Titanic 생존자 예측 과제 보고서

17010668 황창현

목차

- 문제정의
- EDA
- Feature engineering
- modeling
- 결과 분석
- Competition 결과
- 시도 했던 방법들

1. 문제정의

타이타닉 호 침몰 사건 당시의 사망자와 생존자를 구분하는 요인 분석을 통해, 승객들의 생존 여부를 예측

생존 여부 구분
(죽었을 것이다 0
살았을 것이다 1)

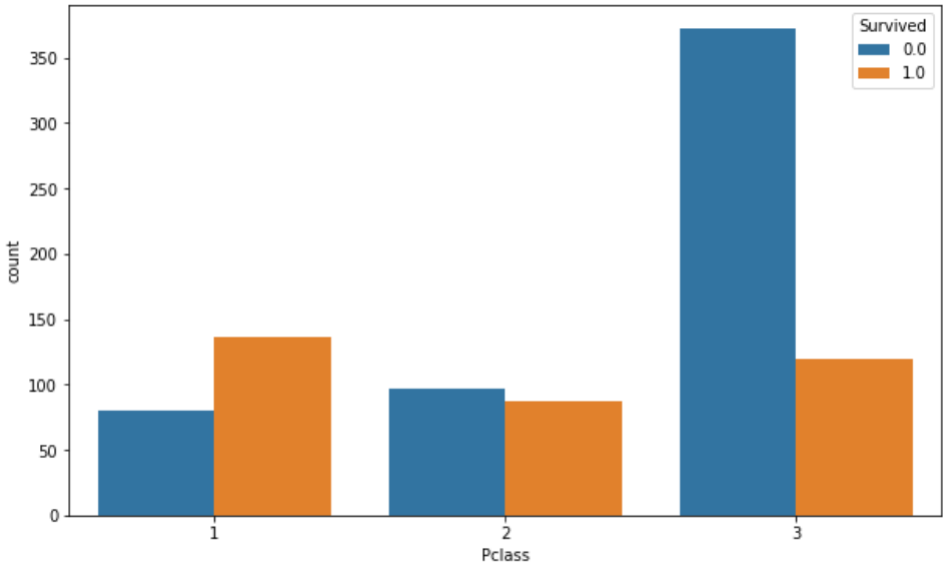
변수간의 상호작용
(죽었을 것이다 0
살았을 것이다 1)

변수상태
(결측값 등 해결)

