인공지능 Assignment4

Kaggle - Titanic: Machine Learning from Disaster

2021/06/04

16010811 김범수

- 문제 정의

- 타이타닉 호 침몰 사건 승객들의 정보를 분석하여 타이타닉 사고로 인한 승객들의 생존 여부를 구분하는 모델을 구축합니다.

- 훈련 데이터 : 타이타닉에 승선한 승객들의 정보 및 생존/사망 여부

- 테스트 데이터 : 타이타닉에 승선한 승객들의 정보

분석과정

1. 데이터 전처리

    1. 데이터 준비

    2. 데이터 변수 확인

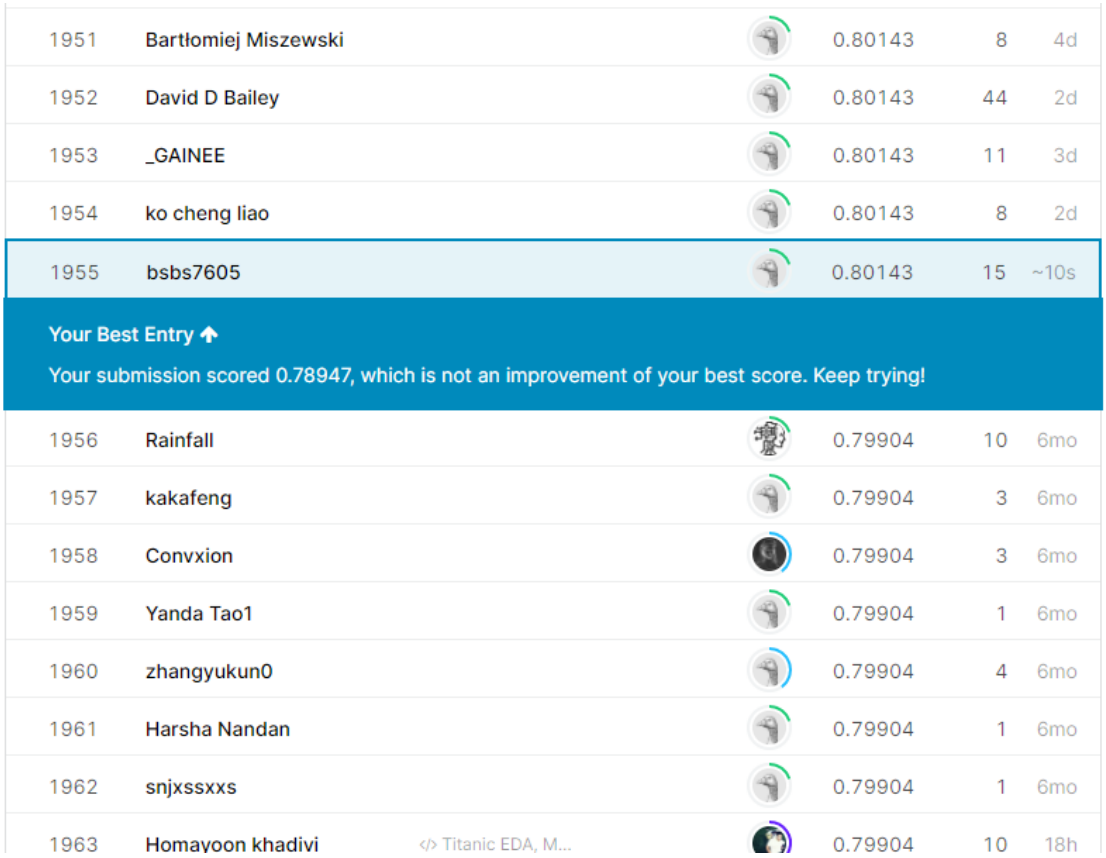
    3. 데이터 시각화 등을 이용한 탐색적 데이터 분석.

