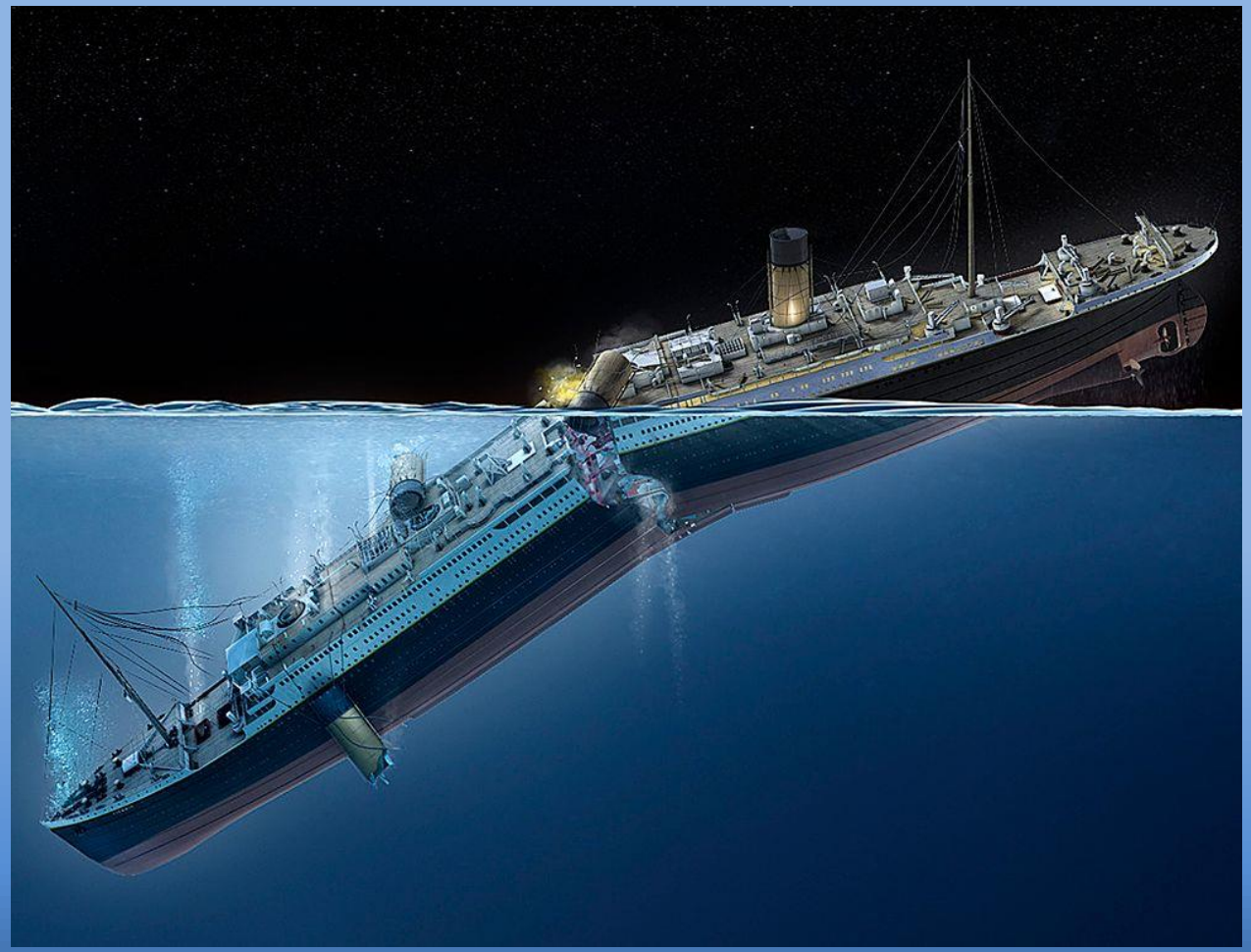


# < Tabular Playground Series – Apr 2021>

## Team: 권정민, 서지훈, 이현정, 전은진

## Date: 2021.04.22

<출처:https://namu.wiki/w/%ED%83%80%EC%9D%B4%ED%83%80%EB%8B%89%ED%98%B8%20%EC%B9%A8%EB%AA%B0%20%EC%82%AC%EA%B3%A0>

# CONTENTS


Part 1 INTRODUCTION

Part 2 EXPLORATORY DATA ANALYSIS (EDA)

Part 3 FEATURE ENGINEERING

Part 4 MODELING

Part5 REFERENCES

# PART 1 INTRODUCTION

## OVERVIEW OF COMPETITION


| | |
|---|---|
| **Competition Timeline** | **April 1st , 2021 ~ April 30th, 2021** |
| Duration of participation | 12 days (April 19th , 2021 ~ April 30th, 2021) |
| Participants | 4 persons |
| Kaggle Notebook | https://www.kaggle.com/tlgks32/0423project |

<출처: https://www.kaggle.com/c/tabular-playground-series-apr-2021>

## Dataset

| | Rows | Cols | Size(MB) |
|---|---|---|---|
| Train | 100,000 | 12 | 5.67 |
| Test | 100,000 | 11 | 5.54 |

➔ 이 대회에서 사용되는 Dataset은 **실제 데이터를 기반**으로 하였으며 CTGAN을 사용하여 생성된 Dataset 입니다.

