파이썬과 판다스를 활용한 실전 데이터 분석

이번 과제는 캐글의 타이타닉(Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic/)) 데이터를 더 깊게 분석합니다.

데이터 사이언티스트로서 데이터를 분석할 때 가장 필수적인 스킬은 프로그래밍 언어 파이썬(Python (https://www.python.org/)), 데이터 분석 패키지 판다스(Pandas (https://pandas.pydata.org/)), 그리고 데이터 시각화 패키지 씨본(Seaborn (https://seaborn.pydata.org))과 matplotlib (https://matplotlib.org)입니다. 데이터 분석가는 언제나 주변 동료들의 요청(ex: 운영팀, 재무팀, 마케팅팀)에 맞게 데이터를 뽑아내 그 통계치를 제공하고 시각화 결과를 전달해줘야 하는데, 파이썬과 판다스, 시각화 스킬이 부족하면 주어진 시간 내에 이를 전달해 줄 수 없습니다.

이런 문제가 생기지 않기 위해, **모든 데이터 사이언티스트는 데이터를 능숙하게 다룰 수 있는 파이썬과 판다스, 시각화 스킬을 필수적으로 보유하고 있어야 합니다.**

이번 노트북에는 타이타닉 데이터를 활용하여, 현업에서 충분히 발생할 만한 사례를 모아 총 12개의 문제를 만들어 보았습니다. 주어진 시간 내에 모든 문제를 해결해보세요. DS School의 내부 테스트 결과, 현업에서 데이터 사이언티스트로 일 하고 있는 분들은 아무리 늦어도 반나절(3~4시간) 내에는 모든 문제를 풀 수 있었습니다. 즉, 3시간 안에 모든 문제를 풀 수 있다면 합격입니다.

문제를 풀 때 다른 자료(DS School 입문반에서 제공한 자료, 또는 판다스, 시각화 전문 서적)를 참고하거나, 구글에 검색하는 것 모두 허용합니다. (문제 중에는 구글에 검색하지 않으면 풀 수 없는 문제도 몇 개 준비해놓았습니다) 관련 자료는 10 minutes to pandas (https://pandas.pydata.org/pandas-docs/stable/10min.html) 를 강력 추천합니다.

In [1]:

```
# 데이터 시각화 패키지 matplotlib에게 inline출력,
# 즉 시각화 결과를 파일로 저장하거나 하지 않고 화면에 바로 출력하도록 명령합니다.
# (Seaborn이 matplotlib를 기반으로 동작하기 때문에, Seaborn에도 동일한 명령이 전달됩니다.)
%matplotlib inline

# 파이썬의 데이터 분석 패키지 판다스(Pandas)를 가져오고, 이를 pd라는 이름의 축약어로 사용합니다.
import pandas as pd

# 파이썬의 데이터 시각화 패키지 시본(Seaborn)을 가져오고, 이를 sns라는 이름의 축약어로 사용합니다.
import seaborn as sns
```

데이터 읽어오기

In [2]:

```
# train.csv 파일을 읽어옵니다. 여기서 PassengerId라는 컬럼을 인덱스(index)로 지정한 뒤, trai
n 변수에 할당합니다.
# 변수에 할당한 결과값을 판다스 전문 용어로 데이터프레임(DataFrame)이라고 부릅니다.
train = pd.read_csv("data/train.csv", index_col="PassengerId")

# train 변수에 할당된 데이터의 행렬 사이즈를 출력합니다.
# 출력은 (row, column) 으로 표시됩니다.
print(train.shape)

# head()로 train 데이터의 상위 5개를 출력합니다.
train.head()
```

```
(891, 11)
```

Out[2]:

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

데이터 정리 + 기본 분석

1. 타이타닉의 train 데이터에서 1) 전체 생존률과 2) 생존자의 총 인원수, 사망자의 총 인원수를 출력해주세요.

1번(생존률)의 경우 약 38.4%가 나와야 하며, 2번(인원수)의 경우 생존자의 총 인원수는 342명, 사망자의 총 인원수는 549명이 나와야 합니다.

