타이타닉 생존자 예측 대회

학습 목표

- Ticket, Gender 피처를 사용한다.
- GridSearchCV를 통해 좋은 변수를 사용한다.
- 데이터 변환을 수행해 본다.
- 새로운 변수 생성을 알아본다.

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/pliptor/titanic-ticket-only-study/notebook (https://www.kaggle.com/code/pliptor/titanic-ticket-only-study/notebook)

In [109]:

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

In [110]:

```
sel_f = ['Ticket', 'Pclass', 'Sex']

train = pd.read_csv("data/titanic/train.csv",
                    usecols=['PassengerId', 'Survived']+sel_f)

test = pd.read_csv("data/titanic/test.csv",
                   usecols=['PassengerId'] + sel_f)

sub = pd.read_csv("data/titanic/gender_submission.csv")
```

In [111]:

```
# 컬럼 추가 및 합치기
test['Survived'] = np.nan
all_df = pd.concat([train, test])
all_df.head()
```

Out[111]:

	PassengerId	Survived	Pclass	Sex	Ticket
0	1	0.0	3	male	A/5 21171
1	2	1.0	1	female	PC 17599
2	3	1.0	3	female	STON/O2. 3101282
3	4	1.0	1	female	113803
4	5	0.0	3	male	373450

02. 티켓 변수 확인

목차로 이동하기

In [112]:

```
all_df.Ticket.unique()
```

```
'113056', '349239', '345774', '349206', '237798', '370373',
'19877', '11967', 'SC/Paris 2163', '349236', '349233', 'PC 17612',
'2693', '113781', '19988', '9234', '367226', '226593', 'A/5 2466',
'17421', 'PC 17758', 'P/PP 3381', 'PC 17485', '11767', 'PC 17608',
'250651', '349243', 'F.C.C. 13529', '347470', '29011', '36928',
'16966', 'A/5 21172', '349219', '234818', '345364', '28551',
'111361', '113043', 'PC 17611', '349225', '7598', '113784',
'248740', '244361', '229236', '248733', '31418', '386525',
'C.A. 37671', '315088', '7267', '113510', '2695', '2647', '345783',
'237671', '330931', '330980', 'SC/PARIS 2167', '2691',

'SOTON/O.Q. 3101310', 'C 7076', '110813', '2626', '14313',
'PC 17477', '11765', '3101267', '323951', 'C 7077', '113503',
'2648', '347069', 'PC 17757', '2653', 'STON/O 2. 3101293',
'349227', '27849', '367655', 'SC 1748', '113760', '350034',
'3101277', '350052', '350407', '28403', '244278', '240929',
'STON/O 2. 3101289', '341826', '4137', '315096', '28664', '347064',
'29106', '312992', '349222', '394140', 'STON/O 2. 3101269',
'343095', '28220', '250652', '28228', '345773', '349254',
'A/5. 13032', '315082', '347080', 'A/4. 34244', '2003', '250655',
```

- 티켓은 순전히 숫자인 것, 그리고 영숫자 접두사, 그리고 승무원인 LINE이 발행.

In [113]:

```
all_df.loc[all_df['Ticket']=='LINE']
```

Out[113]:

	PassengerId	Survived	Pclass	Sex	Ticket
179	180	0.0	3	male	LINE
271	272	1.0	3	male	LINE
302	303	0.0	3	male	LINE
597	598	0.0	3	male	LINE

- 다른 데이터와 비슷하게 만들기 위해 LINE 0 으로 변경

In [114]:

```
all_df['Ticket'] = all_df['Ticket'].replace('LINE', 'LINE 0')
all_df[all_df['Ticket']=='LINE 0']
```

Out[114]:

	PassengerId	Survived	Pclass	Sex	Ticket
179	180	0.0	3	male	LINE 0
271	272	1.0	3	male	LINE 0
302	303	0.0	3	male	LINE 0
597	598	0.0	3	male	LINE 0

티켓의 중복 확인

In [115]:

```
dup_tickets = all_df.groupby('Ticket').size()
dup_tickets
```

Out[115]:

```
Ticket
110152         3
110413         3
110465         2
110469         1
110489         1
              ..
W./C. 6608     5
W./C. 6609     1
W.E.P. 5734    2
W/C 14208      1
WE/P 5735      2
Length: 929, dtype: int64
```

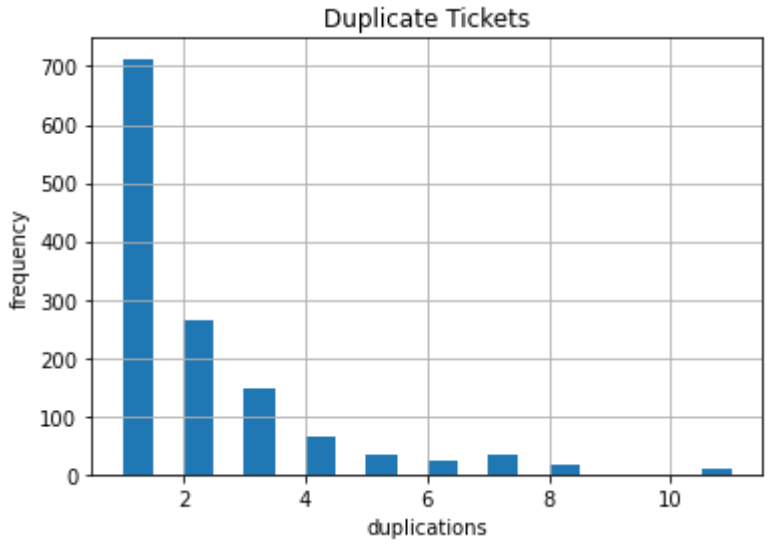
중복 티켓 장수

In [116]:

```
all_df['중복티켓수'] = all_df['Ticket'].map(dup_tickets)
plt.xlabel('duplications')
plt.ylabel('frequency')
plt.title('Duplicate Tickets')
all_df['중복티켓수'].hist(bins=20)
```

Out[116]:

```
<AxesSubplot:title={'center':'Duplicate Tickets'}, xlabel='duplications', ylabel='fr
equency'>
```



- 유일한 티켓이 압도적으로 많다.

티켓의 값 처리

- '.', '/' 을 공백으로 변경

In [117]:

```
all_df['Ticket'] = all_df['Ticket'].apply(lambda x: x.replace('.','').replace('/','').lower())
all_df.head()
```

Out[117]:

	PassengerId	Survived	Pclass	Sex	Ticket	중복티켓수
0	1	0.0	3	male	a5 21171	1
1	2	1.0	1	female	pc 17599	2
2	3	1.0	3	female	stono2 3101282	1
3	4	1.0	1	female	113803	2
4	5	0.0	3	male	373450	1

Ticket의 값을 공백으로 분리 후, 앞의 이름을 갖는 변수 만들기

In [118]:

```
"aaaaa 000000".split(' ')[0][0]    # 첫번째 단어의 맨 앞 첫글자
```

Out[118]:

```
'a'
```

In [119]:

```
def get_prefix(ticket):
    lead = ticket.split(' ')[0][0]

    # 알파벳인지 확인
    if lead.isalpha():
        return ticket.split(' ')[0]
    else:
        return 'NoPrefix'

all_df['Prefix'] = all_df['Ticket'].apply(lambda x: get_prefix(x))
all_df.head()
```

Out[119]:

	PassengerId	Survived	Pclass	Sex	Ticket	중복티켓수	Prefix
0	1	0.0	3	male	a5 21171	1	a5
1	2	1.0	1	female	pc 17599	2	pc
2	3	1.0	3	female	stono2 3101282	1	stono2
3	4	1.0	1	female	113803	2	NoPrefix
4	5	0.0	3	male	373450	1	NoPrefix

In [120]:

```
"a5 21171".split(' ')[-1]
```

Out[120]:

'21171'

In [121]:

```
str("a5 21171")[0]
```

Out[121]:

'a'

In [122]:

```
val = int( "a5 21171".split(' ')[-1] )
str(val)
```

Out[122]:

'21171'

- TNumeric : 숫자로 변경
- TNlen : TNumeric의 길이
- LeadingDigit : TNumeric의 맨 앞글자
- TGroup : Ticket의 뒷부분의 문자로 변경

In [123]:

```
all_df['TNumeric'] = all_df['Ticket'].apply(lambda x: int(x.split(' ')[-1])//1)
all_df['TNlen'] = all_df['TNumeric'].apply(lambda x : len(str(x)))
all_df['LeadingDigit'] = all_df['TNumeric'].apply(lambda x : int(str(x)[0]))
all_df['TGroup'] = all_df['Ticket'].apply(lambda x: str(int(x.split(' ')[-1])//10))
all_df.head()
```

Out[123]:

	PassengerId	Survived	Pclass	Sex	Ticket	중복티켓수	Prefix	TNumeric	TNlen	LeadingDigi
0	1	0.0	3	male	a5 21171	1	a5	21171	5	2
1	2	1.0	1	female	pc 17599	2	pc	17599	5	1
2	3	1.0	3	female	stono2 3101282	1	stono2	3101282	7	3
3	4	1.0	1	female	113803	2	NoPrefix	113803	6	1
4	5	0.0	3	male	373450	1	NoPrefix	373450	6	3

In [124]:

```
pd.crosstab(all_df['Pclass'],all_df['LeadingDigit'])
```

Out[124]:

LeadingDigit	0	1	2	3	4	5	6	7	8	9
Pclass										
1	0	288	8	18	0	5	4	0	0	0
2	0	32	205	37	0	1	0	2	0	0
3	4	22	136	476	22	4	17	18	5	5

In [125]:

```
all_df = all_df.drop(columns=['Ticket', 'TNumeric', 'Pclass'])
all_df
```

Out[125]:

	PassengerId	Survived	Sex	중복티켓수	Prefix	TNlen	LeadingDigit	TGroup
0	1	0.0	male	1	a5	5	2	2117
1	2	1.0	female	2	pc	5	1	1759
2	3	1.0	female	1	stono2	7	3	310128
3	4	1.0	female	2	NoPrefix	6	1	11380
4	5	0.0	male	1	NoPrefix	6	3	37345
...	...	...	...	...	...	...	...	...
413	1305	NaN	male	1	a5	4	3	323
414	1306	NaN	female	3	pc	5	1	1775
415	1307	NaN	male	1	sotonoq	7	3	310126
416	1308	NaN	male	1	NoPrefix	6	3	35930
417	1309	NaN	male	3	NoPrefix	4	2	266

1309 rows × 8 columns

In [126]:

```
all_df['Prefix']
```

Out[126]:

```
0            a5
1            pc
2        stono2
3      NoPrefix
4      NoPrefix
         ...
413          a5
414          pc
415     sotonoq
416    NoPrefix
417    NoPrefix
Name: Prefix, Length: 1309, dtype: object
```

In [127]:

```
all_df = pd.concat([pd.get_dummies(all_df[['Prefix','TGroup']]),
            all_df[['PassengerId','Survived','중복티켓수','TNlen','LeadingDigit', 'Sex']]],
            axis=1)

all_df
```

Out[127]:

Prefix_casoton	...	TGroup_847	TGroup_85	TGroup_923	TGroup_954	PassengerId	Survived	중복티켓수
0	...	0	0	0	0	1	0.0	1
0	...	0	0	0	0	2	1.0	2
0	...	0	0	0	0	3	1.0	1
0	...	0	0	0	0	4	1.0	2
0	...	0	0	0	0	5	0.0	1
...	...	...	...	...	...	...	...	..
0	...	0	0	0	0	1305	NaN	1
0	...	0	0	0	0	1306	NaN	3
0	...	0	0	0	0	1307	NaN	1
0	...	0	0	0	0	1308	NaN	1
0	...	0	0	0	0	1309	NaN	3

In [135]:

```
dict_s = {'male':0, 'female':1}
all_df['Sex'] = all_df['Sex'].map(dict_s)
```

In [136]:

```
predictors = sorted(list(set(all_df.columns) - set(['PassengerId','Survived'])))
predictors
```

Out[136]:

```
['LeadingDigit',
 'Prefix_NoPrefix',
 'Prefix_a',
 'Prefix_a4',
 'Prefix_a5',
 'Prefix_aq3',
 'Prefix_aq4',
 'Prefix_as',
 'Prefix_c',
 'Prefix_ca',
 'Prefix_casoton',
 'Prefix_fa',
 'Prefix_fc',
 'Prefix_fcc',
 'Prefix_line',
 'Prefix_lp',
 'Prefix_pc',
```

In [137]:

```
all_df2 = all_df[predictors + ['Survived']]
all_df2.head()
```

Out[137]:

	LeadingDigit	Prefix_NoPrefix	Prefix_a	Prefix_a4	Prefix_a5	Prefix_aq3	Prefix_aq4	Prefix_as
0	2	0	0	0	1	0	0	
1	1	0	0	0	0	0	0	
2	3	0	0	0	0	0	0	
3	1	1	0	0	0	0	0	
4	3	1	0	0	0	0	0	

5 rows × 450 columns

In [138]:

```
df_train = all_df2.loc[all_df2['Survived'].isin([np.nan]) == False]
df_test  = all_df2.loc[all_df2['Survived'].isin([np.nan]) == True]

print(df_train.shape)
df_train.head()
```

(891, 450)

Out[138]:

	LeadingDigit	Prefix_NoPrefix	Prefix_a	Prefix_a4	Prefix_a5	Prefix_aq3	Prefix_aq4	Prefix_a
0	2	0	0	0	1	0	0	
1	1	0	0	0	0	0	0	
2	3	0	0	0	0	0	0	
3	1	1	0	0	0	0	0	
4	3	1	0	0	0	0	0	

5 rows × 450 columns

In [139]:

```
print(df_test.shape)
df_test.head()
```

(418, 450)

Out[139]:

	LeadingDigit	Prefix_NoPrefix	Prefix_a	Prefix_a4	Prefix_a5	Prefix_aq3	Prefix_aq4	Prefix_a
0	3	1	0	0	0	0	0	
1	3	1	0	0	0	0	0	
2	2	1	0	0	0	0	0	
3	3	1	0	0	0	0	0	
4	3	1	0	0	0	0	0	

5 rows × 450 columns

03. 모델링

목차로 이동하기

In [140]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
```

In [141]:

```
model = KNeighborsClassifier(n_neighbors=11, metric = 'manhattan')

param_grid = ({'n_neighbors':[6,7,8,9,11],
               'metric':['manhattan','minkowski'],
               'p':[1,2]})

grs = GridSearchCV(model, param_grid,
                   cv = 28,
                   n_jobs=1,
                   return_train_score = True,
                   pre_dispatch=1)

grs.fit(np.array(df_train[predictors]), np.array(df_train['Survived']))
```

Out[141]:

```
GridSearchCV(cv=28,
             estimator=KNeighborsClassifier(metric='manhattan', n_neighbors=11),
             n_jobs=1,
             param_grid={'metric': ['manhattan', 'minkowski'],
                         'n_neighbors': [6, 7, 8, 9, 11], 'p': [1, 2]},
             pre_dispatch=1, return_train_score=True)
```

In [142]:

```
print("Best parameters " + str(grs.best_params_))
gpd = pd.DataFrame(grs.cv_results_)
print("정확도 :{0:1.4f}".format(gpd['mean_test_score'][grs.best_index_]))
```

```
Best parameters {'metric': 'manhattan', 'n_neighbors': 9, 'p': 1}
정확도 :0.7969
```

04. 예측

목차로 이동하기

In [144]:

```
pred_knn = grs.predict(np.array(df_test[predictors]))

sub = pd.DataFrame({'PassengerId':test['PassengerId'],'Survived':pred_knn})
sub.to_csv('ticket__sex_knn.csv', index = False, float_format='%1d')
sub.head()
```

Out[144]:

	PassengerId	Survived
0	892	0.0
1	893	1.0
2	894	0.0
3	895	0.0
4	896	1.0

0.72009

실습

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

In [ ]:

```

```