    4. Feature Engineering

2. 모델 구축 및 적용

3. 모델 평가

- **Competition 결과**



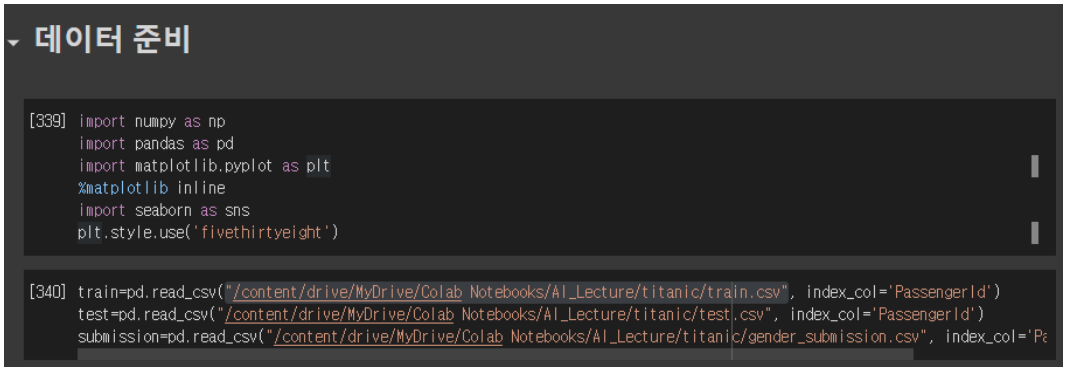
1951	Bartłomiej Miszewski		0.80143	8	4d
1952	David D Bailey		0.80143	44	2d
1953	_GAINEE		0.80143	11	3d
1954	ko cheng liao		0.80143	8	2d
1955	bsbs7605		0.80143	15	~10s

Your Best Entry ↑

Your submission scored 0.78947, which is not an improvement of your best score. Keep trying!

1956	Rainfall		0.79904	10	6mo
1957	kakafeng		0.79904	3	6mo
1958	Convxion		0.79904	3	6mo
1959	Yanda Tao1		0.79904	1	6mo
1960	zhangyukun0		0.79904	4	6mo
1961	Harsha Nandan		0.79904	1	6mo
1962	snjxssxxs		0.79904	1	6mo
1963	Homayoon khadivi	</> Titanic EDA, M...	0.79904	10	18h

- 데이터 준비



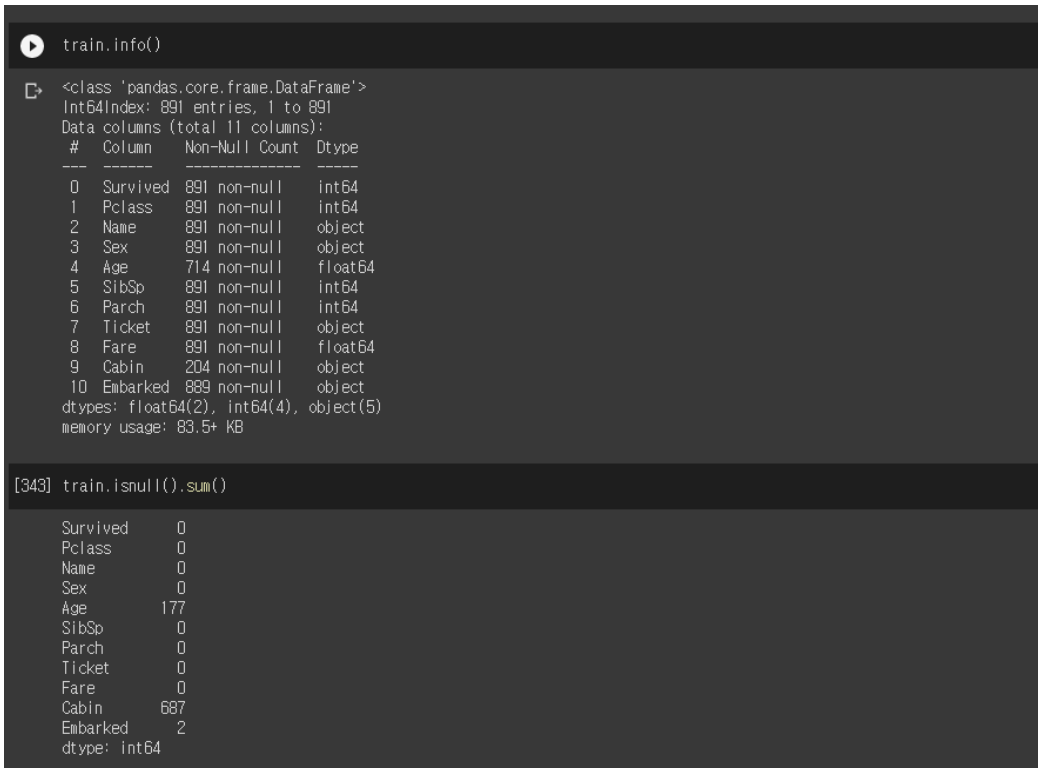
```
데이터 준비

[339] import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      %matplotlib inline
      import seaborn as sns
      plt.style.use('fivethirtyeight')

[340] train=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/AI_Lecture/titanic/train.csv", index_col='PassengerId')
      test=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/AI_Lecture/titanic/test.csv", index_col='PassengerId')
      submission=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/AI_Lecture/titanic/gender_submission.csv", index_col='Pa
```

