타이타닉 분석 (Python)

In [2]:

```
# 데이터분석
import pandas as pd
import numpy as np
import random as rnd

# 시각화
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# 머신러닝
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

1.데이터 입력

Pandas DataFrames에 train 와 test 데이터셋을 입력 후 합친다

In [3]:

```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
combine = [train_df, test_df]
```

2.데이터 분석 와 수집

In [4]:

```
print(train_df.columns.values)
```

```
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

In [5]:

```
# 데이터 탐색
train_df.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8

범주형 데이터: Survived, Sex, Embarked

서수형 데이터: Pclass

연속형 데이터: Age, Fare. Discrete: SibSp, Parch

혼합형 데이터: Ticket(글자+숫자)

문자형 데이터: Cabin

복합형 데이터: Name(에러, 오타, 숫자, 타이틀, 이름 포함)

In [6]:

```
train_df.tail()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

Cabin > Age > Embarked: train 데이터셋에 null 값을 가지고 있다.

Cabin > Age: test 데이터셋이 완전하지 않다.

integer, float 컬럼: 7개 (6개, test 데이터셋)

string 컬럼: 5개

In [7]:

```
train_df.info()
print('_'*40)
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
________________________________________
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

숫자 데이터

샘플 데이터: 891 (실제 탑승 승객(2,224)의 40%에 해당한다)

Survived 컬럼: 0 or 1 값을 갖는다.

38% 샘플 survived 값은 실제 생존비율 32% 에 해당한다.

승객 중 75% 이상은 부모 또는 아이없이 혼자 여행을 했다.

승객 중 30% 는 형제자매와 함께 했다.

Fare는 1% 미만이 $512 이상을 지불한 고객이다.

노령의 고객(65~80)은 1% 미만이다.

In [8]:

```
train_df.describe()
#train_df.describe(percentiles=[.61, .62]) - Survived 률
#train_df.describe(percentiles=[.75, .8]) - Parch 분포
#train_df.describe(percentiles=[.68, .69]) - SibSp 분포
#train_df.describe(percentiles=[.1, .2, .3, .4, .5, .6, .7, .8, .9, .99]) - Age 와 Fare
```

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.0
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.9
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512

문자 데이터

Name: 데이터셋에서 unique 값을 갖는다. (count=unique=891)

Sex: 65% 가 남성. (top=male, freq=577/count=891).

Cabin: 중복 값을 갖는다. 여러명이 같은 cabin을 사용했다.

Embarked: 세개의 값을 갖는다. S항이 제일 많다.

Ticket: 중복 값을 갖는다. 22% (unique=681).

In [9]:

```
train_df.describe(include=['O'])
```

Out[9]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Davidson, Mr. Thornton	male	1601	G6	S
freq	1	577	7	4	644

데이터 분석을 통한 예측

상관관계:

Survival 컬럼과 관련 있는 컬럼을 찾아 관련 모델을 생성한다.

완료:

Age 컬럼과 Survival 컬럼 관계 찾기

Embarked 컬럼과 Survival 컬럼 관계 찾기

수정:

Ticket 컬럼은 높은 중복율(22%)을 갖고 있어 Survival 컬럼과 관련이 없어 분석데이터에서 삭제하기

Cabin 컬럼은 train 와 test 데이터에 null값을 많고 불완전해 분석데이터에서 삭제하기

PassengerId 컬럼은 Survival과 관련이 없어 분석데이터에서 삭제하기

Name 컬럼은 값들이 표준화 되있지 않다. Survival 컬럼에 직접적으로 관련이 없어 삭제하기

생성:

탑승한 가족수 Parch 와 SibSp 컬럼을 기반으로 새로운 컬럼 생성하기

Name 컬럼에서 Title 을 축출하기 위하 새로운 컬럼 생성하기

Age 구역에 대한 새로운 컬럼 생성하기

Fare 범위를 나타내는 새로운 컬럼 생성하기

분류:

여성(Sex=femal) 은 대부분 생존했다.

아이들(Age<?) 은 대부분 생존했다.

상위-클라스 탑승객(Pclass=1)은 대부분 생존했다.

피벗 기능을 통한 분석

Pclass 컬럼: Pclass=1 와 Survived 컬럼와의 관계가 (>0.5) 이상이다. 모델에 컬럼을 추가한다.

Sex 컬럼: Sex=female은 74% 의 높은 생존률을 가지고 있다.

SipSp 와 Parch 컬럼: 컬럼값중에는 관계가 0으로 나오는 것들이 있어 각 컬럼을 합쳐 하나의 컬럼으로 만들어 이점을 끌어낸다.

In [10]:

