Kaggle1

목차 및 출처

1. 데이터셋 확인 - 대부분의 캐글 데이터들은 잘 정제되어 있습니다. 하지만 가끔 null data가 존재합니다. 이를 확인하고, 향후 수정합니다.

2. 탐색적 데이터 분석(exploratory data analysis) - 여러 feature 들을 개별적으로 분석하고, feature 들 간의 상관관계를 확인합니다. 여러 시각화 툴을 사용하여 insight를 얻습니다.

- 출처 : https://kaggle-kr.tistory.com/17?category=868316
- https://kaggle-kr.tistory.com/18?category=868316

0. 기본적인 import

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

plt.style.use('seaborn')
sns.set(font_scale=2.5) # 이 두줄은 본 필자가 항상 쓰는 방법입니다. matplotlib 의 기본 scheme 말고 seaborn scheme 을 세팅하고, 일일이 graph 의 font size 를 지정할 필요 없이 seaborn 의 font_scale 을 사용하면 편합니다.
import missingno as msno

#ignore warnings
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

1. 데이터셋 확인(기본정보)

```
<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: 83.7+ KB
```

train.info()

	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

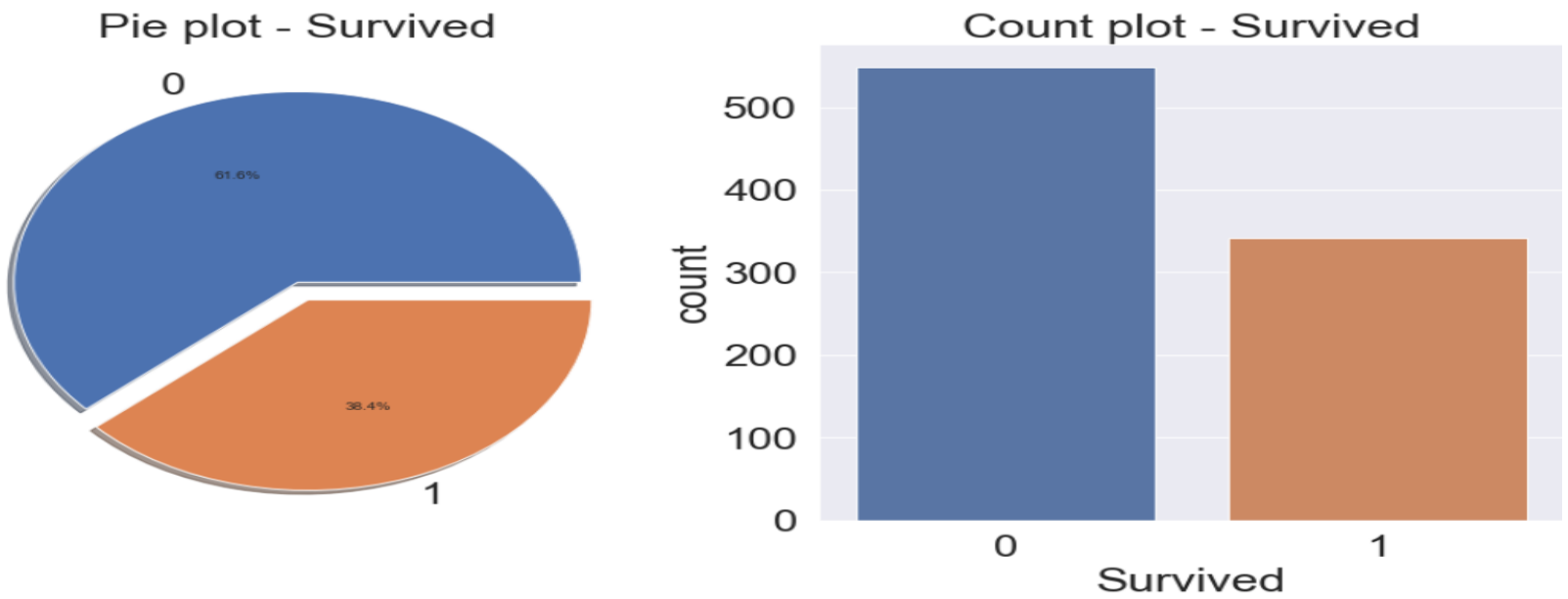
train.describe()

