

- Hidden Layer의 node 개수가 이 64개일 때의 정확도: 0.4551

- 대조군과의 정확도 차이: 0.0735

**4) 실험군-4: Activation 활성함수 변경**





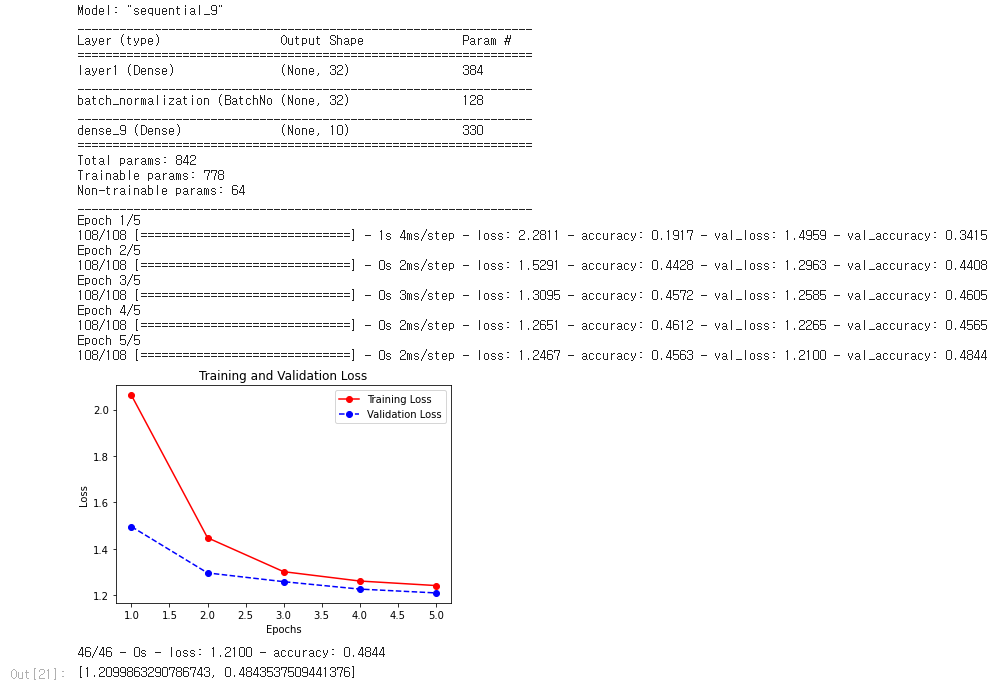
- Activation 활성함수를 ‘ReLU’에서 ‘Sigmoid’로 변경하여 정확도의 변화를 기록합니다.

- Activation 활성함수가 ‘Sigmoid’ 일 때의 정확도: 0.4537

- 대조군과의 정확도 차이: 0.0721

**5) 실험군-5: 정규화(Normalization 사용)**





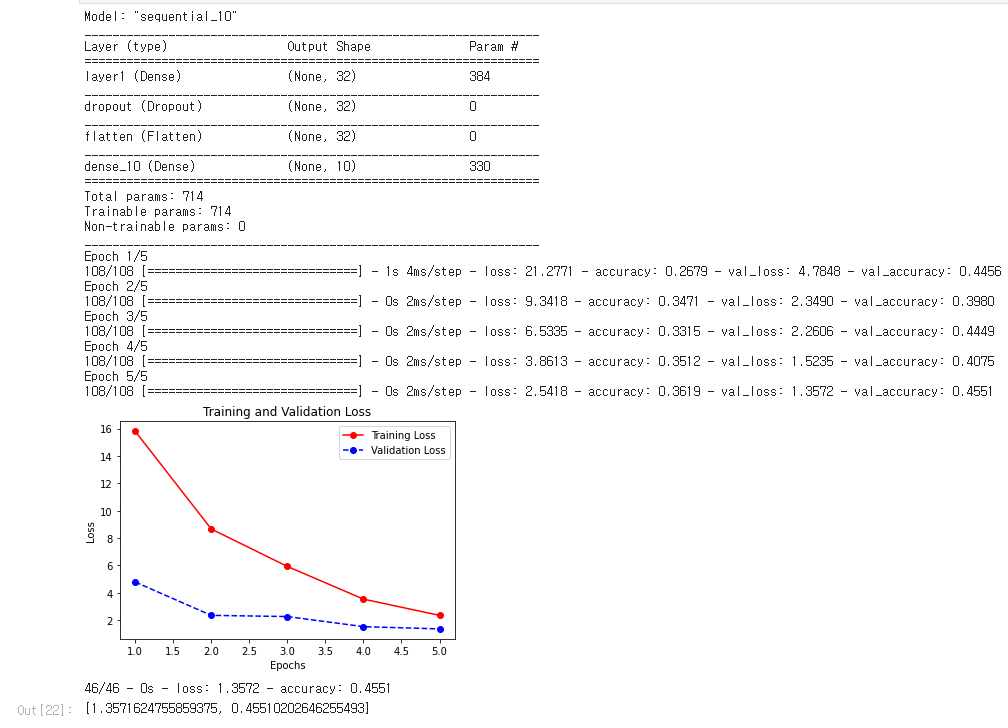
- 데이터 정규화를하여 정확도의 변화를 기록합니다.

