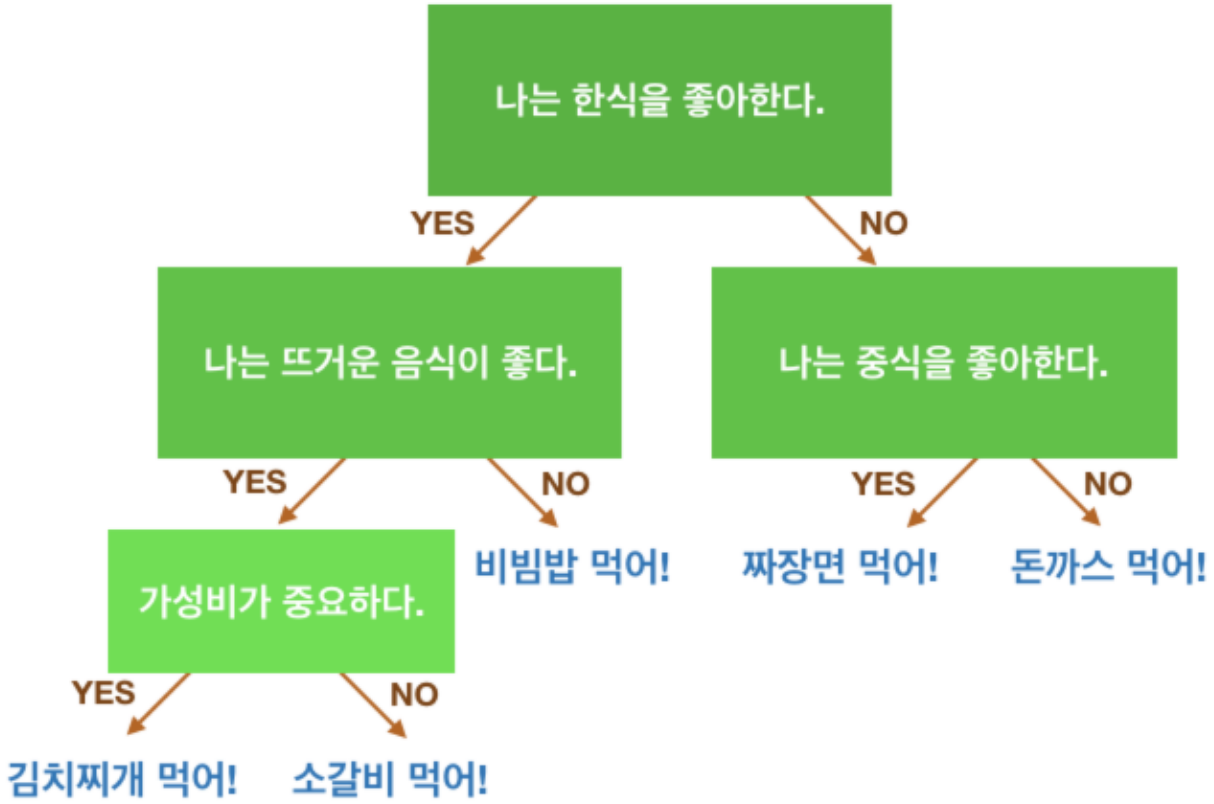
Week 3

Decision Tree로 타이타닉 생존자 찾기

Decision Tree = 컴퓨터의 사고 체계

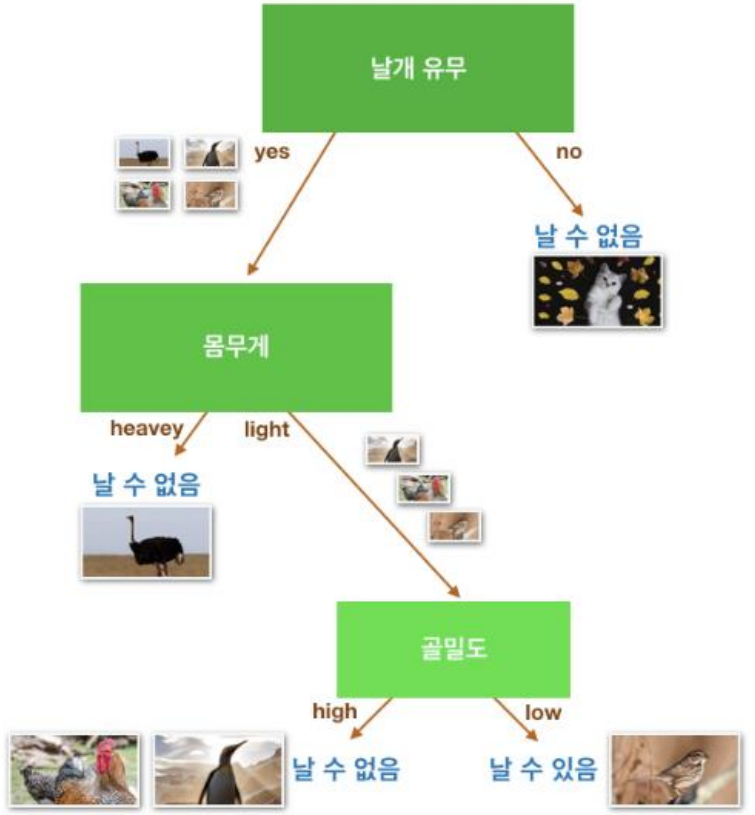


Stage 1. Decision Tree 배우기

Feature Engineering

	날개 유무	몸무게	골밀도	날 수 있는
	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

Train Data



함께 실습 No.1

```
In [16]: data = {
             'wing': [False, True, True, True, True],
             'weight': ['light', 'heavy', 'light', 'heavy', 'light'],
             'density': ['low', 'high', 'high', 'high', 'low'],
             'fly': [False, False, False, False, True]
         }
```

executed in 3ms, finished 16:42:40 2019-01-04

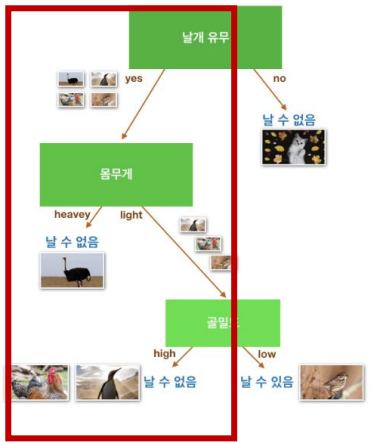
```
In [15]: # 코드작성이 끝나면 target_index를 0부터 4까지 변경하면서 실행해보세요.
         target_index = 0
         target_names = ['고양이', '펭귄', '닭', '타조', '참새']

         print(target_names[target_index], ': 날 수 있는지 확인합니다.')
         # 날개 유무
         if data['wing'][target_index]: # 날개 있음
             # 몸무게
             if data['weight'][target_index] == 'heavy': # 몸무게 heavy
                 print('날 수 없다')
