

전자정보통신공학과 인공지능 1 분반

Term project: 타이타닉 호 생존자 예측하기



17010708 이슬기

2021-6-8


1. 문제 정의

Kaggle Titanic은 주어진 데이터를 학습시켜 생존자 예측 모델을 만드는 것이다.
<https://www.kaggle.com/c/titanic>에서 train 데이터와 test 데이터를 다운받을 수 있다.

2. Competition 결과

2944	YeeSeulki		0.79425	10	1h
Your Best Entry 					
Your submission scored 0.78229, which is not an improvement of your best score. Keep trying!					

(최고점만 기록되는 것 같습니다. 마지막으로 제출한 파일에 맞는 점수는 아래입니다.)

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
submission_17010708.csv	just now	1 seconds	0 seconds	0.78229
Complete				
Jump to your position on the leaderboard 				

(이후 보고서에서 서술하는 코드는 위의 0.78229점에 해당하는 코드입니다.)

3. 데이터 가공, Feature 선택, 적용모델 설명

(1) 데이터 확인하기

데이터를 분석하기 위해 필요한 패키지들을 임포트하고 데이터를 불러온다.

```
In [434]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

In [435]: train=pd.read_csv('C:/Users/doobe/OneDrive/바탕 화면/Titanic/train.csv')
test=pd.read_csv('C:/Users/doobe/OneDrive/바탕 화면/Titanic/test.csv')
```

데이터를 학습시키기 전에 전처리를 진행하여야 하는데, 이를 위해서 데이터의 형태와 크기, 결측치들을 파악해보자.

```
In [5]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId      891 non-null    int64
1   Survived         891 non-null    int64
2   Pclass          891 non-null    int64
3   Name             891 non-null    object
4   Sex              891 non-null    object
5   Age              714 non-null    float64
6   SibSp            891 non-null    int64
7   Parch            891 non-null    int64
8   Ticket           891 non-null    object
9   Fare             891 non-null    float64
10  Cabin            204 non-null    object
11  Embarked         889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
In [6]: test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId      418 non-null    int64
1   Pclass           418 non-null    int64
2   Name             418 non-null    object
3   Sex              418 non-null    object
4   Age              332 non-null    float64
5   SibSp            418 non-null    int64
6   Parch            418 non-null    int64
7   Ticket           418 non-null    object
8   Fare             417 non-null    float64
9   Cabin            91 non-null     object
10  Embarked         418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

결측치는 아래와 같이 확인할 수 있다.