- BatchNormalization()을 추가했을 때의 정확도: 0.4844

- 대조군과의 정확도 차이: 0.1068

**6) 실험군-6: Dropout(), Flatten() 사용**



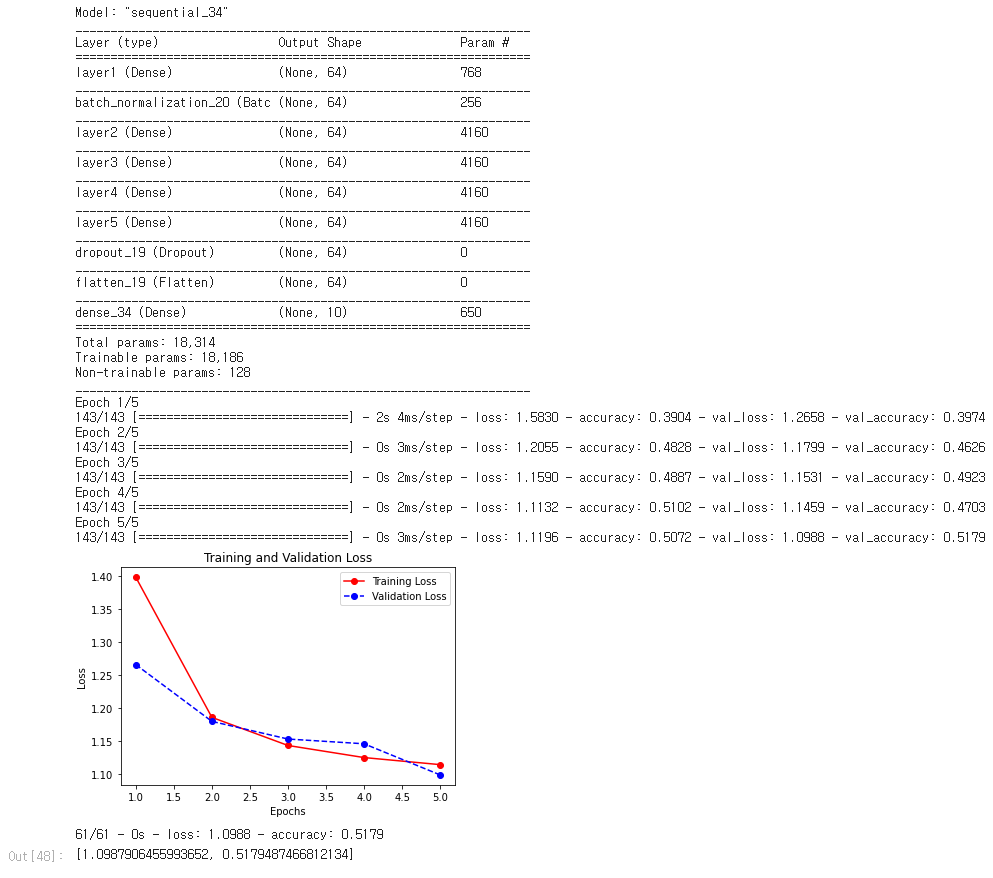


- Hidden Layer의 node를 변화시킨 후 정확도의 변화를 기록합니다.