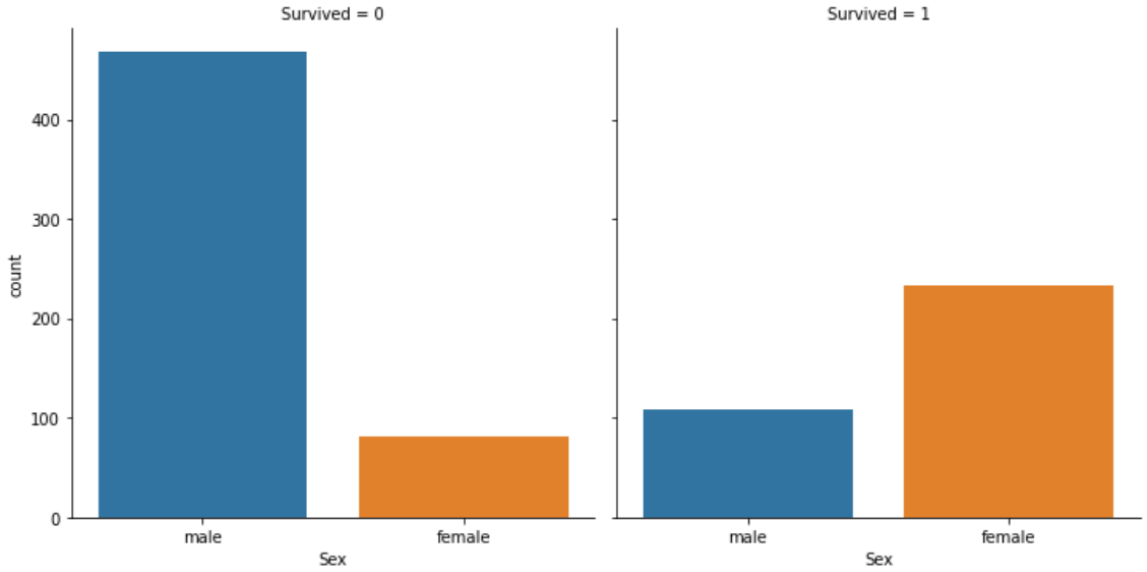
2-0. 데이터 설명

변수	정의	값
Survived	Survival	0 = No, 1 = Yes category
Pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd category, ordinal
Sex	Sex	category
Age	Age in years	numeric
SibSp	# of siblings / spouses aboard the Titanic	numeric
Parch	# of parents / children aboard the Titanic	numeric
Ticket	Ticket number	numeric, char
Fare	Passenger fare	numeric
Cabin	Cabin number	char
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton category

2-1. EDA

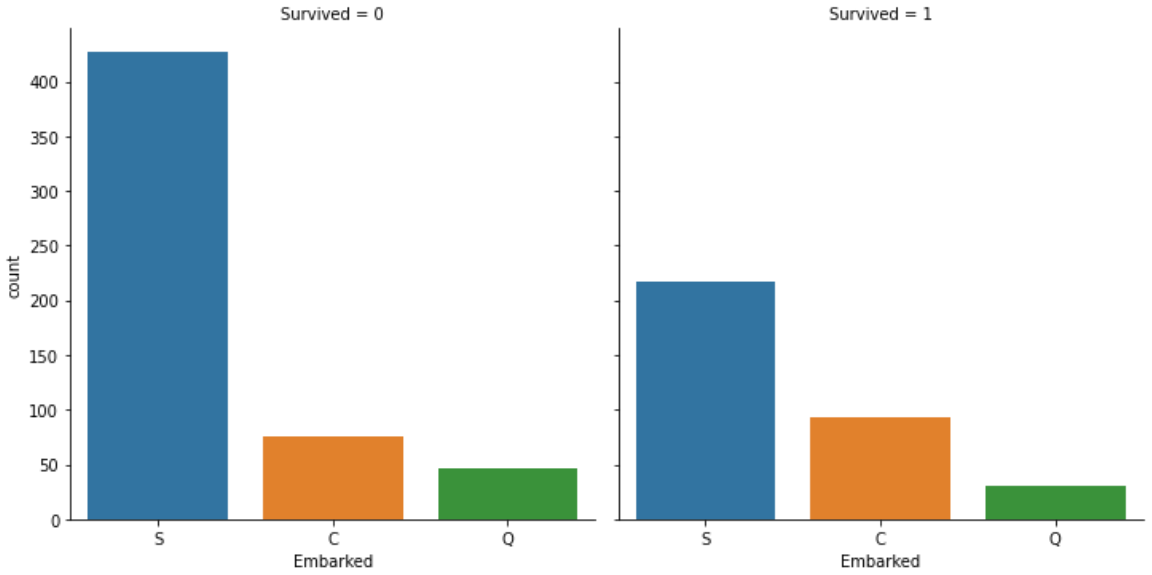


Pclass에 따라 생존수가 다름을 알 수 있다.



