타이타닉 생존자 예측 대회

학습 목표

- Cabin 피처를 사용한다.
- 원핫 인코딩을 이해한다.
- GridSearchCV에 대해 이해한다.

목차

데이터

Data Fields

구분	설명	값
Survival	생존 여부	Survival. 0 = No, 1 = Yes
Pclass	티켓의 클래스	Ticket class. 1 = 1st, 2 = 2nd, 3 = 3rd
Sex	성별(Sex)	남(male)/여(female)
Age	나이(Age in years.)	
SibSp	함께 탑승한 형제와 배우자의 수 /siblings, spouses aboard the Titanic.	
Parch	함께 탑승한 부모, 아이의 수	# of parents / children aboard the Titanic.
Ticket	티켓 번호(Ticket number)	(ex) CA 31352, A/5. 2151
Fare	탑승료(Passenger fare)	
Cabin	객실 번호(Cabin number)	
Embarked	탑승 항구(Port of Embarkation)	C = Cherbourg, Q = Queenstown, S = Southampton

- siblings : 형제, 자매, 형제, 의붓 형제
- spouses : 남편, 아내 (정부와 약혼자는 무시)
- Parch : Parent(mother, father), child(daughter, son, stepdaughter, stepson)

01. 데이터 불러오기

목차로 이동하기

참고 노트북

- titanic 전체 노트북
  - https://www.kaggle.com/code/pliptor/how-am-i-doing-with-my-score/report (https://www.kaggle.com/code/pliptor/how-am-i-doing-with-my-score/report)
  - https://www.kaggle.com/code/ccastleberry/titanic-cabin-features/notebook (https://www.kaggle.com/code/ccastleberry/titanic-cabin-features/notebook)

In [1]:

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

In [2]:

```
train = pd.read_csv("data/titanic/train.csv", index_col='PassengerId')
test = pd.read_csv("data/titanic/test.csv", index_col='PassengerId')
sub = pd.read_csv("data/titanic/gender_submission.csv")
```

데이터 합치기

In [3]:

```
train_results = train["Survived"].copy()
train.drop("Survived", axis=1, inplace=True, errors="ignore")
titanic = pd.concat([train, test])
traindex = train.index
testdex = test.index
```

Cabin 확인

In [4]:

```
titanic.isnull().sum()
```

Out[4]:

```
Pclass         0
Name           0
Sex            0
Age          263
SibSp          0
Parch          0
Ticket         0
Fare           1
Cabin       1014
Embarked       2
dtype: int64
```

In [5]:

```
titanic['Cabin'].value_counts()
```

Out[5]:

```
C23 C25 C27        6
G6                 5
B57 B59 B63 B66    5
C22 C26            4
F33                4
                  ..
A14                1
E63                1
E12                1
E38                1
C105               1
Name: Cabin, Length: 186, dtype: int64
```

02. Cabin 데이터 전처리

목차로 이동하기

Cabin 데이터 전처리

In [16]:

```
cabin_only = titanic[["Cabin"]].copy()
cabin_only["Cabin_Data"] = cabin_only["Cabin"].isnull().apply(lambda x: not x)
```

In [17]:

```
cabin_only["Deck"] = cabin_only["Cabin"].str.slice(0,1)
cabin_only["Room"] = cabin_only["Cabin"].str.slice(1,5).str.extract("([0-9]+)",
                                                        expand=False).astype("float")
cabin_only[cabin_only["Cabin_Data"]]
```

Out[17]:

	Cabin	Cabin_Data	Deck	Room
PassengerId				
2	C85	True	C	85.0
4	C123	True	C	123.0
7	E46	True	E	46.0
11	G6	True	G	6.0
12	C103	True	C	103.0
...	...	...	...	...
1296	D40	True	D	40.0
1297	D38	True	D	38.0
1299	C80	True	C	80.0
1303	C78	True	C	78.0
1306	C105	True	C	105.0

295 rows × 4 columns

In [18]:

```
cabin_only['Deck'].value_counts()
```

Out[18]:

```
C    94
B    65
D    46
E    41
A    22
F    21
G     5
T     1
Name: Deck, dtype: int64
```

In [19]:

```
cabin_only['Deck'].isnull().sum()
```

Out[19]:

```
1014
```

Cabin, Cabin_data 삭제

In [20]:

```
cabin_only.drop(["Cabin", "Cabin_Data"], axis=1, inplace=True, errors="ignore")
```

In [21]:

```
cabin_only["Deck"] = cabin_only["Deck"].fillna("N")
cabin_only["Room"] = cabin_only["Room"].fillna(cabin_only["Room"].mean())
```

In [22]:

```
cabin_only.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 1 to 1309
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Deck    1309 non-null   object 
 1   Room    1309 non-null   float64
dtypes: float64(1), object(1)
memory usage: 30.7+ KB
```

```
    '''
    선택된 컬럼의 원핫을 수행
    Args:
        df: 판다스 데이터 프레임
        label: 변경할 컬럼명
        drop_col: boolean to decide if the chosen column should be dropped
    Returns:
        원핫이 추가된 데이터 프레임
    '''
```

In [23]:

```
def one_hot_column(df, label, drop_col=False):
    one_hot = pd.get_dummies(df[label], prefix=label)
    if drop_col:
        df = df.drop(label, axis=1)
    df = df.join(one_hot)
    return df
```

```
    '''
    This function will one hot encode a list of columns.
    Args:
        df: Pandas dataframe
        labels: list of the columns to encode
        drop_col: boolean to decide if the chosen column should be dropped
    Returns:
        pandas dataframe with the given encoding
    '''
```

In [24]:

```
def one_hot(df, labels, drop_col=False):

    for label in labels:
        df = one_hot_column(df, label, drop_col)
    return df
```

In [25]:

```
cabin_only = one_hot(cabin_only, ["Deck"],drop_col=True)
```

In [26]:

```
cabin_only.head()
```

Out[26]:

	Room	Deck_A	Deck_B	Deck_C	Deck_D	Deck_E	Deck_F	Deck_G	Deck_
PassengerId									
1	49.615917	0	0	0	0	0	0	0	
2	85.000000	0	0	1	0	0	0	0	
3	49.615917	0	0	0	0	0	0	0	
4	123.000000	0	0	1	0	0	0	0	
5	49.615917	0	0	0	0	0	0	0	

Pclass와 Cabin 데이터 셋의 상관관계

In [27]:

```
for column in cabin_only.columns.values[1:]:
    titanic[column] = cabin_only[column]
```

In [28]:

```
titanic.drop(["Ticket","Cabin"], axis=1, inplace=True)
```

In [29]:

```
corr = titanic.corr()
```

In [30]:

```
corr["Pclass"].sort_values(ascending=False)
```

Out[30]:

```
Pclass     1.000000
Deck_N     0.713857
SibSp      0.060832
Deck_G     0.052133
Parch      0.018322
Deck_F     0.013122
Deck_T    -0.042750
Deck_A    -0.202143
Deck_E    -0.225649
Deck_D    -0.265341
Deck_B    -0.353414
Age       -0.408106
Deck_C    -0.430044
Fare      -0.558629
Name: Pclass, dtype: float64
```

In [31]:

```
# Train
train_df = cabin_only.loc[traindex, :]
train_df['Survived'] = train_results

# Test
test_df = cabin_only.loc[testdex, :]
```

In [32]:

```
test_df.head()
```

Out[32]:

	Room	Deck_A	Deck_B	Deck_C	Deck_D	Deck_E	Deck_F	Deck_G	Deck_N
PassengerId									
892	49.615917	0	0	0	0	0	0	0	1
893	49.615917	0	0	0	0	0	0	0	1
894	49.615917	0	0	0	0	0	0	0	1
895	49.615917	0	0	0	0	0	0	0	1
896	49.615917	0	0	0	0	0	0	0	1

03. 모델 구축 및 모델 평가

목차로 이동하기

In [33]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn import metrics
from sklearn.model_selection import cross_val_score
import scipy.stats as st
import numpy as np
```

In [34]:

```
model = RandomForestClassifier()
```

In [35]:

```
X = train_df.drop("Survived", axis=1).copy()
y = train_df["Survived"]
```

In [36]:

```
param_grid ={'max_depth': st.randint(6, 11),
             'n_estimators':st.randint(300, 500),
             'max_features':np.arange(0.5,.81, 0.05),
             'max_leaf_nodes':st.randint(6, 10)}

grid = RandomizedSearchCV(model,
                    param_grid, cv=10,
                    scoring='accuracy',
                    verbose=1,n_iter=20)

grid.fit(X, y)
```

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
```

Out[36]:

```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=20,
                   param_distributions={'max_depth': <scipy.stats._distn_infrastruct
ure.rv_frozen object at 0x000002571D175C40>,
                                        'max_features': array([0.5 , 0.55, 0.6 , 0.6
5, 0.7 , 0.75, 0.8 ]),
                                        'max_leaf_nodes': <scipy.stats._distn_infras
tructure.rv_frozen object at 0x000002571C83D700>,
                                        'n_estimators': <scipy.stats._distn_infrastr
ucture.rv_frozen object at 0x000002571CA05B20>},
                   scoring='accuracy', verbose=1)
```

In [37]:

```
grid.best_estimator_
```

Out[37]:

```
RandomForestClassifier(max_depth=6, max_features=0.5, max_leaf_nodes=6,
                       n_estimators=389)
```

In [38]:

```
grid.best_score_
```

Out[38]:

```
0.690274656679151
```

In [39]:

```
pred = grid.best_estimator_.predict(test_df)
```

In [45]:

```
results_df = pd.DataFrame()
results_df["PassengerId"] = test_df.index
results_df["Survived"] = pred
```

In [46]:

```
results_df.head()
```

Out[46]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0

In [47]:

```
results_df.to_csv("pred_2211.csv", index=False)
```

0.67703

실습

- 1. 다른 feature도 추가한 이후에 제출해 보기
    - 'PassengerId', 'Pclass', 'SibSp', 'Parch'
- 2. Age, Fare, Sex, Embarked를 추가한 이후에 제출해 보기