- 20%의 Dropout 진행 후의 정확도: 0.4551

- 대조군과의 정확도 차이: 0.0735

**7) 위의 변경 사항들을 사용하여 white\_wine 최종 개선 모델 설계 및 학습**

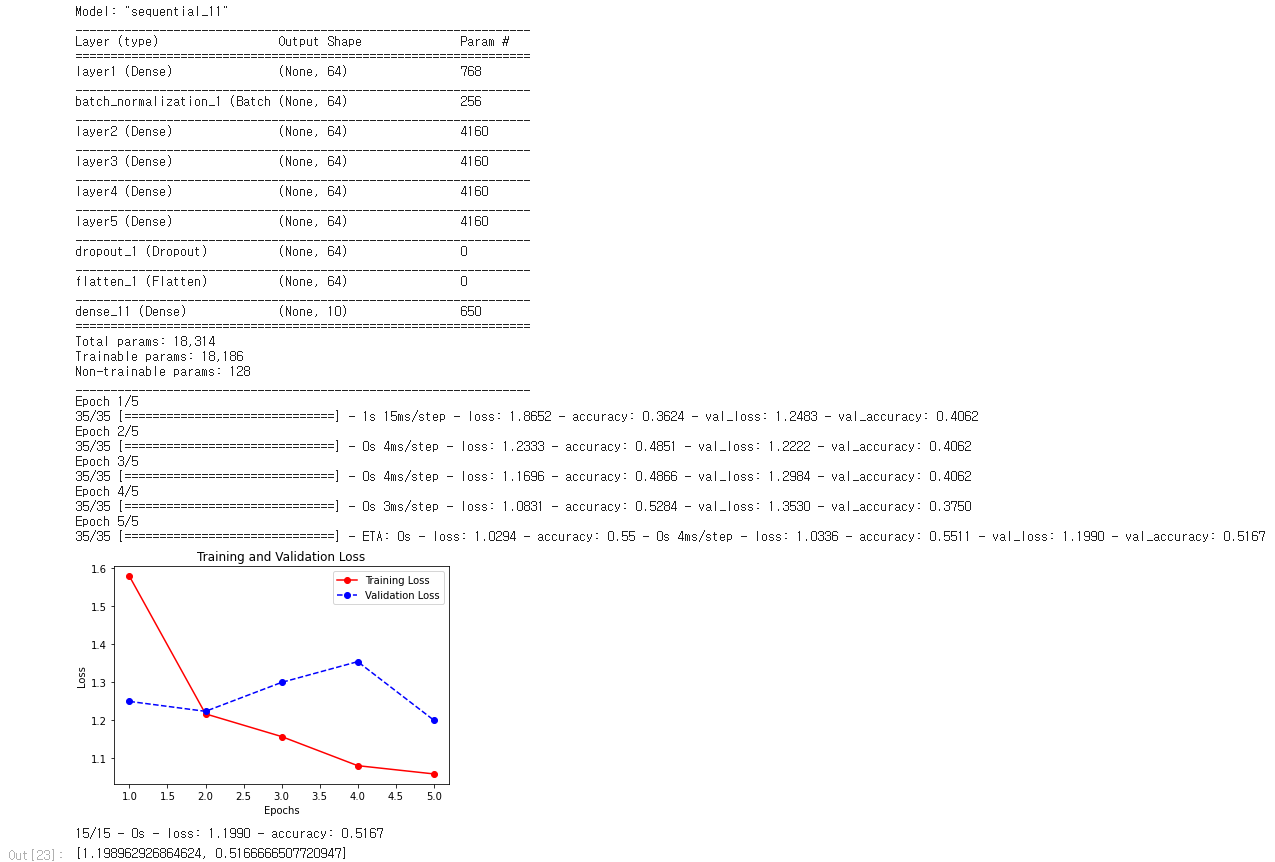


- 위 6가지의 변경사항을 적절히 추가한 후 정확도의 변화를 기록합니다.