➔ 실제 데이터: Titanic (https://www.kaggle.com/c/titanic/overview)

<출처: https://www.kaggle.com/c/tabular-playground-series-apr-2021>

# Description

For the most part, most of the well-known people on board were first-class passengers. Researcher Chuck Anesi crunched the numbers, breaking down the demographics of the survivors. He found that 97.22% of the 144 female first-class passengers were rescued, while only 32.57% of their 175 male counterparts were saved.

Ultimately, he found that male second-class passengers fared the worse in terms of survival, with only 14 out of 168 making it out alive. The total survival rate for women was 74%, while the male survival rate was 20%.

Here are 12 of the most famous victims of the Titanic disaster— and 11 prominent people who survived:

5. (갑판 아래 갇힌) 3등석 탑승객들

캐머런의 영화에서 가장 감정을 자극하는 장면 중 하나는, 3등석 승객들이 강제로 갑판 아래에 감금되고 구명보트 탑승이 금지되는 장면이다. 그러나 리처드 하웰은 이 역시 역사적 증거가 없다고 반박한다.

당시 타이타닉호에는 3등석 탑승객을 다른 승객들로부터 갈라놓는 '문'은 존재하지 않았다. 당시 영국 조사위 보고서도 3등석 승객들이 갑판 아래 감금됐다는 것은 거짓말이라고 지적했다.

다만, 3등석에 탔던 중국, 네덜란드, 이탈리아, 러시아, 스칸디나비아, 시리아 출신의 이민자들은 구조 편의가 아니라 '전염병에 대한 우려' 때문에 '미국 이민법'에 따라 분리됐다. 하웰스는 "미국 이민법 때문에 이민자들은 건강검사를 하고 이민절차를 밟을 때까지 분리됐다"고 전했다. 또 모든 탑승객들은 1·2·3등석 별로 해당 갑판에서 구명보트에 탑승했다. 다만, 결정적으로 배의 3등석 갑판에는 구명보트가 없었다. 따라서 3등석 탑승객들은 1·2등석 탑승객들과 달리 미로같은 복도와 계단을 통해 살 길을 찾아야 했다.

http://www.hani.co.kr/arti/international/international_general/526929.html

<출처: https://www.businessinsider.com/titanic-famous-survivors-victims-2018-4#:~:text=The%20Titanic%20%E2%80%94%20billed%20as%20an,and%20survivors%20were%20famous%20people..>