In [3]:

```
# 타이타닉의 train데이터에서 Survived 컬럼의 평균을 구합니다.
# 그 결과를 survived_rate라는 이름의 변수에 저장합니다.
survived_rate = train["Survived"].mean()

# survived_rate는 현재 0.0 ~ 1.0 사이의 값을 갖습니다.
# 하지만 퍼센티지(%)는 0 ~ 100.0 사이의 값을 가지므로, survived_rate에 100을 곱해줍니다.
survived_rate = survived_rate * 100

# survived_rate를 출력합니다. 결과는 38.4%가 나와야 합니다.
print(f"생존률 = {survived_rate:.1f}%")
```

```
생존률 = 38.4%
```

In [4]:

```
# pandas의 value_counts를 활용하여 생존자의 총 인원수와 사망자의 총 인원수를 출력합니다.
# 생존자의 총 인원수(1)은 342명, 사망자의 총 인원수(0)는 549명이 나와야 합니다.
train["Survived"].value_counts()
```

Out[4]:

```
0    549
1    342
Name: Survived, dtype: int64
```

2. Survived 컬럼에 들어가 있는 값을 쉬운 표현으로 바꿔주세요.

Survived 컬럼에는 0(사망)이라는 값과 1(생존)이라는 값이 있습니다. 이 표현은 직관적이지 않기 때문에, 데이터 분석을 원활하게 하기 위해서는 사람이 읽기 쉬운 표현을 쓰는 것이 좋습니다.

In [5]:

```
# Survived 컬럼의 상위 5개의 값을 출력합니다.
# 결과값은 0과 1이 나오는데, Survived 컬럼에 대한 사전 설명(가령 0이 어떤 값을 나타내는지, 1
이 어떤 값을 나타내는지)
# 을 듣지 않으면 이 값이 어떠한 의미를 가지는지 직관적으로 이해하기 어렵습니다.
train["Survived"].head()
```

Out[5]:

```
PassengerId
1    0
2    1
3    1
4    1
5    0
Name: Survived, dtype: int64
```

가령 저라면 **Survived(humanized)**라는 새로운 컬럼을 만들겠습니다. 이 컬럼에는 0(사망), 1(생존)이 아닌, Perish(사망), Survived(생존)이라는 값이 들어가 있다면 좋겠습니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	Survived	Survived(humanized)
1	0	Perish
2	1	Survived
3	1	Survived
4	1	Survived
5	0	Perish

In [6]:

```
# 먼저 Survived 컬럼이 0인 승객을 색인합니다. 이후 Survived(humanized)라는 이름의
# 새로운 컬럼을 만들어 여기에 Perish 라는 값을 넣습니다.
train.loc[train["Survived"] == 0, "Survived(humanized)"] = "Perish"

# 비슷하게 Survived 컬럼이 1인 승객을 찾아 Survived(humanized)에 Survived라는 값을 넣습니다.
train.loc[train["Survived"] == 1, "Survived(humanized)"] = "Survived"

# train 변수에 할당된 데이터의 행렬 사이즈를 출력합니다.
# 출력은 (row, column) 으로 표시됩니다.
print(train.shape)

# Survived 컬럼과 Survived(humanized) 컬럼 두 개를 출력하여 비교합니다.
train[["Survived", "Survived(humanized)"]].head()
```

```
(891, 12)
```

Out[6]:

PassengerId	Survived	Survived(humanized)
1	0	Perish
2	1	Survived
3	1	Survived
4	1	Survived
5	0	Perish

내지는 이런 방식을 사용할 수 있습니다.

In [7]:

```
# Survived 컬럼이 0인 값을 Perish로, 1인 값을 Survived로 대체(replace)합니다.
train["Survived(humanized)"] = train["Survived"].replace(0, "Perish").replace(1, "Survived")

# train 변수에 할당된 데이터의 행렬 사이즈를 출력합니다.
# 출력은 (row, column) 으로 표시됩니다.
print(train.shape)

# Survived 컬럼과 Survived(humanized) 컬럼 두 개를 출력하여 비교합니다.
train[["Survived", "Survived(humanized)"]].head()
```

```
(891, 12)
```

Out[7]:

PassengerId	Survived	Survived(humanized)
1	0	Perish
2	1	Survived
3	1	Survived
4	1	Survived
5	0	Perish

또한 이번에는 Survived 컬럼이 아닌 아닌 새롭게 만든 Survived(humanized) 컬럼으로 생존자의 총 인원수와 사망자의 총 인원수를 출력해 주세요. 앞서 사용한 `value_counts` 를 그대로 사용하면 될 것 같습니다.