```
train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[10]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [11]:

```
train_df[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[11]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

In [12]:

```
train_df[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[12]:

	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

In [13]:

```
train_df[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[13]:

	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

시각화 데이터 분석

숫자 컬럼와의 관계

관찰:

영유아(Age<=4)는 생존률이 높다.

최고령자(Age=80)가 생존했다.

상당수의 15~25살은 생존하지 못했다.

탑승객 대부분은 15~35살이다.

판단:

모델에 Age컬럼 추가를 고려해야 한다.

Age 컬럼에 null 값을 채운다.

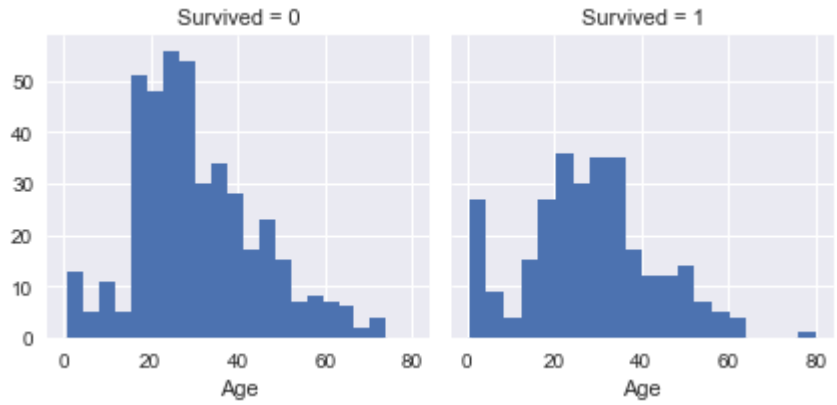
Age 컬럼을 나이로 그룹짓는다.

In [14]:

```
g = sns.FacetGrid(train_df, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[14]:

<seaborn.axisgrid.FacetGrid at 0x26fe14a6780>



서수 컬럼와의 관계

관찰:

Pclass=3 은 대부분 생존하지 못 했다.

Pclass=2 와 Pclass=3 의 영유아 탑승객은 대부분 생존했다.

Pclass=1 탑승객은 대부분 생존했다.

