Titanic

목표는 '다양한 사람들의 생존 정보를 이용하여 생존 여부를 예측'하는 것.

Library

```
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)

import missingno as msno

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

목표는 '다양한 사람들의 생존 정보를 이용하여 생존 여부를 예측'하는 것.

1. Dataset 확인.

```
df_train = pd.read_csv('../ ~ ')
df_test  = pd.read_csv('../ ~ ')
```

1.1 Null data 확인.

동일하게
df_test도
확인

```
for col in df_train.columns :
    msg = 'Columns : {:>10}\t Percentage of NAN value : {:.2f}%'.
            format(col, 100 * (df_train[col].isnull().sum() /
                                df_train[col].shape[0] ) )
    print(msg)
```

```
msno.matrix( df = df_train.iloc[:, :], figsize = (8, 8),
             color = (.8, .5, .2) )
```
`bar` 또도 표현 가능.

1.2 Target label 확인
; target label의 distribution을 확인 (모델 평가 방법을 정하기 위함.)

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

파이 그래프 그리기

```
df_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=df_train, ax=ax[1])
ax[1].set_title('Count plot - Survived')

plt.show()
```

2. EDA
: 여러 feature 들을 개별적으로 분석, ⎫→ Insight 도출
  feature 간 상관 관계 확인 ⎭

2.1 Pclass.
- 'Ordinal' type (카테고리이면서 순서가 있음)
- 'Pclass'와 '생존률' 간의 상관 관계 여부 확인.

```
df_train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).count()
df_train                    "                    , sum()
                                            (→ 열 합 축기)
→ pd.crosstab( df_train['Pclass'], df_train['Survived'], margins=True).
    style.background_gradient(cmap='summer_r')

df_train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=True).
    mean().sort_values(by='Survived', ascending=False).plot.bar()
```

(+) Count plot (Seaborn)

```
y_position = 1.02

f, ax = plt.subplots (1, 2, figsize=(18,8) )
df_train ['Pclass'].value_counts ( ).plot.bar (color=[    ,    ,    ,],
                                    ax=ax[0] )

ax[0].set_title ('Num of Passenger by pclass', y=y_position )
ax[0].set_ylabel ('Count')
```

```
sns.countplot ('Pclass', hue='Survived', data=df_train, ax=ax[1] )
ax[1].set_title ('The num of the survived by pclass', y=y_position)
```

2.2. Sex.
   - 성별에 따라 생존율이 어떤 차이를 보이는 지 확인.

```
f, ax = plt.subplots (1, 2, figsize=(18.8) )
df_train [['Sex', 'Survived']].groupby (['Sex'], as_index=True).mean().
       plot.bar (ax=ax[0])
sns.countplot ('Sex', hue='Survived', data=df_train, ax=ax[1] )
ax[0].set_title ('The ratio of Survived by Sex' )
ax[1].set_title ('Sex : Survived vs Dead' )
plt.show( )
```

2.3 Pclass & Sex
   - 'Pclass'와 'Sex'가 '생존율'와 어떤 상관 관계를 보이는 지 확인.

```
sns.factorplot ('Pclass', 'Survived', hue='Sex', data=df_train,
              size=6, aspect=1.5 )
```

2.4 Age.

```
print (f'❝ The oldest passenger : { df_train ['Age'].max()): -1f 3\n
              The youngest .   : {   "   , min() "  3 "
              The average of age :     "    .mean()  "   ')
```

Kdeplot : 'kernel density estimation'의 약자로,
         discrete 한 변수를 Continuous 해서 밀도의 확률 값으로 추정.

```
fig, ax = plt. Subplots (1, 1, figsize =(9,5) )
sns. Kdeplot (df_train [df_train ['Survived'] == 1][ 'Age'], ax=ax )
          "                [ '  " ']==0    "
plt. legend ( ['Survived', 'Dead'] )
plt. show ( )              Random
          ⇒ 'Age'를 Continuous Variables로 변환하여 Density 관점에서
            확률을 정의 (분포)
```

Age distribution by Pclass
```
plt. figure ( figsize = (8,6) )
~~df. train~~
for i in df_train ['Pclass']. values :
    df_train ['Age'][df_train ['Pclass'] == i]. plot (Kind= 'kde' )
```

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

?

Survival rate Change depending on range of Age.

풀이?

cummulative

한 말변표

가 무슨 의미

가 있는지?

↓

```
cummulate_survival_ratio = [ ]
for i in range(1, 80):
    cummulate_survival_ratio.append(
```

idea) Discrete한 Age의 survival rate를 구하고, 이를 interpolation한 뒤에 시각화 하는 것이 본 취지와 더 가깝다고 생각함.