In [8]:

```
# pandas의 value_counts를 활용하여 생존자의 총 인원수와 사망자의 총 인원수를 출력합니다.
# 여기서 생존 여부는 Survived가 아닌 Survived(humanized) 컬럼을 사용합니다.
# 생존자의 총 인원수(Survived)은 342명, 사망자의 총 인원수(Perish)는 549명이 나와야 합니다.
train["Survived(humanized)"].value_counts()
```

Out[8]:

```
Perish      549
Survived    342
Name: Survived(humanized), dtype: int64
```

3. Pclass 컬럼에 들어가 있는 값을 읽기 쉬운 표현으로 바꿔주세요.

Pclass도 마찬가지로 1, 2, 3이라는 표현은 직관적이지 않기 때문에, 사람이 이해하기 쉬운 표현으로 바꿔주고 싶습니다.

In [9]:

```
# pandas의 pivot_table을 활용하여 Pclass별 생존률을 출력합니다.
# 여기서 Pclass값이 1, 2, 3이 나오는데, Pclass 컬럼에 대한 사전 설명을 듣지 않으면 이해하기 어렵습니다.
# 그러므로 Pclass값을 조금 더 직관적으로 바꿔준다면 pivot_table로 분석하기 편할 것입니다.
pd.pivot_table(data=train, index="Pclass", values="Survived")
```

Out[9]:

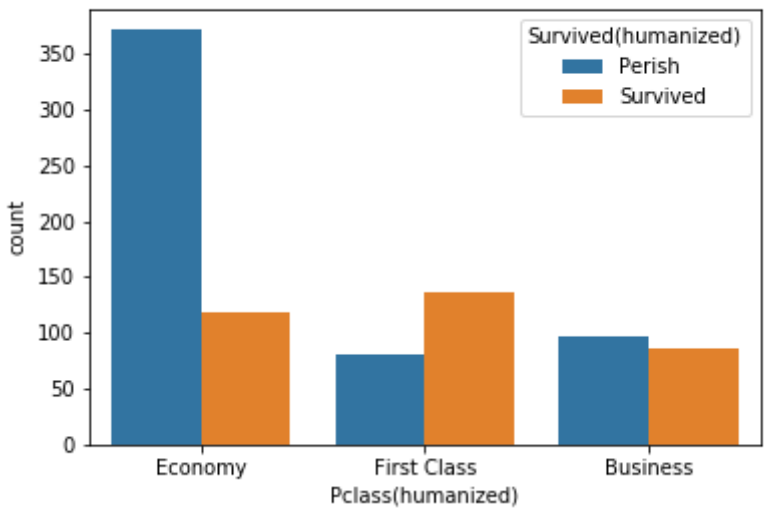
Pclass	Survived
1	0.629630
2	0.472826
3	0.242363

(판다스 pivot table에 대한 설명은 다음의 링크 (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.pivot_table.html)에서 살펴봐주세요)

이번에는 **Pclass(humanized)**라는 새로운 컬럼을 만들어주세요. 이 컬럼에는 1, 2, 3이 아닌 First Class, Business, Economy 라는 값이 들어가 있다면 좋겠습니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	Pclass	Pclass(humanized)
1	3	Economy
2	1	First Class
3	3	Economy
4	1	First Class
5	3	Economy

또한 위 내용을 바탕으로 **Pclass(humanized)**별 생존자와 사망자의 차이를 시각화해주세요. 최종적으로는 다음의 결과가 나와야 합니다.



In [ ]:

```
# Write your code here!
```

4. Embarked 컬럼에 들어가 있는 값을 읽기 쉬운 표현으로 바꿔주세요.

Embarked 컬럼도 마찬가지로 C, S, Q라는 표현은 직관적이지 않습니다. 저라면 사람이 조금 더 이해하기 쉽게끔 C는 Cherbourg 라는 표현으로, S는 Southampton 이라는 표현으로, 그리고 Q는 Queenstown 이라는 표현으로 바꾸겠습니다.

