

Titanic - Advanced Feature Engineering Tutorial.

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid')

```

} For EDA

For Model.

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold

```

```

import string
import warnings
warnings.filterwarnings('ignore')
SEED = 42

```

} Default.

0. Loading Data

```
def concat_df(train_data, test_data):
```

```

    # Index 32
    # Index drop.
    return pd.concat([train_data, test_data], sort=True).reset_index(drop=True)

```

```
def divide_df(all_data)
```

```

    # change
    return all_data.iloc[:, :890], all_data.iloc[891:, :].drop(['Survived'], axis=1)

```

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

```
df_test = ''
```

```
df_all = concat_df(df_train, df_test)
```

```
df_train.name = 'Training Set'
```

```
df_test.name = 'Test Set'
```

```
df_all.name = 'All Set'
```

```
dfs = [df_train, df_test]
```

각 Set의 Shape, Columns 확인.

1. EDA.

1.1 Overview the dataset

1.2 Missing Values.

```
def display_missing(df):
```

```
    for col in df.columns.tolist():
```

```
        print(f'{col} columns missing values : {df[col].isnull().sum()}')
```

```
for df in dfs:
```

```
    print(df.name)
```

```
    display_missing(df)
```


1.2.1 Age.

- How to fill NA in 'Age'? By using high correlation with other features.

```
df_all_corr = df_all.corr().abs().unstack().sort_values(kind='quicksort',
                                                    ascending=False).reset_index()
```

df_all_corr의 type : Series → DataFrame (by 'reset_index' method)
(이름 변경) , inplace=True.

```
df_all_corr.rename(columns={'level_0': 'Feature_1', ... 3})
```

```
df_all_corr[df_all_corr['Feature_1'] == 'Age']
```

'Age'와 'Pclass'의 상관 관계가 가장 높음. 하지만 이것만으로 feature engineering
하기엔 너무 부족함. 정확도를 높이기 위해 'Sex' feature도 고려하여 missing value 처리.

groupby
age-by-pclass-sex = df_all[['Pclass', 'Sex']].median()['Age']

```
for pclass in df_all['Pclass'].unique().tolist():
```

```
    for sex in df_all['Sex']:
```

```
        print(f"median age of {pclass} {sex} : ", age-by-pclass-sex[pclass][sex])
```

```
df_all['Age'] = df_all.groupby(['Pclass', 'Sex'])['Age'].apply(lambda x: x.fillna(x.median()))
```

'lambda'는 DataFrame 객체의 Index 값을 받음. 위의 경우인 (1, 'female')과
같은 이름 인덱스로 x로 받음.

1.2.2 Embarked

- Need to fill two missing values. (Using outside information)

```
df_all['Embarked'] = df_all['Embarked'].fillna('S')
```

1.2.3 Fare

- Need to fill one missing value.

- The passenger is male, third-class, and no family.

```
med_fare = df_all.groupby(['Pclass', 'Parch', 'Sibsp'])['Fare'].  
            median()[3][0][0]
```

```
df_all['Fare'] = df_all['Fare'].fillna(med_fare)
```

1.2.4 Cabin

```
df_all['Deck'] = df_all['Cabin'].apply(lambda s: s[0] if pd.notnull(s)  
                                         else 'M')
```

→ 'Cabin'의 첫 문자가 Deck을 의미함. Deck 별 Pclass 상자를 확인

```
df_all_decks = df_all.groupby(['Pclass', 'Deck']).count().  
                drop(columns=[' ~ '])
```

```
def get_pclass_dist(df):
```

```
    deck_counts = {'A': {3, ... 3}
```

```
    decks = df.columns.levels[0]
```


for deck in decks:

try: for pclass in range(1, 4):
 count = df[deck][pclass][0]
 deck_counts[deck][pclass] = count

except KeyError:

deck_counts[deck][pclass] = 0.

df_deck = pd.DataFrame(deck_counts)

~~df~~ deck_percentage = {}

for col in df_deck.columns:

deck_percentage[col] = [(count / df_deck[col].sum()) * 100

for count in df_deck[col]]

return deck_counts, deck_percentage.

def display_pclass_dist(percentages):

df_percentages = pd.DataFrame(percentages).transpose()

deck_names = ('A', ..., 'T')

bar_count = np.arange(len(deck_names)) } # Xticks labeled with 1 to 20

bar_width = 0.85

pclass1 = df_percentages[0]

" 2 " [1]

" 3 " [2]

plt.figure(figsize=(20, 10))

```
plt.bar ( bar_count, pclass1, color = ,
          width = bar_width, edgecolor = ,
          label = 'pclass1' )
```

```
plt.bar ( bar_count, pclass2, bottom = pclass1,
          " " )
```

```
plt.bar ( bar_count, pclass3, bottom = pclass1 + pclass2,
          " " )
```

```
plt.xlabel ( 'Deck' )
```

```
plt.ylabel ( 'Pclass percentage' )
```

```
plt.xticks ( bar_count, deck_names )
```

```
plt.legend ( loc = 'Upper left', bbox_to_anchor = (1, 1), prop = {'size': 15} )
```

```
plt.title ( ' ' )
```

```
plt.show ( )
```

```
all_deck_count, all_deck_per = get_pclass_dist (df_all_decks)
display_pclass_dist (all_deck_per)
```

```
# change 'T' deck to 'A' deck.
```

```
idx = df_all [df_all ['Deck'] == 'T'].index
```

```
df_all.loc [idx, 'Deck'] = 'A'
```

* Survived & Deck & 위의 과정의 동일성