```
In [7]: train.isnull().sum()
```

```
Out[7]: PassengerId      0
Survived                0
Pclass                  0
Name                    0
Sex                     0
Age                     177
SibSp                   0
Parch                   0
Ticket                  0
Fare                    0
Cabin                   687
Embarked                2
dtype: int64
```

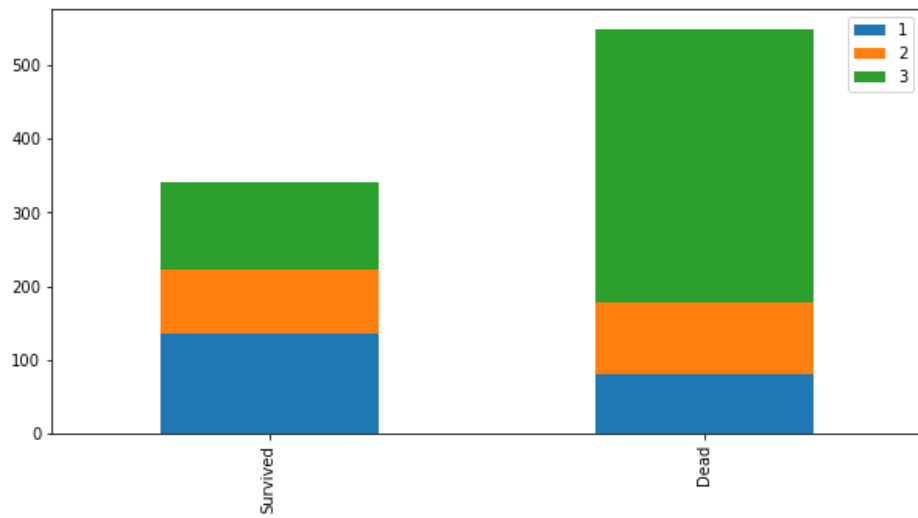
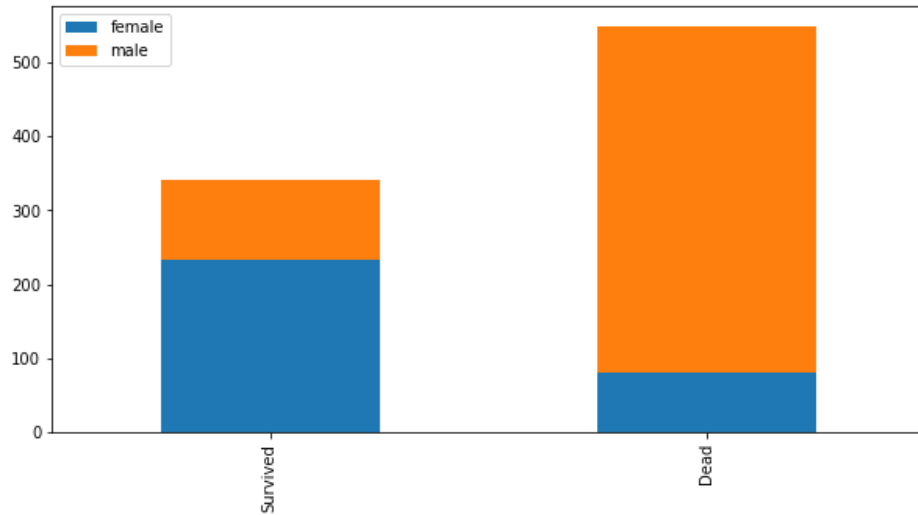
```
In [8]: test.isnull().sum()
```

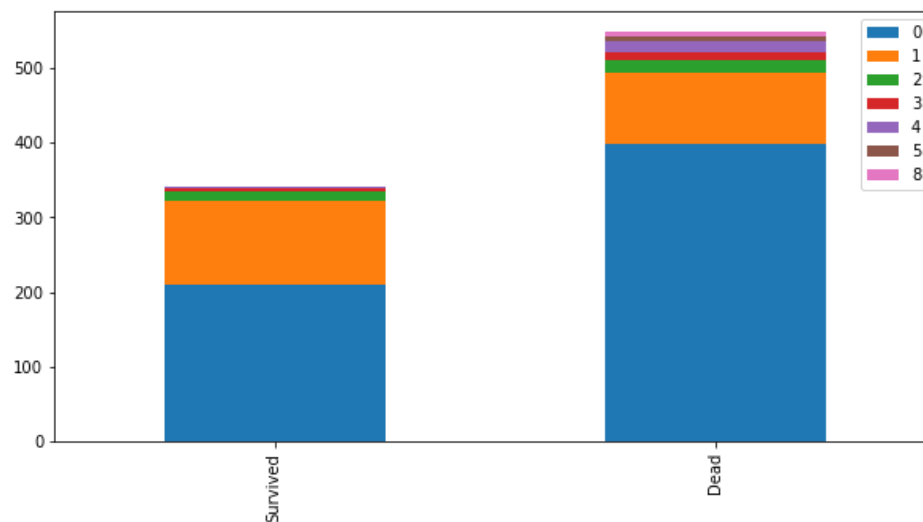
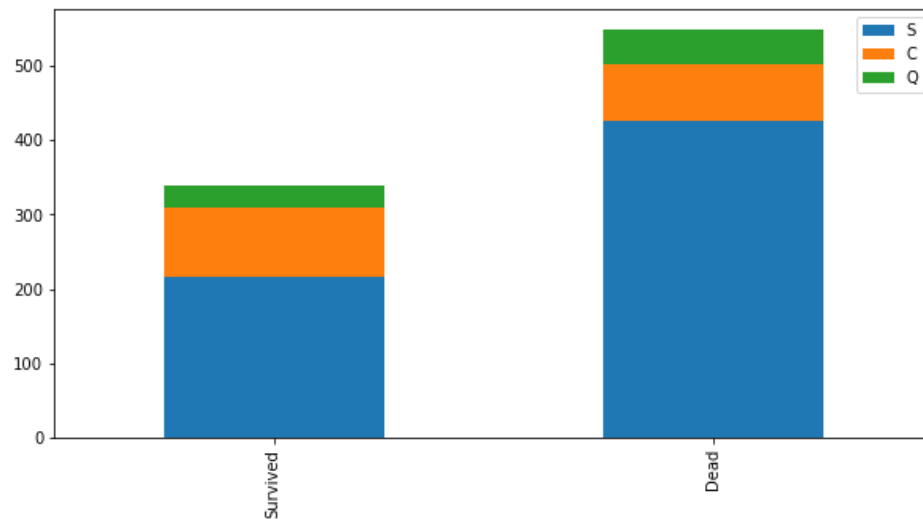
```
Out[8]: PassengerId      0
Pclass                  0
Name                    0
Sex                     0
Age                     86
SibSp                   0
Parch                   0
Ticket                  0
Fare                    1
Cabin                   327
Embarked                0
dtype: int64
```

데이터 가공작업을 거치며 결측치들을 적당한 값들로 메꿔줘야 할 것이다.

(2) 데이터를 그래프로 확인하기

각 column들을 시각화하여 대략적인 분포를 파악하고 생존여부와 상관관계를 유추한다. 막대그래프를 그려주는 함수를 하나 정의하고 이산형 데이터에 해당하는 feature들의 그래프를 그려본다. 순서대로 Sex, Pclass, Embarked, SibSp에 해당한다.





데이터를 시각화해서 본 결과 남성보단 여성의 생존율이 높으며, 1등급 승객, 가족이 있는 승객들의 생존율이 높았다. 탑승지역에선 비율상으로 S 승객들이 많이 사망했다.

(3) 데이터 전처리

(3.1) Name Feature

일단 test 데이터와 train 데이터 모두 전처리과정이 진행돼야 하므로 둘을 `data = [train, test]`로 묶어줬다.

이름 자체가 생존율과 유의미한 관계가 있다고 보기는 힘들지만, train 데이터를 살펴보면 이름은 Mr, Mrs, Miss등으로 혼인여부와 성별을 포함한다. 위에서 확인했듯 성별은 생존율과 관계가 있기 때문에 해당 부분만 추출한다.