# PART 2

# EXPLORATORY DATA ANALYSIS (EDA)

## Summary

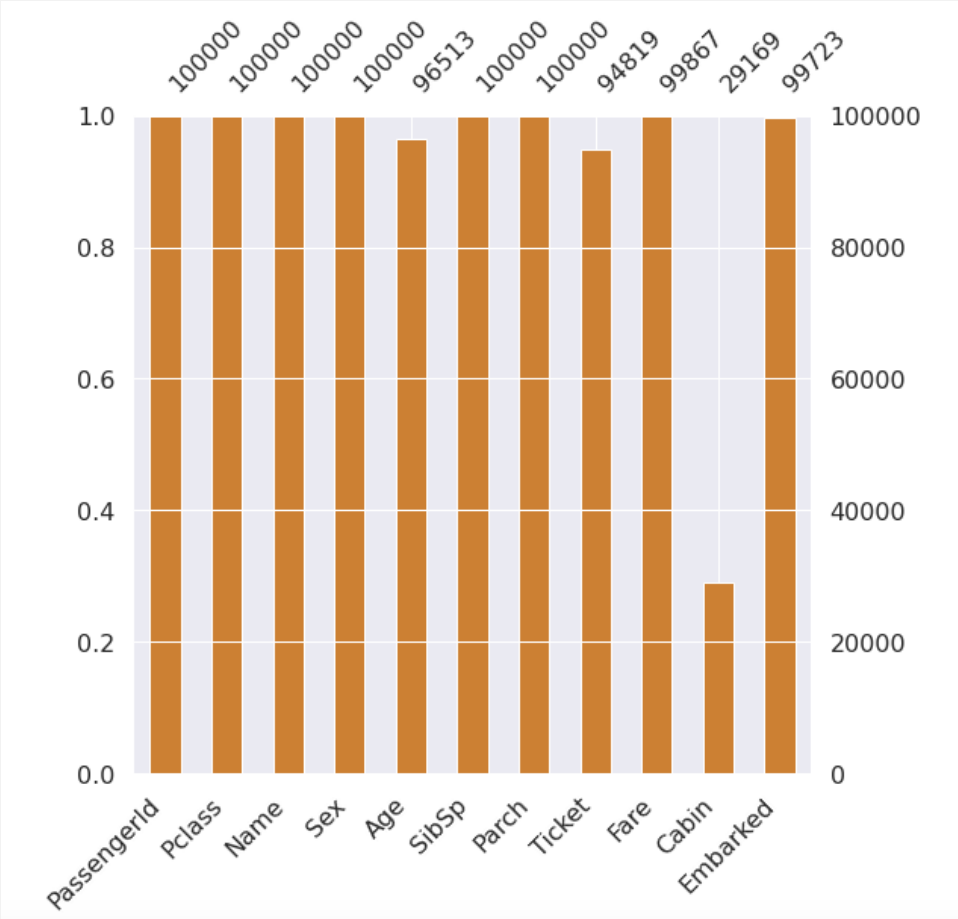
### 이 프로젝트의 목적

주어진 독립 변수들 사이에서 유의미한 변수를 찾고, 모델에 도움이 되지않는 변수는 제외한다.

### 결론

독립 변수 Parch는 회귀 선형성을 위반하는 분포를 띄고있어 제외한다.
다른 변수들은 유의미하다고 판단하여 적절한 인코딩을 통해 모델에 포함시킨다.

