

Titanic

NO.

표준은 '당연한 사람들의 생사 정보를 이용하여 생존 여부를 예측하는 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 msn
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
%matplotlib inline
```

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} | 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))
```

for 35 표기 가능

1.2 Target label 확인

: target label의 distribution을 확인 (모델 평가 방법을 정하기 위함.)

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

```
df_train['Survived'].value_counts(), plt.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, \rightarrow 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=False).count()
```

```
df_train["Pclass"].value_counts()
```

(정렬 포함하기)

```
pd.crosstab(df_train['Pclass'], df_train['Survived'], margins=True)
```

```
style.background_gradient(cmap='summer_r')
```

```
df_train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).
```

```
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 생존률이 어떤 차이를 보이는지 확인.

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

df_train[['Sex', 'Survived']].groupby(['Sex'], as_index=False).mean().
plot.bar(ax=ax[0])

sns.countplot('Sex', hue='Survival', 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 '생존률'과 어떤 상관 관계를 보이는지 확인.

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

?

Survival rate change depending on range of Age.

굳이?

cumulate_survival_ratio = []

cumulative

for i in range(1, 80):

한 번도 보지

cumulate_survival_ratio.append(

가 무슨 의미

가 있는지?

↓

idea) Discrete 한 Age의 Survival rate를 구하고, 이를 interpolation한 그래프를 시각화하는 것이 본 취지와 더 가깝다고 생각함.

25 Pclass, Sex, Age.

• Boxplot + Kernel density (분포 플롯) \Rightarrow Violin-plot.

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

 sns.violinplot("Pclass", "Age", hue="Survived", data=df_train,
 scale="count", split=True, ax=ax[0])

ax[0].set_title('Pclass & Age vs Survived')

ax[0].set_yticks(range(0, 110, 10))

sns.violinplot('Sex', 'Age', hue='Survived', data=df_train,

scale='count', split=True, ax=ax[1])

plt.show()

2.6 Embarked

```
f, ax = plt.subplots(1, 1, figsize=(7, 7))
df_train[['Embarked', 'Survived']].groupby(['Embarked'], as_index=True).
mean().sort_values(by='Survived', ascending=False).plot.bar(ax=ax)
```

```
f, ax = plt.subplots(2, 2, figsize=(20, 15))
sns.countplot('Embarked', data=df_train, ax=ax[0, 0])
ax[0, 0].set_title('(1) No. of passengers boarded')
sns.countplot('Embarked', hue='Sex', ...)
```

} " ('Survived' & 'Pclass')

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

2.7 Family - Sibsp + Parch.

```
df_train['FamilySize'] = df_train['Sibsp'] + df_train['Parch'] + 1
df_test
```

• FamilySize 와 Survived 관계 시각화.

```
f, ax = plt.subplots(1, 3, figsize=(40, 10))
sns.countplot('FamilySize', data=df_train, ax=ax[0])
ax[0].set_title()
```

```
sns.countplot('FamilySize', hue='Survived', data=df_train, ax=ax[1])
ax[1].set_title(
```

```
df_train df_train[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False)
    .mean().sort_values(by='Survived', ascending=False).plot.bar(ax=ax[2])
ax[2].set_title(
```

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

2.8 Fare

• Continuous Feature.

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

→ 'High skewness' → log 스케일링

```
df_test.loc[df_test.Fare.isnull(), 'Fare'] = df_test['Fare'].mean()
```

```
df_train['Fare'] = df_train['Fare'].map(lambda i: np.log(i) if i > 0 else 0)
df_test
```

11

3. Feature engineering.

3.1 Fill Null value

3.1.1 Fill Null in Age using Title

문자열의 종류가

`df_train['Initial'] = df_train['Name'].str.extract(`
`'([A-Za-z]+)\.'`

`df_test`

'Initial'과 'Sex'의 관계를 crosstab

`pd.crosstab(df_train['Initial'], df_train['Sex'], T.`

`style.background_gradient(cmap='summer_r')`

~~Age~~ Initial

→ return 값을 바탕으로, 데이터 값을 replace

3.1.2 Fill Null in Embarked.

`print('Embarked has', sum(df_train['Embarked'].isnull()), 'null values')`

• 2 Null values in 'Embarked' feature.

`df_train['Embarked'].fillna('S', inplace=True)`

3.2 Change 'Age' feature from Continuous to Category

• Caution: Continuous feature를 Categorical로 바꾸면서

information loss 발생 가능 (Trial에 의의)

• 10 이상으로 Category 분류

• 'loc' or 'Apply' 활용


```
X_train = df_train.drop('Survived', axis=1).values
```

```
target_values = df_train['Survived'].values
```

```
X_test = df_test.values
```

```
X_tr, X_val, y_tr, y_val = train_test_split(X_train, target_values,  
                                             random_state=42,  
                                             test_size=0.3)
```

4.2 Model generation & Prediction.

```
model = RandomForestClassifier()
```

```
model.fit(X_tr, y_tr)
```

```
prediction = model.predict(X_val)
```

```
print('Acc: {:.2f}'.format(metrics.accuracy_score(prediction, y_val)))
```

4.3 Feature Importance.

```
from pandas import Series
```

```
feature_importance = model.feature_importances_
```

```
Series_feat_imp = Series(feature_importance, index=df_test.columns)
```