타이타닉 데이터 셋을 살펴보기

학습 내용

- 타이타닉 데이터 셋을 살펴보며, 데이터 전처리에 대해서 알아본다.

In [ ]:
```
import pandas as pd

print("pandas 버전 ", pd.__version__)
```

```
pandas 버전  1.1.3
```

In [ ]:
```
test = pd.read_csv("../data/titanic/test.csv")
train = pd.read_csv("../data/titanic/train.csv")
all_df = [train, test]

test.shape, train.shape
```

Out[ ]:
```
((418, 11), (891, 12))
```

데이터 셋 요약값 살펴보기

In [ ]:
```
train.describe()
```

Out[ ]:

	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

```
'b'       boolean
'i'       (signed) integer
'u'       unsigned integer
'f'       floating-point
'c'       complex-floating point
'O'       (Python) objects
'S', 'a'  (byte-)string
'U'       Unicode
'V'       raw data (void)
```

In [ ]:
```
train.describe(include=['O'])
```

Out[ ]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3

	Name	Sex	Ticket	Cabin	Embarked
top	Doling, Miss. Elsie	male	1601	G6	S
freq	1	577	7	4	644

Group별 통계에 대해 알아보자.

- groupby()

Pclass별 Survived를 알아보자.

In [ ]:
```
train[ ['Pclass', 'Survived']].groupby(['Pclass']).count()
```

Out[ ]:

	Survived
Pclass	
1	216
2	184
3	491

Pclass별 몇 %나 생존했을까?

In [ ]:
```
train[ ['Pclass', 'Survived']].groupby(['Pclass']).mean()
```

Out[ ]:

	Survived
Pclass	
1	0.629630
2	0.472826
3	0.242363

(실습) 남성, 여성의 생존 비율을 알아보자

In [ ]:
```
train[["Sex", "Survived"]].groupby(['Sex']).mean()
# train[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean()
```

Out[ ]:

	Survived
Sex	
female	0.742038
male	0.188908

값의 정렬에 대해 알아보기

- sort_values()

sort_values를 이용하여 값을 정렬시키기

- 그렇다면 SibSp에 대한 생존 비율은 어떻게 될까?

In [ ]:
```
tmp_val = train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mea
```

```
tmp_val.sort_values(by='Survived', ascending=False)
```

Out[ ]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

[실습] Parch 항목에 대해 sort_values로 활용하여 생존율을 정렬해 보자.

In [ ]:
```
tmp_val = train[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mea
tmp_val.sort_values(by='Survived', ascending=False)
```

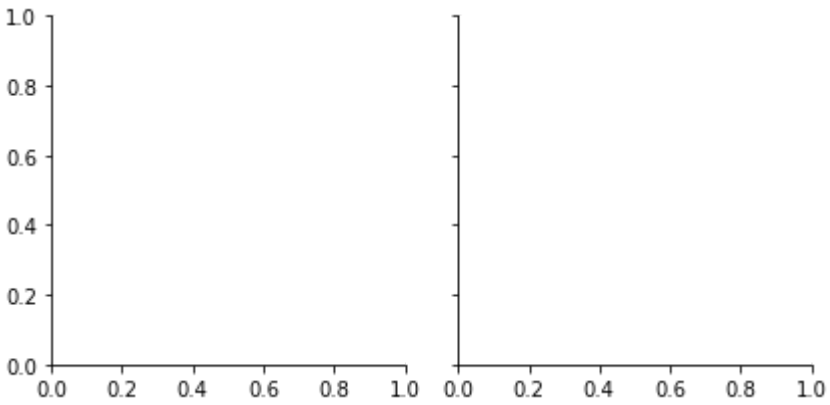
Out[ ]:

	Parch	Survived
3	3	0.600000
1	1	0.550847
2	2	0.500000
0	0	0.343658
5	5	0.200000
4	4	0.000000
6	6	0.000000

In [ ]:
```
import seaborn as sns
```

In [ ]:
```
g = sns.FacetGrid(train, col='Survived')
type(g)
```

Out[ ]:
```
seaborn.axisgrid.FacetGrid
```



- http://seaborn.pydata.org/generated/seaborn.FacetGrid.map.html#seaborn.FacetGrid.map

