타이타닉 데이터 셋 - 데이터 처리

학습 목표

- 다양한 추가적인 변수를 추가해 본다.

01. Feature Engineering & 모델링

In [27]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [28]:

```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

```
Numerical Features: Age (Continuous), Fare (Continuous), SibSp (Discrete), Parch (Discret
e)
Categorical Features: Survived, Sex, Embarked, Pclass
Alphanumeric Features: Ticket, Cabin
```

02. Age 나이대별로 확인해 보자.

In [29]:

```
train["Age"] = train["Age"].fillna(train['Age'].median())
test["Age"] = test["Age"].fillna(test['Age'].median())
```

In [30]:

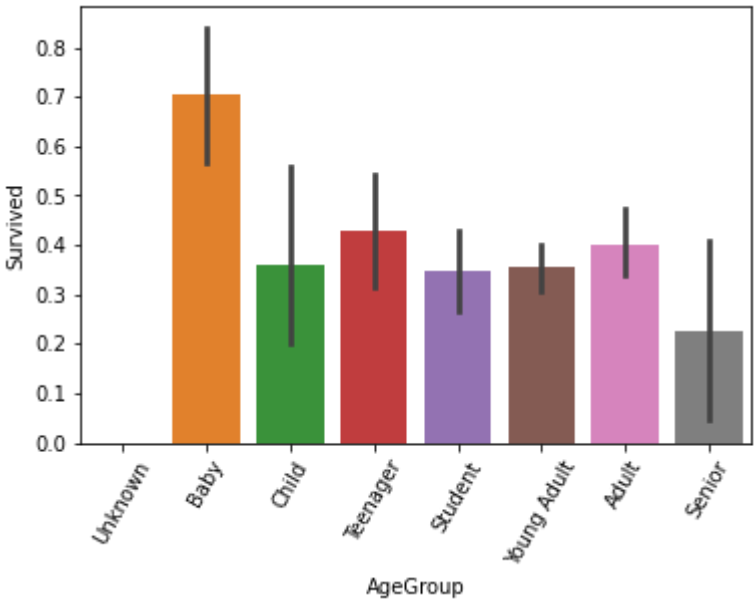
```
print( train.info(), test.info() )
```

```
<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          891 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
<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          418 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
None None
```

In [32]:

```
bins = [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]  # 나이대 구분
labels = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Senior']
train['AgeGroup'] = pd.cut(train["Age"], bins, labels = labels)
test['AgeGroup'] = pd.cut(test["Age"], bins, labels = labels)

# 변경된 내용 막대 그래프 그리기
sns.barplot(x="AgeGroup", y="Survived", data=train)
plt.xticks(rotation=60)
plt.show()
```



03. Cabin Feature 확인

- Cabin이 null이 아닐 경우

In [33]:

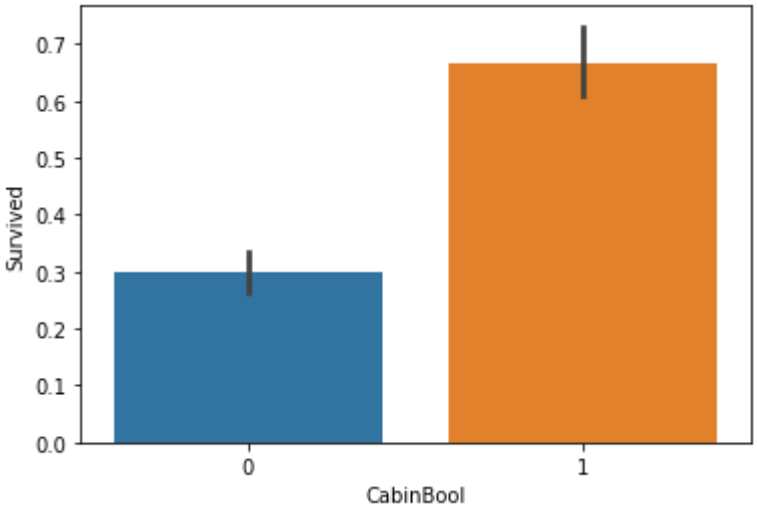
```
train["CabinBool"] = (train["Cabin"].notnull().astype('int'))
test["CabinBool"] = (test["Cabin"].notnull().astype('int'))
```

In [34]:

```
sns.barplot(x="CabinBool", y="Survived", data=train)
```

Out[34]:

```
<AxesSubplot:xlabel='CabinBool', ylabel='Survived'>
```



In [35]:

```
train["CabinBool"].value_counts()
```

Out[35]:

```
0    687
1    204
Name: CabinBool, dtype: int64
```

04. 컬럼 삭제

Cabin Feature drop

In [36]:

```
train = train.drop(['Cabin'], axis = 1)
test = test.drop(['Cabin'], axis = 1)
```

Ticket Feature drop

In [37]:

```
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
```

05. 승선항 결측치 처리

In [38]:

```
print( train['Sex'].value_counts() )
print( train['Embarked'].value_counts() )
train = train.fillna({"Embarked": "S"})
```

```
male      577
female    314
Name: Sex, dtype: int64
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

06. 데이터 문자열 처리 - 이름에 대한 구분자 가져오기

- [].str.extract()

In [39]:

```
#create a combined group of both datasets
combine = [train, test]

#extract a title for each Name in the train and test datasets
for dataset in combine:
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)W.', expand=False)

dataset['Title']
```

Out[39]:

```
0          Mr
1         Mrs
2          Mr
3          Mr
4         Mrs
        ...
413        Mr
414      Dona
415        Mr
416        Mr
417    Master
Name: Title, Length: 418, dtype: object
```

In [40]:

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

Out[40]:

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

06. 데이터 문자열 처리 - 이름 구분 - 변경

- [].replace()

In [42]:

```
# 다양한 이름을 변경하기
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Capt', 'Col',
                                            'Don', 'Dr', 'Major', 'Rev', 'Jonkheer', 'Dona'], 'Rare')

    dataset['Title'] = dataset['Title'].replace(['Countess', 'Lady', 'Sir'], 'Royal')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

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

Out[42]:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Rare	0.285714
5	Royal	1.000000

07. 수치형 값으로 그룹을 매핑하기

FamilySize feature

In [43]:

```
train['FamilySize'] = train['SibSp'] + train['Parch'] + 1
test['FamilySize'] = test['SibSp'] + test['Parch'] + 1
```

08. NameLength feature

In [44]:

```
train["NameLength"] = train["Name"].apply(lambda x: len(x))
test["NameLength"] = test["Name"].apply(lambda x: len(x))
```

In [45]:

```
#map each of the title groups to a numerical value
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Royal": 5, "Rare": 6}
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

train.head()
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S

In [50]:

```
train[train["Title"] == 1]["AgeGroup"].mode()
```

Out[50]:

```
0    Young Adult
Name: AgeGroup, dtype: category
Categories (8, object): ['Unknown' < 'Baby' < 'Child' < 'Teenager' < 'Student' < 'Yo
ung Adult' < 'Adult' < 'Senior']
```

In [51]:

```
# fill missing age with mode age group for each title
mr_age = train[train["Title"] == 1]["AgeGroup"].mode()      # Young Adult
miss_age = train[train["Title"] == 2]["AgeGroup"].mode()    # Student
mrs_age = train[train["Title"] == 3]["AgeGroup"].mode()     # Adult
master_age = train[train["Title"] == 4]["AgeGroup"].mode() # Baby
royal_age = train[train["Title"] == 5]["AgeGroup"].mode()  # Adult
rare_age = train[train["Title"] == 6]["AgeGroup"].mode()   # Adult

age_title_mapping = {1: "Young Adult", 2: "Student", 3: "Adult", 4: "Baby", 5: "Adult", 6: "Adult"}
```

In [52]:

```
for x in range(len(train["AgeGroup"])):
    if train["AgeGroup"][x] == "Unknown":
        train["AgeGroup"][x] = age_title_mapping[train["Title"][x]]

for x in range(len(test["AgeGroup"])):
    if test["AgeGroup"][x] == "Unknown":
        test["AgeGroup"][x] = age_title_mapping[test["Title"][x]]
```

In [53]:

```
# 수치형 값으로 Age 컬럼을 매핑
age_mapping = {'Baby': 1, 'Child': 2, 'Teenager': 3, 'Student': 4, 'Young Adult': 5, 'Adult': 6, 'Se
train['AgeGroup'] = train['AgeGroup'].map(age_mapping)
test['AgeGroup'] = test['AgeGroup'].map(age_mapping)

train.head()

# Age 컬럼을 삭제
train = train.drop(['Age'], axis = 1)
test = test.drop(['Age'], axis = 1)
```

In [54]:

```
# drop the name feature since it contains no more useful information.
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)
```

In [55]:

```
# map each Sex value to a numerical value
sex_mapping = {"male": 0, "female": 1}
train['Sex'] = train['Sex'].map(sex_mapping)
test['Sex'] = test['Sex'].map(sex_mapping)

train.head()
```

Out[55]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Fare	Embarked	AgeGroup	CabinBo
0	1	0	3	0	1	0	7.2500	S	4.0	
1	2	1	1	1	1	0	71.2833	C	6.0	
2	3	1	3	1	0	0	7.9250	S	5.0	
3	4	1	1	1	1	0	53.1000	S	5.0	
4	5	0	3	0	0	0	8.0500	S	5.0	