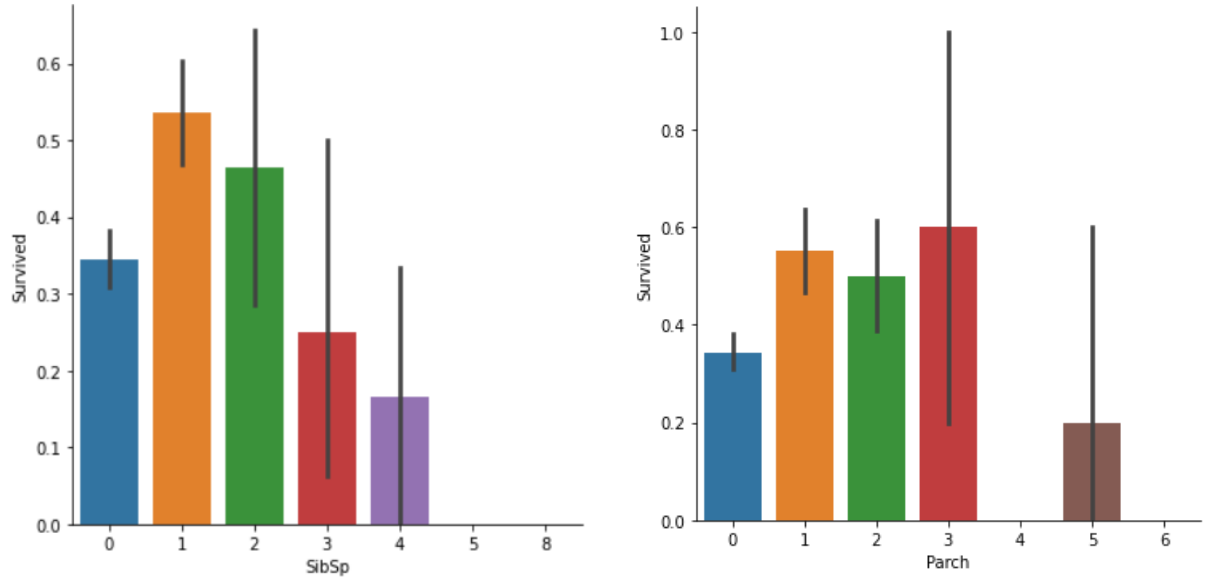
성별에 따라 생존수가 다름을 알 수 있다.

2-2. EDA



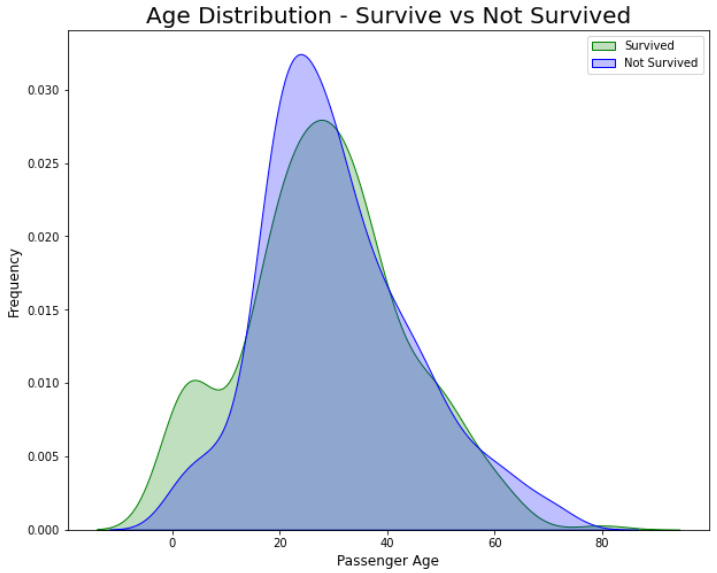
Embarked는 살아남은 사람과 죽은 사람의 비율차이가 크지 않다.

다만 S가 많다.

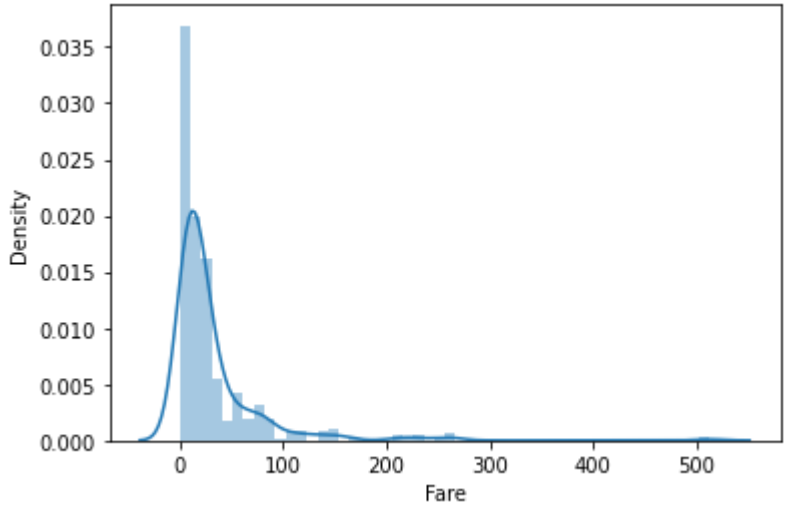


형제,배우자의 총 합이 1이나 2일때 많이 살았다. Parch도 비슷하다.

