의사결정트리 (Titanic)

#01. 패키지 참조

```
import numpy as np
import seaborn as sb
import re   # 정규표현식 연산 패키지

from pandas import read_csv, DataFrame
from matplotlib import pyplot as plt

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import GridSearchCV
from sklearn.tree import plot_tree
import dtreeviz
```

#02. 데이터 가져오기

csv파일 읽기

```
train = read_csv("./train.csv")
test = read_csv("./test.csv")

print(train.shape, test.shape)
train.head()
```

```
(891, 12) (418, 11)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

데이터 프레임 원본 백업

```
original_train = train.copy()
original_test = test.copy()
```

훈련 데이터와 검증 데이터를 병합하여 전체 데이터셋 구성

```
full_data = [train, test]
print(type(full_data[0]))
full_data
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
[     PassengerId  Survived  Pclass  \
 0              1         0       3
 1              2         1       1
 2              3         1       3
 3              4         1       1
 4              5         0       3
 ..           ...       ...     ...
 886          887         0       2
 887          888         1       1
 888          889         0       3
 889          890         1       1
 890          891         0       3

                                                 Name     Sex   Age  SibSp  \
 0                             Braund, Mr. Owen Harris    male  22.0      1
 1    Cumings, Mrs. John Bradley (Florence Briggs Th ...  female  38.0      1
 2                             Heikkinen, Miss. Laina  female  26.0      0
 3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
 4                           Allen, Mr. William Henry    male  35.0      0
 ..                                               ...     ...   ...    ...
 886                            Montvila, Rev. Juozas    male  27.0      0
 887                     Graham, Miss. Margaret Edith  female  19.0      0
 888         Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
 889                            Behr, Mr. Karl Howell    male  26.0      0
 890                              Dooley, Mr. Patrick    male  32.0      0
```

```
     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...       ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns],
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           ...     ...                                           ...
413         1305       3                           Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex   Age  SibSp  Parch              Ticket      Fare Cabin Embarked
0      male  34.5      0      0              330911    7.8292   NaN        Q
1    female  47.0      1      0              363272    7.0000   NaN        S
2      male  62.0      0      0              240276    9.6875   NaN        Q
3      male  27.0      0      0              315154    8.6625   NaN        S
4    female  22.0      1      1             3101298   12.2875   NaN        S
..      ...   ...    ...    ...                 ...       ...   ...      ...
413    male   NaN      0      0           A.5. 3236    8.0500   NaN        S
414  female  39.0      0      0            PC 17758  108.9000  C105        C
415    male  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416    male   NaN      0      0              359309    8.0500   NaN        S
417    male   NaN      1      1                2668   22.3583   NaN        C

[418 rows x 11 columns]]
```

#03. 데이터 전처리

객실 유무

```
train['Has_Cabin'] = train['Cabin'].apply(lambda x: 0 if type(x) == float else 1)
test['Has_Cabin'] = test['Cabin'].apply(lambda x: 0 if type(x) == float else 1)
```

가족 구성원 수

```
train['FamilySize'] = train['SibSp'] + train['Parch'] + 1
test['FamilySize'] = test['SibSp'] + test['Parch'] + 1
```

혼자 탑승했는지 여부

```
train['ISAlone'] = 0
train.loc[train['FamilySize'] == 1, 'ISAlone'] = 1

test['ISAlone'] = 0
test.loc[test['FamilySize'] == 1, 'ISAlone'] = 1
```

탑승지 결측치 처리

소수의 결측치 데이터(train기준 2건)를 최빈값인 `S`로 대체

```
train['Embarked'] = train['Embarked'].fillna('S')
test['Embarked'] = test['Embarked'].fillna('S')
```

요금 데이터의 결측치 처리

```
train['Fare'] = train['Fare'].fillna(train['Fare'].median())
test['Fare'] = test['Fare'].fillna(test['Fare'].median())
```

나이 데이터의 결측치 처리

훈련데이터

```
age_avg = train['Age'].mean()
age_std = train['Age'].std()
age_null_count = train['Age'].isnull().sum()
age_null_rnd_list = np.random.randint(age_avg - age_std,
                        age_avg + age_std, size=age_null_count)