## Summary of Data

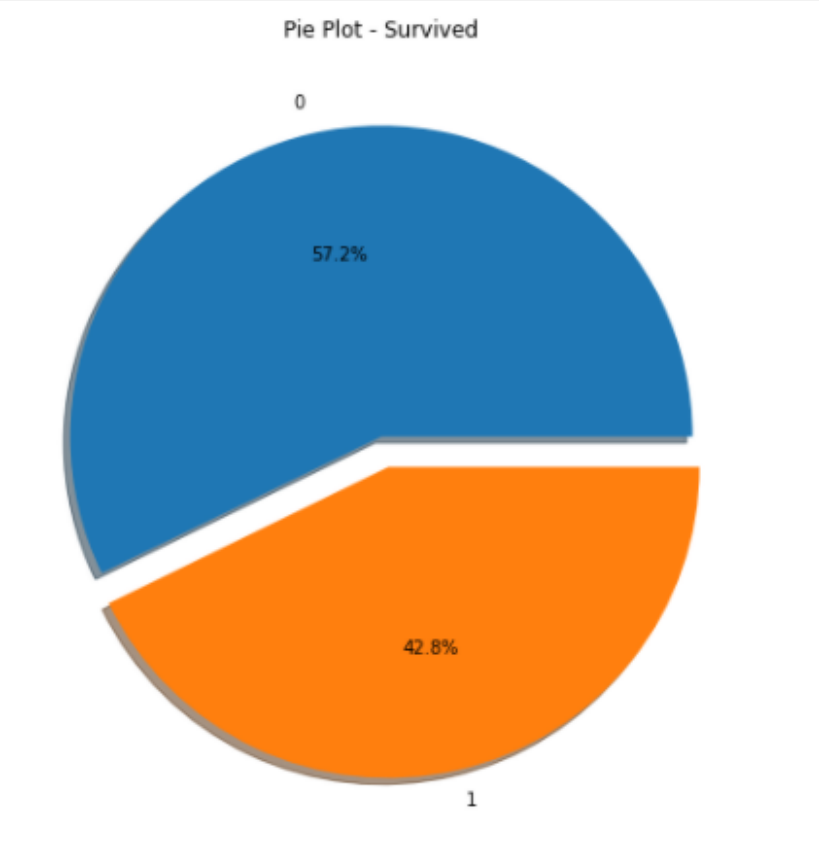



Null values

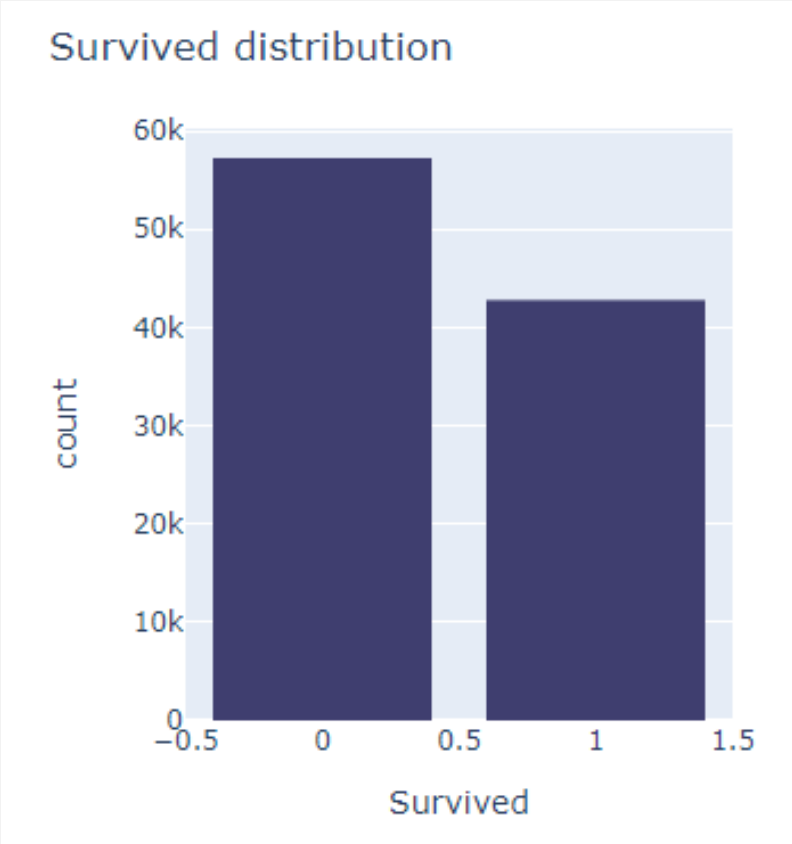
| Features | 설명 | 타입 | Null값 개수(Train) |
|---|---|---|---|
| Survived(생존 여부) | Target label(0,1) | integer | |
| PassengerId | | integer | |
| Pclass(객실) | 1st(1), 2nd(2), 3rd(3) | integer | |
| Name(이름) | Last / First name | string | |
| Sex(성별) | Male, Female | string | |
| Age(나이) | continuous | integer | 3,292 |
| Sibsp (형제, 배우자 수) | Quantitative | integer | |
| Parch (부모, 자식 수) | Quantitative | integer | |
| Ticket(티켓 번호) | Alphabet+integer | string | 4,623 |
| Fare(탑승료) | Continuous | integer | 134 |
| Cabin(갑판 번호) | Alphabet+integer | string | 67,866 |
| Embarked (탑승항구) | C(Cherbourg), Q(Queenstown), S(Southhampton) | string | 250 |

