타이타닉 전처리

생존자 예측 머신러닝

2212523
소비자경제학과
주정빈

01

타이타닉 EDA

가설

젊은 남성의 생존율이 가장 낮을 것이다.

In [7]:
```
p1=train[['Age_cat','Sex','Survived']]
p2=p1.value_counts().sort_index()
p2
```

Out[7]:

Age_cat	Sex	Survived	
child	female	0	16
		1	33
	male	0	29
		1	22
middle	female	0	14
		1	54
	male	0	98
		1	22
old	female	0	17
		1	36
	male	0	118
		1	17
prime	female	0	2
		1	22
	male	0	37
		1	8
young	female	0	32
		1	88
	male	0	186
		1	40

dtype: int64

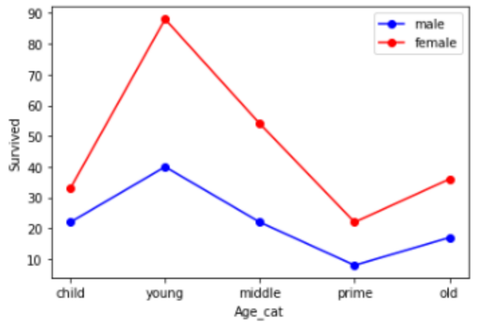
가설

젊은 남성의 생존율이 가장 낮을 것이다.

결과: 남성의 생존율이 여성에 비해
낮은 것은 맞았지만
대체적으로
나이가 어릴수록, 젊을수록
생존율이 높은 결과가 나타남.

->가설 틀림

In [15]:
```
def get_category(age):

    cat=''

    if age <=16 : cat= 'child'

    elif age <=32 : cat = 'young'

    elif age <=48 : cat = 'middle'

    elif age <=64 : cat = 'prime'

    else : cat ='old'

    return cat

train["Age_cat"] = train['Age'].apply(lambda x :get_category(x))

Age_cat=['child','young','middle','prime','old']

Survived_m=(22,40,22,8,17)
Survived_f=(33,88,54,22,36)

plt.plot(Age_cat, Survived_m, data=train, color='blue', marker='o', linestyle='solid', label='male')
plt.plot(Age_cat, Survived_f, data=train, color='red', marker='o', linestyle='solid',label='female')

plt.xlabel('Age_cat')
plt.ylabel('Survived')

plt.legend()
plt.show()
```



02

타이타닉 전처리

01 데이터 전처리

In [32]:
```
train_and_test = [train,test]
train_and_test
```

Out[32]:
```
[     PassengerId  Survived  Pclass  ₩
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  ₩
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 ...     ...   ...    ...
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/02. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns],
     PassengerId  Pclass                                          Name  ₩
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
```

02 Name

In [33]:
```
for dataset in train_and_test:
    dataset['Title']=dataset['Name'].str.extract('([A-Za-z]+)₩.' ,expand=False)
```

In [99]:
```
# Title을 가진 승객이 몇 명 존재하는지 확인

## crosstab 함수를 사용해 성별을 기준으로 Title 데이터의 개수를 구함.
```

In [34]:
```
pd.crosstab(train['Title'], train["Sex"])
```

Out [34]:

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

02 Name

In [ ]:
```
# Title 데이터의 출력 형태 설정

## replace 함수를 사용해 흔하지 않은 Title은 Other로 대체하고, 중복되는 표현은 통일함.
```

In [35]:
```
for dataset in train_and_test:
    dataset['Title']=dataset['Title'].replace(['Capt' , 'Col' , 'Countess' , 'Don' ,'Dona' ,'Dr' , 'Jonkheer' ,
                                               'Lady' , 'Major' , 'Rev' , 'Sir'] , 'Other')
    dataset['Title']=dataset['Title'].replace('Mlle' , 'Miss')
    dataset['Title']=dataset['Title'].replace('Ms' , 'Miss')
    dataset['Title']=dataset['Title'].replace('Mme' , 'Mrs')
```

In [ ]:
```
# Title 데이터가 제대로 추출됐는지 확인

## groupby 함수를 사용해 Title을 기준으로 그룹을 나누어 Title별로 생존율의 평균을 구함.
## index를 사용하고 싶은 않은 경우, as_index=False 를 설정
```

In [36]:
```
train[['Title' , 'Survived']].groupby(['Title'] , as_index=False).mean()
```

Out [36]:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Other	0.347826

03 Sex

In [103]:
```
# Sex 데이터의 데이터 타입 설정
```

In [38]:
```
for dataset in train_and_test:
    dataset['Sex']=dataset['Sex'].astype(str)
```