# 결측치에 해당하는 행에 랜덤값을 채워준다.
train.loc[np.isnan(train['Age']), 'Age'] = age_null_rnd_list

train['Age'] = train['Age'].astype(int)
```

검증데이터

```
age_avg = test['Age'].mean()
age_std = test['Age'].std()
age_null_count = test['Age'].isnull().sum()
age_null_rnd_list = np.random.randint(age_avg - age_std,
                        age_avg + age_std, size=age_null_count)

# 결측치에 해당하는 행에 랜덤값을 채워준다.
test.loc[np.isnan(test['Age']), 'Age'] = age_null_rnd_list
```

```
test['Age'] = test['Age'].astype(int)
```

정규표현식 처리 함수

이름에서 알파벳을 제외한 나머지 글자를 제외하고 어절단위로 리스트로 묶은 후 인덱스가 1인 위치를 리턴하는 함수

```
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)

    if title_search:
        return title_search.group(1)

    return ""
```

```
train['Title'] = train['Name'].apply(get_title)
print(list(train['Title'].value_counts().index))
```

```
['Mr', 'Miss', 'Mrs', 'Master', 'Dr', 'Rev', 'Mlle', 'Major', 'Col', 'Countess', 'Capt
```

```
train['Title'] = train['Title'].replace(['Dr', 'Rev',
                'Mlle', 'Major', 'Col', 'Countess', 'Capt', 'Ms',
                'Sir', 'Lady', 'Mme', 'Don', 'Jonkheer'], "Rare")
```

```
test['Title'] = test['Name'].apply(get_title)
test['Title'] = test['Title'].replace(['Dr', 'Rev',
                'Mlle', 'Major', 'Col', 'Countess', 'Capt', 'Ms',
                'Sir', 'Lady', 'Mme', 'Don', 'Jonkheer'], "Rare")
```

데이터 라벨링

성별

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

호칭

```
train['Title'] = train['Title'].map({"Mr": 1, "Master": 2, "Mrs": 3,
                                     "Miss": 4, "Rare": 5})
test['Title'] = test['Title'].map({"Mr": 1, "Master": 2, "Mrs": 3,
                                   "Miss": 4, "Rare": 5})
```

탑승지

```
train['Embarked'] = train['Embarked'].map({"S": 0, "C": 1, "Q": 2})
```

```
test['Embarked'] = test['Embarked'].map({"S": 0, "C": 1, "Q": 2})
```

연령대 분할

```
train.loc[ train['Age'] ⩽ 16, "Age"] = 0
train.loc[ (train['Age'] > 16) & (train['Age'] ⩽ 32), "Age"] = 1
train.loc[ (train['Age'] > 32) & (train['Age'] ⩽ 48), "Age"] = 2
train.loc[ (train['Age'] > 48) & (train['Age'] ⩽ 64), "Age"] = 3
train.loc[ train['Age'] > 64, "Age"] = 4

test.loc[ test['Age'] ⩽ 16, "Age"] = 0
test.loc[ (test['Age'] > 16) & (test['Age'] ⩽ 32), "Age"] = 1
test.loc[ (test['Age'] > 32) & (test['Age'] ⩽ 48), "Age"] = 2
test.loc[ (test['Age'] > 48) & (test['Age'] ⩽ 64), "Age"] = 3
test.loc[ test['Age'] > 64, "Age"] = 4
```

탑승 요금 분할

탑승요금의 최대/최소, 사분위 수 확인

```
train['Fare'].quantile([0, 0.25, 0.5, 0.75, 1.0])
```

```
0.00      0.0000
0.25      7.9104
0.50     14.4542
0.75     31.0000
1.00    512.3292
Name: Fare, dtype: float64
```