```
# 혼인여부, 성별 파악 가능
for dataset in data:
    dataset['Title'] = dataset['Name'].str.extract('([Ww]+)W.', expand=False)
```

Name 열에서 정규표현식 ([Ww]+)W.에 해당하는 부분을 추출하여 Title이라는 series를 만들었다.

```
pd.crosstab(train['Title'], train['Sex'])
```

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

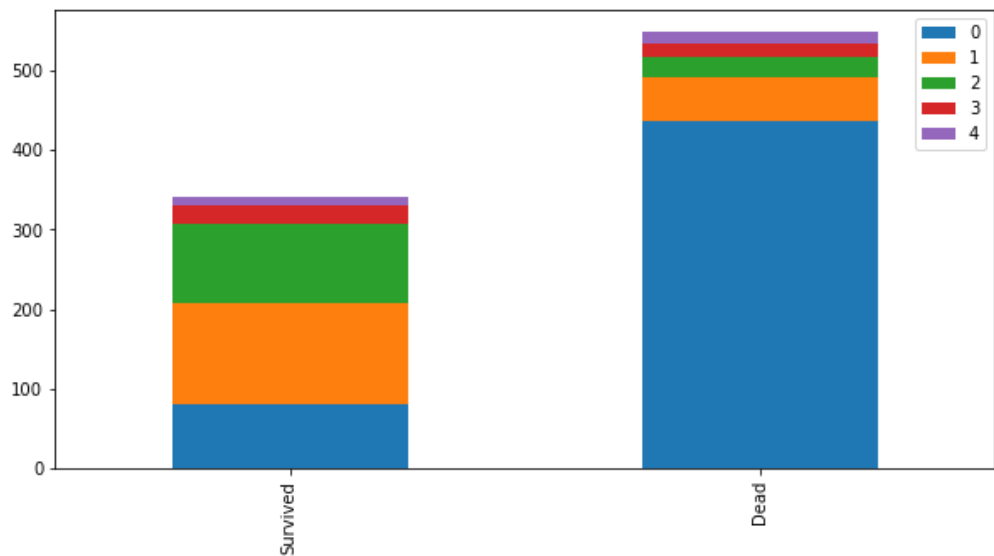
위의 표를 확인하면 Mr, Mrs, Miss, Master가 대부분을 차지하고 있음을 알 수 있다. 따라서 이 네 개를 제외한 것들은 하나로 취급하기로 한다. 컴퓨터가 조금 더 알아보기 쉽게 숫자에 매핑한다.