- white\_wine 최종 개선 모델의 정확도: 0.5179

- 대조군과의 정확도 차이: 0.1363

**8) 위의 변경 사항들을 사용하여 red\_wine 최종 개선 모델 설계 및 학습**



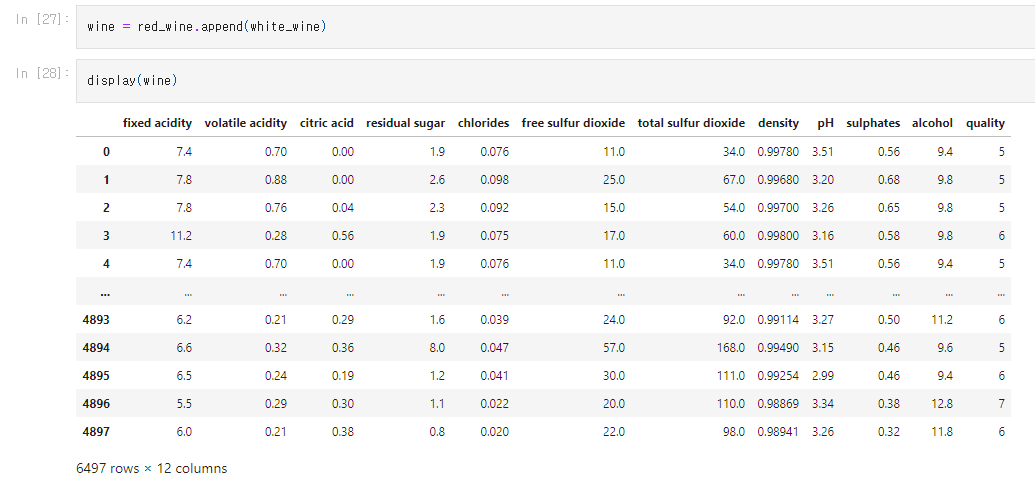
- white\_wine과 마찬가지로 6가지의 변경사항을 적절히 추가한 후 정확도의 변화를 기록합니다.