데이터 시각화 과정에서 필요한 library 들을 import 하고
train, test 데이터 불러오는데 PassengerId 가 고유값이므로 index 로 사용하여 읽어 왔습니다.

- 데이터 변수 확인



```
train.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB

[343] train.isnull().sum()

Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Embarked 의 경우 패딩하여 사용하겠지만 결측치가 너무 많은 Cabin 의 경우 사용하지 않는게 좋을 것 이라 예상하였습니다.

```
train.describe(include=['O'])
```

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Danbom, Mr. Ernst Gilbert	male	347082	B96 B98	S
freq	1	577	7	4	644

Name 은 miss, mr 처럼 결혼 여부 등의 정보가 들어있을 것이라 고유값이어도 사용함이 좋을 것 같지만 빈도 적은 변수가 많아 변형해서 사용해야할 것이라 예상하였습니다.

Sex 는 male 이 577 로 많다는 사실을 확인하였습니다.

Ticket, Cabin 은 의미 없는 고유값으로 Drop 할 예정입니다.

Embarked 는 승객의 대다수인 644 명이 S 에서 탑승한 사실을 확인하였습니다.

```
[345] train.describe()
```

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Describe 명령어를 통해 Survived(생존률)이 약 38%임을 확인하였고,

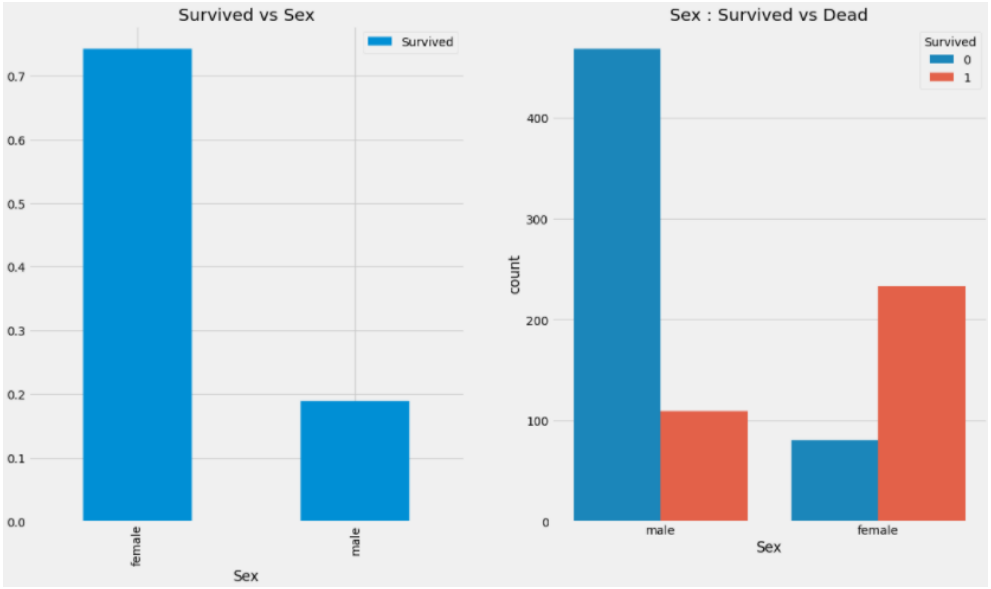
Age 에서 2~30 대 승객 다수이며,

SibSp, Parch 비슷한 성격의 변수, 대부분 승객이 홀로탄 사실을 확인하였습니다.

또한,
Fare 에서 평균은 32 이나 max 값이 512 이므로 최대치를 낮게 조정해줄 필요가 있다고 생각하였습니다.

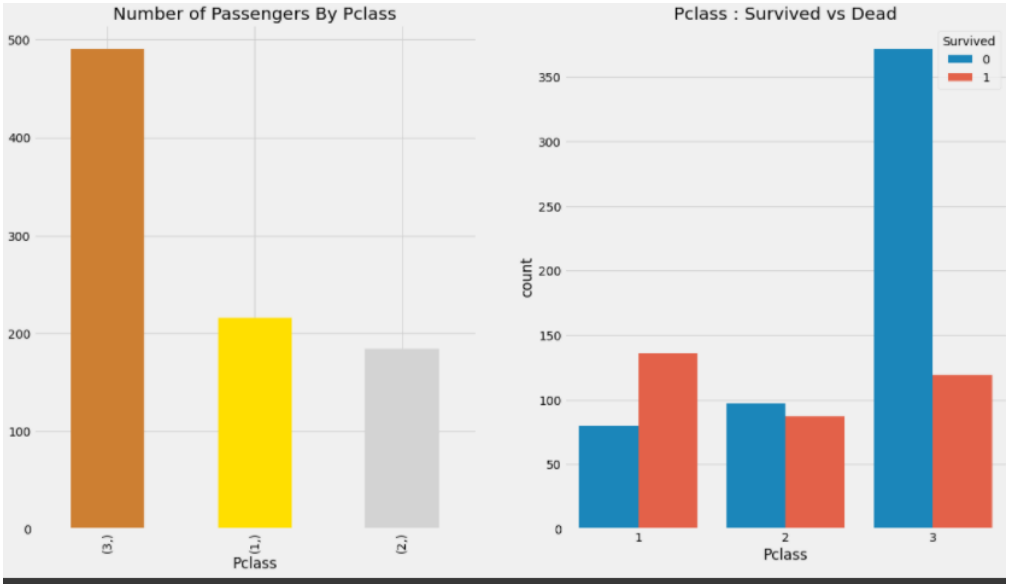