1. 데이터셋 확인(NULL 확인)

```
# NULL check train columns
for col in train.columns:
    msg = 'column: {:>10}₩t Percent of NaN value: {:.2f}%'.format(col, 100 * (train[col].isnull().sum() / train[col].shape[0]))
    print(msg)

# NULL check test columns
for col in test.columns:
    msg = 'column: {:>10}₩t Percent of NaN value: {:.2f}%'.format(col, 100 * (train[col].isnull().sum() / train[col].shape[0]))
    print(msg)
```

```
column: PassengerId         Percent of NaN value: 0.00%
column:    Survived         Percent of NaN value: 0.00%
column:      Pclass         Percent of NaN value: 0.00%
column:        Name         Percent of NaN value: 0.00%
column:         Sex         Percent of NaN value: 0.00%
column:         Age         Percent of NaN value: 19.87%
column:       SibSp         Percent of NaN value: 0.00%
column:       Parch         Percent of NaN value: 0.00%
column:      Ticket         Percent of NaN value: 0.00%
column:        Fare         Percent of NaN value: 0.00%
column:       Cabin         Percent of NaN value: 77.10%
column:    Embarked         Percent of NaN value: 0.22%
column: PassengerId         Percent of NaN value: 0.00%
column:      Pclass         Percent of NaN value: 0.00%
column:        Name         Percent of NaN value: 0.00%
column:         Sex         Percent of NaN value: 0.00%
column:         Age         Percent of NaN value: 19.87%
column:       SibSp         Percent of NaN value: 0.00%
column:       Parch         Percent of NaN value: 0.00%
column:      Ticket         Percent of NaN value: 0.00%
column:        Fare         Percent of NaN value: 0.00%
column:       Cabin         Percent of NaN value: 77.10%
column:    Embarked         Percent of NaN value: 0.22%
```

1. 데이터셋 확인(목표 label)



0(죽은사람), 1(생존한사람)
죽은사람이 38.4%이다.
데이터가 균일함

```
# Check target label
f, ax = plt.subplots(1, 2, figsize=(18, 8))

train['Survived'].value_counts().plot.pie(explode=[0, 0.1], autopct='%1.1f%%', ax=ax[0], shadow=True)
ax[0].set_title('Pie plot - Survived')
ax[0].set_ylabel('')
sns.countplot('Survived', data=train, ax=ax[1])
ax[1].set_title('Count plot - Survived')

plt.show()
```

2. Exploratory data analysis(Pclass)

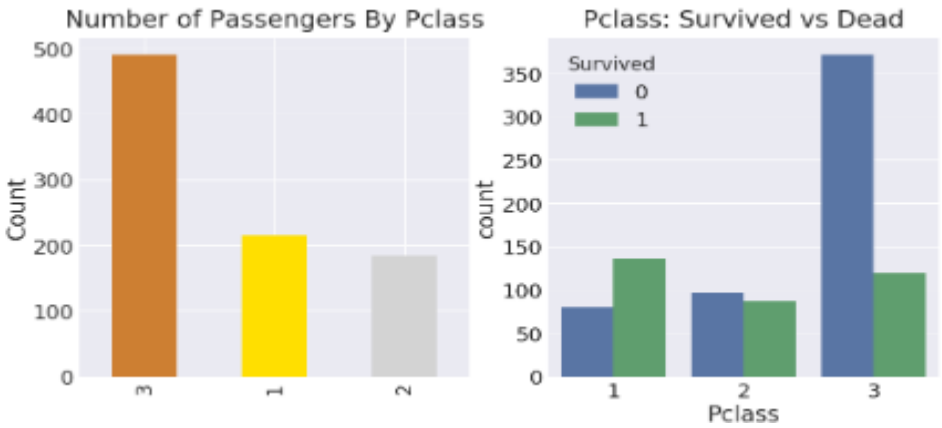
- Pclass와 Survived 관계
  - Pclass에 속한 총 사람 수, 생존한 사람 수

```
# Part2 - exploratory data analysis
# Pclass & Survival
print(train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).count())
print(train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).sum())
pd.crosstab(train['Pclass'], train['Survived'], margins=True).style.background_gradient(cmap='summer_r')
```

```
        Survived
Pclass
1             216
2             184
3             491
        Survived
Pclass
1             136
2              87
3             119
```

Survived	0	1	All
Pclass			
1	80	136	216
2	97	87	184
3	372	119	491
All	549	342	891

```
y_position = 1.02
f, ax = plt.subplots(1, 2, figsize=(18, 8))
train['Pclass'].value_counts().plot.bar(color=['#CD7F32','#FFDF00','#D3D3D3'], ax=ax[0])
ax[0].set_title('Number of Passengers By Pclass', y=y_position)
ax[0].set_ylabel('Count')
sns.countplot('Pclass', hue='Survived', data=train, ax=ax[1])
ax[1].set_title('Pclass: Survived vs Dead', y=y_position)
plt.show()
```