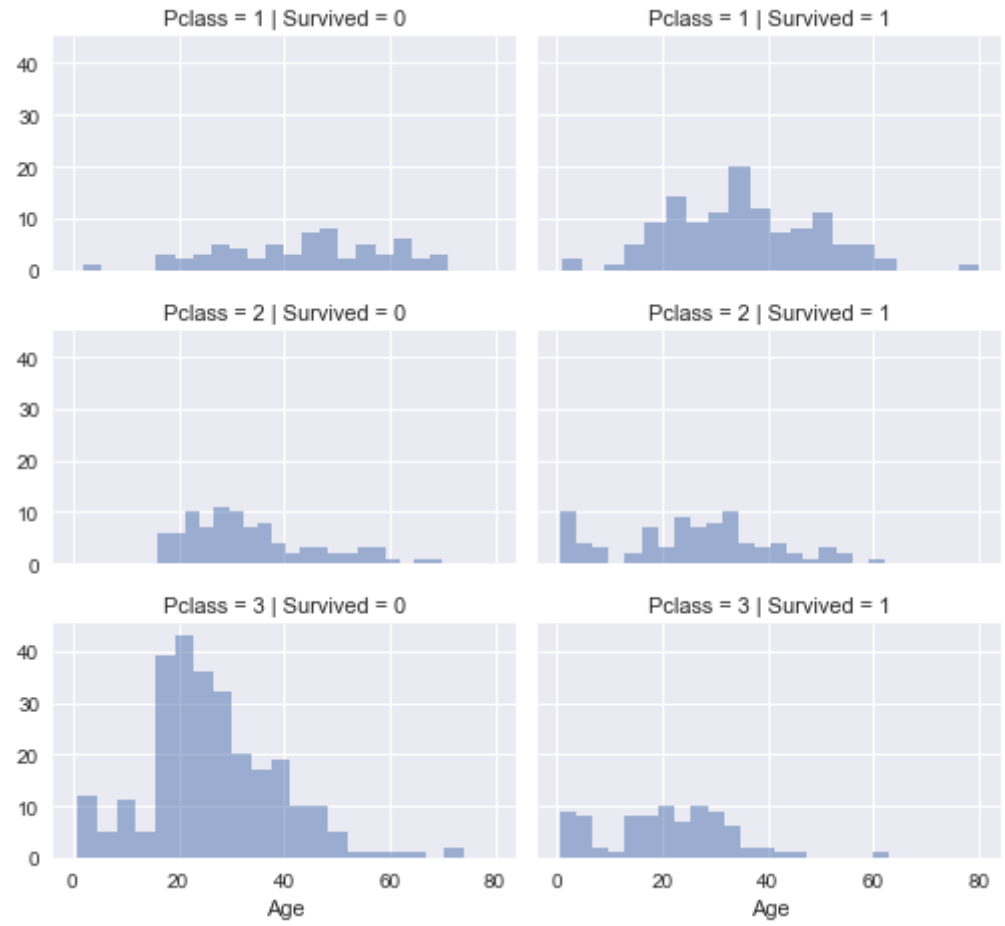
Pclass을 탑승객 나이 분포로 변경한다.

판단:

모델에 Pclass 컬럼 추가를 고려한다.

In [15]:

```
#grid = sns.FacetGrid(train_df, col='Pclass', hue='Survived')
grid = sns.FacetGrid(train_df, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```



범주형 컬럼와의 관계

관찰:

Female 탑승객은 male 보다 생존률이 더 높다.

Embarked=C를 제외하고 male 탑승객의 생존률이 높다. Pclass 와 Embarked 와 관련이 있어 보이고 Pclass 와 Survived 도 관련이 있어 보인다.

C 와 Q 항구에 대한 Plcass=2에 비해 Pclass=3의 male 탑승객은 비교적 생존률이 높다.

승선항구를 male 탑승객 와 Pclass=3에 대한 생존률로 바꾼다.

판단:

모델에 Sex 컬럼 추가한다.

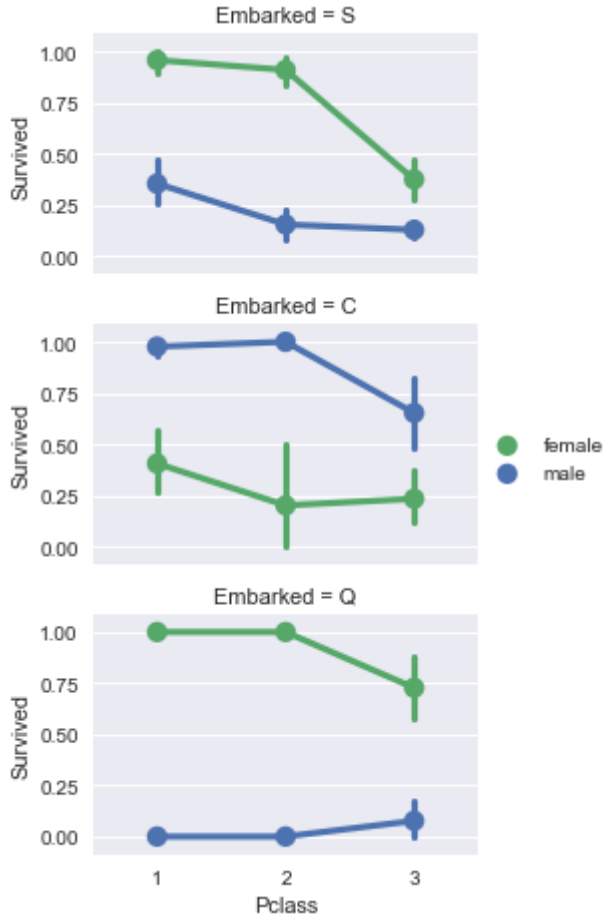
모델에 Embarked 컬럼을 추가 한다.

In [16]:

```
# grid = sns.FacetGrid(train_df, col='Embarked')
grid = sns.FacetGrid(train_df, row='Embarked', size=2.2, aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

Out[16]:

<seaborn.axisgrid.FacetGrid at 0x26fe68c1a58>



범주형 와 숫자형 컬럼 관계

Embarked(범주형), Sex(범주형), Fare(숫자형), Survived(범주형 숫자)

관찰:

더 많이 지불한 Fare 탑승객이 더 많이 생존했다.

승선 항구 와 생존률은 관계가 있다.

판단:

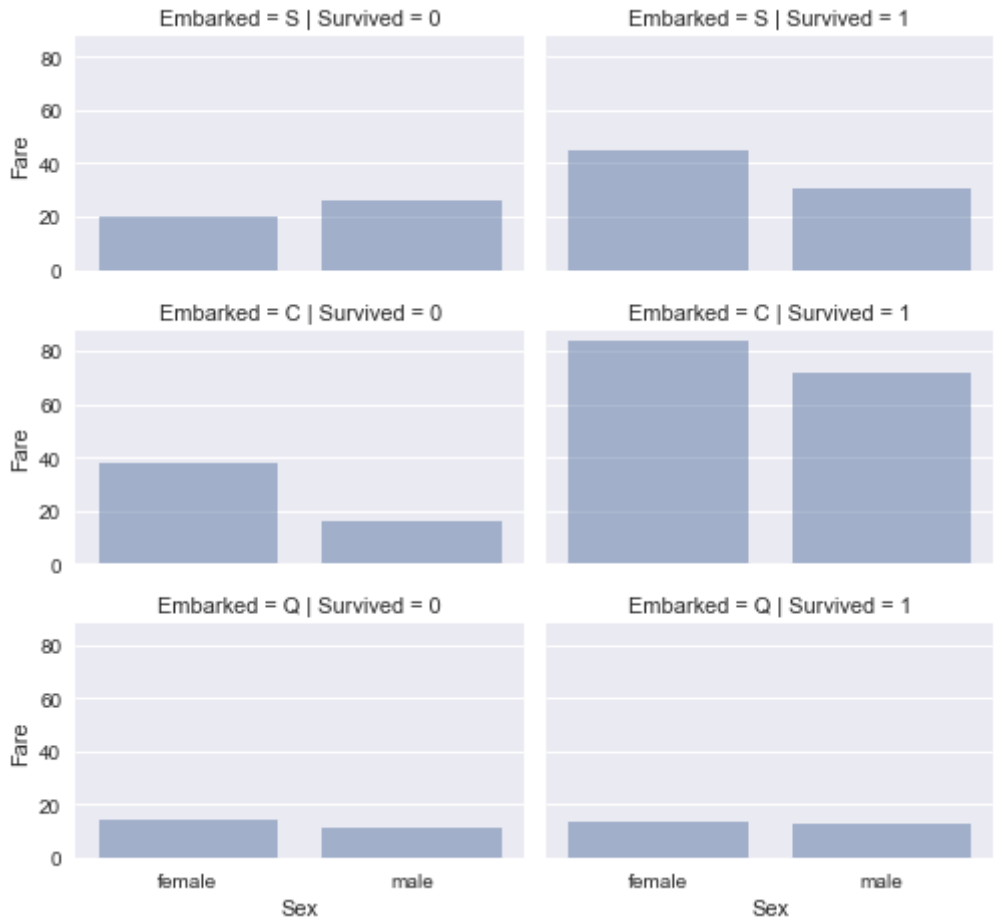
Fare 컬럼 구분을 고려한다.