```
for dataset in data:
    dataset['Title'] = dataset['Title'].apply(lambda x:
        0 if x=="Mr"
        else 1 if x=="Miss"
        else 2 if x=="Mrs"
        else 3 if x=="Master"
        else 4)

train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2 3101282	7.9250	NaN	S	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0

제일 우측에 Title에 매핑이 된 것을 확인할 수 있다. 이를 막대그래프로 수치를 확인해 보자.



0에 해당하는 Mr가 사망률이 높은 것을 확인할 수 있다. 이를 통해서 또 남자의 사망률이 높다는 사실을 알 수 있다.

(3.2) Sex Feature

성별은 숫자 0과 1에 매핑시켜주는 과정만을 거치면 된다.

```
sex_mapping = {"male":0, "female":1}
for dataset in data:
    dataset['Sex'] = dataset['Sex'].map(sex_mapping)

train.head()
```

train.head()로 성별이 0과 1로 잘 매핑된 것을 확인할 수 있다.

(3.3) Embarked Feature

```
train['Embarked'].value_counts()

S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

S지역에서 탑승한 사람들이 압도적으로 많은 것을 확인할 수 있다. 승객 대부분이 S 지역에서 탑승했으므로, 결측치는 S로 채워준다.

```
for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
```

위 예제와 마찬가지로 숫자와 매핑 시켜준다.

```
embarked_mapping={'S':0, 'C':1, 'Q':2}
for dataset in data:
    dataset['Embarked']=dataset['Embarked'].map(embarked_mapping)

train.head()
```

(3.4) Age Feature

나이 정보에도 마찬가지로 결측치가 존재하는데 위의 Name Feature에서 구한 Title에서 어느 정도의 나이대를 유추할 수 있다. (혼인 여부등을 알 수 있으므로) 따라서 각 Title에 해당하는 그룹의 중간 값으로 결측치를 채운다.

```
for dataset in data:
    dataset['Age'].fillna(dataset.groupby('Title')['Age'].transform('median'),inplace=True)
```

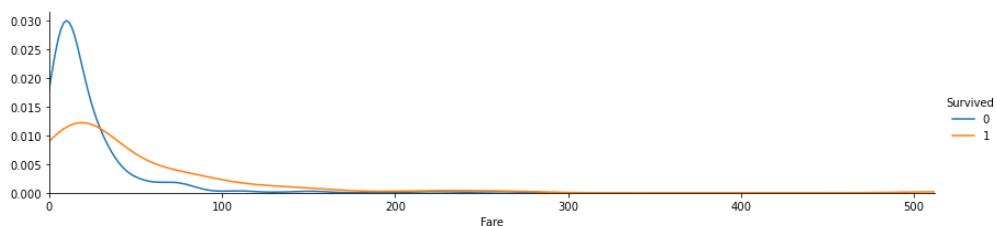
나이대별로 그룹화를 하기위해 흔히 사용하는 기준인 "청소년, 청년, 중년, 장년, 노년" 5개의 집단으로 나눈다. 그 후 Ageband에 저장한다. 길이를 기준으로 데이터를 나누는 pandas의 cut method를 사용하였다.

(3.5) Fare Feature

탑승요금은 높은 등급에 탑승한 승객일수록 높다. Fare역시 결측치가 존재하는데, 이 역시 각 등급 등급별의 중간 값으로 채운다.