```
train.loc[ train['Fare'] ⩽ 7.91, "Fare"] = 0
train.loc[ (train['Fare'] > 7.91) & (train['Fare'] ⩽ 14.454), "Fare"] = 1
train.loc[ (train['Fare'] > 14.454) & (train['Fare'] ⩽ 31), "Fare"] = 2
train.loc[ train['Fare'] > 31, "Fare"] = 3

test.loc[ test['Fare'] ⩽ 7.91, "Fare"] = 0
test.loc[ (test['Fare'] > 7.91) & (test['Fare'] ⩽ 14.454), "Fare"] = 1
test.loc[ (test['Fare'] > 14.454) & (test['Fare'] ⩽ 31), "Fare"] = 2
test.loc[ test['Fare'] > 31, "Fare"] = 3
```

불필요한 필드 제거

```
drop_elements = ['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp']
train = train.drop(drop_elements, axis = 1)
test  = test.drop(drop_elements, axis = 1)

train.head()
```

	Survived	Pclass	Sex	Age	Parch	Fare	Embarked	Has_Cabin	FamilySize	ISAlone	T
0	0	3	1	1	0	0.0	0	0	2	0	1

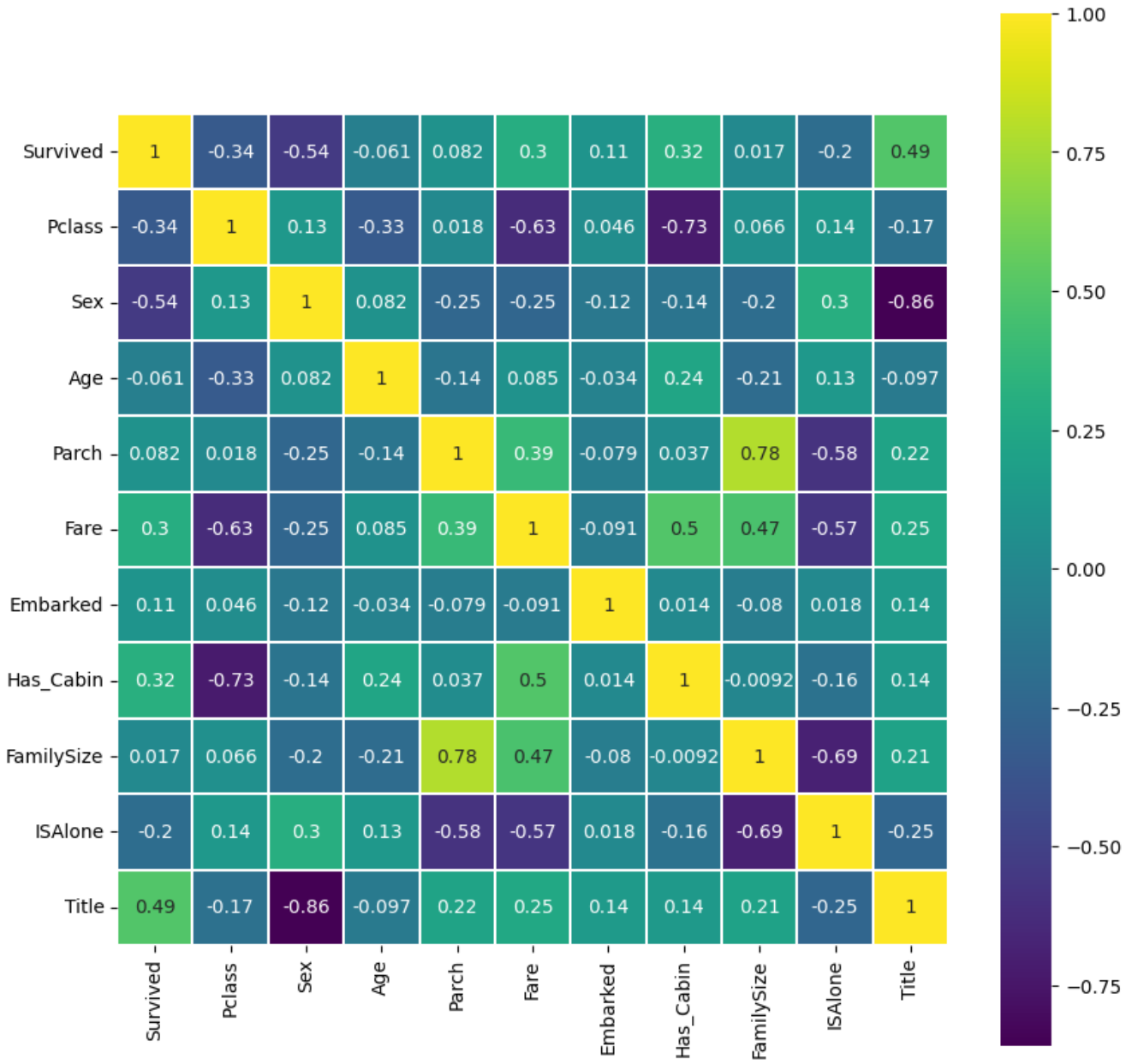
	Survived	Pclass	Sex	Age	Parch	Fare	Embarked	Has_Cabin	FamilySize	ISAlone	T
1	1	1	0	2	0	3.0	1	1	2	0	3
2	1	3	0	1	0	1.0	0	0	1	1	4
3	1	1	0	2	0	3.0	0	1	2	0	3
4	0	3	1	2	0	1.0	0	0	1	1	1

전처리 결과 확인

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

sb.heatmap(train.astype(float).corr(),linewidths=0.1,vmax=1.0, square=True,
           cmap=plt.cm.viridis, linecolor='white', annot=True)

plt.show()
plt.close()
```