In [10]:

```
# pandas의 pivot_table을 활용하여 Embarked 별 생존률을 출력합니다.
# 여기서도 Embarked 컬럼이 C, S, Q라는 다소 직관적이지 않은 값이 나옵니다.
# 그러므로 Embarked 컬럼의 값도 Pclass 처럼 직관적으로 바꿔주고 싶습니다.
pd.pivot_table(data=train, index="Embarked", values="Survived")
```

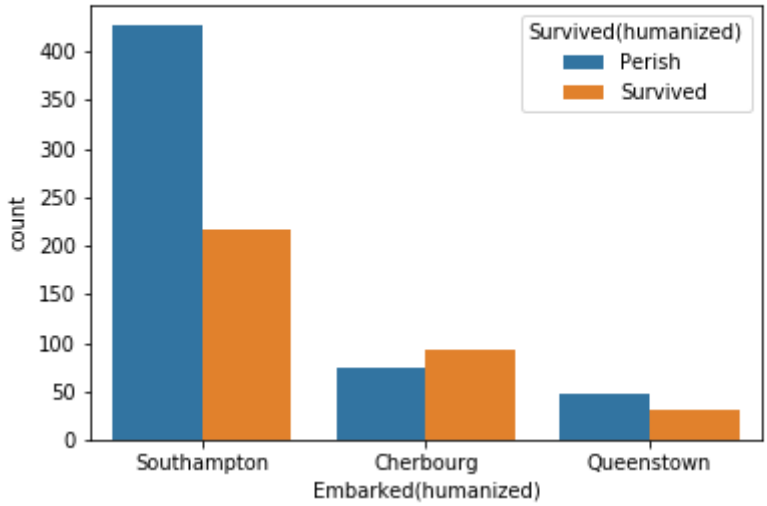
Out[10]:

Embarked	Survived
C	0.553571
Q	0.389610
S	0.336957

Survived(humanized), Pclass(humanized)와 마찬가지로, Embarked 컬럼도 **Embarked(humanized)**라는 이름의 새로운 컬럼을 만들어주세요. 이 컬럼에는 C, S, Q가 아닌 Cherbourg, Southampton, Queenstown이라는 값이 들어갑니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	Embarked	Embarked(humanized)
1	S	Southampton
2	C	Cherbourg
3	S	Southampton
4	S	Southampton
5	S	Southampton

또한 위 내용을 바탕으로 **Embarked(humanized)**별 생존자와 사망자의 차이를 시각화해주세요. 최종적으로는 다음의 결과가 나와야 합니다.



In [ ]:

```
# Write your code here!
```

나이(Age) 컬럼 분석

5. 나이(Age) 컬럼에서 다음의 정보를 출력해주세요.

- 평균(mean)
- 가장 나이가 많은 사람. (max)
- 가장 나이가 적은 사람. (min)

가령 평균은 약 29.7세, 가장 어린 사람은 0.42세(약 생후 4개월), 가장 나이가 많은 사람은 80세가 나와야 합니다.

In [ ]:

```
# Write your code here!
```

6. 객실 등급별 나이(Age) 컬럼의 평균을 보여주세요.

이번에는 전체 평균이 아닌 객실 등급(Pclass)별 평균을 보고 싶습니다.

가령 전체 승객의 평균 나이는 약 29.7세이지만, 1등급 승객의 평균 나이는 약 38.2세가 나와야 합니다. 비슷한 방식으로 2등급과 3등급 승객의 평균 나이를 알 수 있다면 좋겠습니다.

In [ ]:

```
# Write your code here!
```

7. 나이를 일정 구역으로 나눠서, 구역마다의 생존률을 보여주세요.

이번에는 나이(Age)별 생존률을 확인하고 싶습니다. 다만 나이 컬럼은 숫자이기 때문에, 그대로 쓰지 않고 일정 구역으로 나눈 뒤 생존률의 통계를 내는 것이 보기 편할 것입니다. 그러므로 나이 컬럼을 다음의 세 구역으로 나눕니다.

1. 나이가 15세 미만인 승객.
2. 나이가 15세 이상이고 30세 미만인 승객.
3. 나이가 30세 이상인 승객.