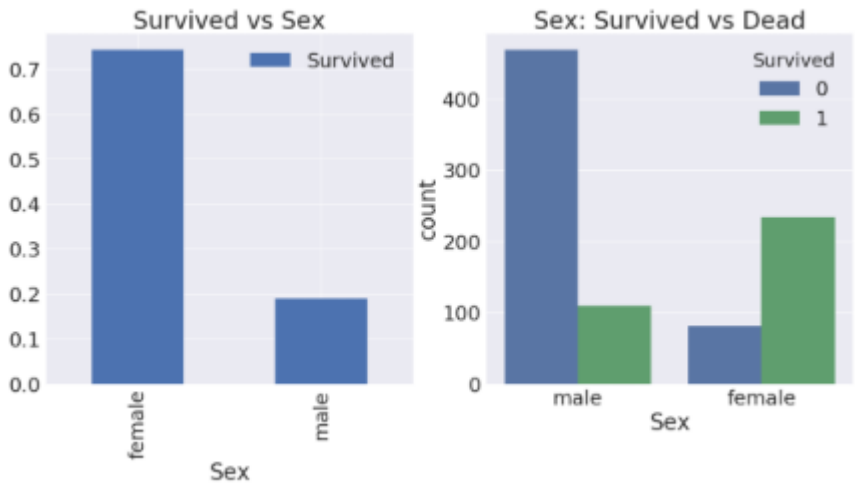


2. Exploratory data analysis(Sex)

```
# Sex & Survival
f, ax = plt.subplots(1, 2, figsize=(18, 8))
train[['Sex', 'Survived']].groupby(['Sex'], as_index=True).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survived vs Sex')
sns.countplot('Sex', hue='Survived', data=train, ax=ax[1])
ax[1].set_title('Sex: Survived vs Dead')
plt.show()

print(train[['Sex', 'Survived']].groupby(['Sex'], as_index=False).mean().sort_values(by='Survived', ascending=False))
pd.crosstab(train['Sex'], train['Survived'], margins=True).style.background_gradient(cmap='summer_r')
```

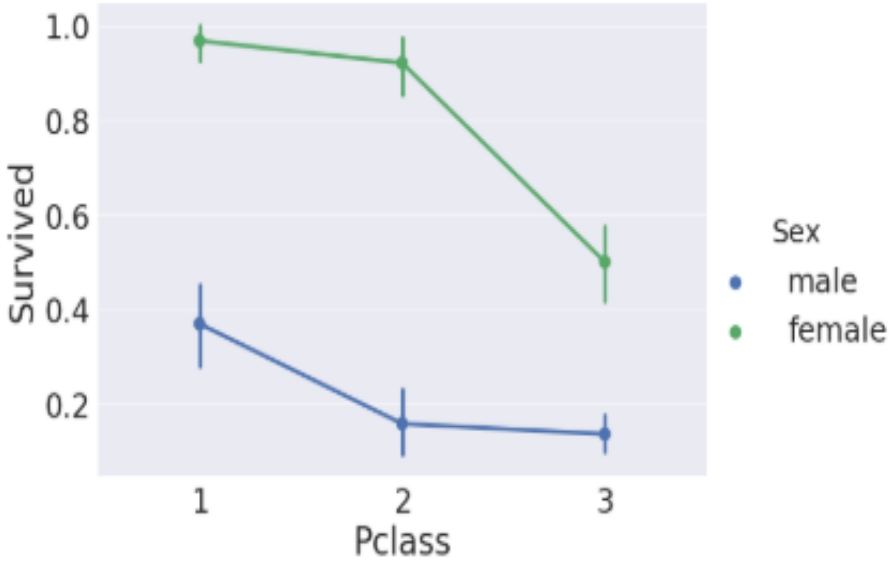
```
       Sex  Survived
0   female  0.742038
1     male  0.188908
```

