# 타이타닉 생존자 예측 대회

## 학습 내용

- 1-1 데이터 불러오기
- 1-2 데이터 탐색하기
- 1-3 결측치 처리
- 1-4 모델 선택 및 평가

## Data Fields

| 구분 | 설명 | 값 |
|---|---|---|
| **Survival** | 생존 여부 | Survival. 0 = No, 1 = Yes |
| **Pclass** | 티켓의 클래스 | Ticket class. 1 = 1st, 2 = 2nd, 3 = 3rd |
| **Sex** | 성별(Sex) | 남(male)/여(female) |
| **Age** | 나이(Age in years.) | |
| **SibSp** | 함께 탑승한 형제와 배우자의 수 /siblings, spouses aboard the Titanic. | |
| **Parch** | 함께 탑승한 부모, 아이의 수 | # of parents / children aboard the Titanic. |
| **Ticket** | 티켓 번호(Ticket number) | (ex) CA 31352, A/5. 2151 |
| **Fare** | 탑승료(Passenger fare) | |
| **Cabin** | 객실 번호(Cabin number) | |
| **Embarked** | 탑승 항구(Port of Embarkation) | C = Cherbourg, Q = Queenstown, S = Southampton |

- siblings : 형제, 자매, 형제, 의붓 형제
- spouses : 남편, 아내 (정부와 약혼자는 무시)
- Parch : Parent(mother, father), child(daughter, son, stepdaughter, stepson)

In [1]:

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

In [2]:

```
train = pd.read_csv("data/titanic/train.csv")
test = pd.read_csv("data/titanic/test.csv")
sub = pd.read_csv("data/titanic/gender_submission.csv")
```

### EDA

## 수치형 변수 확인 및 요약값

In [3]:

```
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 [8]:

```
train['Survived'].dtype
```

Out[8]:

```
dtype('int64')
```

In [4]:

```
num_cols = [col for col in train.columns[:12]
              if train[col].dtype in ['int64', 'float64'] ]

print(num_cols)
train[num_cols].describe()
```

```
['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
```

Out[4]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

## 범주형 변수 살펴보기

In [9]:

```
cat_cols = [col for col in train.columns[:12]
              if train[col].dtype in ['O'] ]

print(cat_cols)

train[cat_cols].describe()
```

```
['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked']
```

Out[9]:

| | Name | Sex | Ticket | Cabin | Embarked |
|---|---|---|---|---|---|
| **count** | 891 | 891 | 891 | 204 | 889 |
| **unique** | 891 | 2 | 681 | 147 | 3 |
| **top** | Garfirth, Mr. John | male | 347082 | G6 | S |
| **freq** | 1 | 577 | 7 | 4 | 644 |

## 범주형 데이터에 대해 확인해 보기

In [11]:

```
import numpy as np
```

In [12]:

```
for col in cat_cols:
    uniq = np.unique(train[col].astype(str))
    print("colname : {}, uniq : {}".format(col, uniq), end="₩n₩n")
```