- 탐색적 데이터 분석

1. 성별(Sex)



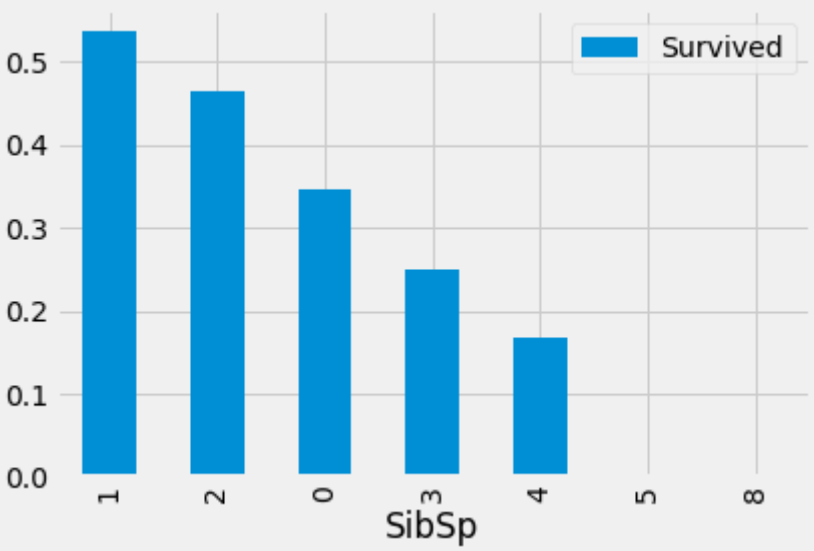
여성의 생존률이 남성에 비해 상대적으로 높음을 확인했습니다.

2. 객실 등급(Pclass)



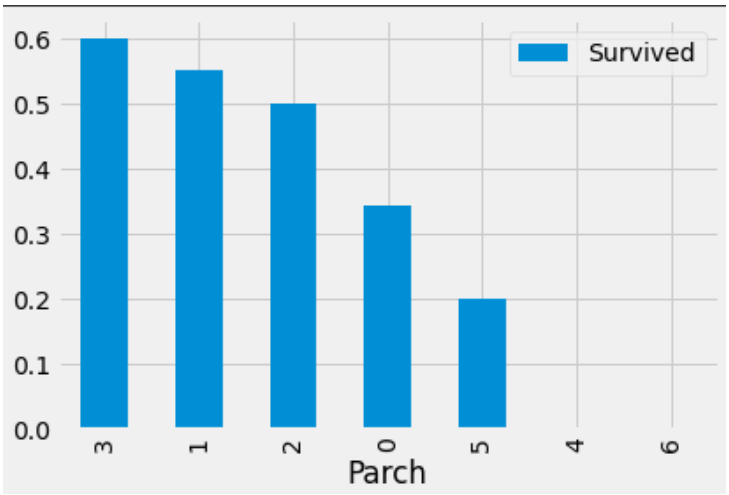
3 > 1 > 2 등석으로 탑승한 인원이 많았고,
1 > 2 > 3 등석으로 생존 비율이 높았음을 확인했습니다

3. 동반 형재 자매(SibSp)



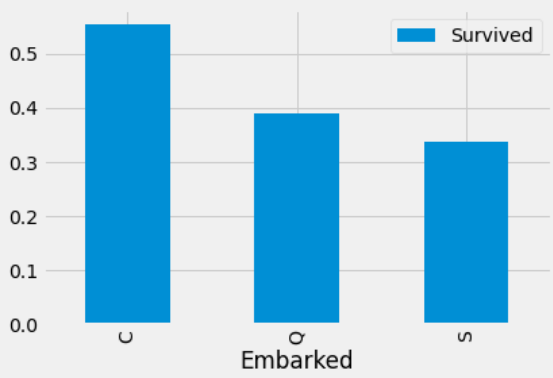
동승한 형제 및 배우자 수가 1 or 2 명일 때 생존률이 높으며, 이후 03456 순으로 확인했습니다.

3. 동반 부모 자녀(Parch)



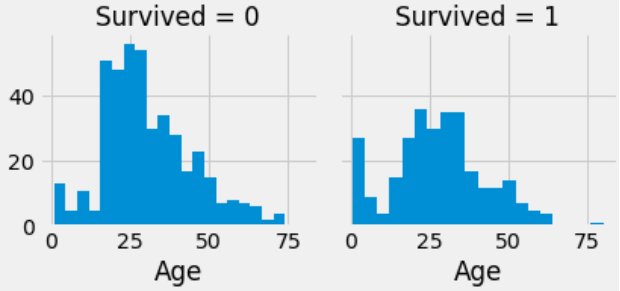
동승한 부모 자식 수가 적을수록 (3120546 순)으로 생존률이 높음을 확인했습니다.

5. 승선 항(Embarked)



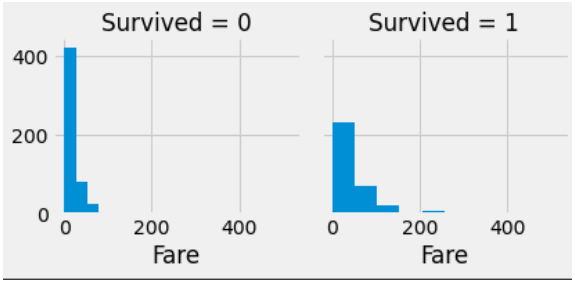