Survived	0	1	All
Sex			
female	81	233	314
male	468	109	577
All	549	342	891



2. Exploratory data analysis(Sex, Pclass)

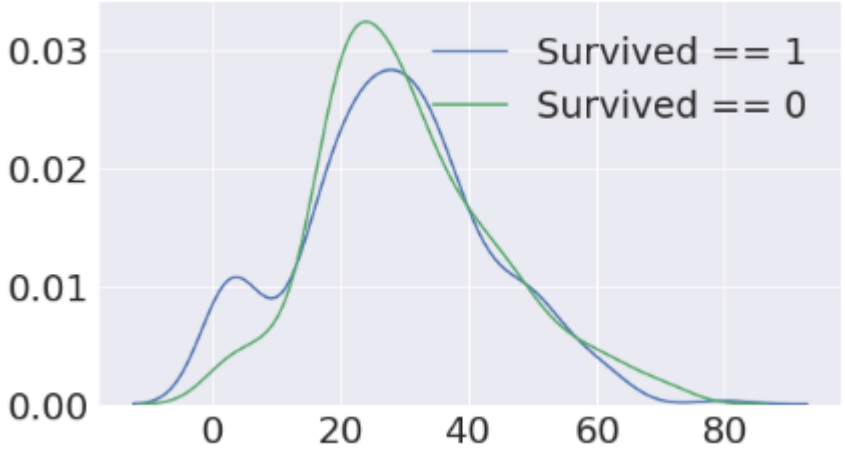
```
# Pclass & Sex & Survival
sns.factorplot('Pclass', 'Survived', hue='Sex', data=train,
               size=6, aspect=1.5)
```

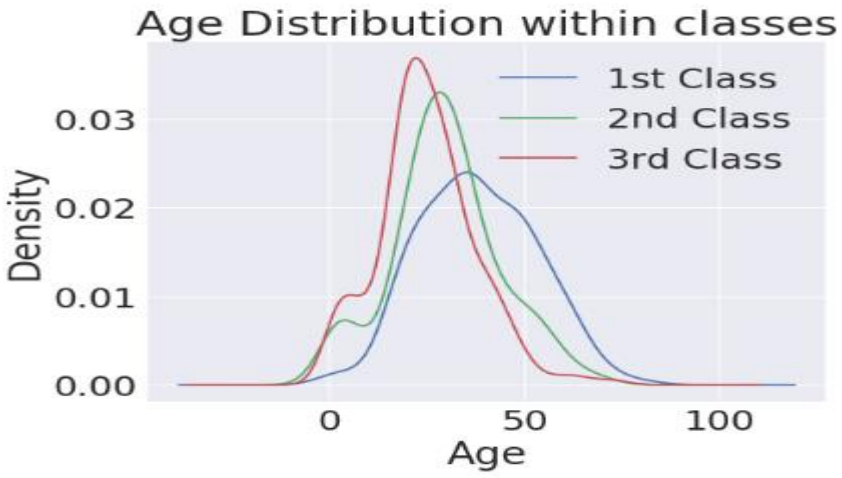


2. Exploratory data analysis(age)

```
# Age
print('oldest : {:.1f} Years'.format(train['Age'].max()))
print('youngest : {:.1f} Years'.format(train['Age'].min()))
print('mean age : {:.1f} Years'.format(train['Age'].mean()))

fig, ax = plt.subplots(1, 1, figsize=(9, 5))
sns.kdeplot(train[df_train['Survived'] == 1]['Age'], ax=ax)
sns.kdeplot(train[df_train['Survived'] == 0]['Age'], ax=ax)
plt.legend(['Survived == 1', 'Survived == 0'])
plt.show()
```