- red\_wine 최종 개선 모델의 정확도: 0.5167

- 대조군과의 정확도 차이: 0.0646

**3.3 화이트 와인과 레드 와인을 하나의 모델만 사용하여 분류**

**1) white\_wine과 red\_wine을 합쳐서(append) wine DataFrame**

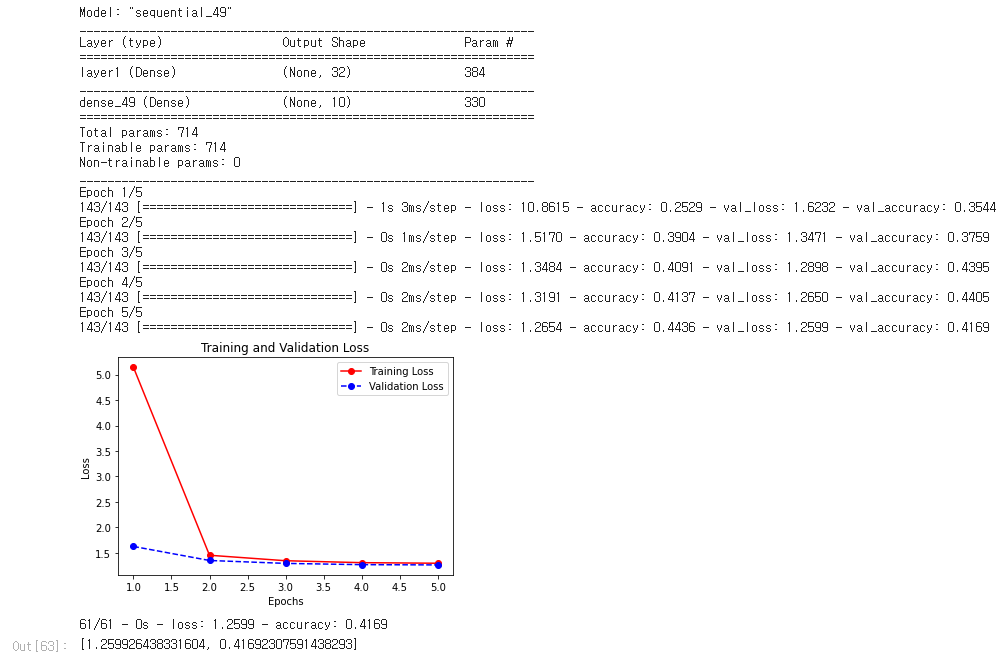


- red\_wine과 white\_wine을 합쳐서 wine Dataset을 만듭니다.

- wine Dataset을 display합니다.