             else: # 몸무게 light
                 # 골밀도
                 if data['density'][target_index] == 'high': # 골밀도 high
                     print('날 수 없다')
                 else: # 골밀도 low
                     print('날 수 있다!!!')
         else: # 날개 없음
             print('날 수 없다')
```

executed in 6ms, finished 16:41:57 2019-01-04

```
고양이 : 날 수 있는지 확인합니다.
날 수 없다
```

숫자를 0부터 4까지 바꾸면서 동물 이름 변경 가능
ex) 0일 때 고양이, 1일 때 펭귄

이전 도표의 좌측 부분에 해당함



Stage 2. Feature Engineering

```
In [74]: import pandas as pd
```
→ pandas로 데이터 불러오기

```
In [75]: df = pd.read_csv('data/train.csv')
         df.head(10)
```
→ 위의 10줄 데이터만 불러오기

Out[75]:

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Stage 2. Feature Engineering

```
In [76]: df[['Sex', 'Age', 'SibSp', 'Parch']].head(6)
```

→ head 안에 아무 숫자도 넣지 않으면 5로 자동 인식함

Out[76]:

	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
5	male	NaN	0	0

대괄호는 두 번 입력해야 함

```
In [77]: #testset 불러오기
         df_test = pd.read_csv('data/test.csv')
         df_test.head()
```

→ train 데이터와 test 데이터의 차이: survived 데이터의 유무!