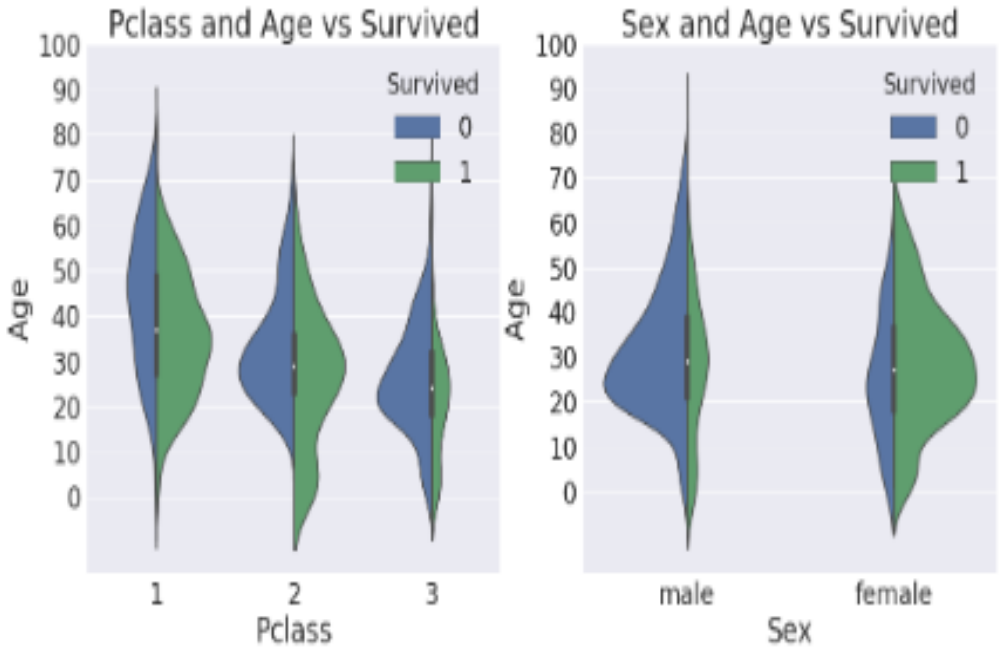
```
# Age distribution withing classes
plt.figure(figsize=(8, 6))
df_train['Age'][train['Pclass'] == 1].plot(kind='kde')
df_train['Age'][train['Pclass'] == 2].plot(kind='kde')
df_train['Age'][train['Pclass'] == 3].plot(kind='kde')

plt.xlabel('Age')
plt.title('Age Distribution within classes')
plt.legend(['1st Class', '2nd Class', '3rd Class'])
```





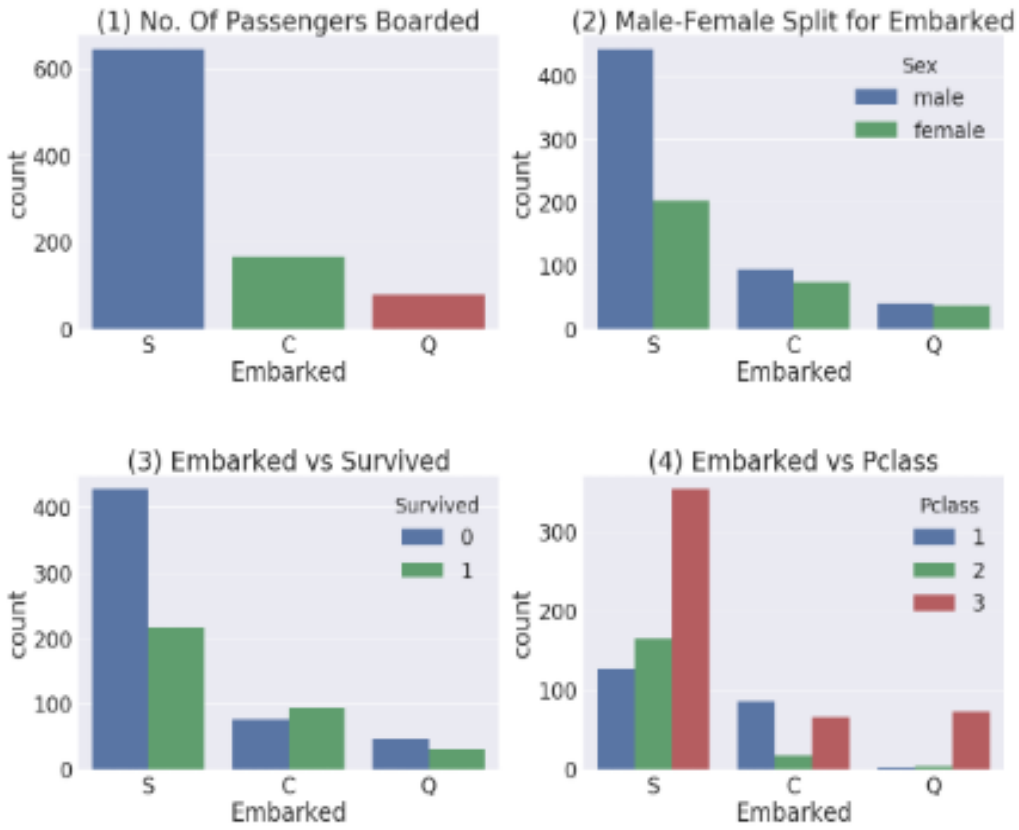
2. Exploratory data analysis(pclass,sex,age)

```
# Pclass & Age & Sex & Survival
f,ax=plt.subplots(1,2,figsize=(18,8))
sns.violinplot("Pclass","Age", hue="Survived", data=train, scale='count', split=True,ax=ax[0])
ax[0].set_title('Pclass and Age vs Survived')
ax[0].set_yticks(range(0,110,10))
sns.violinplot("Sex","Age", hue="Survived", data=train, scale='count', split=True,ax=ax[1])
ax[1].set_title('Sex and Age vs Survived')
ax[1].set_yticks(range(0,110,10))
plt.show()
```