FacetGrid.map()

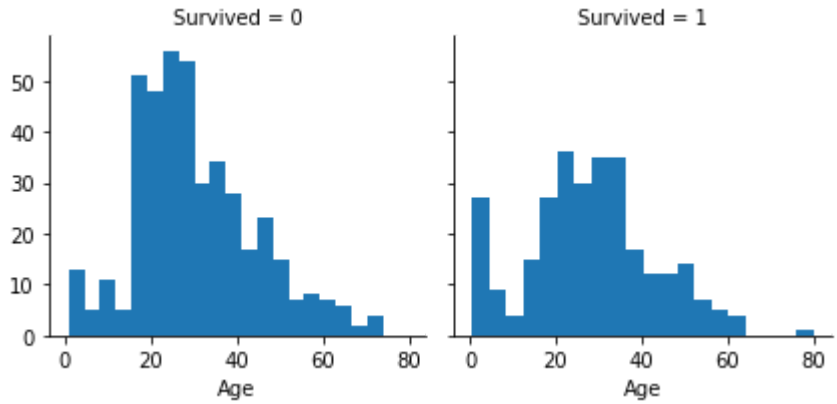
- FacetGrid.map(self, func, *args, **kwargs)

- func : A plotting function
- *args : 데이터의 열이름
- **kwargs : 전달되는 인수

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
g.map(plt.hist, 'Age', bins=20)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fdae13a71f0>
```

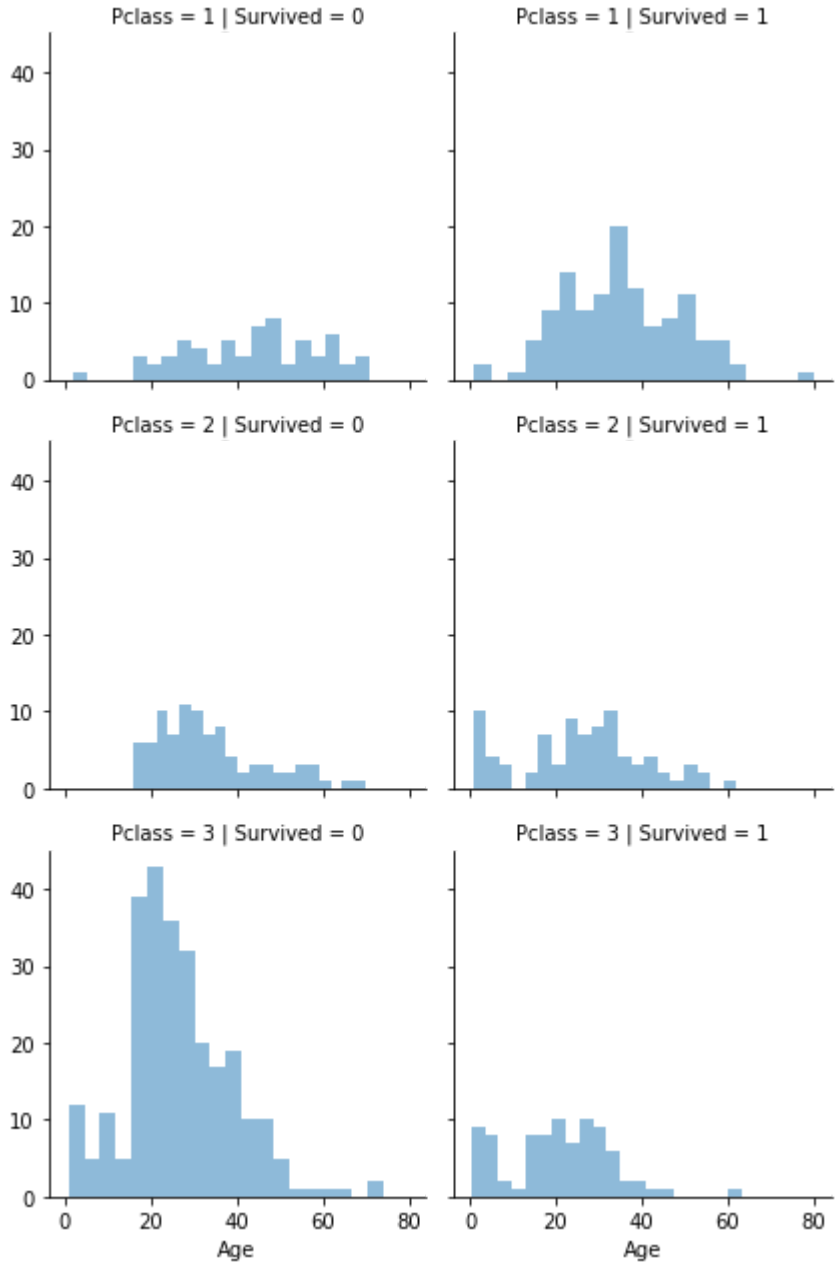


Pclass별 생존에 대해서 나이대별 분포를 살펴보자.

In [ ]:
```
plt.figure(figsize=(15,15))
grid = sns.FacetGrid(train, col='Survived', row='Pclass')
grid.map(plt.hist, 'Age', alpha=0.5, bins=20)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fdae4132070>
```