Out[77]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

함께 실습 3. Age의 빈칸(NaN)을 평균값으로 채우기

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0
5	6	0	3	Moran, Mr. James	male	NaN	0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0

* train 셋과 test 셋이 동일한 값을 가질 수 있도록
해당 과정을 df_test 셋에도 동일하게 적용해주어야 함.

fillna는 데이터 값을 영구적으로 변경하지 않기 때문에
앞에 df['Age'] = 을 꼭 추가해야 함

```
In [70]: df['Age'] = df['Age'].fillna(df['Age'].mean())
```

나이 평균값을 의미

함께 실습 4. Age의 범위 묶기

In [80]:
```
df_test['Age'] = df_test['Age'].fillna(df_test['Age'].mean())
```

In [81]:
```
#df.loc[조건,열] = 넣고싶은 값

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 [83]:
```
#df_test.loc[조건,열] = 넣고싶은 값

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
```

df_test도 동일하게 실행

In [84]:
```
df
```

* 여러 셀 동일하게 변경하기

: edit → find and replace → df 입력
→ df_test로 변경

함께 실습 4 .새로운 열 만들기 & 필요 없는 열 삭제하기

In [85]:
```
df['familysize'] = df['SibSp'] + df['Parch']
df.head()
```

Out[85]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	familysize
0	1	0	3	Braund, Mr. Owen Harris	male	2.0	1	0	A/5 21171	7.2500	NaN	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	3.0	1	0	PC 17599	71.2833	C85	C	1
2	3	1	3	Heikkinen, Miss. Laina	female	2.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	3.0	1	0	113803	53.1000	C123	S	1
4	5	0	3	Allen, Mr. William Henry	male	3.0	0	0	373450	8.0500	NaN	S	0

In [87]:
```
train = df[['Survived', 'Sex', 'Age', 'familysize']]
test = df_test[['Sex', 'Age', 'familysize']]

train.head()
```

원하는 데이터 이름만 나열

Out[87]:

	Survived	Sex	Age	familysize
0	0	male	2.0	1
1	1	female	3.0	1
2	1	female	2.0	0
3	1	female	3.0	1
4	0	male	3.0	0

함께 실습 7. 빈칸 정보, 값 요약 보기

```
In [93]: df.isnull().sum()
```

isnull = 빈칸 찾기

```
Out[93]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age              0
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         familysize       0
         dtype: int64
```

0 = 빈칸이 없음

```
In [94]: df_test.isnull().sum()
```

```
Out[94]: PassengerId      0
         Pclass           0
         Name             0
         Sex              0
         Age              0
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          327
         Embarked         0
         familysize       0
         dtype: int64
```

```
In [95]: df['Embarked'].value_counts()
```

값 찾기

```
Out[95]: S    644
         C    168
         Q     77
         Name: Embarked, dtype: int64
```

Challenge 1. 문자를 숫자 데이터로 변경하기

해당 과정을 진행하기 전 데이터를 복사한 후 Kernal → Restart & Run을 눌러 모든 셀을 다시 시작해 줌

In [26]:
```
#df.loc[조건, 선택할 열] = 값
df.loc[df['Sex'] == 'male', 'Sex'] = 0
df.loc[df['Sex'] == 'female', 'Sex'] = 1

df
```

파이썬에서는 ==이 '같다'는 의미로 통용됨!

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	familysize
0	1	0	3	Braund, Mr. Owen Harris	0	2.0	1	0	A/5 21171	7.2500	NaN	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	3.0	1	0	PC 17599	71.2833	C85	C	1
2	3	1	3	Heikkinen, Miss. Laina	1	2.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	3.0	1	0	113803	53.1000	C123	S	1
4	5	0	3	Allen, Mr. William Henry	0	3.0	0	0	373450	8.0500	NaN	S	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	0	2.0	0	0	211536	13.0000	NaN	S	0
887	888	1	1	Graham, Miss. Margaret Edith	1	1.0	0	0	112053	30.0000	B42	S	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	1	2.0	1	2	W./C. 6607	23.4500	NaN	S	3
889	890	1	1	Behr, Mr. Karl Howell	0	2.0	0	0	111369	30.0000	C148	C	0
890	891	0	3	Dooley, Mr. Patrick	0	3.0	0	0	370376	7.7500	NaN	Q	0

해당 과정을 Embarked(S,C,Q)에도 동일하게 적용하고, df_test 셋도 다시 한 번 돌려줌

Stage 3. 사이킷런 배우기

* 데이터를 먼저 두 부분으로 나누는 과정이 필요함
output = target = y_data
input = x_data