**2) wine 분류 모델 설계 및 학습**

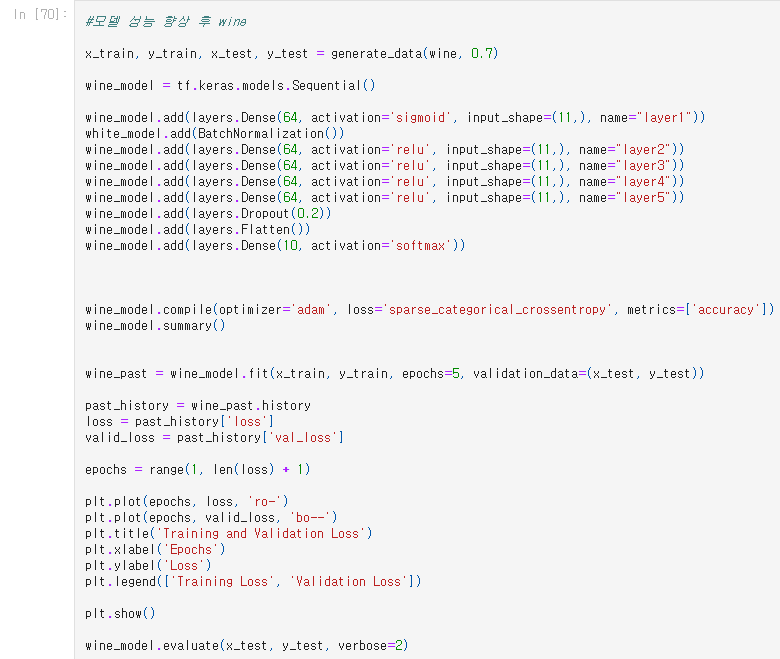


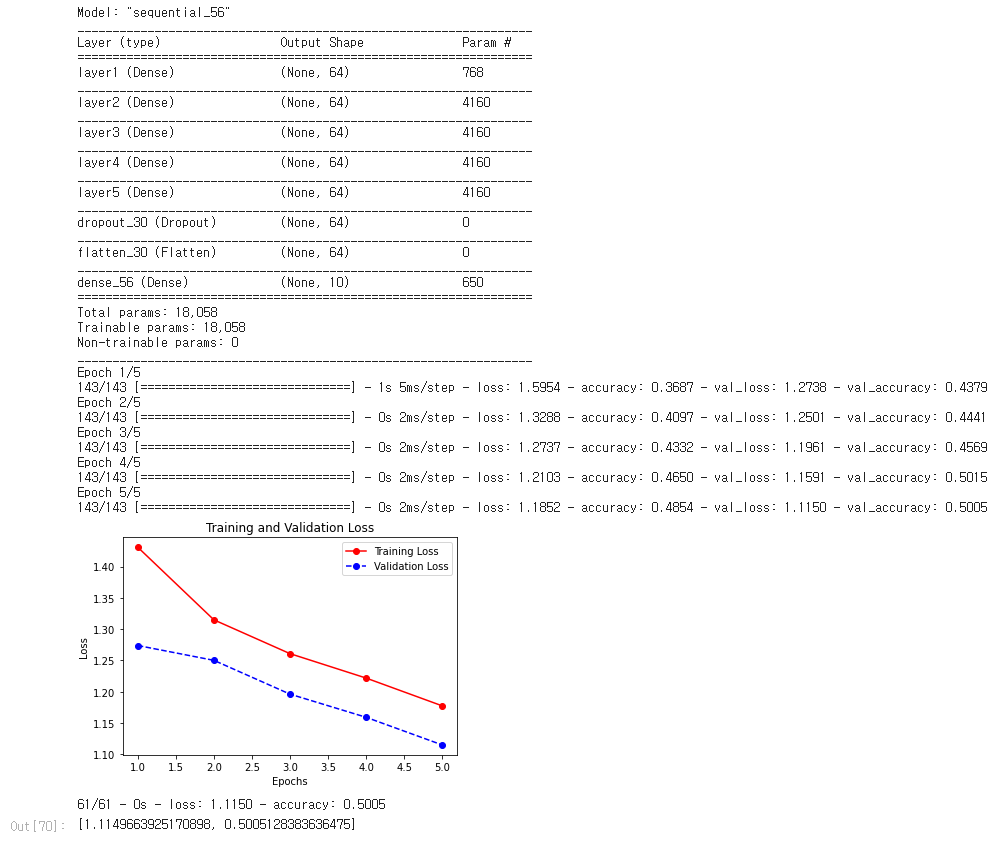


- Node가 32개인 Hidden Layer 한 개가 있는 wine 분류 모델을 설계 및 학습하고, 시각화하여 출력하여 대조군을 만듭니다.

**-** wine 대조군의 정확도: 0.4169

**3) 개선된 분류 모델 설계 및 학습**





- white\_wine과 마찬가지로 6가지의 변경사항을 적절히 추가한 후 정확도의 변화를 기록합니다.

- 최종 개선 모델의 정확도: 0.5005

- 대조군과의 정확도 차이: 0.0836

**Wine Classifier 과제를 하며 느낀 점**

이전의 과제들은 이미 있는 예시를 따라치고, 모르는 점은 구글에서 검색하면서 JupyterNotebook과 Python에 익숙해지고, Machine Learning의 겉핥기만 하는 느낌이었다. 기계적으로 따라하며 단순하게 어떻게 CSV 파일을 읽고, 시각화하는지만 알 수 있었다.

반면, 이번 과제를 하면서 Machine Learning의 좀더 본질적인 목적을 느낄 수 있었고, 나의 지식이 얼마나 얕은지 새삼 느낄 수 있었다. 첫 번째 관문은 generate\_data() 함수를 만들 때였다. 함수의 parameter을 받는거야 CPP와 비슷해서 어렵지 않았지만 train\_test\_split를 찾아보며, 어떻게 parameter들을 대입하고 변수에 저장하는지 알아내기 위해 많은 시간을 투자하였다. 또한 모델들의 성능을 높이기 위해서 다양한 가능성을 열어두고 테스트를 진행하였다. 각 실험군 마다 100번 이상 compile을 하면서, 왜 이렇게 성능이 개선이 안될까, 내가 잘 못 생각한 부분이 있었다 고민을 했었다. 성능 개선을 이루어냈지만, 생각하던 것보다 개선이 많이 되진 않아서 아쉬움이 따른다.

이 보고서를 쓰면서 내 코드에 심각한 오류가 있다는 것을 발견하여 코드를 다시 손보면서, 보고서의 중요성을 깨달았다. 단순히 과제로 제출하거나, 조교님께서 채점하기 쉽게 하라는 것이 아니라 보고서로 정리를 하고 스스로 평가를 하면서 나의 실수를 발견할 수 있다는 것을 알아냈다.

이번에 내가 가장 어려움을 겪은 것은 Python의 문법과 Tensorflow, Keras의 함수들인 것 같다. Python이야 쉬운 언어라지만, 딱히 연습해보거나 배워보지 않은 나로써는 처음에 어려움을 겪었다. Tensorflow와 Keras의 문법은 구글에 검색함으로써 해결해나가야 할 듯 싶다.