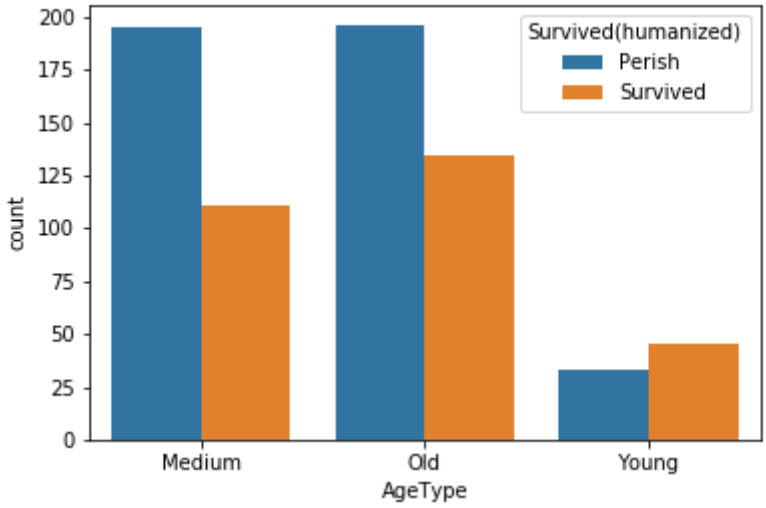
최종적으로는 다음의 결과가 나와야 합니다.

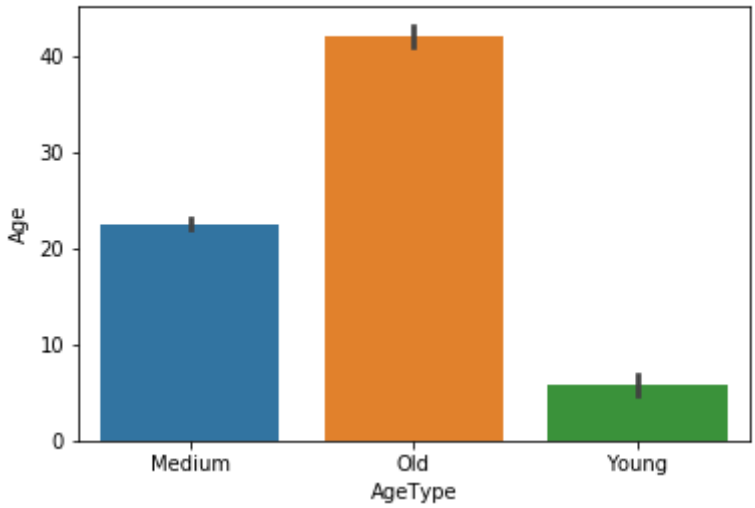
PassengerId	Age	AgeType
1	22.0	Medium
2	38.0	Old
3	26.0	Medium
4	35.0	Old
5	35.0	Old
6	NaN	NaN
7	54.0	Old
8	2.0	Young
9	27.0	Medium
10	14.0	Young

또한, 위 조건에서 1번, 2번, 3번 구역에 해당하는 승객의 평균 생존률을 구하고 싶습니다.

가령 1번 구역(나이가 15세 미만)에 해당하는 승객의 평균 생존률은 약 57.7%가 나와야 합니다.

마지막으로 이를 활용해 1) 구역별 생존자와 사망자의 차이, 2) 구역별 평균 나이를 시각화 해주세요. 최종적으로는 다음의 결과가 나와야 합니다.





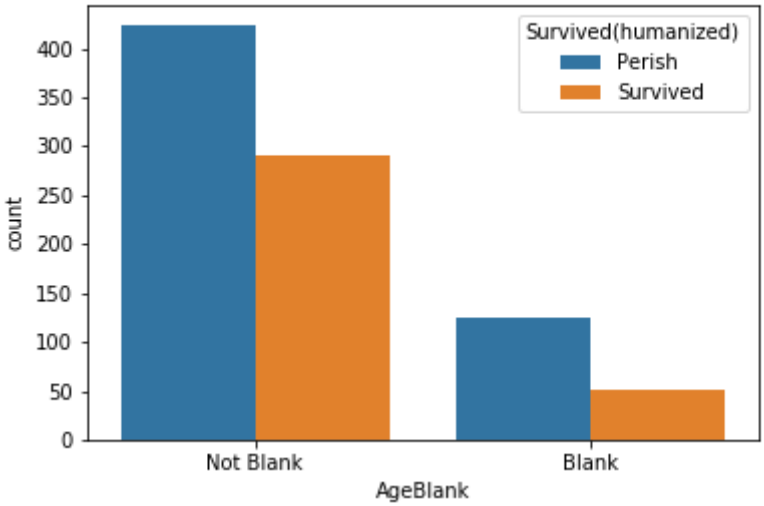
In [ ]:

```
# Write your code here!
```

8. 나이가 비어있는 승객과 비어있지 않은 승객의 생존률 차이를 보여주세요.