```
In [34]: x_train = train[['Sex', 'Age', 'familysize', 'Fare', 'Embarked']]
         y_train = train['Survived']

         x_train
```

x 데이터에 해당하는 값만 출력

Out[34]:

	Sex	Age	familysize	Fare	Embarked
0	0	2.0	1	7.2500	1
1	1	3.0	1	71.2833	2
2	1	2.0	0	7.9250	1
3	1	3.0	1	53.1000	1
4	0	3.0	0	8.0500	1
...	...	...	...	...	...
886	0	2.0	0	13.0000	1
887	1	1.0	0	30.0000	1
888	1	2.0	3	23.4500	1
889	0	2.0	0	30.0000	2
890	0	3.0	0	7.7500	0

```
In [35]: y_train
```

y 데이터에 해당하는 survived 값만 출력

```
Out[35]: 0      0
         1      1
         2      1
         3      1
         4      0
                ..
         886    0
         887    1
         888    0
         889    1
         890    0
         Name: Survived, Length: 891, dtype: int64
```

Stage 3. 데이터 결과 예측하기

```
In [36]: from sklearn.tree import DecisionTreeClassifier

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

         tree.score(x_train, y_train)
```

```
Out[36]: 0.9450056116722784
```

→ 1주차 때 배웠던 황금계수 찾는 과정과 동일

```
In [37]: x_test = test[['Sex', 'Age', 'familysize', 'Fare', 'Embarked']]

         x_test
```

Out[37]:

	Sex	Age	familysize	Fare	Embarked
0	0	3.0	0	7.8292	0
1	1	4.0	1	7.0000	1
2	0	5.0	0	9.6875	0
3	0	2.0	0	8.6625	1
4	1	2.0	2	12.2875	1
...	...	...	...	...	...
413	0	3.0	0	8.0500	1
414	1	3.0	0	108.9000	2
415	0	3.0	0	7.2500	1
416	0	3.0	0	8.0500	1
417	0	3.0	2	22.3583	2

```
In [40]: prediction = tree.predict(x_test)
         prediction
```

```
Out[40]: array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
                1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
                1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 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, 1,
                0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 1, 1, 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, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
                0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
                1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
                1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
                1, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
                0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
                1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 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, 1, 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, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0],
               dtype=int64)
```

0 = 사망
1 = 생존 ←

Stage 3. kaggle에 정확한 결과 제출하기

In [44]:
```
submit = pd.DataFrame ({
    'PassengerId' : df_test['PassengerId'],
    'Survived' : prediction
})
submit.to_csv('submit.csv', index=False)
```

앞에서 작성한 prediction 값 추가

csv 파일로 변환 후 제출

In [46]:
```
my_prediction = pd.read_csv('submit.csv')
my_prediction.head()
```

Out[46]:

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

Stage 4. Count Plot으로 나이에 따른 생존 여부 히스토그램 얻기

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

→ 데이터 불러오기

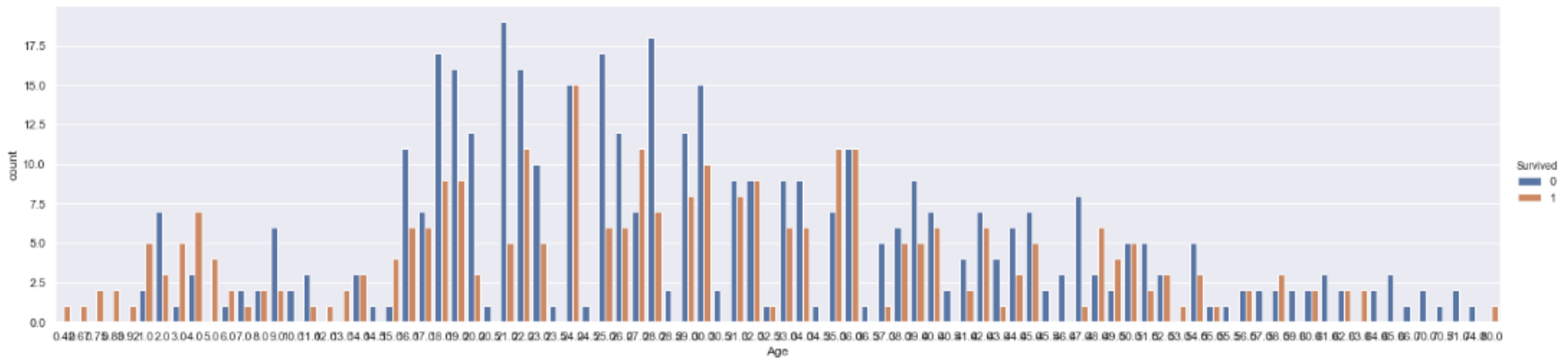
```
In [3]: df = pd.read_csv('data/train.csv')
        df
```