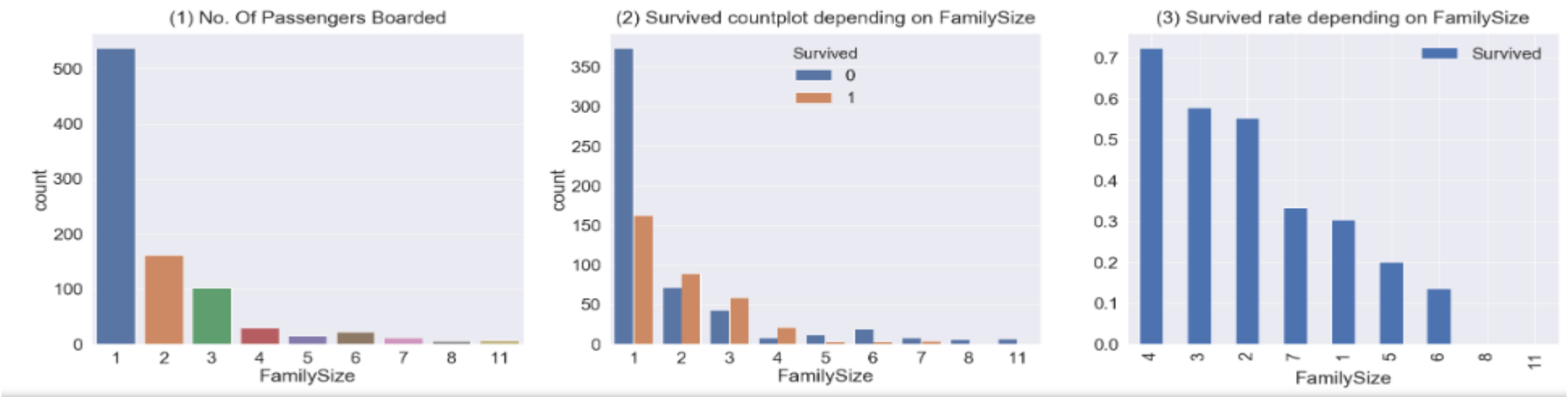
2. Exploratory data analysis(Embarked)

```
# Embarked
f,ax=plt.subplots(2, 2, figsize=(20,15))
sns.countplot('Embarked', data=train, ax=ax[0,0])
ax[0,0].set_title('(1) No. Of Passengers Boarded')
sns.countplot('Embarked', hue='Sex', data=train, ax=ax[0,1])
ax[0,1].set_title('(2) Male-Female Split for Embarked')
sns.countplot('Embarked', hue='Survived', data=train, ax=ax[1,0])
ax[1,0].set_title('(3) Embarked vs Survived')
sns.countplot('Embarked', hue='Pclass', data=train, ax=ax[1,1])
ax[1,1].set_title('(4) Embarked vs Pclass')
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



2. Exploratory data analysis(Family)

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

f,ax=plt.subplots(1, 3, figsize=(40,10))
sns.countplot('FamilySize', data=train, ax=ax[0])
ax[0].set_title('(1) No. Of Passengers Boarded', y=1.02)

sns.countplot('FamilySize', hue='Survived', data=train, ax=ax[1])
ax[1].set_title('(2) Survived countplot depending on FamilySize',  y=1.02)

train[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=True).mean().sort_values(by='Survived',
                                                                                            ascending=False).plot.bar(ax=ax[2])
ax[2].set_title('(3) Survived rate depending on FamilySize',  y=1.02)

plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```



2. Exploratory data analysis(Fare)

```
# Fare(NAN을 Fare 전체의 평균으로 채움), outlier 영향 줄이기 위해 log취함
test.loc[test.Fare.isnull(), 'Fare'] = test['Fare'].mean()

train['Fare'] = train['Fare'].map(lambda i: np.log(i) if i > 0 else 0)
test['Fare'] = test['Fare'].map(lambda i: np.log(i) if i > 0 else 0)

fig, ax = plt.subplots(1, 1, figsize=(8, 8))
g = sns.distplot(train['Fare'], color='b', label='Skewness : {:.2f}'.format(train['Fare'].skew()), ax=ax)
g = g.legend(loc='best')
```