이번에는 다른 방식으로 생존률의 차이를 보겠습니다. 타이타닉 데이터의 나이(Age) 컬럼을 자세히 보면 나이가 비어있는 데이터가 있습니다. 판다스에서는 이를 NaN(Not a Number의 약자)으로 표현합니다.

타이타닉 데이터에서 나이 컬럼이 비어있는 승객과 비어있지 않은 승객의 생존률을 각각 찾아서 출력해주세요. 또한 이를 시각화로 비교해주세요. 최종적으로 다음의 결과가 나와야합니다.



In [ ]:

```
# Write your code here!
```

9. Pclass별 나이(Age)의 평균을 구한 뒤 빈 값에 채워주세요.

이번에는 나이(Age) 컬럼의 빈 값을 채우고 싶습니다. 일반적으로 가장 많이 하는 방식은 나이의 평균(mean) 값을 구한 뒤 이를 빈 값에 채워넣는 것입니다. 하지만 이번에는 다른 방식으로 빈 값을 채우고 싶은데, 바로 객실 등급(Pclass)에 따라 다르게 나이의 빈 값을 채워주고 싶습니다. 가령

1. 객실 등급(Pclass)이 1등급인 승객의 평균 나이를 구해서, 해당 승객 중 나이(Age)컬럼값이 비어있는 승객을 찾아 빈 나이 값을 채워줍니다.
2. 객실 등급(Pclass)이 2등급인 승객의 평균 나이를 구해서, 해당 승객 중 나이(Age)컬럼값이 비어있는 승객을 찾아 빈 나이 값을 채워줍니다.
3. 객실 등급(Pclass)이 3등급인 승객의 평균 나이를 구해서, 해당 승객 중 나이(Age)컬럼값이 비어있는 승객을 찾아 빈 나이 값을 채워줍니다.

위와 같은 방식을 사용하면, 단순히 전체 평균을 사용하는 것 보다 조금 더 원래 값에 근접하게 평균을 채워줄 수 있을 것 같습니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	Pclass	Age	Age(fill)
1	3	22.0	22.00000
2	1	38.0	38.00000
3	3	26.0	26.00000
4	1	35.0	35.00000
5	3	35.0	35.00000
6	3	NaN	25.14062
7	1	54.0	54.00000
8	3	2.0	2.00000
9	3	27.0	27.00000
10	2	14.0	14.00000
11	3	4.0	4.00000
12	1	58.0	58.00000
13	3	20.0	20.00000
14	3	39.0	39.00000
15	3	14.0	14.00000
16	2	55.0	55.00000
17	3	2.0	2.00000
18	2	NaN	29.87763
19	3	31.0	31.00000
20	3	NaN	25.14062

In [ ]:

```
# Write your code here!
```

SibSp, Parch 컬럼 분석

10. 타이타닉호에 동승한 형제, 자매, 배우자(SibSp)도 없고, 부모와 자식(Parch)도 없는 사람을 구해주세요.

해당 사용자를 싱글(Single)이라고 가정하겠습니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	SibSp	Parch	Single
1	1	0	False
2	1	0	False
3	0	0	True
4	1	0	False
5	0	0	True

또한 싱글(Single)인 사람과 그렇지 않은 사람간의 생존률의 차이도 알고 싶습니다. 최종적으로는 다음의 결과가 나와야 합니다.

Single	Survived
False	0.505650
True	0.303538

마지막으로 이를 시각화를 통해 비교해주세요. 최종적으로는 다음의 결과가 나와야합니다.