## Survived

Train set 의 생존율




Train set 의 생존율

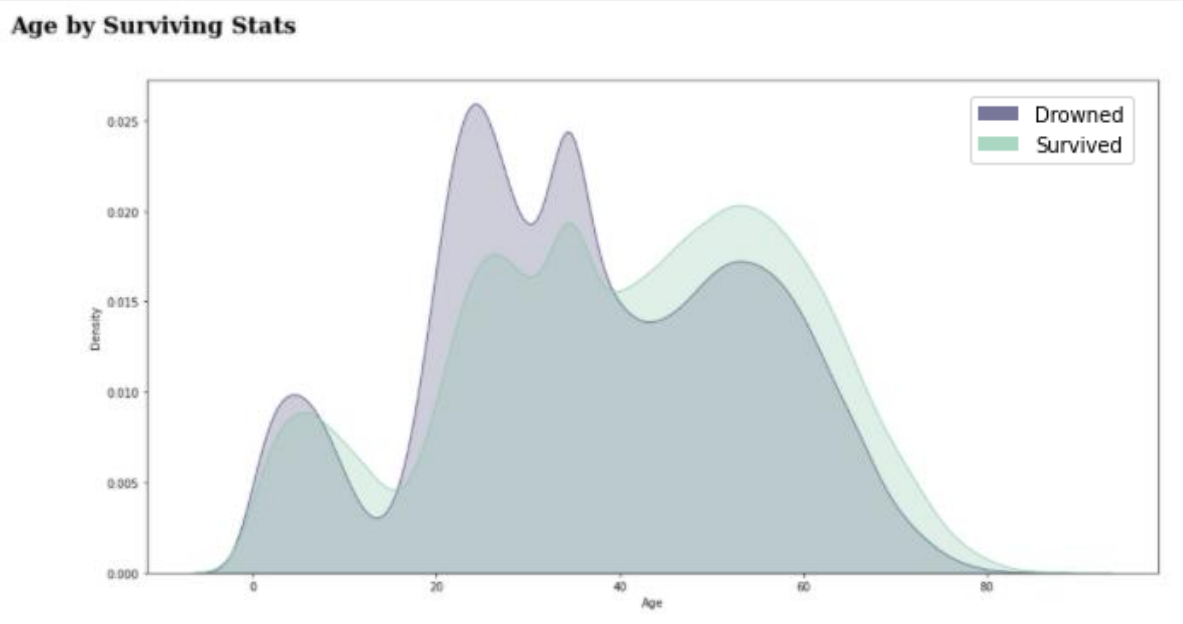


## Age

Summary

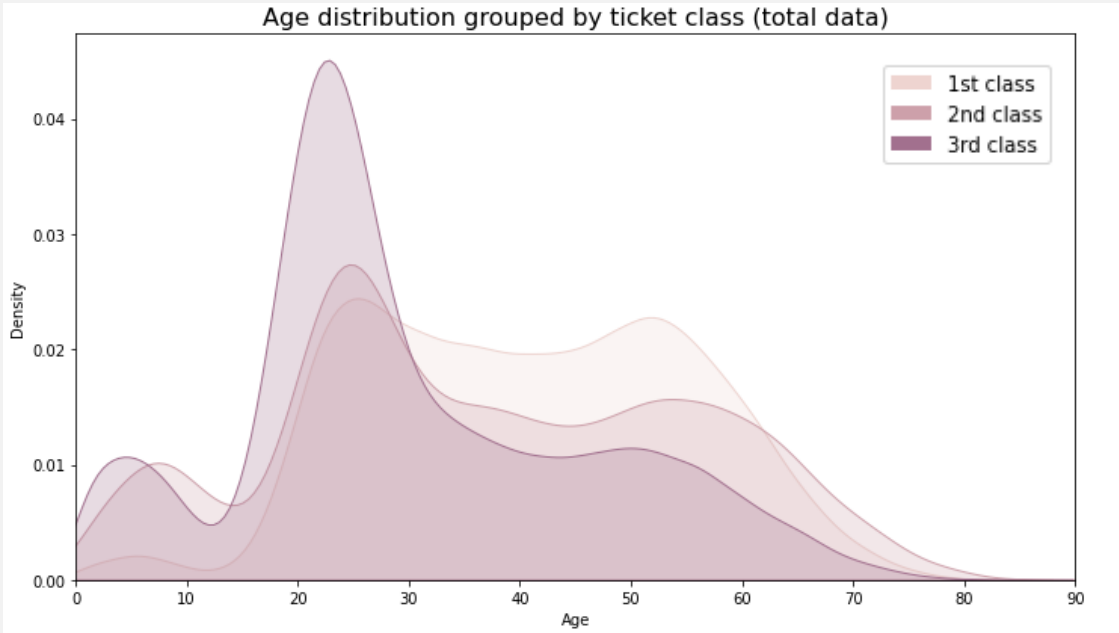
노년 층의 생존율이 더 높다.

### 나이와 생존율의 관계




➔ 젊은 사람들에 비해 노년층의 생존율이 더 많은 것을 볼 수 있다.