04 Age

In [105]:
```
# Age 특성: 결측치를 중앙값으로 채우는 방법 사용
```

In [ ]:
```
# Age 데이터의 중앙값으로 결측치 채우기
# Age 데이터의 데이터 타입 설정 (순서 상관 x)
# Age 데이터의 나이대 구분
## train 데이터에 AgeBand 열을 추가한 후, pd.cut()을 사용해 Age 데이터를 같은 길이의 구간을 가지는 5개의 그룹으로 나눔.
```

In [39]:
```
for dataset in train_and_test:
    dataset['Age'].fillna(dataset['Age'].median() , inplace=True)
    dataset['Age']=dataset['Age'].astype(int)
    train['AgeBand']=pd.cut(train['Age'],5)
```

In [ ]:
```
# 구분한 나이대 별로 생존율의 평균 확인

## groupby 함수를 사용해 AgeBand를 기준으로 그룹을 나누어 AgeBand별로 생존율의 평균을 구함.
```

In [40]:
```
train[['AgeBand' , 'Survived']].groupby(['AgeBand'],as_index=False).mean()
```

Out [40]:

	AgeBand	Survived
0	(-0.08, 16.0]	0.550000
1	(16.0, 32.0]	0.344762
2	(32.0, 48.0]	0.403226
3	(48.0, 64.0]	0.434783
4	(64.0, 80.0]	0.090909

04 Age

In [ ]:
```
# 결측치 사라졌는지 확인
```

In [41]:
```
train.isnull().sum()
```

Out[41]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
Title            0
AgeBand          0
dtype: int64
```

In [108]:
```
# Age 데이터의 출력 형태 설정

## loc 함수를 사용해 조건에 맞는 Age열이 설정한 값으로 출력되도록 설정
## 예로, 16살 이하의 Age열의 값들은 0으로 출력되도록 설정
## map 함수를 사용해 Age열의 값들이 새로 지정한 문자 형태로 출력되도록 설정
## 예로, 0은 Child로 출력되도록 설정
```

In [42]:
```
for dataset in train_and_test:
    dataset.loc[dataset['Age']<=16, 'Age']=0
    dataset.loc[(dataset['Age']>16)&(dataset['Age']<=32), 'Age']=1
    dataset.loc[(dataset['Age']>32)&(dataset['Age']<=48), 'Age']=2
    dataset.loc[(dataset['Age']>48)&(dataset['Age']<=64), 'Age']=3
    dataset.loc[dataset['Age']>64, 'Age']=4
    dataset['Age']=dataset['Age'].map({0:'child' , 1:'Young' , 2:'Middle' , 3:'Prime' , 4:'Old'}).astype(str)
```

05 Embarked

In [110]:
```
# Embarked 특성: 결측치를 최빈값으로 채우는 방법 사용
```

In [ ]:
```
# Embarked에서 결측치를 가진 행을 추출
```

In [43]:
```
train[train['Embarked'].isnull()==True]
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	AgeBand
61	62	1	1	Icard, Miss. Amelie	female	Middle	0	0	113572	80.0	B28	NaN	Miss	(32.0, 48.0]
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	Prime	0	0	113572	80.0	B28	NaN	Mrs	(48.0, 64.0]

In [ ]:
```
# Embarked의 최빈값 구하기

## value_counts 함수 사용해서 Embarked 데이터의 개수 확인
```

In [44]:
```
train['Embarked'].value_counts()
```

Out[44]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [114]:
```
# Embarked 데이터의 최빈값으로 결측치 채우기
# Embarked 데이터의 데이터 타입 설정
```

In [45]:
```
for dataset in train_and_test:
    dataset['Embarked']=dataset['Embarked'].fillna('S')
    dataset['Embarked']=dataset['Embarked'].astype(str)
```

06 Fare

In [115]:
```
# Fare 특성: 결측치를 같은 Pclass인 승객들의 평균 Fare값으로 채우는 방법 사용
```

In [ ]:
```
# Fare에서 결측치를 가진 행을 추출
```

In [46]:
```
test[test['Fare'].isnull()==True]
```

Out[46]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
152	1044	3	Storey, Mr. Thomas	male	Prime	0	0	3701	NaN	NaN	S	Mr

In [ ]:
```
# Pclass별로 Fare의 평균값 구하기

## groupby 함수를 사용해서 Pclass를 기준으로 그룹을 나눔.
```

In [47]:
```
train[['Pclass','Fare']].groupby(['Pclass'] , as_index=False).mean()
```

Out[47]:

	Pclass	Fare
0	1	84.154687
1	2	20.662183
2	3	13.675550

In [118]:
```
# 같은 Pclass인 승객들의 평균 Fare값으로 결측치 채우기
```

In [48]:
```
for dataset in train_and_test:
    dataset['Fare']=dataset['Fare'].fillna(13.675)
```