```
for dataset in data:
    dataset["Fare"].fillna(dataset.groupby("Pclass")["Fare"].transform("median"),inplace=True)
```

Fare역시 나이와 같이 그룹화를 진행해준다. 먼저 그래프를 확인하자면



승객별로 탑승요금의 편차가 크고 우측에는 거의 존재하지 않는 것처럼 보인다. 나이와는 다르게 길이가 아닌 개수를 기준으로 나누는 pandas의 qcut method를 이용하고 Fareband에 저장해준다.

(3.6) SibSp & Parch Feature (Family)

두 feature가 의미하는 바는 결국 가족의 수이다. 이 두개를 Family로 묶어준다. 또한 동승자의 여부를 판단하는 IsAlone이라는 변수를 생성한다. 가족수가 1보다 많으면 동승자가 있으므로 IsAlone에는 0을 저장해준다.

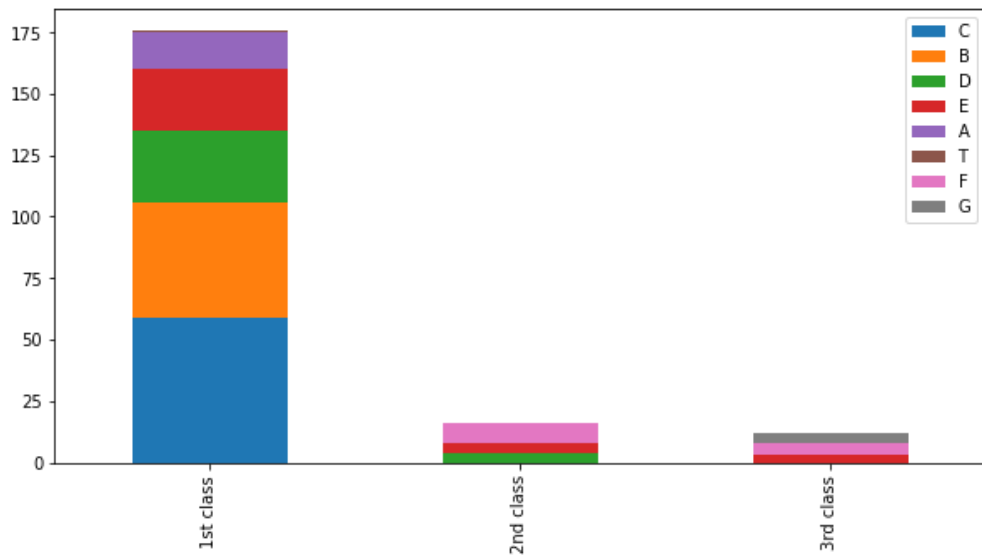

```
for dataset in data:
    dataset['Family']=dataset['Parch']+dataset['SibSp']+1
    dataset['IsAlone']=1
    dataset.loc[dataset['Family']>1, 'IsAlone']=0

train.head()
```

이후 전처리가 잘 됐는지 확인해보려면 train.head()를 이용하면 된다.

(3.7) Cabin Feature

Cabin은 객실정보를 담고 있다. Train.Cabin.value_counts()로 데이터가 어떤 형태를 띄고 있는지 확인하면 C23, C25와 같이 알파벳과 숫자로 이루어져 있음을 알 수 있다. 데이터를 시각화하여 좀 더 자세히 알아보자.



객실의 정보와 Pclass(등급)에는 밀접한 관계가 있다고 생각했다. 등급이 높을수록 좋은 객실을 배정받았을 것이기 때문이다. 따라서 위와 같이 Pclass와 cabin을 연관 지어서 데이터를 시각화 하였다. 1등급에는 ABCDET, 2등급에는 DEF, 3등급은 EFG로 이루어져 있다. 다음과 같이 매핑시켜주었다.

```
cabin_mapping = {'A':0,
                  'B':0.4,
                  'C':0.8,
                  'D':1.2,
                  'E':1.6,
                  'F':2,
                  'G':2.4,
                  'T': 2.8}

for dataset in data:
    dataset['Cabin'] = dataset['Cabin'].map(cabin_mapping)
```

또한 Cabin의 결측치는 위에서 말했듯 Pclass(등급)와 밀접한 관계가 있으므로 각 Pclass(등급)별 cabin의 중간값을 넣어주도록 한다.

```
train['Cabin'].fillna(train.groupby('Pclass')['Cabin'].transform('median'),
                      inplace=True)
test['Cabin'].fillna(test.groupby('Pclass')['Cabin'].transform('median'),
                     inplace=True)
```

(4) 기타 데이터 정리

전처리가 끝났거나 훈련에 이용하지 않을 부분들을 삭제해준다.