```
 'Van Impe, Mr. Jean Baptiste'
 'Van Impe, Mrs. Jean Baptiste (Rosalie Paula Govaert)'
 'Van der hoef, Mr. Wyckoff' 'Vande Velde, Mr. Johannes Joseph'
 'Vande Walle, Mr. Nestor Cyriel' 'Vanden Steen, Mr. Leo Peter'
 'Vander Cruyssen, Mr. Victor' 'Vander Planke, Miss. Augusta Maria'
 'Vander Planke, Mr. Leo Edmondus'
 'Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)'
 'Vestrom, Miss. Hulda Amanda Adolfina' 'Vovk, Mr. Janko'
 'Waelens, Mr. Achille' 'Walker, Mr. William Anderson' 'Ward, Miss. Anna'
 'Warren, Mrs. Frank Manley (Anna Sophia Atkinson)'
 'Watson, Mr. Ennis Hastings'
 'Watt, Mrs. James (Elizabeth "Bessie" Inglis Milne)'
 'Webber, Miss. Susan' 'Webber, Mr. James' 'Weir, Col. John'
 'Weisz, Mrs. Leopold (Mathilde Francoise Pede)' 'Wells, Miss. Joan'
 'West, Miss. Constance Mirium' 'West, Mr. Edwy Arthur'
 'West, Mrs. Edwy Arthur (Ada Mary Worth)' 'Wheadon, Mr. Edward H'
 'White, Mr. Percival Wayland' 'White, Mr. Richard Frasar'
 'Wick, Miss. Mary Natalie' 'Wick, Mrs. George Dennick (Mary Hitchcock)'
 'Widegren, Mr. Carl/Charles Peter' 'Widener, Mr. Harry Elkins'
 'Wiklund, Mr. Jakob Alfred' 'Wilhelms, Mr. Charles' 'Willey, Mr. Edward'
```

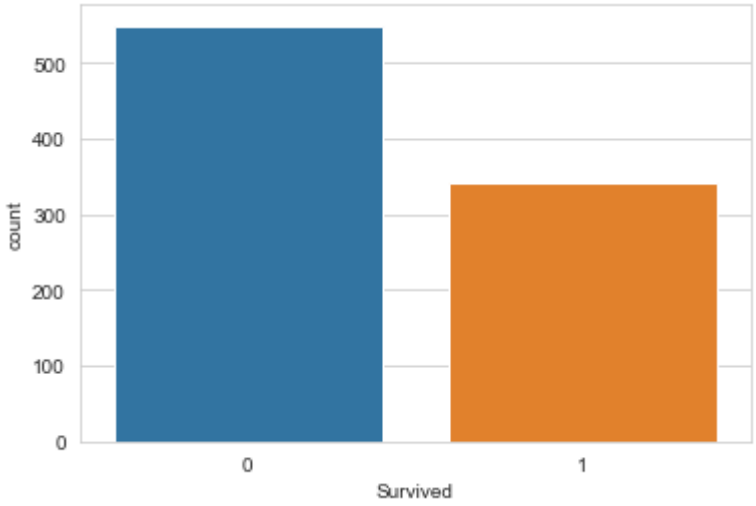
## EDA - 시각화

In [13]:

```
sns.set_style('whitegrid')   # seaborn 스타일 지정
sns.countplot(x='Survived', data=train)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f457592370>
```

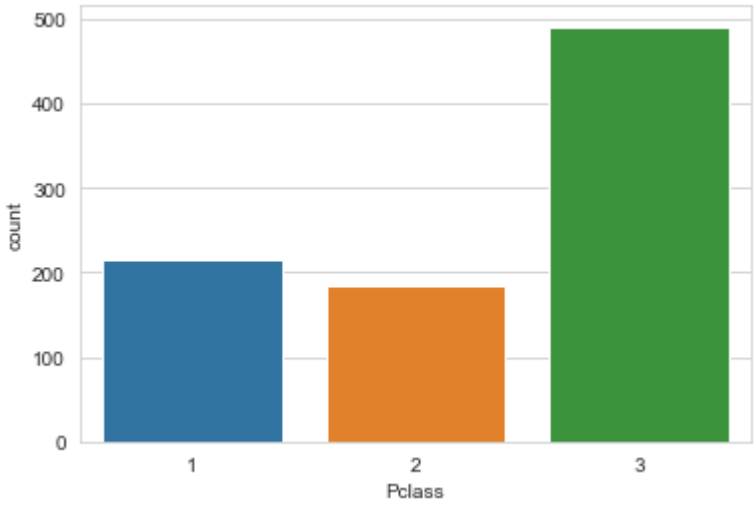



In [14]:

```
## 해보기 : PClass 별 Count
sns.countplot(x='Pclass', data=train)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f4575b4940>
```

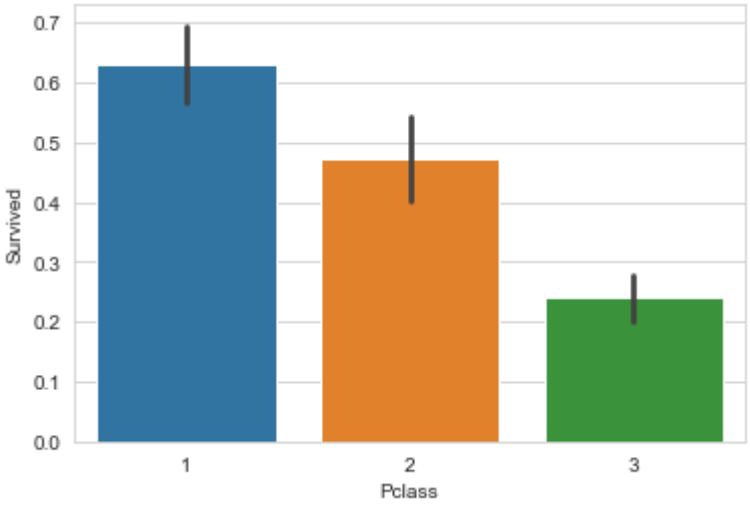


In [15]:

```
sns.set_style('whitegrid')   # seaborn 스타일 지정
sns.barplot(x='Pclass', y='Survived', data=train)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f457ef4bb0>
```

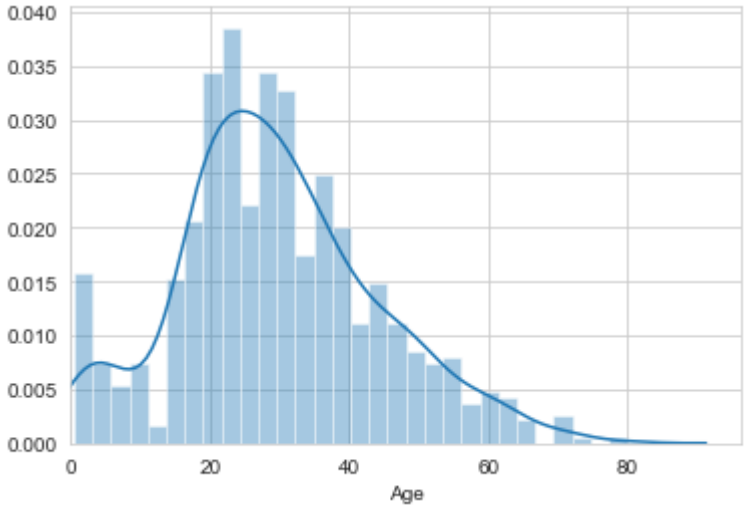



In [16]:

```
sns.distplot(train['Age'].dropna(), bins=30).set_xlim(0,)
```

Out[16]:

```
(0.0, 96.421405713925)
```

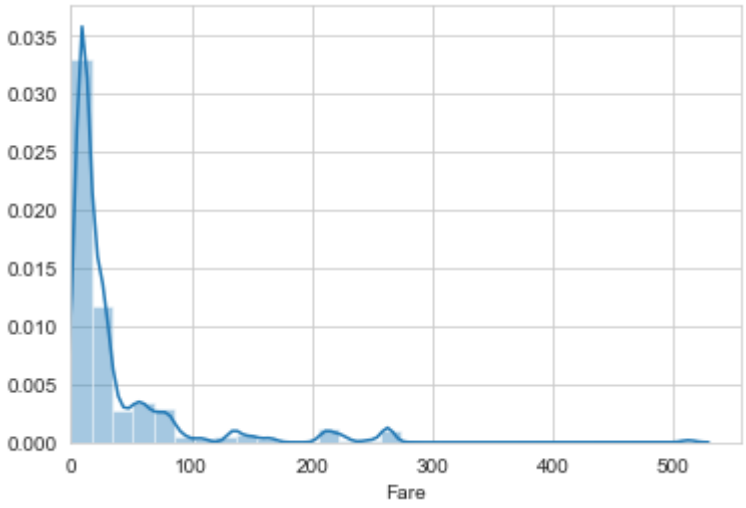


In [17]:

```
## 해보기 Fare
sns.distplot(test['Fare'].dropna(), bins=30).set_xlim(0,)
```

Out[17]:

(0.0, 556.2418231843072)

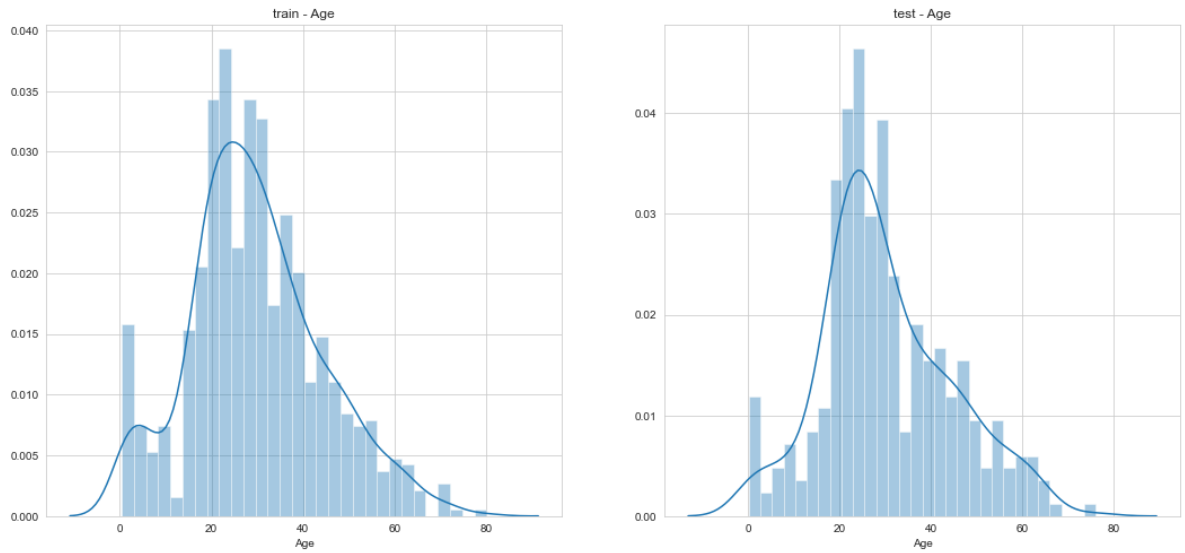



In [18]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))

# 첫번째 그래프
sns.distplot(train['Age'].dropna(), bins=30, ax=ax[0])
ax[0].set_title('train - Age')

# 두번째 그래프
sns.distplot(test['Age'].dropna(), bins=30, ax=ax[1])
ax[1].set_title('test - Age')
plt.show()
```



