Titanic: Machine Learning from Disaster

Goal: Information about passengers is given. Information contains name, sex, Pclass, Fare, cabin, and so on... With this given data, making prediction of survival is goal of this practice.

**First Trial**

-Keras를 통해 신경망을 학습

-matplotlib, seaborn을 통해서 전처리 과정을 일부 시각화

-pandas, numpy를 통해서 데이터 전처리

**데이터 전처리 과정**

-우선 탑승자의 생존 여부와 중요하지 않다고 생각되는 자료는 삭제하였다.

Name, PassengerId(primary key 역할을 하는 column), Ticket(Ticket number)를 삭제하였다.

-첫 시도이기 때문에 처리하기 까다로운(?) 데이터는 삭제하였다.

Cabin(Cabin number), Embarked(Port of Embarkation)은 non-numeric and categorical data 이다. Non-numeric data는 수치화해서 전처리를 진행해야 하는데, 라이브러리 숙달이 부 족하고, 정신이 없어서 우선은 넘어갔다.

-데이터 중간에 비어있는 부분이 있는데, 이 부분을 fillna method를 통해서 해당 column의 평균으로 채웠다. (Age, Fare) 이 방법이 빈 칸을 채우기 위한 좋은 방법이라는 확신은 없다. 다만, 빈 칸을 0과 같은 숫자로 채운다면 0과 빈칸이 가지고 있는 정보가 0 주변에 있는 숫자에 영향을 줄 수 있다고 판단해서 평균을 넣어보았다.

-성별은 0과 1로 바꾸었다.

-Age와 Fare의 일의 자리 숫자를 버렸다. 기존의 숫자를 넣는 것과 해당 방법 중 어느 것이 더 효과적인지는 모르겠다.

**훈련 과정**

-Training set을 800개의 샘플로 정하였고, 나머지를 development set으로 설정하였다.

-Test set file을 test set으로 설정하였다.

def reluModel():

model=models.Sequential()

model.add(layers.Dense(64, activation='relu', input\_shape=(6,)))

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

hist=model.fit(x\_train, y\_train, epochs=100, validation\_data=(x\_val, y\_val))

Epoch 91/100

800/800 [==============================] - 0s 84us/step - loss: 0.4085 - acc: 0.8313 - val\_loss: 0.3576 - val\_acc: 0.8571

Epoch 92/100

800/800 [==============================] - 0s 86us/step - loss: 0.4228 - acc: 0.8113 - val\_loss: 0.3997 - val\_acc: 0.8462

Epoch 93/100

800/800 [==============================] - 0s 86us/step - loss: 0.4320 - acc: 0.8175 - val\_loss: 0.3643 - val\_acc: 0.8462

Epoch 94/100

800/800 [==============================] - 0s 83us/step - loss: 0.4122 - acc: 0.8237 - val\_loss: 0.3772 - val\_acc: 0.8462

Epoch 95/100

800/800 [==============================] - 0s 81us/step - loss: 0.4097 - acc: 0.8250 - val\_loss: 0.3578 - val\_acc: 0.8571

Epoch 96/100

800/800 [==============================] - 0s 80us/step - loss: 0.4048 - acc: 0.8275 - val\_loss: 0.3816 - val\_acc: 0.8571

Epoch 97/100

800/800 [==============================] - 0s 81us/step - loss: 0.4466 - acc: 0.8313 - val\_loss: 0.3787 - val\_acc: 0.8571

Epoch 98/100

800/800 [==============================] - 0s 81us/step - loss: 0.4178 - acc: 0.8100 - val\_loss: 0.3639 - val\_acc: 0.8352

Epoch 99/100

800/800 [==============================] - 0s 82us/step - loss: 0.4193 - acc: 0.8200 - val\_loss: 0.3624 - val\_acc: 0.8462

Epoch 100/100

800/800 [==============================] - 0s 87us/step - loss: 0.4083 - acc: 0.8250 - val\_loss: 0.3791 - val\_acc: 0.8571

test\_score, test\_score=model.evaluate(x\_test, y\_test)

418/418 [==============================] - 0s 39us/step

0.9401913875598086

-아쉬운 점

1. 다른 사람들이 해결한 것을 보면 각 column 별로 Survival(goal)과의 상관관계를 확인한 후에 가공을 거친다. 다음에 할 때는 이 부분을 추가하면 좋을 듯.

2. Non-numeric data에 대한 가공 방법 구상

3. 훈련 과정의 진척도를 히스토그램으로 확인해보면 좋을 듯 하다.

4. 여러 모델을 사용하여 비교하면 좋을 것 같다.

**Second Trial**

-다양한 신경망 사용

-