선정된 최종 변수

성별과 나이의 경우 Title에 대한 상관계수가 지나치게 높기 때문에 Title과 함께 사용하는 것은 적절하지 않다고 판단

```
x_train = train.drop(['Sex', 'Age', 'Survived'], axis=1)
y_train = train["Survived"]
```

#04. 의사결정트리 구현

최적의 파라미터 찾기

```
dtree = DecisionTreeClassifier(random_state=777)

params = {
    'max_depth': [3, 5, 7, 9],
    'min_samples_split': [2, 3, 4],      # 노드를 분할하는데 필요한 최소 샘플 수
    'splitter': ['best', 'random']       # 각 노드에서 분할을 선택하는데 사용되는 전략
}

grid_dt = GridSearchCV(dtree, param_grid=params, cv=5, n_jobs=-1)

grid_dt.fit(x_train, y_train)
print(grid_dt.best_params_, grid_dt.best_score_)

result = DataFrame(grid_dt.cv_results_['params'])
result['mean_test_score'] = grid_dt.cv_results_['mean_test_score']
result.sort_values(by='mean_test_score', ascending=False)
```

```
{'max_depth': 3, 'min_samples_split': 2, 'splitter': 'best'} 0.824882304940054
```

	max_depth	min_samples_split	splitter	mean_test_score
0	3	2	best	0.824882
2	3	3	best	0.824882
4	3	4	best	0.824882
7	5	2	random	0.820400
11	5	4	random	0.820400
9	5	3	random	0.820400
10	5	4	best	0.815969
8	5	3	best	0.815969
6	5	2	best	0.815969
20	9	3	best	0.810345
22	9	4	best	0.808097
19	9	2	random	0.805838
18	9	2	best	0.803616

	max_depth	min_samples_split	splitter	mean_test_score
5	3	4	random	0.803565
3	3	3	random	0.803565
1	3	2	random	0.803565
13	7	2	random	0.802492
23	9	4	random	0.802467
16	7	4	best	0.801381
21	9	3	random	0.801331
14	7	3	best	0.799134
12	7	2	best	0.795763
15	7	3	random	0.790126
17	7	4	random	0.783397

검증 데이터를 훈련 데이터와 동일하게 필터링

```
test_df = test.filter(x_train.columns)
test_df
```

	Pclass	Parch	Fare	Embarked	Has_Cabin	FamilySize	ISAlone	Title
0	3	0	0.0	2	0	1	1	1.0
1	3	0	0.0	0	0	2	0	3.0
2	2	0	1.0	2	0	1	1	1.0
3	3	0	1.0	0	0	1	1	1.0
4	3	1	1.0	0	0	3	0	3.0
...	...	...	...	...	...	...	...	...
413	3	0	1.0	0	0	1	1	1.0
414	1	0	3.0	1	1	1	1	NaN
415	3	0	0.0	0	0	1	1	1.0
416	3	0	1.0	0	0	1	1	1.0
417	3	1	2.0	1	0	3	0	2.0