In [17]:

```
#grid = sns.FacetGrid(train_df, col='Embarked', hue='Survived', palette={0: 'k', 1: 'w'})
grid = sns.FacetGrid(train_df, row='Embarked', col='Survived', size=2.2, aspect=1.6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
```

Out[17]:

<seaborn.axisgrid.FacetGrid at 0x26fe6feb240>



데이터 수집

삭제 컬럼 관계

관련 없는 데이터를 삭제해 분석을 쉽게하고 분석 속도를 높인다.

위 예측과 판단을 기반으로 Cabin 와 Ticket 컬럼을 삭제한다.

In [18]:

```
print("Before", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape)

train_df = train_df.drop(['Ticket', 'Cabin'], axis=1)
test_df = test_df.drop(['Ticket', 'Cabin'], axis=1)
combine = [train_df, test_df]

"After", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape
```

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

Out[18]:

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

기본 컬럼에서 추출해 새로운 컬럼 생성

Name 와 PassengerId 컬럼을 삭제하기 전에 title 와 survival 사이의 관계를 확인한다.

Title 컬럼은 정규표현식을 사용한다.

관찰:

Title은 Age 그룹을 정확힌 구분한다. 예: Master는 Age 5살을 의미한다.

Title Age 그룹간의 생존은 살짝 바꾼다.

특정 title은 대부분 생존한다. (Mme,Lady,Sir) 또는 대부분 생존하지 않는다.(Don,Rev,Jonkheer)

판단:

모델에 새로운 Title 컬럼을 추가하기로 결정한다.

In [20]:

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

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

Out[20]:

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

여러 title을 일반 이름으로 변경하거나 Rare로 분류한다.

In [21]:

```
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col',₩
        '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[21]:

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

범주형 title을 서수형으로 변경한다.

In [22]:

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

train_df.head()
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Em
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

이제 Name 컬럼은 train 와 test 데이터셋에서 삭제한다.

train 데이터셋에 PassengerId 컬럼을 삭제한다.

In [23]:

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

Out[23]:

```
((891, 9), (418, 9))
```

범주형 컬럼 변경

문자에 숫자의미를 값는 컬럼을 변경한다. 모델 알고리즘에 필요하다.

Sex 컬럼을 female=1 와 male=0을 갖는 Gender 컬럼으로 변경한다.

In [24]:

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

train_df.head()
```

Out[24]:

	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

숫자형 연속 컬럼을 완성

misssing 값이 이나 null 값을 갖는 컬럼을 보완한다.

숫자형 연속 컬럼을 완성하기 위해 3가지 방법을 고려한다.