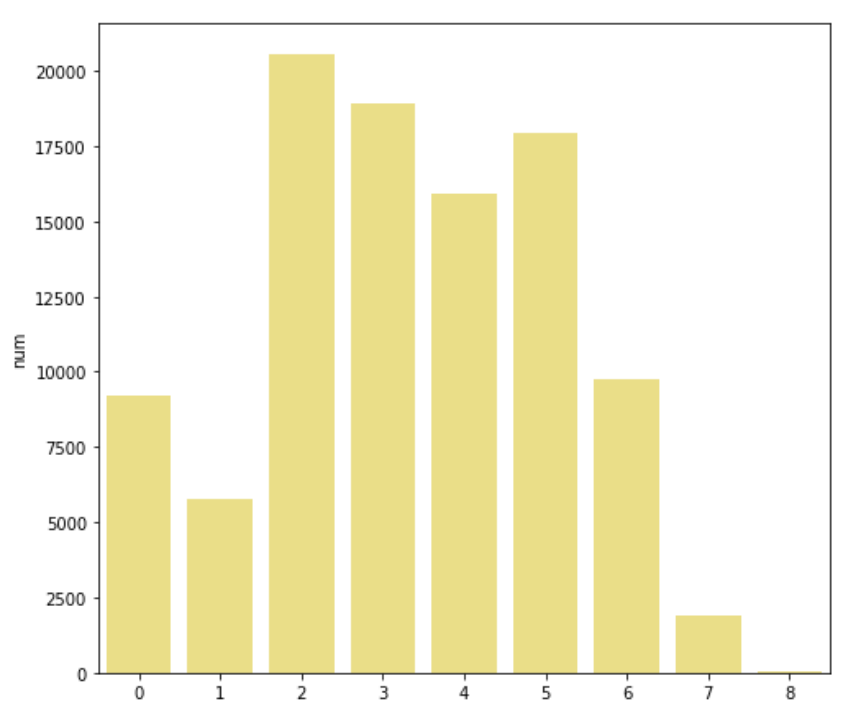
### Age by Pclass



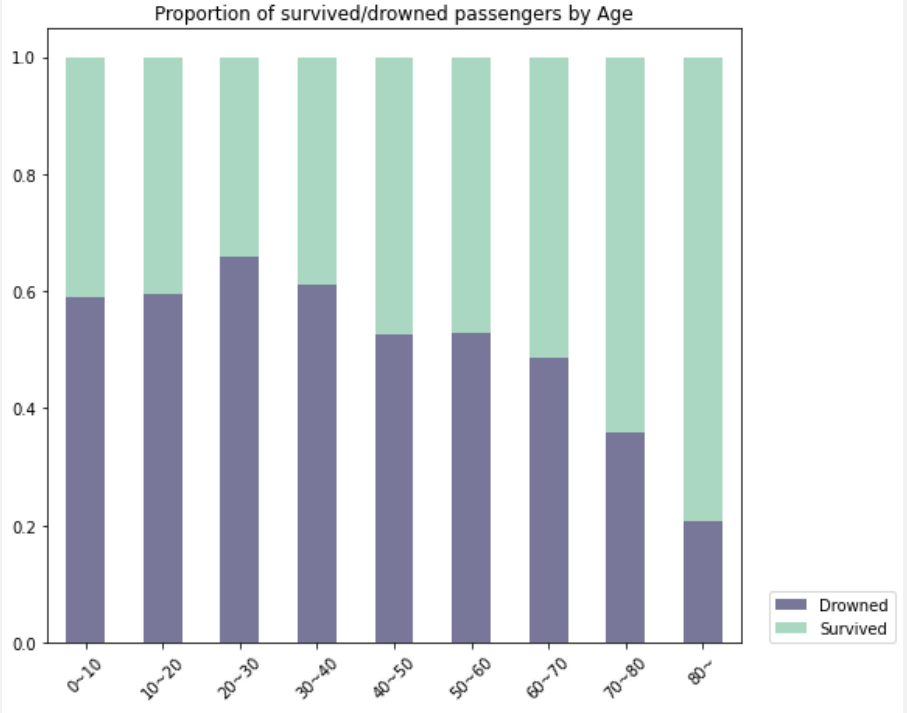

➔ 3등석에는 젊은 사람들이 대부분임

# Age

## 연령대별 나이 분포도




## 연령대에 따른 생존율




➔ 80대의 생존율이 가장 높음

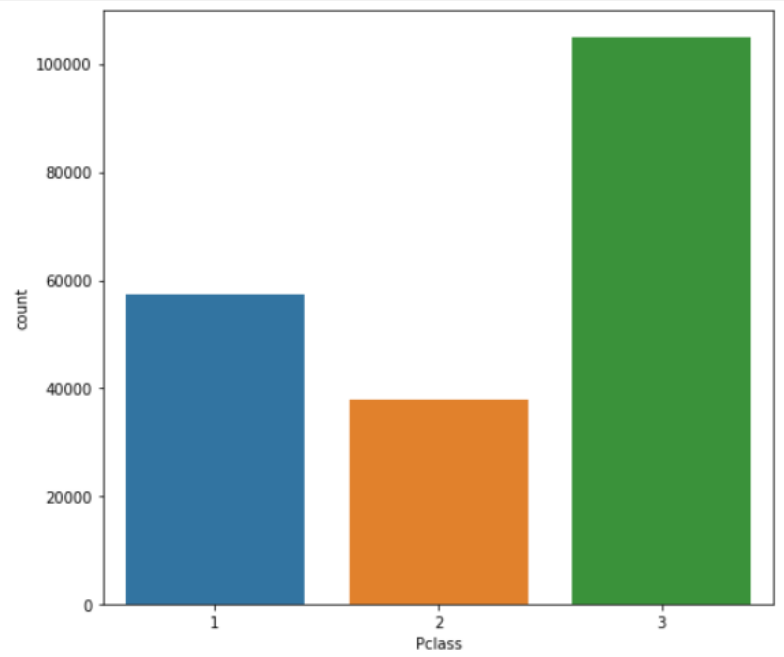
➔ 20대의 생존율이 가장 낮음

## Pclass

Summary

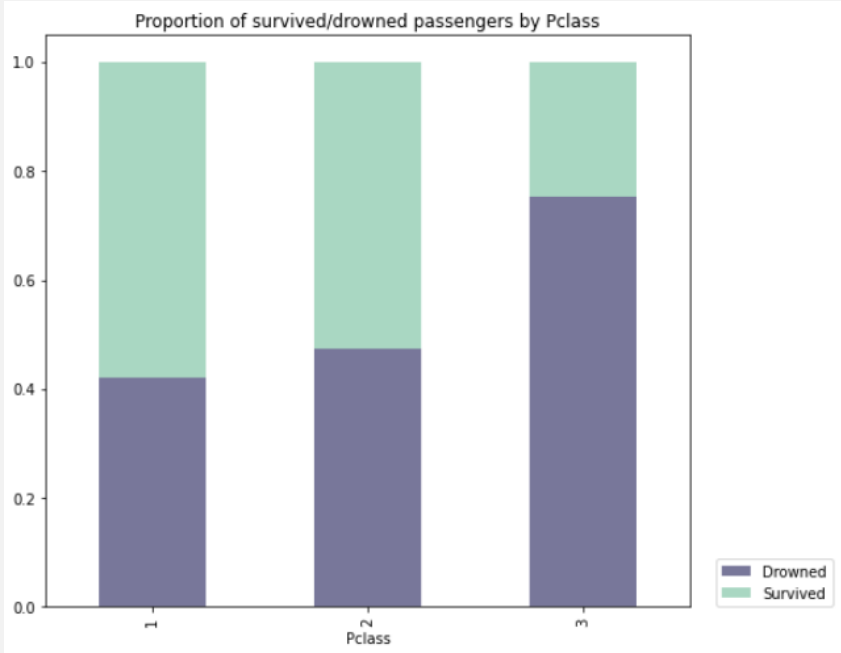
더 높은 등급일 수록 생존율도 높아진다.