```
CQS 순으로 생존률을 보임을 확인했습니다.
```

6. 나이(Age)




```
4 세 이하 생존율 높음
15~25 생존율 높음
나이에 결측치(NULL)이 존재하고 연속 데이터이므로 변형 필요할 것으로 예상하였습니다.
```

7. 요금(Fare)




```
높은 요금과 생존이 비례하는 관계는 아닌 것으로 예상되고 별다른 상관관계를 찾지
못하였습니다
결측치가 존재하고 연속 데이터이므로 변형 필요할 것으로 예상하였습니다.
```

- Feature Engineering

Cabin, Ticket, Age

```
[357] train=train.drop(columns='Cabin')
      test=test.drop(columns='Cabin')
      train=train.drop(columns='Ticket')
      test=test.drop(columns='Ticket')
      train=train.drop(columns='Age')
      test=test.drop(columns='Age')
```

고유값으로 사용하지않을 Cabin, Ticket 변수 제거하였습니다.

Age 변수를 범주형으로 나눠 여러 예측을 해보았으나 성능 저하로 제거하기로 하였습니다.

Sex

```
train.loc[train['Sex']=='male', 'Sex']=0
train.loc[train['Sex']=='female','Sex']=1
test.loc[test['Sex']=='male','Sex']=0
test.loc[test['Sex']=='female','Sex']=1
```

성별은 male & female 로 되어 있던 data 를 numeric 하게 바꿔주었습니다.

Pclass

```
[359] train['Pclass_3']=(train['Pclass']==3)
      train['Pclass_2']=(train['Pclass']==2)
      train['Pclass_1']=(train['Pclass']==1)

      test['Pclass_3']=(test['Pclass']==3)
      test['Pclass_2']=(test['Pclass']==2)
      test['Pclass_1']=(test['Pclass']==1)

      train=train.drop(columns='Pclass')
      test=test.drop(columns='Pclass')
```

각각의 Pclass 를 나타내는 변수로 원-핫 인코딩을 진행하였습니다

Fare

```
[360] train.loc[train['Fare'].isnull(),'Fare']=0
      test.loc[test['Fare'].isnull(),'Fare']=0
```

Fare 변수 또한 범주형 변수로 밴드로 나눠 시도해봤으나 성능 향상이 되지 않았기에 Fare 결측치 0 으로 패딩하는 과정만을 진행하였습니다.

SibSp, Parch

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

[362] train['Single']=train['FamilySize']==1
      train['Mid']=(2<=train['FamilySize']) & (train['FamilySize']<=4)
      train['Big']=train['FamilySize']>=5

      test['Single']=test['FamilySize']==1
      test['Mid']=(2<=test['FamilySize']) & (test['FamilySize']<=4)
      test['Big']=test['FamilySize']>=5

[363] train=train.drop(columns=['Single','Big','SibSp','Parch','FamilySize'])
      test=test.drop(columns=['Single','Big','SibSp','Parch','FamilySize'])
```

비슷한 성질을 가진 SibSp, Parch 변수를 동해 동반 승객의 크기를 나타내는 FamilySize 를 나타내었고 FamilySize 변수 크기에 따라 나눠 학습을 진행해보니 중간 크기인 Mid 변수를 이용하는 것이 가장 좋은 성능을 나타내 나머지 변수들을 제거해주었습니다.

Embarked

```
[364] train['EmbarkedC']=train['Embarked']=='C'
      train['EmbarkedS']=train['Embarked']=='S'
      train['EmbarkedQ']=train['Embarked']=='Q'
      test['EmbarkedC']=test['Embarked']=='C'
      test['EmbarkedS']=test['Embarked']=='S'
      test['EmbarkedQ']=test['Embarked']=='Q'

      train=train.drop(columns='Embarked')
      test=test.drop(columns='Embarked')
```

Embarked 변수 또한 Pclass 변수와 같은 과정으로 원-핫인코딩을 진행하였습니다.

Name

```
train['Rename'] = train['Name'].str.extract(' ([A-Za-z]+)₩.', expand=False)
test['Rename'] = test['Name'].str.extract(' ([A-Za-z]+)₩.', expand=False)

train['Rename'] = train['Rename'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkhe

train['Rename'] = train['Rename'].replace('Mlle', 'Miss')
train['Rename'] = train['Rename'].replace('Ms', 'Miss')
train['Rename'] = train['Rename'].replace('Mme', 'Mrs')

test['Rename'] = test['Rename'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer

test['Rename'] = test['Rename'].replace('Mlle', 'Miss')
test['Rename'] = test['Rename'].replace('Ms', 'Miss')
test['Rename'] = test['Rename'].replace('Mme', 'Mrs')

train['Rename_Numeric'] = train['Rename'].astype('category').cat.codes
test['Rename_Numeric'] = test['Rename'].astype('category').cat.codes


train=train.drop(columns='Name')
test=test.drop(columns='Name')
train=train.drop(columns='Rename')
test=test.drop(columns='Rename')
```

Name 변수 또한 고유값으로 생각할 수도 있으나, Mr, Ms 등 결혼 등의 정보를 나타내는 성질이 있기 때문에 제거하지 않았고,
수가 별로 없는 Name 데이터들은 단일화시켜 5 가지로 줄인 후 Numeric 하게 바꿔주었습니다.