```
In [4]: import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set()
```

→ 그래프 그리기

```
In [5]: sns.catplot(data=df, x='Age', hue='Survived', kind='count', aspect=4)
```

```
Out[5]: <seaborn.axisgrid.FacetGrid at 0x26115eaa308>
```

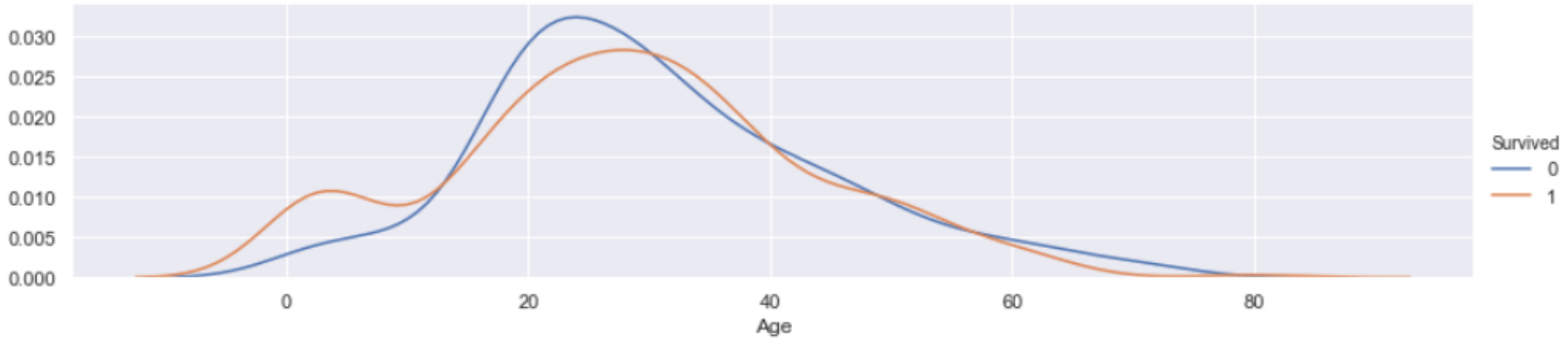


문제점 1: x축의 값이 너무 많이 가시성이 떨어짐
문제점 2: x축의 값이 고르게 분포 되지 않음

Stage 4. Count Plot으로 나이에 따른 생존 여부 히스토그램 얻기

```
In [8]: facet = sns.FacetGrid(df, hue='Survived', aspect=4)
        facet.map(sns.kdeplot, 'Age')
        facet.add_legend()

        plt.show()
```

조금 더 매끄러운 분포의 그래프를 그릴 수 있는 패키지



Challenge 2. boxplot 그리기

```
In [3]: import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set()
```

kind를 box로 변경하면 됨

```
In [6]: sns.catplot(data=df, x='Embarked', y='Age', hue='Sex', kind='violin', aspect=4)
Out[6]: <seaborn.axisgrid.FacetGrid at 0x1aea8677b08>
```