2-3. EDA



Age가 적으면 살아남을 확률이 높다.



Fare의 분포를 봤지만 큰 의미를 얻을 수는 없다.

2-4. 가정

생각해야 할 점

1. 성별에 따라 살아남은 정도가 다르다. -> 성별은 중요한 변수가 될 것이다.

2. 그렇다면 이름은? 이름 또한 성별을 가를수 있게 도와주는 변수가 될것이다.

3. 배우자와 형제(sibsp)의 수, 부모와 아이의수(parch)는 가족을 대표하는 변수로 상관계수가 0.41로 조금 높은편이라 합쳤다.

4. 그리고 특정 가족의 수 일때 (2~4명 즉 가족이 적을때 살아남을 확률이 높았다. 가족의 힘 그런거라고 생각했다.) 차이가 있으므로 따로 묶었다.

5. Embarked가 비어있는 2명의 사람은 구글링을 통해 Southampton에서 온것을 알았다. 그대로 채워준다.

6. Ticket은 종류가 너무 다양했지만 규칙은 영어와 숫자가 있는 티켓, 숫자만 있는 티켓이있었다. 그리고 티켓의 앞부분만 있다면 생각했다.
   실제로 그룹별 차이가 있었고 괜찮은 변수가 될 것이라고 가정했다.

7. Fare는 testset에 한 개만 비어있다. 이것을 train만 보고 유추해야하므로 row의 같은 요소들을 골라서 대체 했다.

8. Age의 결측값은 이름을 통해 추출한 변수로 그룹을 지어 평균으로 대체했다.

모델 빌드 할 때

1. 더미화

2. 정규화

3. Data Pipeline

Name
title(성) 추출

Age 결측값
title 별 평균

Sex
0,1 치환

fsize는
Parch+
SibSp+1

TicketHead
ticket의 앞

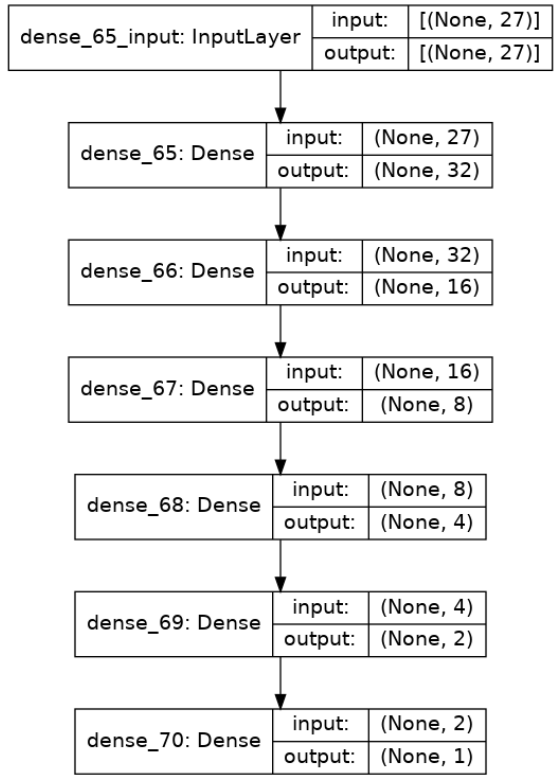
fsizeBin
2<=fsize<=4

Fare
결측값
6.2375

Embarked
결측값 S

더미화

표준화



3-0. Feature engineering

Name – title 추출

```
## Data.groupby(['Title'])['Age'].mean()
## 그러던 중 Master는 나이가 어린 사람에게 쓰는것 같다고 생각했다.
## 또, Data.groupby(['Title'])['Survived'].mean() 을 했을 때 약 55%나 살았다.

## 이름은 4개의 카테고리로 나누기로 했다.

Data['title']=Data['title'].replace('Mlle','Miss')
Data['title']=Data['title'].replace(['Mme','Lady','Ms'],'Mrs')
Data['title'].loc[(Data['title']!='Master')&(Data['title']!='Mr')&
                  (Data['title']!='Miss')&(Data['title']!='Mrs')]='Others'
```