- 모델 구축 및 적용 및 평가

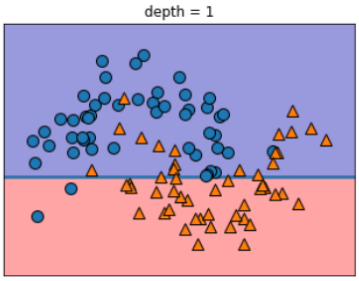
```
Y_train=train['Survived']
feature=list(test)
X_train=train[feature]
X_test=test[feature]
```

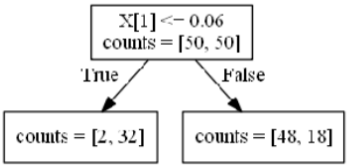
```
[517] from sklearn.tree import DecisionTreeClassifier

      model=DecisionTreeClassifier(max_depth=11, random_state=18)
      model.fit(X_train, Y_train)
      predictions=model.predict(X_test)
      submission['Survived']=predictions
      submission.to_csv('Submit.csv')
      submission.head()
```

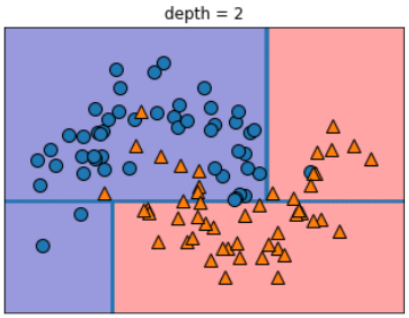
여러가지 모델로 평가를 진행해보았는데 그 중 DecisionTreeClassifier가 가장 좋은 성능을 보여 해당 모델로 학습을 진행하게 되었습니다.

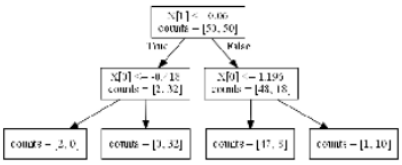
결정 트리 학습법(DecisionTreeLearning)은 수업에서 배운 Logistic 같은 분류 알고리즘으로



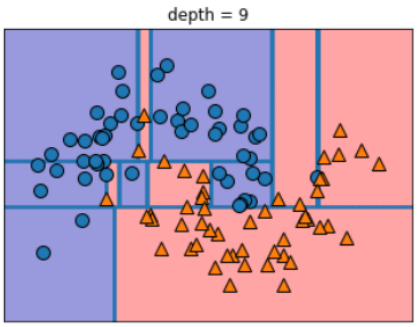


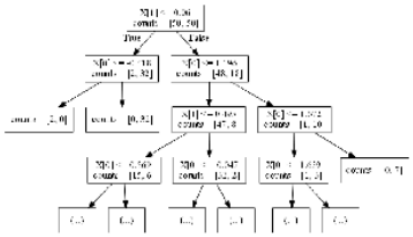
첫번째 관문 (depth=1)을 통과시키면서 True or False에 나눠주고





다음 관문(depth)에서도 같은 과정을 통해 나눠져





계속 깊이를 증가시켜 분류하는 알고리즘입니다.

Depth의 깊이를 매우 크게 설정하면 100% 학습테스트가 가능하지만 그만큼 과적합이 발생하는 알고리즘입니다.

```
[518] model.score(X_train, Y_train)

      0.9135802469135802
```

여러 max_depth로 학습을 진행해보니 max_depth=11로 설정하고 진행한 학습의 결과가 가장 좋았습니다. 하지만 kaggle과의 accuracy를 비교한 결과를 살펴보니 오차가 많이 발생하는 것으로 보아 과적합이 발생하고 있을 것이라 예상됩니다.

- Discussion & Lession

데이터 전처리 과정에서 Fare, Age 변수들을 범주형 변수화시켜 범위에 따라 카테고리화하여 학습을 진행한 결과들에서 모두 성능 향상이 나오지 않아 해당 과정을 모두 제거하게 되었습니다. 해당 과정에 대한 여운이 남아 프로젝트 이후에 더 공부하여 진행해보고 싶은 아쉬움이 남습니다.

또한, 데이터 분석하는 과정이 미숙하여 기본적인 패딩이나 원-핫 인코딩 외에는 별다른 기법이 들어 있지 않아 아쉽고, 변수들 간에도 존재하는 상관 관계라던가 모델 구축 과정에서 알게 된 앙상블 기법 등을 사용하기에는 과정 내에 숙지하지 못한 부분이 너무 많고 이후에 더 학습하고 적용해보고 싶은 생각이 남습니다.