In [ ]:

```
# Write your code here!
```

11. SibSp 컬럼과 Parch 컬럼을 활용하여 가족 수(FamilySize)라는 새로운 컬럼을 만들어주세요.

형제, 자매, 배우자(SibSp) 컬럼과 부모 자식(Parch) 컬럼은 얼핏 달라 보이지만 실은 가족 관계를 나타내는 것이라고 볼 수 있습니다. 그러므로 두 컬럼을 하나로 합쳐서 **가족 수(FamilySize)**라는 새로운 컬럼을 만들면 승객의 가족관계를 더 편리하게 분석할 수 있을 것입니다.

형제, 자매, 배우자(SibSp) 컬럼과 부모 자식(Parch) 컬럼을 더해서 가족 수(FamilySize) 컬럼을 만들어주세요. 단 가족 수를 계산할때는 언제나 나 자신을 포함해서 계산하는데, 나 자신은 SibSp 컬럼에도 Parch 컬럼에도 들어가있지 않습니다. 그러므로 가족 수(FamilySize) 컬럼은 언제나 SibSp 컬럼과 Parch 컬럼을 더한 값에서 하나가 더 많아야 합니다.

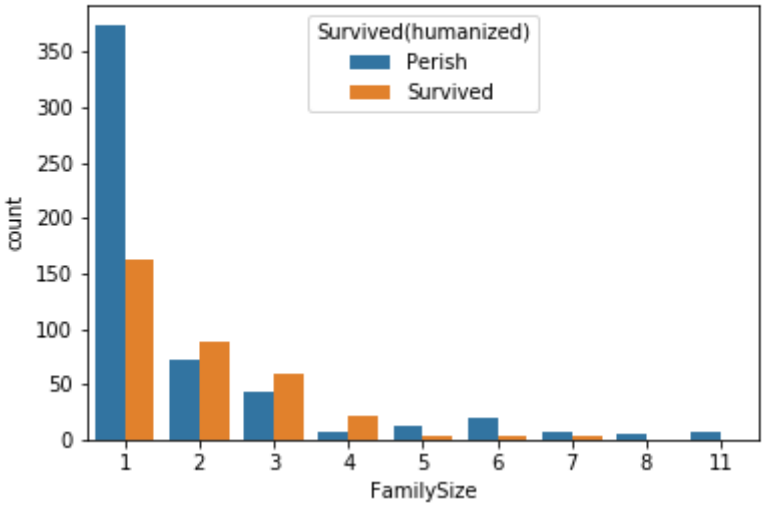
그러므로 최종적으로 다음의 결과가 나와야 합니다.

PassengerId	SibSp	Parch	FamilySize
1	1	0	2
2	1	0	2
3	0	0	1
4	1	0	2
5	0	0	1
6	0	0	1
7	0	0	1
8	3	1	5
9	0	2	3
10	1	0	2

또한 가족 수(FamilySize) 컬럼을 구한 뒤, 가족 수 별 생존률의 차이도 알고 싶습니다. 가족 수(ex: 1명 ~ 11명) 마다의 생존률을 구해서 출력해주세요. 최종적으로 다음의 결과가 나와야 합니다.

FamilySize	Survived
1	0.303538
2	0.552795
3	0.578431
4	0.724138
5	0.200000
6	0.136364
7	0.333333
8	0.000000
11	0.000000

마지막으로 이를 시각화를 통해 보여주세요. 최종적으로는 다음의 결과가 나와야 합니다.



In [ ]:

```
# Write your code here!
```

12. 가족 수(FamilySize) 컬럼의 구역을 나눠주세요.

가족 수(FamilySize) 컬럼을 기준으로 pivot_table로 분석을 해본 결과, 경우의 수가 너무 많아서(가족 수가 1명일 때 ~ 11명일 때) 분석 결과가 너무 잘게 쪼개지는 것 같습니다.

그러므로 가족 수(FamilySize) 컬럼을 세 구역으로 나누고 싶습니다. 구체적으로는 다음과 같습니다.

- **싱글(Single)** - 동승한 가족이 아무도 없고, 나 혼자 탑승한 경우입니다.
- **핵가족(Nuclear)** - 동승한 가족이 나 자신을 포함해 2명 이상 5명 미만인 경우입니다.
- **대가족(Big)** - 동승한 가족이 나 자신을 포함 5명 이상인 경우입니다.