06 Fare

In [ ]:
```
# 결측치 사라졌는지 확인
```

In [49]:
```
test.isnull().sum()
```

Out[49]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
Title            0
dtype: int64
```

In [ ]:
```
# Fare 데이터의 요금 가격대 구분

## train 데이터에 FareBand 열을 추가한 후, pd.qcut 함수를 사용해 Fare 데이터를 같은 개수의 구간을 가지는 5개의 그룹으로 나눔.
```

In [50]:
```
for dataset in train_and_test:
    train['FareBand']=pd.qcut(train['Fare'],5)
```

In [ ]:
```
# 가격대 별로 Fare의 평균 확인

## groupby 함수를 사용해 FareBand를 기준으로 그룹을 나누어 FareBand별로 Fare의 평균을 구함.
```

In [51]:
```
train[['FareBand','Fare']].groupby(['FareBand'] , as_index=False).mean()
```

Out[51]:

	FareBand	Fare
0	(-0.001, 7.854]	6.822908
1	(7.854, 10.5]	8.623997
2	(10.5, 21.679]	15.215019
3	(21.679, 39.688]	28.922592
4	(39.688, 512.329]	102.629451

06 Fare

In [ ]:
```
# Fare 데이터의 출력 형태 설정
# Fare 데이터의 데이터 타입 설정

## loc 함수를 사용해 조건에 맞는 Fare열이 설정한 값으로 출력되도록 설정
## 예로, 7.854 이하의 Age열의 값들은 0으로 출력되도록 설정
```

In [52]:
```
for dataset in train_and_test:
    dataset.loc[dataset['Fare']<=7.854 ,'Fare']=0
    dataset.loc[(dataset['Fare']>7.854)&(dataset['Fare']<=10.5),'Fare']=1
    dataset.loc[(dataset['Fare']>10.5)&(dataset['Fare']<=21.679),'Fare']=2
    dataset.loc[(dataset['Fare']>21.679)&(dataset['Fare']<=39.688),'Fare']=3
    dataset.loc[dataset['Fare']>39.688,'Fare']=4
    dataset['Fare']=dataset['Fare'].astype(int)
```

07 SibSp&Parch(Family)

In [125]:
```
# SibSp와 Parch, 두 개의 특성을 합쳐서 Family라는 새로운 특성으로 만들기
# Family 데이터의 데이터 타입 설정
```

In [53]:
```
for dataset in train_and_test:
    dataset['Family']=dataset['Parch']+dataset['SibSp']
    dataset['Family']=dataset['Family'].astype(int)
```

In [ ]:
```
# 최종 데이터 확인 (train)
```

In [54]:
```
train.head()
```

Out [54]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	AgeBand	FareBand	Family
0	1	0	3	Braund, Mr. Owen Harris	male	Young	1	0	A/5 21171	0	NaN	S	Mr	(16.0, 32.0]	(-0.001, 7.854]	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	Middle	1	0	PC 17599	4	C85	C	Mrs	(32.0, 48.0]	(39.688, 512.329]	1
2	3	1	3	Heikkinen, Miss. Laina	female	Young	0	0	STON/O2. 3101282	1	NaN	S	Miss	(16.0, 32.0]	(7.854, 10.5]	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	Middle	1	0	113803	4	C123	S	Mrs	(32.0, 48.0]	(39.688, 512.329]	1
4	5	0	3	Allen, Mr. William Henry	male	Middle	0	0	373450	1	NaN	S	Mr	(32.0, 48.0]	(7.854, 10.5]	0

In [ ]:
```
# 최종 데이터 확인 (test)
```

In [55]:
```
test.head()
```

Out [55]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Family
0	892	3	Kelly, Mr. James	male	Middle	0	0	330911	0	NaN	Q	Mr	0
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	Middle	1	0	363272	0	NaN	S	Mrs	1
2	894	2	Myles, Mr. Thomas Francis	male	Prime	0	0	240276	1	NaN	Q	Mr	0
3	895	3	Wirz, Mr. Albert	male	Young	0	0	315154	1	NaN	S	Mr	0
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	Young	1	1	3101298	2	NaN	S	Mrs	2

08 특성 추출 및 나머지 전처리 학습시킬 때 제외시킬 Feature 제거

In [57]:
```
# Survived label을 예측하는 문제이므로 train에서 Survived 열 제거
train_label=train['Survived']
train=train.drop('Survived',axis=1)
# drop() : 행 또는 열을 삭제
# axis=0: 행을 삭제, axis=1: 열을 삭제
```

In [4]:
```
# 머신러닝을 할 때 필요도가 낮은 특성 제거
```

In [58]:
```
features1_drop=['Name','Ticket','Cabin','SibSp','Parch','PassengerId']
train=train.drop(features1_drop , axis=1)
test=test.drop(features1_drop , axis=1)
```

In [ ]:
```
# 전처리 과정에서 만든 임시 특성 제거
```

In [59]:
```
features2_drop=['AgeBand','FareBand']
train=train.drop(features2_drop , axis=1)
```

In [ ]:
```
# 잘 제거 되었는지 확인
```

In [60]:
```
print(train.head())
print(test.head())
```