```
<Figure size 1080x1080 with 0 Axes>
```

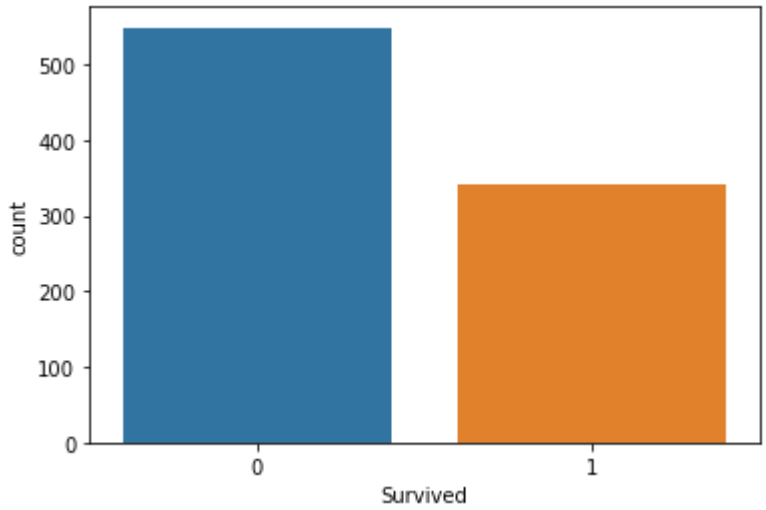


- Pclass=3의 젊은 사람들이 많이 사망하였다.
- Pclass=1의 상대적으로 많은 사람들이 생존했다.

Pclass별 생존시에 어떤 항구에 탄 사람들이 많이 생존했을까?

In [ ]:
```
sns.countplot(x='Survived', data=train)
```

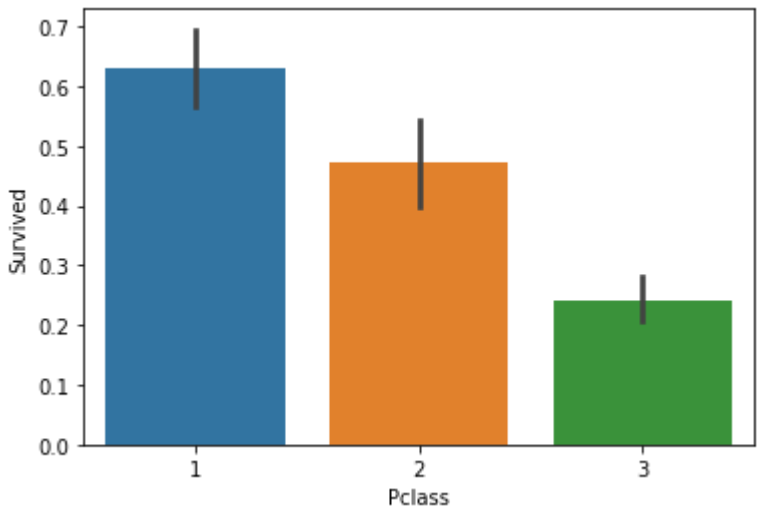
Out[ ]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [ ]:
```
sns.barplot('Pclass', 'Survived', data=train)
```