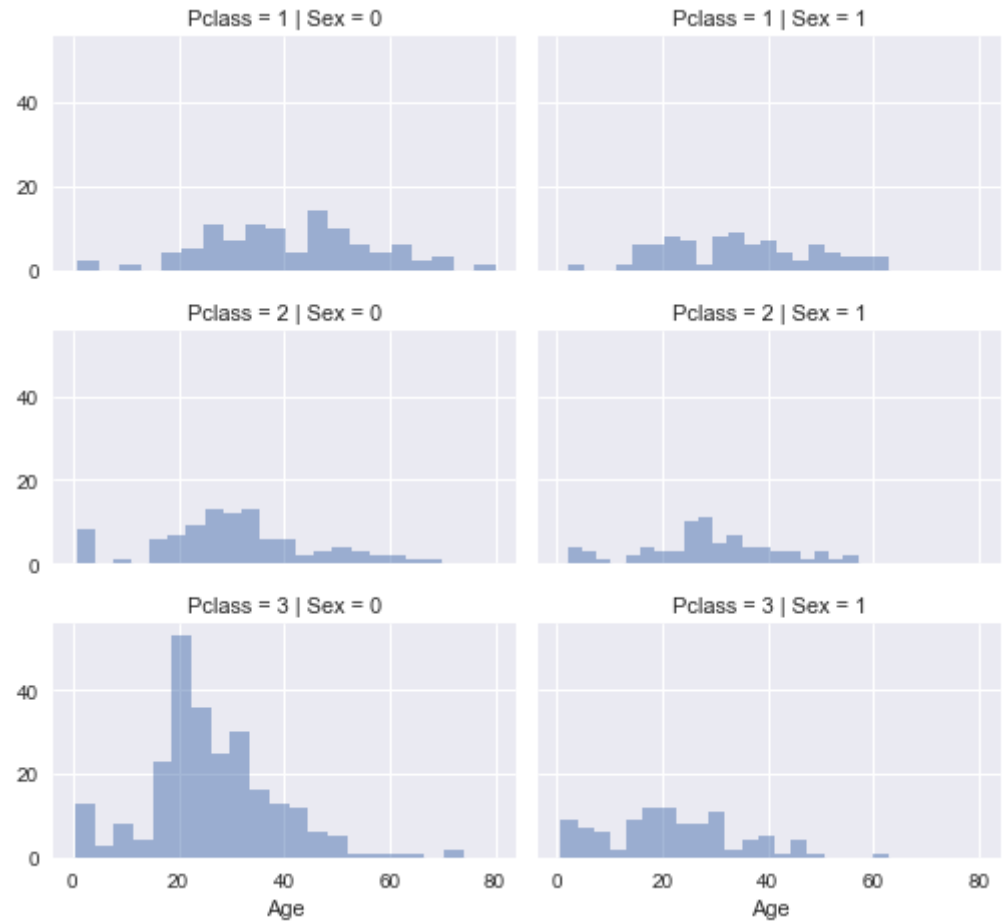
1. 평균 와 표준편차 사이의 랜덤 숫자를 생성한다.
2. missing 값 추측의 정확한 방법은 다른 관련 컬럼을 사용하는 것이다. Age, Gender 와 Pclass 사이의 관계를 주의한다. Pclass 와 Gender 컬럼의 합에서 Age에 대한 값으로 중앙값을 이용해 예측한다. Pclass=1 와 Gender=0은 중앙값 Age, 등등..
3. 방법 1와2를 합친다. 중앙값으로 age값을 예측하는거 대신, 평균 와 표준편차 간의 random 숫자를 사용한다. Pclass 와 Gender 합을 기반으로 삼는다.

In [26]:

```
# grid = sns.FacetGrid(train_df, col='Pclass', hue='Gender')
grid = sns.FacetGrid(train_df, row='Pclass', col='Sex', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
```

Out[26]:

<seaborn.axisgrid.FacetGrid at 0x26fe7391a20>



Pclass x Gender 합을 기반으로 예측된 Age 값을 얻기 위해 빈 배열을 준비한다.

In [27]:

```
guess_ages = np.zeros((2,3))
guess_ages
```

Out[27]:

```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

Sex(0 or 1) 와 Pclass (1,2,3)을 이용해 Age 값을 예측하기 위해 반복실행한다.

In [28]:

```
for dataset in combine:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & ₩
                               (dataset['Pclass'] == j+1)]['Age'].dropna()

            # age_mean = guess_df.mean()
            # age_std = guess_df.std()
            # age_guess = rnd.uniform(age_mean - age_std, age_mean + age_std)

            age_guess = guess_df.median()

            # Convert random age float to nearest .5 age
            guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

    for i in range(0, 2):
        for j in range(0, 3):
            dataset.loc[ (dataset.Age.isnull()) & (dataset.Sex == i) & (dataset.Pclass == j+1),₩
                    'Age'] = guess_ages[i,j]

    dataset['Age'] = dataset['Age'].astype(int)

train_df.head()
```

Out[28]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22	1	0	7.2500	S	1
1	1	1	1	38	1	0	71.2833	C	3
2	1	3	1	26	0	0	7.9250	S	2
3	1	1	1	35	1	0	53.1000	S	3
4	0	3	0	35	0	0	8.0500	S	1

Suvived와 관계를 결정하고 Age 구분을 생성한다.

In [29]:

```
train_df['AgeBand'] = pd.cut(train_df['Age'], 5)
train_df[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=True)
```

Out[29]:

	AgeBand	Survived
0	(-0.08, 16.0]	0.550000
1	(16.0, 32.0]	0.337374
2	(32.0, 48.0]	0.412037
3	(48.0, 64.0]	0.434783
4	(64.0, 80.0]	0.090909

이들 구분을 기반으로 Age를 서수형으로 바꾼다.

In [30]:

```
for dataset in combine:
    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']
train_df.head()
```

Out[30]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title	AgeBand
0	0	3	0	1	1	0	7.2500	S	1	(16.0, 32.0]
1	1	1	1	2	1	0	71.2833	C	3	(32.0, 48.0]
2	1	3	1	1	0	0	7.9250	S	2	(16.0, 32.0]
3	1	1	1	2	1	0	53.1000	S	3	(32.0, 48.0]
4	0	3	0	2	0	0	8.0500	S	1	(32.0, 48.0]

AgeBand 컬럼을 삭제한다.

In [31]:

```
train_df = train_df.drop(['AgeBand'], axis=1)
combine = [train_df, test_df]
train_df.head()
```

Out[31]:

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

기본 컬럼과 합쳐 새로운 컬럼을 만든다.

Parch 와 SibSp 컬럼을 합쳐 FamilySize란 새로운 컬럼을 생성한다.

데이터셋에서 Parch 와 SibSp를 삭제한다.

In [32]:

```
for dataset in combine:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1

train_df[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean().sort_values(
by='Survived', ascending=False)
```

Out[32]:

	FamilySize	Survived
3	4	0.724138
2	3	0.578431
1	2	0.552795
6	7	0.333333
0	1	0.303538
4	5	0.200000
5	6	0.136364
7	8	0.000000
8	11	0.000000

IsAlone 라는 새로운 컬럼을 생성한다.

In [33]:

```
for dataset in combine:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1

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

Out[33]:

	IsAlone	Survived
0	0	0.505650
1	1	0.303538

Parch, SibSp 와 FamilySize 컬럼을 삭제하고 IsAlone만 남긴다.

In [34]:

```
train_df = train_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
test_df = test_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
combine = [train_df, test_df]

train_df.head()
```

Out[34]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone
0	0	3	0	1	7.2500	S	1	0
1	1	1	1	2	71.2833	C	3	0
2	1	3	1	1	7.9250	S	2	1
3	1	1	1	2	53.1000	S	3	0
4	0	3	0	2	8.0500	S	1	1

Pclass 와 Age 컬럼을 합친 가변컬럼을 생성한다.

In [35]:

```
for dataset in combine:
    dataset['Age*Class'] = dataset.Age * dataset.Pclass

train_df.loc[:, ['Age*Class', 'Age', 'Pclass']].head(10)
```

Out[35]:

	Age*Class	Age	Pclass
0	3	1	3
1	2	2	1
2	3	1	3
3	2	2	1
4	6	2	3
5	3	1	3
6	3	3	1
7	0	0	3
8	3	1	3
9	0	0	2

범주형 컬럼 완성

승선 항구를 기반으로 하는 Embarked 컬럼은 S,Q,C 값을 갖는다. train 데이터셋은 missing값이 2개 이다. 최빈값을 입력한다.

In [36]:

```
freq_port = train_df.Embarked.dropna().mode()[0]
freq_port
```

Out[36]:

```
'S'
```

In [37]:

```
for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)

train_df[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean().sort_values(by=
'Survived', ascending=False)
```

Out[37]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

범주형 컬럼을 숫자형 컬럼으로 변경

Embarked 컬럼을 새로운 숫자형 항구 컬럼으로 변경한다.

In [38]:

```
for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

train_df.head()
```

Out[38]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	7.2500	0	1	0	3
1	1	1	1	2	71.2833	1	3	0	2
2	1	3	1	1	7.9250	0	2	1	3
3	1	1	1	2	53.1000	0	3	0	2
4	0	3	0	2	8.0500	0	1	1	6

숫자형 컬럼으로 변경

Fare 컬럼을 완성하기 위해 최빈값을 test 데이터셋의 missing값에 입력한다.