title 로 Age 결측값 채우기

```
## Age 결측값 채우기
## title로 유추 할 수 있다고 가정.
def titleAge(title):
    if title=='Master':
        return 4
    elif title=='Miss':
        return 22
    elif title=='Mr':
        return 29
    elif title=='Mrs':
        return 35
    else:
        return 47

Data['Age']=Data.apply(lambda row:titleAge(row['title']) if pd.isnull(row['Ag
e']) else row['Age'],axis=1)
```

fsize 파생변수

```
## SipSp와 Parch를 가족을 대표하는 변수들을 하나로 합친다.
Data['fsize']=Data['SibSp']+Data['Parch']+1
```

fsizeBin 생성

```
## 가족이 살아남을 확률은 보통 2명, 그리고 4명 즉, 배우자 및 자녀의 수 합이 2부터 4일때 살아
남는 경우가 있다.
## 2부터 4만 따로 묶는다.
def fsizeBand(size):
    if size == 1:
        return 0
    elif size <= 4:
        return 1
    else:
        return 2

Data['fsizeBin']=Data['fsize'].map(fsizeBand)
```

Embarked 결측값 채우기 – 데이터 수집

```
## Embarked train에만 비어있는 이 값은 이름을 구글에 치면 Southampton 에서 탑승했다는 기
록이 있다.
Data['Embarked']=Data['Embarked'].fillna('S')
```

Sex 0,1 치환

```
## 성별 - 단순 치환
Data['Sex']=Data['Sex'].map({'male':0,'female':1})
```

3-1. Feature engineering

Fare 결측값 채우기

```
## Fare는 testset에 결측값이 있는데 train의 정보로만 판단해야하므로
## train의 남자,부모형제없고, embarked가 S면서 나이대 비슷한 사람과 연관지어서 값을 넣는다.
## 가정이다.

Data.loc[Data['Fare'].isna(),'Fare']=6.2375
```

TicketHead 추출

```
## Ticket은 앞자리가 같으면 비슷하지 않을까 가정.
##
Data['TicketHead']=Data['Ticket'].apply(lambda x:x[0])
```

drop 전처리끝난 column

```
## Cabin은 결측값이 많아서 채우기보다 버리기로 판단한다.
## 그 외 처리했던 변수들 제거 한다.


Data=Data.drop(['Ticket','Cabin','Name','SibSp','Parch'],axis=1)
```

더미화

```
## 마무리
Data=pd.get_dummies(Data,drop_first=True)
```

표준화 – 인공신경망은 스케일이 중요하다

```
from sklearn.preprocessing import StandardScaler

X=train.drop(['Survived'],axis=1).values

sc=StandardScaler()
X=sc.fit_transform(X)
test=sc.transform(test)
y=train['Survived'].values
```

4. modeling

결과를 다시 얻기 위한 seed 고정.

```
## reproducibility
import os
import random
import tensorflow as tf

seed=71

def seedEvery(seed):
    os.environ['PYTHONHASHSEED']=str(seed)
    random.seed(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)

seedEvery(seed)
```

weights와 bias 초기 설정 xavier

```
from keras.layers import Dense
from keras.models import Sequential

ANN=Sequential()
ANN.add(Dense(32,input_dim=27,activation='relu',kernel_initializer=tf.keras.initializers.Glorot
Normal()))
ANN.add(Dense(16,activation='relu',kernel_initializer=tf.keras.initializers.GlorotNormal()))
ANN.add(Dense(8,activation='relu',kernel_initializer=tf.keras.initializers.GlorotNormal()))
ANN.add(Dense(4,activation='relu',kernel_initializer=tf.keras.initializers.GlorotNormal()))
ANN.add(Dense(2,activation='relu',kernel_initializer=tf.keras.initializers.GlorotNormal()))
ANN.add(Dense(1,activation='sigmoid'))

ANN.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

인공신경망 모델 빌드

```
from keras.layers import Dense
from keras.models import Sequential

ANN=Sequential()
ANN.add(Dense(32,input_dim=27,activation='relu'))
ANN.add(Dense(16,activation='relu'))
ANN.add(Dense(8,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(2,activation='relu'))
ANN.add(Dense(1,activation='sigmoid'))