```
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

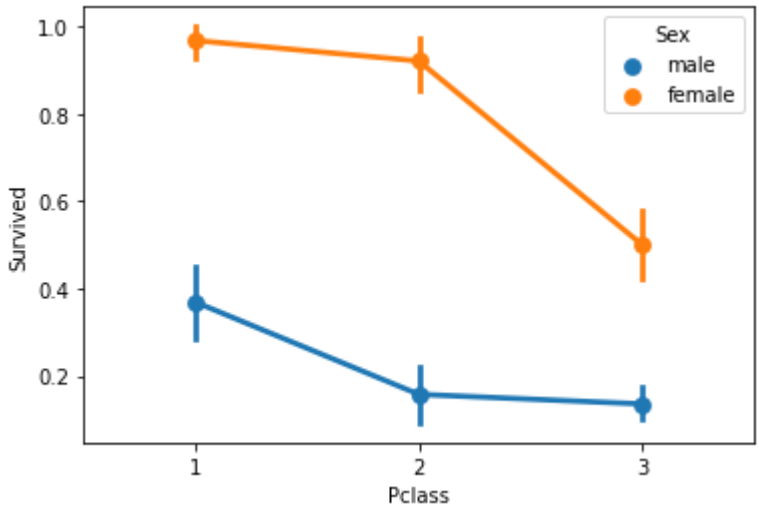
Out[ ]: <AxesSubplot:xlabel='Pclass', ylabel='Survived'>



In [ ]:
```
sns.pointplot('Pclass', 'Survived', hue='Sex', data=train)
```

```
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

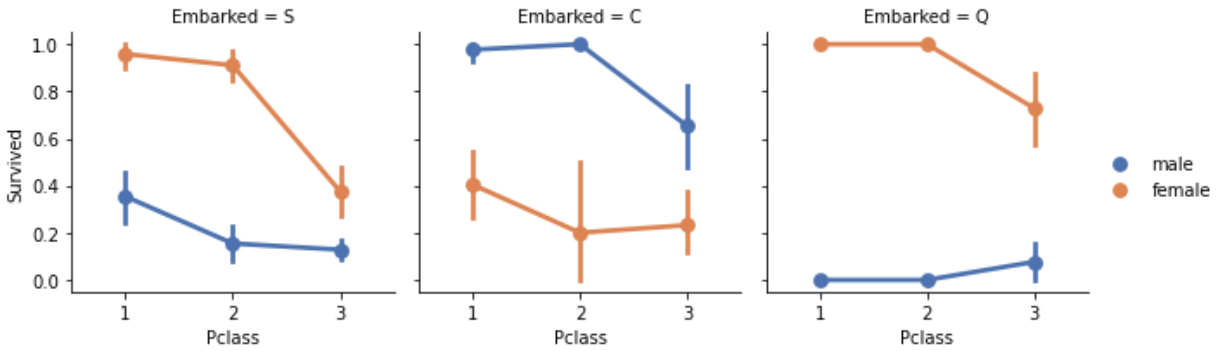
Out[ ]: <AxesSubplot:xlabel='Pclass', ylabel='Survived'>



In [ ]:
```
grid = sns.FacetGrid(train, col='Embarked')
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

```
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/seaborn/axisgrid.p
y:645: UserWarning: Using the pointplot function without specifying `order` is
likely to produce an incorrect plot.
  warnings.warn(warning)
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/seaborn/axisgrid.p
y:650: UserWarning: Using the pointplot function without specifying `hue_order
` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fdae48a1130>
```



In [ ]:
```
print("Before", train.shape, test.shape, all_df[0].shape, all_df[1].shape)
```

```
Before (891, 12) (418, 11) (891, 12) (418, 11)
```

데이터 전처리

- Ticket, Cabin 열 삭제하기 - drop()

In [ ]:
```
train_df = train.drop(['Ticket', 'Cabin'], axis=1)
test_df = test.drop(['Ticket', 'Cabin'], axis=1)
all_df = [train_df, test_df]
```

In [ ]:
```
print("After", train_df.shape, test_df.shape, all_df[0].shape, all_df[1].shape
```

```
After (891, 10) (418, 9) (891, 10) (418, 9)
```

이름에서 중요정보를 뽑기

In [ ]:
```
all_df[0].Name
```

Out[ ]:
```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                              ...
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

crosstab() 함수에 대해 알아보기

- crosstab() 함수는 특정 데이터 그룹에 대한 빈도수를 나타내는 표시해 준다.

- 정규 표현식 ()
  - ()안에 넣으면 그룹화가 된다.

In [ ]:
```
for dataset in all_df:
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=Fals

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

Out[ ]:

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

- Miss와 Mr가 많은 수를 차지한다.

이 데이터를 5개의 그룹으로 변경하자

In [ ]:
```
for dataset in all_df:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', '
        'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')

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

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

Out[ ]:

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

타이틀을 수치로 변경하자.

In [ ]:
```
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for dataset in all_df:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

train_df.head()
```

Out[ ]:

	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

Name 컬럼과 PassengerId를 없애기

In [ ]:
```
train_df = train_df.drop(['Name', 'PassengerId'], axis=1)
test_df = test_df.drop(['Name'], axis=1)
```

```
all_df = [train_df, test_df]
train_df.shape, test_df.shape
```

Out[ ]:
```
((891, 9), (418, 9))
```

In [ ]:
```
for dataset in all_df:
    dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(in

train_df.head()
```

Out[ ]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	S	1
1	1	1	1	38.0	1	0	71.2833	C	3
2	1	3	1	26.0	0	0	7.9250	S	2
3	1	1	1	35.0	1	0	53.1000	S	3
4	0	3	0	35.0	0	0	8.0500	S	1

In [ ]:
```
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          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
<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
```

Out[ ]:
```
(None, None)
```

In [ ]:
```
print(train_df.isnull().sum())
print(test_df.isnull().sum())
```

```
Survived      0
```

```
Pclass         0
Sex            0
Age          177
SibSp          0
Parch          0
Fare           0
Embarked       2
Title          0
dtype: int64
PassengerId     0
Pclass          0
Sex             0
Age            86
SibSp           0
Parch           0
Fare            1
Embarked        0
Title           0
dtype: int64
```

Age의 결측치 채우기

In [ ]:
```
for dataset in all_df:
    dataset['Age'] = dataset['Age'].fillna(dataset['Age'].mean())
```

결측치 처리 Embarked(승선항)

In [ ]:
```
train_df['Embarked'] = train_df['Embarked'].fillna('S')
```

결측치 처리-Fare(요금)

In [ ]:
```
test_df['Fare'] = test_df['Fare'].fillna(test_df['Fare'].mean())
```

In [ ]:
```
print(train_df.isnull().sum())
print(test_df.isnull().sum())
```

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
Title       0
dtype: int64
PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
Title          0
dtype: int64
```

In [ ]:
```
train_df.head()
```

Out[ ]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	S	1
1	1	1	1	38.0	1	0	71.2833	C	3

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
2	1	3	1	26.0	0	0	7.9250	S	2
3	1	1	1	35.0	1	0	53.1000	S	3
4	0	3	0	35.0	0	0	8.0500	S	1

In [ ]:
```
test_df.head()
```

Out[ ]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	892	3	0	34.5	0	0	7.8292	Q	1
1	893	3	1	47.0	1	0	7.0000	S	3
2	894	2	0	62.0	0	0	9.6875	Q	1
3	895	3	0	27.0	0	0	8.6625	S	1
4	896	3	1	22.0	1	1	12.2875	S	3