In [39]:

```
test_df['Fare'].fillna(test_df['Fare'].dropna().median(), inplace=True)
test_df.head()
```

Out[39]:

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	892	3	0	2	7.8292	2	1	1	6
1	893	3	1	2	7.0000	0	3	0	6
2	894	2	0	3	9.6875	2	1	1	6
3	895	3	0	1	8.6625	0	1	1	3
4	896	3	1	1	12.2875	0	3	0	3

FareBand를 생성한다.

In [40]:

```
train_df['FareBand'] = pd.qcut(train_df['Fare'], 4)
train_df[['FareBand', 'Survived']].groupby(['FareBand'], as_index=False).mean().sort_values(by=
'FareBand', ascending=True)
```

Out[40]:

	FareBand	Survived
0	(-0.001, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31.0]	0.454955
3	(31.0, 512.329]	0.581081

FareBand 을 기반으로 Fare 컬럼을 서수형 값으로 변경한다.

In [41]:

```
for dataset in combine:
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare'] = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   = 2
    dataset.loc[ dataset['Fare'] > 31, 'Fare'] = 3
    dataset['Fare'] = dataset['Fare'].astype(int)

train_df = train_df.drop(['FareBand'], axis=1)
combine = [train_df, test_df]

train_df.head(10)
```

Out[41]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	0	0	1	0	3
1	1	1	1	2	3	1	3	0	2
2	1	3	1	1	1	0	2	1	3
3	1	1	1	2	3	0	3	0	2
4	0	3	0	2	1	0	1	1	6
5	0	3	0	1	1	2	1	1	3
6	0	1	0	3	3	0	1	1	3
7	0	3	0	0	2	0	4	0	0
8	1	3	1	1	1	0	3	0	3
9	1	2	1	0	2	1	3	0	0

test 데이터셋도 바꾼다.

In [42]:

```
test_df.head(10)
```

Out[42]:

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	892	3	0	2	0	2	1	1	6
1	893	3	1	2	0	0	3	0	6
2	894	2	0	3	1	2	1	1	6
3	895	3	0	1	1	0	1	1	3
4	896	3	1	1	1	0	3	0	3
5	897	3	0	0	1	0	1	1	0
6	898	3	1	1	0	2	2	1	3
7	899	2	0	1	2	0	1	0	2
8	900	3	1	1	0	1	3	1	3
9	901	3	0	1	2	0	1	0	3

모델 생성 과 예측

여러 예측 모델 알고리즘을 이용해 Survived 값을 예측한다.

알고리즘:

Logistic Regression

KNN or k-Nearest Neighbors

Support Vector Machines

Naive Bayes classifier

Decision Tree

Random Forrest

Perceptron

Artificial neural network

RVM or Relevance Vector Machine

In [43]:

```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test  = test_df.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

Out[43]:

```
((891, 8), (891,), (418, 8))
```

로지스틱 회귀모델을 이용한다.

범주형 종속변수 와 여러개의 독립변수 간의 관계를 측정한다. 일반적인 회귀 분석의 목표와 동일하게 종속 변수와 독립 변수간의 관계를 구체적인 함수로 나타내어 향후 예측 모델에 사용하는 것이다. 이는 독립 변수의 선형 결합으로 종속 변수를 설명한다는 관점에서는 선형 회귀 분석과 유사하다. 하지만 로지스틱 회귀는 선형 회귀 분석과는 다르게 종속 변수가 범주형 데이터를 대상으로 하며 입력 데이터가 주어졌을 때 해당 데이터의 결과가 특정 분류로 나뉘기 때문에 일종의 분류 (classification) 기법으로도 볼 수 있다.

train 데이터셋을 기반으로 만든 모델로 생성한 예측 점수를 주의한다.

In [44]:

```
# 로지스틱 회귀모형 (Logistic Regression)

logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

Out[44]:

```
80.359999999999999
```

Sex 컬럼이 높은 상관계수를 갖는다. Suvivied=1 가능성을 높인다. Pclass 증가에서 Survived=1의 가능성은 낮아진다.

Age * Class 가변컬럼이 Survived 상관계수에 좋은 점수를 갖는 모델을 만든다.

Title 컬럼은 두번째 높은 상관계수를 갖는다.