### Pclass에 따른 승객 수




➔ 가장 많은 인원수가 탑승한 3등석

### Pclass에 따른 생존율

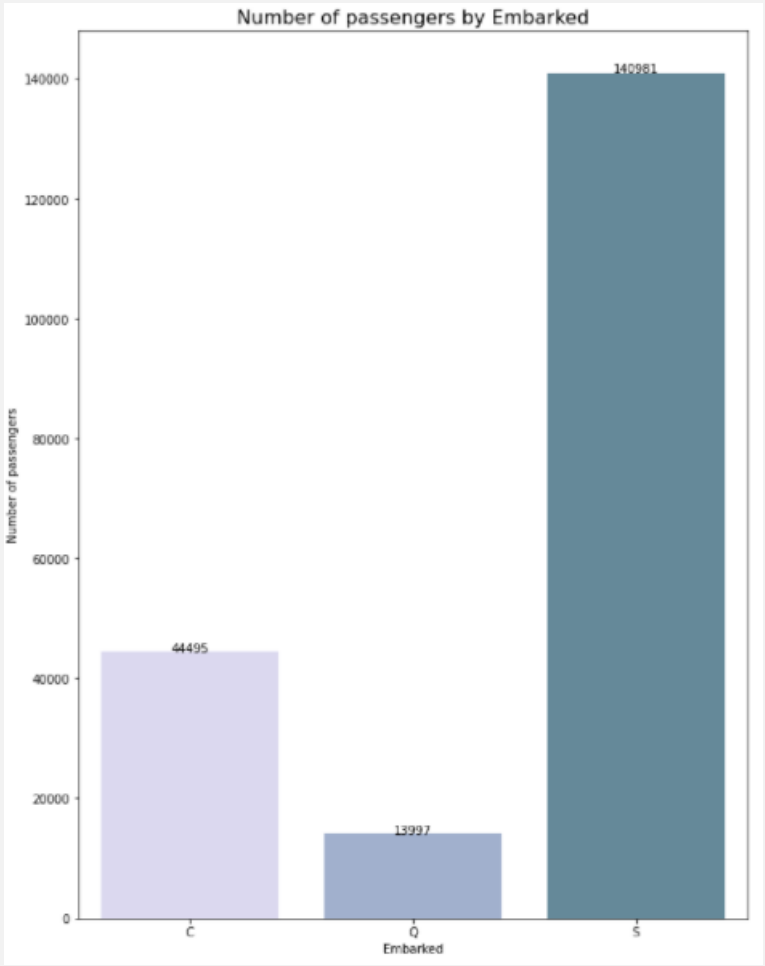



➔ 그러나 가장 낮은 생존율을 보이는 3등석

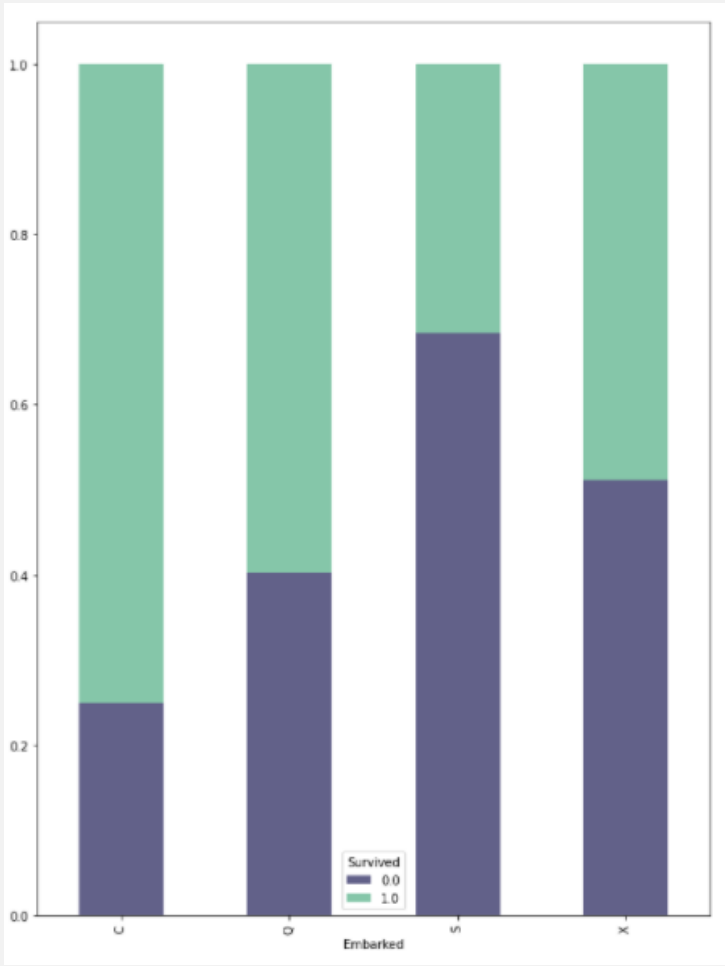