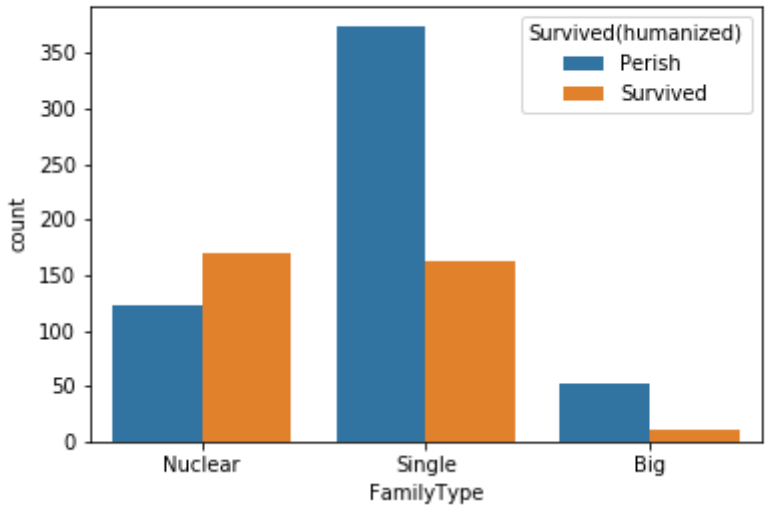
위의 정보를 활용하여, 가족 형태(FamilyType)라는 새로운 컬럼을 만들어 주세요. 이 컬럼에는 앞서 설명한 Single, Nuclear, 그리고 Big이 들어갑니다. 최종적으로는 다음의 결과가 나와야 합니다.

PassengerId	FamilySize	FamilyType
1	2	Nuclear
2	2	Nuclear
3	1	Single
4	2	Nuclear
5	1	Single
6	1	Single
7	1	Single
8	5	Big
9	3	Nuclear
10	2	Nuclear

또한 가족 수(FamilySize)와 마찬가지로 가족 형태(FamilyType) 별 생존률의 차이도 구해주세요. 최종적으로 다음의 결과가 나와야 합니다.

FamilyType	Survived
Big	0.161290
Nuclear	0.578767
Single	0.303538

마지막으로 이를 시각화를 통해 비교해주세요. 최종적으로는 다음의 결과가 나와야합니다.



In [ ]:

```
# Write your code here!
```

마무리하며

지금까지 프로그래밍 언어 파이썬(Python (https://python.org/))과 파이썬의 데이터 분석 패키지 판다스(Pandas (https://pandas.pydata.org/)), 데이터 시각화 패키지 씨본(Seaborn (https://seaborn.pydata.org))과 matplotlib (https://matplotlib.org)를 활용한 실전 예제를 살펴보았습니다. 앞서 말씀드린대로, 위 문제를 실전에서 반나절(3~4시간) 안에 해결할 수 있다면 현업에서 데이터 사이언티스트로서 일 할 수 있는 충분한 판다스 스킬을 보유했다고 볼 수 있습니다.

반면 1) 앞으로 데이터 분석을 업무에 활용하고자 하는 분들, 또는 2) 앞으로 데이터 사이언티스트로 취업이나 이직, 전직을 노리는 분 중, 위 문제를 반나절 안에 풀지 못한 분들은 판다스를 추가 학습해야 할 필요가 있다고 생각하시면 됩니다. 그런 분들에게는 다음의 자료를 추천합니다.

- 10 minutes to pandas (https://pandas.pydata.org/pandas-docs/stable/10min.html)
- Pandas Cookbook (http://github.com/jvns/pandas-cookbook)
- Python for Data Science (http://wavedatalab.github.io/datawithpython/)
- Modern Pandas (http://tomaugspurger.github.io/modern-1-intro.html)
- Seaborn Gallery (https://seaborn.pydata.org/examples/index.html)

모든 문제를 풀이하셨다면 **해당 Level 채널**에 풀이한 과제 파일(.ipynb)을 올려주시거나 전담 튜터에게 **DM(Direct Message)**로 풀이한 과제 파일(.ipynb)을 주시면 됩니다. 기타 수업 관련 문의 사항은 슬랙의 전담 튜터에게 Direct Messages로 말씀해주세요!

In [ ]: