WEEK 3

타이타닉에서는 누가 살아남았을까?

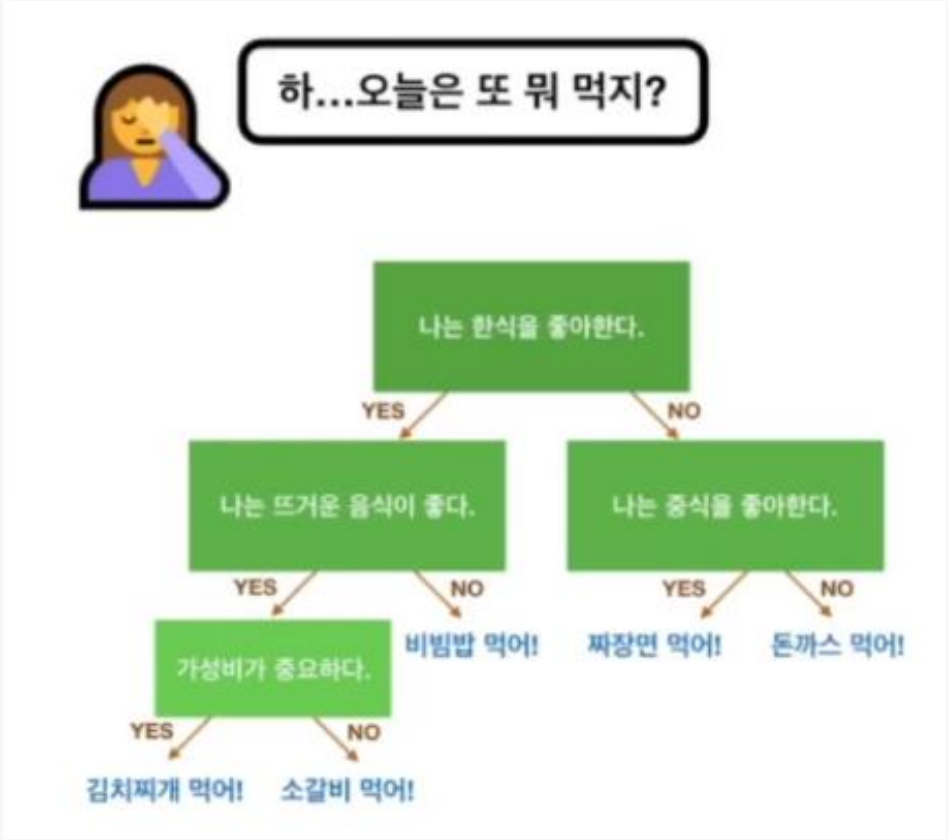
발표자 : 이하린

2020.06.24(WED)

CONTENTS

CONTENTS

01 Decision Tree



나무처럼 보여서 의사결정 나무(Decision Tree)

선택지를 둘로 남기고 여러 단계에 걸쳐 답하는 방식!

01 Decision Tree

패턴화 하는 사고 체계가 model
이를 만드는 과정은 learning
사고 체계를 decision tree로
만들 것.
데이터 테이블은 모델에 미리
제공해주어야 한다.

01 Decision Tree

카테고리 데이터 기반의 TREE 만들기

Feature Engineering

	날개 유무	몸무게	골밀도	날 수 있는가?
	no	light	low density	no
	yes	heavy	high density	no
	yes	light	high density	no
	yes	heavy	high density	no
	yes	light	low density	yes

```
In [1]:
data = {
    'name' : ['고양이','펭귄','닭','타조','참새'],
    'wing':[False,True,True,True,True],
    'weight' :['light','heavy','light','heavy','light'],
    'density':['low','high','high','high','low'],
    'fly' :[False,False,False,False,True]
}

In [2]:
print(data)

{'name': ['고양이', '펭귄', '닭', '타조', '참새'], 'wing': [False
'heavy', 'light'], 'density': ['low', 'high', 'high', 'high', 'lo

In [3]:
print('고양이는 날개가 있을까?',data['wing'][0])

고양이는 날개가 있을까? False
```

데이터 테이블에서 인덱스 맞추어서 print해주면 원하는 결과 프린트 가능

```
print(data['wing'][1],
      data['weight'][1],
      data['density'][1],
      data['fly'][1])

True heavy high False
```

01 Decision Tree

In [18]:
```
target_index = 2

print(data['name'][target_index],': 날 수 있는지 확인합니다.')

#날개 / 몸무게 / 골밀도

# 날개 유무
if data['wing'][target_index] == True :
    #날개가 있는 경우
    if data['weight'][target_index] == 'heavy':
        print("날 수 없다.")
    else:
        #몸무게가 가볍다.
        if data['density'][target_index] == 'high':
            print('날 수 없다.')
        else:
            print("날 수 있다!!")
else:
    #날개가 없는 경우
    print('날 수 없다.')
```

```
닭 : 날 수 있는지 확인합니다.
날 수 없다.
```

Target_index 이용해서 Decision Tree 구현가능

02 Feature Engineering

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.read_csv('data/train.csv')
```

```
In [4]: df.head(5)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
In [6]: df[['Sex','Age','SibSp','Parch']].head()
```

Out[6]:

	Sex	Age	SibSp	Parch
0	male	22.0	1	0
1	female	38.0	1	0
2	female	26.0	0	0
3	female	35.0	1	0
4	male	35.0	0	0

1. 판다스 불러오기
2. Csv 파일 ‘df’로 불러오고 출력.
3. 4가지 데이터만 출력하기.

02 Feature Engineering

```
In [9]: df['Age']

Out[9]: 0      22.0
        1      38.0
        2      26.0
        3      35.0
        4      35.0
               ...
        886    27.0
        887    19.0
        888     NaN
        889    26.0
        890    32.0
        Name: Age, Length: 891, dtype: float64
```

1. Test set(시험용 데이터 → 잘 만들어졌는지 확인)
2. Age의 빈칸을 평균으로 대체하기

→ NaN 표시 : 이 칸이 비어있다는 의미.

→ .fillna: 빈칸을 x로 채우기
[원본을 바꾸지는 않음]

```
In [18]: df['Age']= df['Age'].fillna(df['Age'].mean())

In [13]: df['Age'].mean()

Out[13]: 29.69911764705882

In [19]: df

Out[19]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	NaN	Q

02 Feature Engineering

```
In [22]: df_test['Age']= df_test['Age'].fillna(df_test['Age'].mean())
```

```
In [25]: df.loc[df['Age']<10,'Age'] = 0
         df.loc[(df['Age'] >= 10) & (df['Age'])<20,'Age'] = 1
         df.loc[(df['Age'] >= 20) & (df['Age'])<30,'Age'] = 2
         df.loc[(df['Age'] >= 30) & (df['Age'])<40,'Age'] = 3
         df.loc[(df['Age'] >= 40) & (df['Age'])<50,'Age'] = 4
         df.loc[df['Age'] >= 50, 'Age'] = 5
```

```
In [26]: df_test.loc[df_test['Age']<10,'Age'] = 0
         df_test.loc[(df_test['Age'] >= 10) & (df_test['Age'])<20,'Age'] = 1
         df_test.loc[(df_test['Age'] >= 20) & (df_test['Age'])<30,'Age'] = 2
         df_test.loc[(df_test['Age'] >= 30) & (df_test['Age'])<40,'Age'] = 3
         df_test.loc[(df_test['Age'] >= 40) & (df_test['Age'])<50,'Age'] = 4
         df_test.loc[df_test['Age'] >= 50, 'Age'] = 5
```

1. df.loc[조건, 열] = 넣고 싶은 값
2. 조건이 2개면?
   ➔ [(조건1)&(조건2),열]
3. df_test에도 적용시키기
4. ctrl+v ➔ edit ➔ find and replace에서
   Df ➔ df_test