## 결측치 처리

- 나이는 평균값
- 결측치 채우기 [].fillna(값)

In [19]:

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

## 해보기
test['Fare'] = test['Fare'].fillna(test['Fare'].mean())

print(train.isnull().sum())
print(test.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

## 결측치 처리 - Embarked(승선항)

- 가장 많이 나온 값으로 결측치 처리를 하자

In [21]:

```
val_mode = train['Embarked'].mode()
print(val_mode[0])
train['Embarked'] = train['Embarked'].fillna(val_mode[0])
```

```
S
```

In [22]:

```
print(train.isnull().sum())
print(test.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

## 라벨 인코딩 및 자료형 변환

In [23]:

```
train['Sex'] = train['Sex'].map( {'female': 0, 'male': 1} ).astype(int)
test['Sex'] = test['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

train['Embarked'] = train['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)
test['Embarked']= test['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)
```

In [24]:

```
## 나이에 대한 int 처리
train['Age'] = train['Age'].astype('int')
test['Age'] = test['Age'].astype('int')
```

## 데이터 나누기

In [25]:

```
sel = ['PassengerId', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex', 'Embarked']

all_X = train[sel]
all_y = train['Survived']

last_X_test = test[sel]
```

In [26]:

```
from sklearn.model_selection import train_test_split
```

In [27]:

```
X_train, X_test, y_train, y_test = train_test_split(all_X,
                                                    all_y,
                                                    stratify=all_y,
                                                    test_size=0.3,
                                                    random_state=77 )
```

In [28]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
```

In [30]:

```
model = [DecisionTreeClassifier(), LogisticRegression(), LinearSVC(), KNeighborsClassifier()]

for model_one in model:
    model = model_one
    model.fit(X_train, y_train)
    acc_tr = model.score(X_train, y_train)
    acc_test = model.score(X_test, y_test)
    print("모델명 : {}, 정확도 {} {} ".format(model, acc_tr, acc_test) )
```

```
모델명 : DecisionTreeClassifier(), 정확도 1.0 0.75
모델명 : LogisticRegression(), 정확도 0.7817014446227929 0.8059701492537313
모델명 : LinearSVC(), 정확도 0.4911717495987159 0.4925373134328358
모델명 : KNeighborsClassifier(), 정확도 0.7335473515248796 0.6268656716417911

C:₩Users₩WJ₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762: Conve
rgenceWarning: 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(
C:₩Users₩WJ₩anaconda3₩lib₩site-packages₩sklearn₩svm₩_base.py:976: ConvergenceWarnin
g: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

## 실습 Logistic 모델을 이용하여 최종 모델을 만들고 제출해 보자.

In [31]:

```
model = LogisticRegression()
model.fit(all_X, all_y)
pred = model.predict(last_X_test)
sub['Survived'] = pred
sub.to_csv("four_lgreg_sub.csv", index=False)
```

```
C:₩Users₩WJ₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762: Conve
rgenceWarning: 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(
```

In [33]:

```
import os
files = os.listdir()
print("파일 유무 확인 : ", "four_lgreg_sub.csv" in files)  # 0.75837
```

```
파일 유무 확인 :  True
```

# 실습

- LogisticRegression 또는 DecisionTree 모델을 선택하여, 파라미터 튜닝을 한 후, 최적의 모델을 선택 후, 이를 제출해 보자.