```
# 전처리가 끝났거나 훈련에 이용하지 않을 column은 삭제
drop_column = ['Name', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare']

for dataset in data:
    dataset = dataset.drop(drop_column, axis=1, inplace=True)
```

	PassengerId	Survived	Pclass	Sex	Cabin	Embarked	Title	Ageband	Fareband	Family	IsAlone
0	1	0	3	0	2.0	0	0	1	0	2	0
1	2	1	1	1	0.8	1	2	2	3	2	0
2	3	1	3	1	2.0	0	1	1	1	1	1
3	4	1	1	1	0.8	0	2	2	3	2	0
4	5	0	3	0	2.0	0	0	2	1	1	1

train.head()로 데이터가 한결 더 간결해진 것을 알 수 있다.

(5) Modeling & Testing

이제 모든 정리가 끝난 파일로 모델링과 테스트를 진행해보자. 모델링에 필요한 것들 것 임포트 해준다.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow import keras
import tensorflow as tf

import numpy
```

```
numpy.random.seed(3)
tf.random.set_seed(3)

drop_column2 = ['PassengerId', 'Survived']
train_data = train.drop(drop_column2, axis=1)
test_data = test.drop("PassengerId", axis=1)
target = train['Survived']
```

PassengerId는 승객들의 번호에 불과하므로 학습에서 빼고, Survived 역시 결과값에 해당하기 때문에 학습에서 제외시킨다. Survived는 target에 저장한다.

```

model=Sequential()

model.add(Dense(128, input_dim=9, activation='relu'))
model.add(BatchNormalization(-1))
model.add(Dropout(0.1))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))

model.add(Dropout(0.1))
opt=keras.optimizers.Adam(learning_rate=0.005)

model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

```

모델 층을 쌓는 것은 수업내용과 책을 참고하였다. 결과 값이 Survived에서 0과 1인 이진분류 데이터이므로 sigmoid 함수를 써줬고 loss function은 binary_crossentropy를 사용해주었다. 정확도를 확인하기 위해 metrics는 accuracy로 하였다.

```

model.fit(train_data, target, epochs=200)
print("\n Accuracy: %.4f" % (model.evaluate(train_data, target)[1]))

```

다음과 같이 모델을 fit하고 evaluate해서 정확도를 확인해 보면 다음과 같다.

Accuracy: 0.8485

이제 kaggle에 제출하기 위한 csv를 만들기 위한 작업만 하면 된다.

```

predict = model.predict_classes(test_data)
predict=np.array(predict).flatten().tolist()

submission = pd.DataFrame({
    'PassengerId' : test['PassengerId'],
    'Survived' : predict})

submission.to_csv('submission_17010708.csv', index=False)
submission.head(10)

```

이를 제출하여 Competition 결과를 확인하였다.

4. 소스코드 캡처

(위의 데이터구성 서술 시 Jupyter Notebook의 소스코드 캡처 본을 사용하였습니다.)

5. Discussion & Lesson

(1) 한계점

결측치들을 채울 때 해당 데이터들을 확인하여 조금 더 좋은 값으로 채울 수 있을 것이다. 특히 Cabin의 경우 결측치가 상당히 많고 단순히 승객의 등급과 관계가 있을

것이라 생각하여 대강 결측치를 채웠지만, 아마 배가 침몰할 때 갑판 쪽에 있던 객실의 경우는 생존율이 높았을 것이다. 하지만 이를 직접 확인할 수 없어서 등급과의 관계만 생각하여 결측치를 대강 채웠으므로 더 많은 정보가 있으면 정확도를 더 높이 측정할 수 있을 것이다.

(2) 시행착오

처음에 Cabin은 결측치가 너무 많아 제외시키고 학습을 시켰다. 하지만 이를 시행하고 Kaggle에 제출하였을 때 점수가 0.75~0.76을 왔다 갔다 하였다. 이에 Cabin도 포함시켜 학습시키기로 결정하였고, 0.78정도의 점수를 얻을 수 있었다. 위의 한계점에서 말했듯이 결측치들을 채울 때 조금 더 정밀한 작업이 있더라면 0.8도 넘지 않았을까 라는 생각이 든다.