```
   Pclass     Sex     Age  Fare Embarked Title  Family
0       3    male   Young     0        S    Mr       1
1       1  female  Middle     4        C   Mrs       1
2       3  female   Young     1        S  Miss       0
3       1  female  Middle     4        S   Mrs       1
4       3    male  Middle     1        S    Mr       0
   Pclass     Sex     Age  Fare Embarked Title  Family
0       3    male  Middle     0        Q    Mr       0
1       3  female  Middle     0        S   Mrs       1
2       2    male   Prime     1        Q    Mr       0
3       3    male   Young     1        S    Mr       0
4       3  female   Young     2        S   Mrs       2
```

In [131]:
```
# 뒤에 나올 get_dummies를 위해 데이터 통합을 하여 오류를 방지

## pd.concat() : 데이터 통합 함수
```

In [62]:
```
all_data=pd.concat([train,test])
```

08 특성 추출 및 나머지 전처리 one-hot encoding

In [1]:
```
#데이터 타입 확인
```

In [63]:
```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Pclass    1309 non-null   int64
 1   Sex       1309 non-null   object
 2   Age       1309 non-null   object
 3   Fare      1309 non-null   int32
 4   Embarked  1309 non-null   object
 5   Title     1309 non-null   object
 6   Family    1309 non-null   int32
dtypes: int32(2), int64(1), object(4)
memory usage: 51.1+ KB
```

In [2]:
```
#.astype()을 이용하여 데이터 타입 바꾸기
```

In [64]:
```
all_data['Pclass']=all_data['Pclass'].astype(str)
all_data['Fare']=all_data['Fare'].astype(str)
all_data['Family']=all_data['Family'].astype(str)
```

In [ ]:
```
#데이터 타입이 잘 바뀌었는지 확인
```

In [65]:
```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Pclass    1309 non-null   object
 1   Sex       1309 non-null   object
 2   Age       1309 non-null   object
 3   Fare      1309 non-null   object
 4   Embarked  1309 non-null   object
 5   Title     1309 non-null   object
 6   Family    1309 non-null   object
dtypes: object(7)
memory usage: 46.0+ KB
```

08 특성 추출 및 나머지 전처리 one-hot encoding

In [66]:
```
#get_dummies 함수를 사용하여 one-hot encoding을 수행

all_data=pd.get_dummies(all_data)

##get_dummies는 문자형에 대해서만 적용이 됨
```

In [3]:
```
#함수가 잘 적용이 되었는지 확인
```

In [67]:
```
all_data.head()
```

Out [67]:

	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_Middle	Age_Old	Age_Prime	Age_Young	Age_child	...	Title_Other	Family_0	Family_1	Fami
0	0	0	1	0	1	0	0	0	1	0	...	0	0	1	
1	1	0	0	1	0	1	0	0	0	0	...	0	0	1	
2	0	0	1	1	0	0	0	0	1	0	...	0	1	0	
3	1	0	0	1	0	1	0	0	0	0	...	0	0	1	
4	0	0	1	0	1	1	0	0	0	0	...	0	1	0	

5 rows × 32 columns

In [136]:
```
# 머신러닝을 위한 데이터 분리
```

In [68]:
```
print(train.shape)
print(test.shape)
```

```
(891, 7)
(418, 7)
```

In [69]:
```
train=all_data[:891]
test=all_data[891:]
```

In [70]:
```
#sklearn 라이브러리에 있는 shuffle 함수 불러오기
#데이터의 순서가 같으면 기계가 순서를 학습하기 때문에 방해가 될 수 있음

from sklearn.utils import shuffle
train, train_label = shuffle(train, train_label, random_state = 5)
```

03

머신러닝

머신러닝

랜덤 포레스트는 약 88%의 정확도
회귀분석은 약 82%의 정확도

>회귀분석보다 랜덤포레스트의
정확도가 약 6% 정도
더 높은 것을 알 수 있음

In [72]:
```
# 머신러닝 방법 1 - 랜덤포레스트
from sklearn.ensemble import RandomForestClassifier

rf_pred = running(RandomForestClassifier(n_estimators=100))

#정확도는 시도할 때 마다 달라질 수 있습니다!
```

```
Accuracy :  88.55 %
```

In [73]:
```
# 머신러닝 방법 2 - 회귀분석
from sklearn.linear_model import LogisticRegression

log_pred=running(LogisticRegression())

#정확도는 시도할 때 마다 달라질 수 있습니다!
```

```
Accuracy :  82.72 %
```