02 Feature Engineering

```
In [27]: df['FamilySize']=df['SibSp'] + df['Parch']
         df.head()
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Se
0	1	0	3	Braund, Mr. Owen Harris	mal
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	femal
2	3	1	3	Heikkinen, Miss. Laina	femal
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	femal
4	5	0	3	Allen, Mr. William Henry	mal

```
In [30]: train = df[['Survived','Sex','Age','FamilySize']]
         test = df[['Sex','Age','FamilySize']]
         train.head()
```

Out[30]:

	Survived	Sex	Age	FamilySize
0	0	male	4.0	1
1	1	female	4.0	1
2	1	female	4.0	0
3	1	female	4.0	1
4	0	male	4.0	0

1. Family size라는 열 새로 만들기
2. 필요 없는 열 삭제하기

➔ 필요한 열만 선택하기

02 Feature Engineering

함께 실습 7

```
In [43]: df.isnull().sum()
```

```
Out[43]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

```
In [49]: df["Embarked"].value_counts()
```

```
Out[49]: S    644
         C    168
         Q     77
         Name: Embarked, dtype: int64
```

1. Isnull ➔ 빈칸을 찾아라
2. Sum ➔ 빈칸의 개수를 세라

Cabin 의 빈칸은 687

3. .value_counts ➔ 값 구해라

03 Scikit-learn

```
In [32]:
x_train = train[['Sex','Age','FamilySize','Fare','Embarked']]
y_train = train["Survived"]

x_train
```

Out[32]:

	Sex	Age	FamilySize	Fare	Embarked
0	0	4.0	1	7.2500	0
1	1	4.0	1	71.2833	1
2	1	4.0	0	7.9250	0
3	1	4.0	1	53.1000	0
4	0	4.0	0	8.0500	0
...	...	...	...	...	...
886	0	4.0	0	13.0000	0
887	1	4.0	0	30.0000	0
888	1	4.0	3	23.4500	0
889	0	4.0	0	30.0000	1
890	0	4.0	0	7.7500	2

891 rows × 5 columns

Scikit – learn : 라이브러리
데이터를 두 부분으로 나누는 것부터 시작!

데이터를 잘 공부해서 살았는지 죽었는지 판단
최종적으로 알고 싶은 데이터가 : output,y,target
나머지 : input,x

➔ input과 output적절히 나누어야 한다!

03 Scikit-learn

```
In [35]: from sklearn.tree import DecisionTreeClassifier

         tree = DecisionTreeClassifier()
         tree.fit(x_train,y_train)

         tree.score(x_train,y_train)

Out[35]: 0.9191919191919192
```

Decision tree 학습
Scikit-learn에서 decision tree해당하는 것만 import

1주차에서 황금계수 찾는 것 4줄로 가능!
Train 데이터 이용하면 91%의 정확도로 생존여부 맞출 수 있다.
X_train: 가지고 있던 부분에서 survived제외한 것
Y-train: survived만

03 Scikit-learn

In [46]:
```
x_test = test[['Sex','Age','FamilySize','Fare','Embarked']]

x_test.head(5)
```

Out[46]:

	Sex	Age	FamilySize	Fare	Embarked
0	0	4.0	0	7.8292	2
1	1	4.0	1	7.0000	0
2	0	4.0	0	9.6875	2
3	0	4.0	0	8.6625	0
4	1	4.0	2	12.2875	0

In [34]:
```
prediction = tree.predict(x_test)
prediction
```

Out[34]:
```
array([0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0])
```

In [35]:
```
x_test
```

Out[35]:

	Sex	Age	FamilySize	Fare	Embarked
0	0	4.0	0	7.8292	2
1	1	4.0	1	7.0000	0
2	0	4.0	0	9.6875	2
3	0	4.0	0	8.6625	0
4	1	4.0	2	12.2875	0
...	...	...	...	...	...
413	0	4.0	0	8.0500	0
414	1	4.0	0	108.9000	1
415	0	4.0	0	7.2500	0
416	0	4.0	0	8.0500	0
417	0	4.0	2	22.3583	1

Tree.predict(____)

0번 → 0

1번 → 1

2번 → 0

03 Scikit-learn

In [36]:
```
submit = pd.DataFrame({
    'PassengerId' : df_test['PassengerId'],
    'Survived' : prediction
})

submit.to_csv('submit.csv', index = False)
```

In [47]:
```
my_prediction = pd.read_csv('submit.csv')
my_prediction
```

Out[47]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

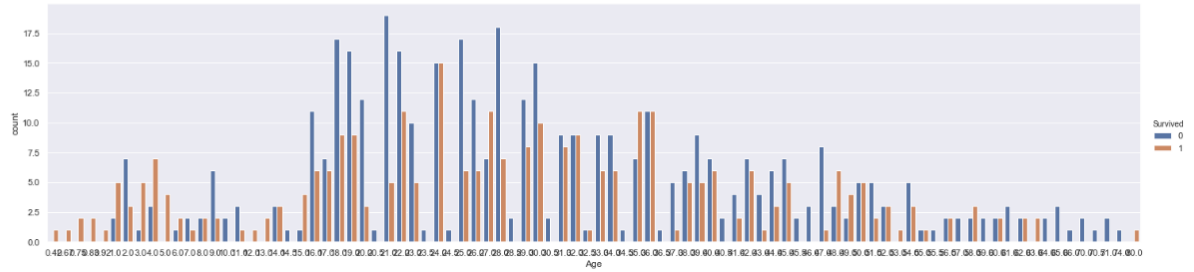
418 rows × 2 columns

CSV 파일 만들고 제출

04 Kdeplot 그리기

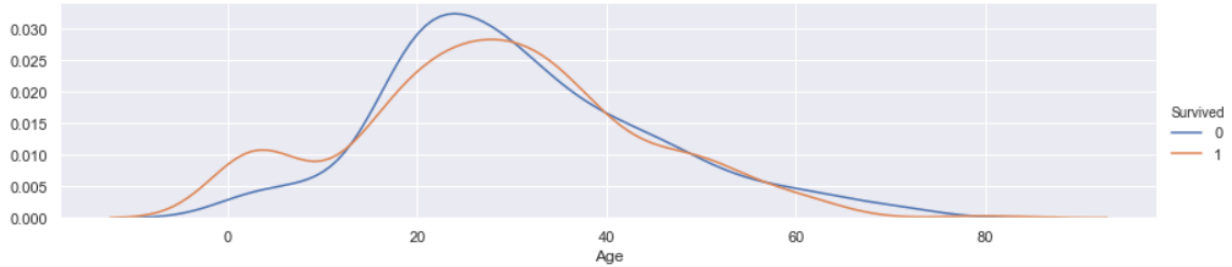
```
In [5]: import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set()

In [7]: sns.catplot(data=df, x='Age', hue ='Survived', kind='count', aspect = 4)
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1719712f208>
```




```
facet = sns.FacetGrid(df,hue='Survived', aspect=4)
facet.map(sns.kdeplot,'Age')
facet.add_legend()

plt.show()
```



나이에 따른 생존여부 히스토그램 그리기

1. Csv 파일 불러오기
2. Catplot 그리기

➔ X 값 읽을 수 없음....

➔ X축이 고르지 않음....

Kdeplot 이용해서 매끄러운 분포의 히스토그램 그리기

0살~11살까지가 1세대
11살 부터 29까지가 2세대
29~40까지가 3세대
밀도 분포가 크게 바뀌는 부분에서 새로운 세대 정의
→ decision tree의 성능에 좋은 영향 미칠 수도!

THANK YOU-