418 rows × 8 columns

최종 모델

```
dtree = DecisionTreeClassifier(random_state=777,
            max_depth=grid_dt.best_params_['max_depth'],
            min_samples_split=grid_dt.best_params_['min_samples_split'],
            splitter=grid_dt.best_params_['splitter'])
dtree.fit(x_train, y_train)
y_pred = dtree.predict(test_df)
```

```
print(dtree.score(x_train, y_train))
y_pred[:5]
```

```
0.8237934904601572
```

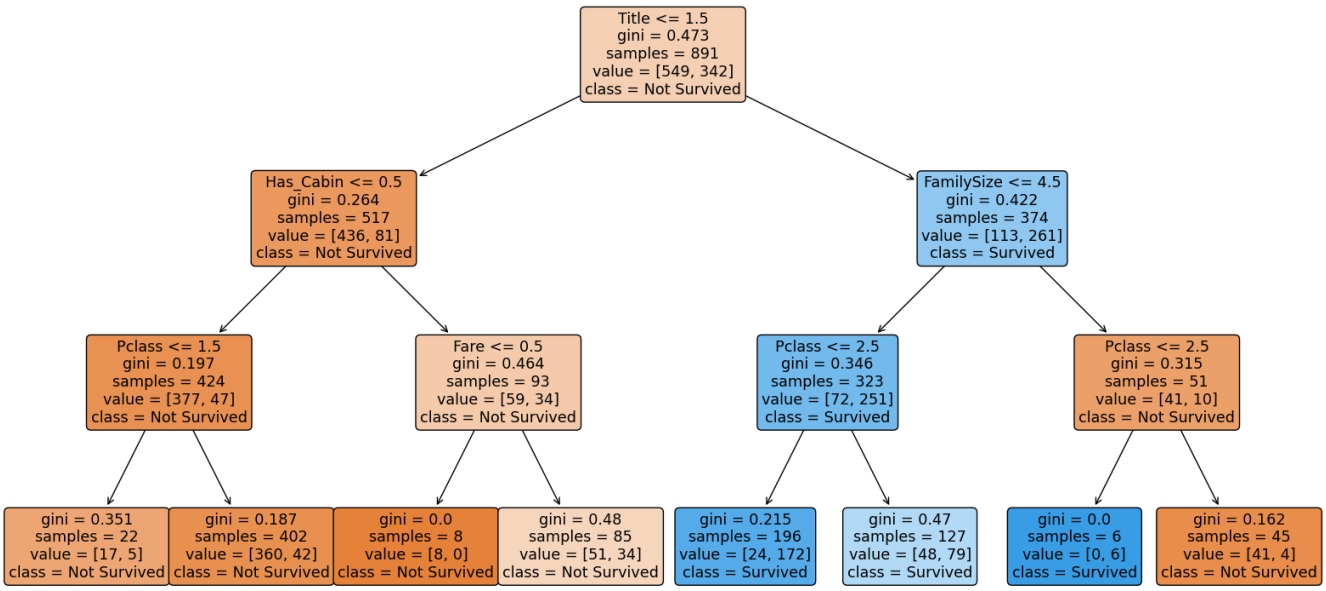
```
array([0, 1, 0, 0, 1], dtype=int64)
```

결과 시각화

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

# 리턴을 안받으면 그래프가 2개 출력된다. 리턴값을 따로 사용할 것이 아니므로 언더바로 받는다.
_ = plot_tree(dtree,
        feature_names=list(x_train.columns),
        class_names=['Not Survived', 'Survived'],
        rounded=True,   # 노드의 모서리를 둥글게
        filled=True)     # 노드의 색상을 다르게)

plt.show()
plt.close()
```




```
viz = dtreeviz.model(dtree,
                     X_train=x_train,
                     y_train=y_train,
                     target_name="titanic",
                     feature_names=x_train.columns,
                     class_names=['Not Survived', 'Survived'])

viz.view(scale=2.0)
```

```
c:\Users\leekh\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.p
```