In [56]:

```
# 수치형 값으로 승선항을 매핑한다.
embarked_mapping = {"S": 1, "C": 2, "Q": 3}
train['Embarked'] = train['Embarked'].map(embarked_mapping)
test['Embarked'] = test['Embarked'].map(embarked_mapping)

train.head()
```

Out[56]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Fare	Embarked	AgeGroup	CabinBo
0	1	0	3	0	1	0	7.2500	1	4.0	
1	2	1	1	1	1	0	71.2833	2	6.0	
2	3	1	3	1	0	0	7.9250	1	5.0	
3	4	1	1	1	1	0	53.1000	1	5.0	
4	5	0	3	0	0	0	8.0500	1	5.0	

In [57]:

```
train.info(), test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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   Sex          891 non-null    int64
 4   SibSp        891 non-null    int64
 5   Parch        891 non-null    int64
 6   Fare         891 non-null    float64
 7   Embarked     891 non-null    int64
 8   AgeGroup     891 non-null    float64
 9   CabinBool    891 non-null    int32
 10  Title        891 non-null    int64
 11  FamilySize   891 non-null    int64
 12  NameLength   891 non-null    int64
dtypes: float64(2), int32(1), int64(10)
memory usage: 87.1 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Sex          418 non-null    int64
 3   SibSp        418 non-null    int64
 4   Parch        418 non-null    int64
 5   Fare         417 non-null    float64
 6   Embarked     418 non-null    int64
 7   AgeGroup     418 non-null    float64
 8   CabinBool    418 non-null    int32
 9   Title        418 non-null    int64
 10  FamilySize   418 non-null    int64
 11  NameLength   418 non-null    int64
dtypes: float64(2), int32(1), int64(9)
memory usage: 37.7 KB
```

Out[57]:

```
(None, None)
```

Fare Feature

In [58]:

```
#fill in missing Fare value in test set based on mean fare for that Pclass
for x in range(len(test["Fare"])):
    if pd.isnull(test["Fare"][x]):
        pclass = test["Pclass"][x] #Pclass = 3
        test["Fare"][x] = round(train[train["Pclass"] == pclass]["Fare"].mean(), 4)
```

```
<ipython-input-58-27ffceb84a8f>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  test["Fare"][x] = round(train[train["Pclass"] == pclass]["Fare"].mean(), 4)
```

In [59]:

```
#map Fare values into groups of numerical values
train['FareBand'] = pd.qcut(train['Fare'], 4, labels = [1, 2, 3, 4])
test['FareBand'] = pd.qcut(test['Fare'], 4, labels = [1, 2, 3, 4])

#drop Fare values
train = train.drop(['Fare'], axis = 1)
test = test.drop(['Fare'], axis = 1)
```

In [60]:

```
train.head()
```

Out[60]:

	PassengerId	Survived	Pclass	Sex	SibSp	Parch	Embarked	AgeGroup	CabinBool	Title
0	1	0	3	0	1	0	1	4.0	0	1
1	2	1	1	1	1	0	2	6.0	1	3
2	3	1	3	1	0	0	1	5.0	0	2
3	4	1	1	1	1	0	1	5.0	1	3
4	5	0	3	0	0	0	1	5.0	0	1

In [61]:

```
test.head()
```

Out[61]:

	PassengerId	Pclass	Sex	SibSp	Parch	Embarked	AgeGroup	CabinBool	Title	FamilySize
0	892	3	0	0	0	3	5.0	0	1	1
1	893	3	1	1	0	1	6.0	0	3	2
2	894	2	0	0	0	3	7.0	0	1	1
3	895	3	0	0	0	1	5.0	0	1	1
4	896	3	1	1	1	1	4.0	0	3	3

모델 선택하기

In [65]:

```
# 'Name', 'Ticket' =>  문자포함으로 제외
sel = ['Pclass', 'Sex', 'SibSp', 'Parch', 'Embarked', 'AgeGroup', 'CabinBool', 'Title', 'FamilySize'

# 학습에 사용될 데이터 준비 X_train, y_train
X_train = train[sel]
y_train = train['Survived']
X_test = test[sel]
```

In [66]:

```
from sklearn.linear_model import LogisticRegression
```

In [69]:

```
from sklearn.linear_model import LogisticRegression
log_r = LogisticRegression()
log_r.fit(X_train, y_train)

# 예측
pred = log_r.predict(X_test)
pred[:15]
```

```
C:WUsersWtotoWanaconda3WlibWsite-packagesWsklearnWlinear_modelW_logistic.py:763: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[69]:

```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1], dtype=int64)
```

In [71]:

```
test_pid = test['PassengerId']
pred = pred.astype(int)
df_pred = pd.DataFrame({'PassengerID':test_pid, 'Survived':pred})
df_pred.to_csv("log_fourth_model.csv", index=False)
```