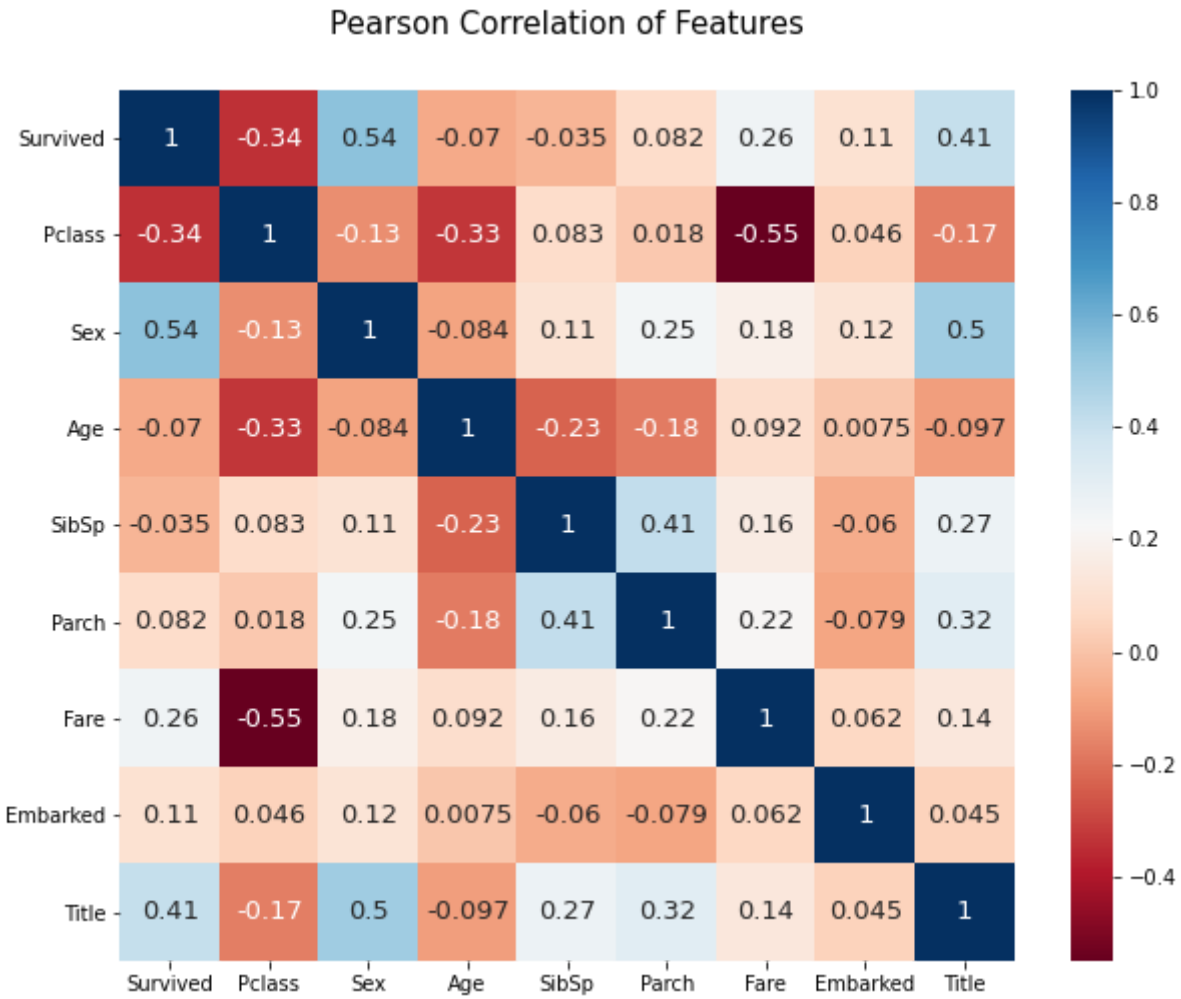
In [ ]:
```
train_df['Embarked'] = train_df['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).a
test_df['Embarked']= test_df['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).asty
```

feature별 상관관계 분석

In [ ]:
```
colormap = plt.cm.RdBu
plt.figure(figsize=(10, 8))
plt.title('Pearson Correlation of Features',
          y=1.05, size=15)

sns.heatmap(train_df.corr(), cmap=colormap, annot=True, annot_kws={"size": 13
```

Out[ ]:
```
<AxesSubplot:title={'center':'Pearson Correlation of Features'}>
```



In [ ]:
```
# 'Name', 'Ticket' =>  문자포함
sel = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked','Title' ]

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

모델 만들고 제출

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
decisiontree = DecisionTreeClassifier()
decisiontree.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeClassifier()
```

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

Out[ ]:
```
array([0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1])
```

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

Ref

- 정규 표현식 : https://nachwon.github.io/regular-expressions/