ANN.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

모델 학습 – 최적의 모델을 위한 callback 정의

```
callbacks=[
    tf.keras.callbacks.EarlyStopping(monitor='loss',patience=3),
    tf.keras.callbacks.ModelCheckpoint(filepath='/kaggle/working/model.h5',
                                       save_weights_only=True,
                                       monitor='accuracy',
                                       mode='max',
                                       save_best_only=True)
]
history=ANN.fit(X,y,epochs=200,steps_per_epoch=50,callbacks=callbacks)
```

5. 결과분석

학습 결과 – 과적합없이 수렴하고 있다.

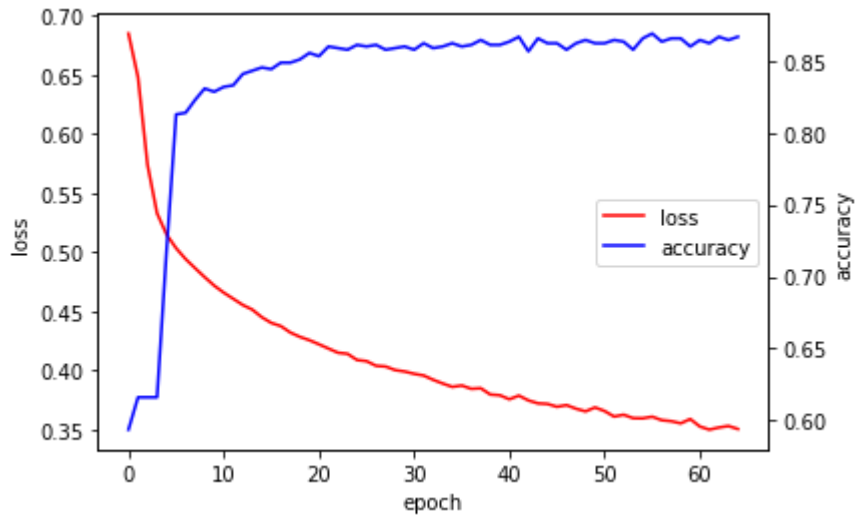
```
fig,loss_ax=plt.subplots()

line1=loss_ax.plot(history.history['loss'],'r',label='loss')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')

acc_ax=loss_ax.twinx()
line2=acc_ax.plot(history.history['accuracy'],'b',label='accuracy')
acc_ax.set_ylabel('accuracy')

lines=line1+line2
labs=[l.get_label() for l in lines]
loss_ax.legend(lines,labs,loc='right')

plt.show()
```



6. Competition 결과

0.79정도의 높은 정확도를 보인다. (상위 약 10%)

submit (2).csv 0.78947

a few seconds ago by 98hyun

add submission details

코드 링크

https://www.kaggle.com/hwangchanghyun/kaggle-ann

7. 시도 했던 방법들

가족의 특성을 살린 파생변수 생성. 다만 과적합으로 인해 버림.

```
Data['New']=Data.apply(lambda row:str(row['fsize'])+'#'+row['Embarked']+
                       '#'+row['TicketHead']+'#'+row['Cabin'],axis=1)
```

validation을 통한 검증. 파라미터를 사용해봤지만 그래도 과적합.

```
callbacks=[
    tf.keras.callbacks.EarlyStopping(monitor='loss',patience=3),
    tf.keras.callbacks.ModelCheckpoint(filepath='/kaggle/working/model.h5',
                                       save_weights_only=True,
                                       monitor='val_loss',
                                       mode='min',
                                       save_best_only=True)
]
history=ANN.fit(X,y,validation_split=0.15,
                epochs=200,steps_per_epoch=50,callbacks=callbacks)
```

코드 링크

https://www.kaggle.com/hwangchanghyun/kaggle-ann

Cabin또한 의미가 있을것이라 판단. 다만, 없이 했을 때보다 점수가 낮아서 제거.

```
## cabin 도 그룹별로 의미가 있다고 생각했다.
# Data.groupby('Cabin')['Survived'].mean()
Data['Cabin']=Data['Cabin'].fillna('unknown').apply(lambda row:row[0])
```

감사합니다.