## Embarked

**Summary** 'S'에 가장 많은 인원이 탑승하였고, 생존율은 가장 낮았다.

### 탑승항구에 따른 승객 수




### 탑승항구에 따른 생존율



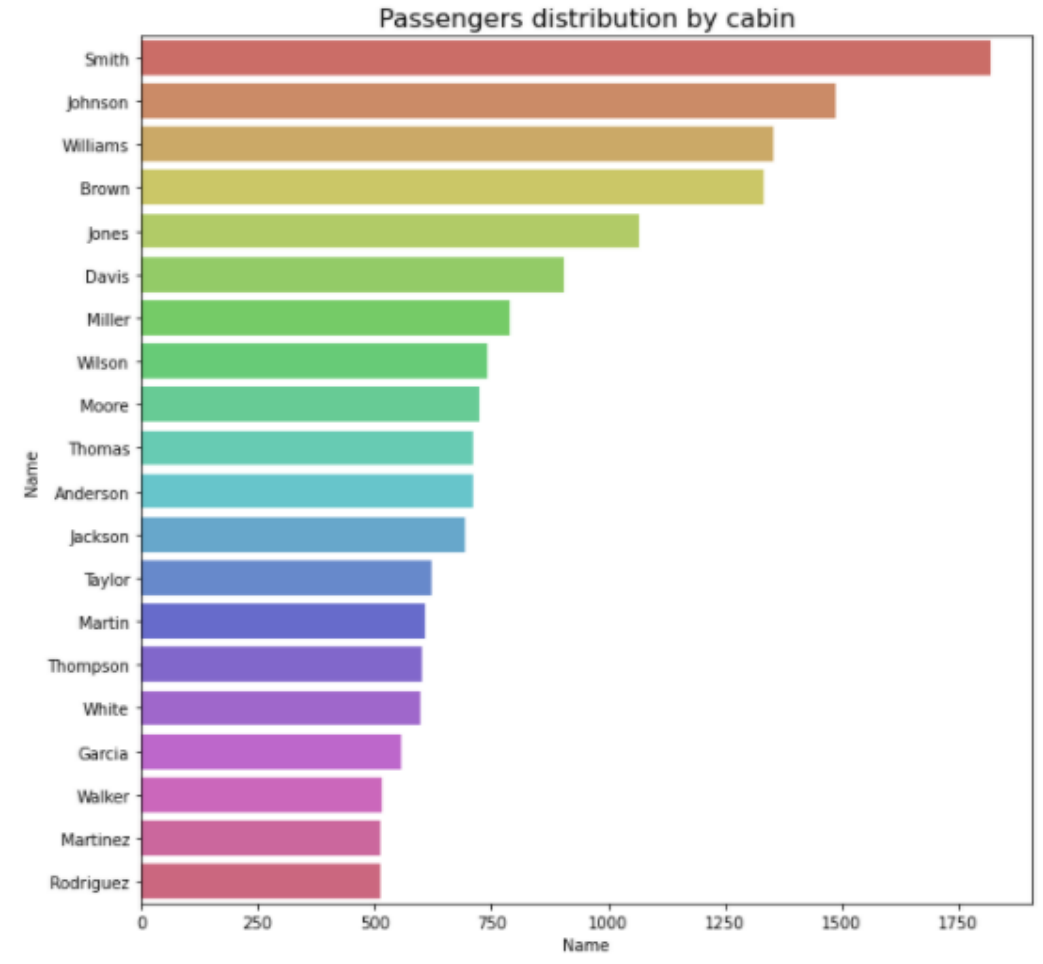

➔ 탑승인원 중 약 70%에 해당하는 승객들은 S(Southhampton)에 해당함

➔ S에서 탑승한 승객들이 탑승인원에 비해 가장 낮은 생존율을 보임

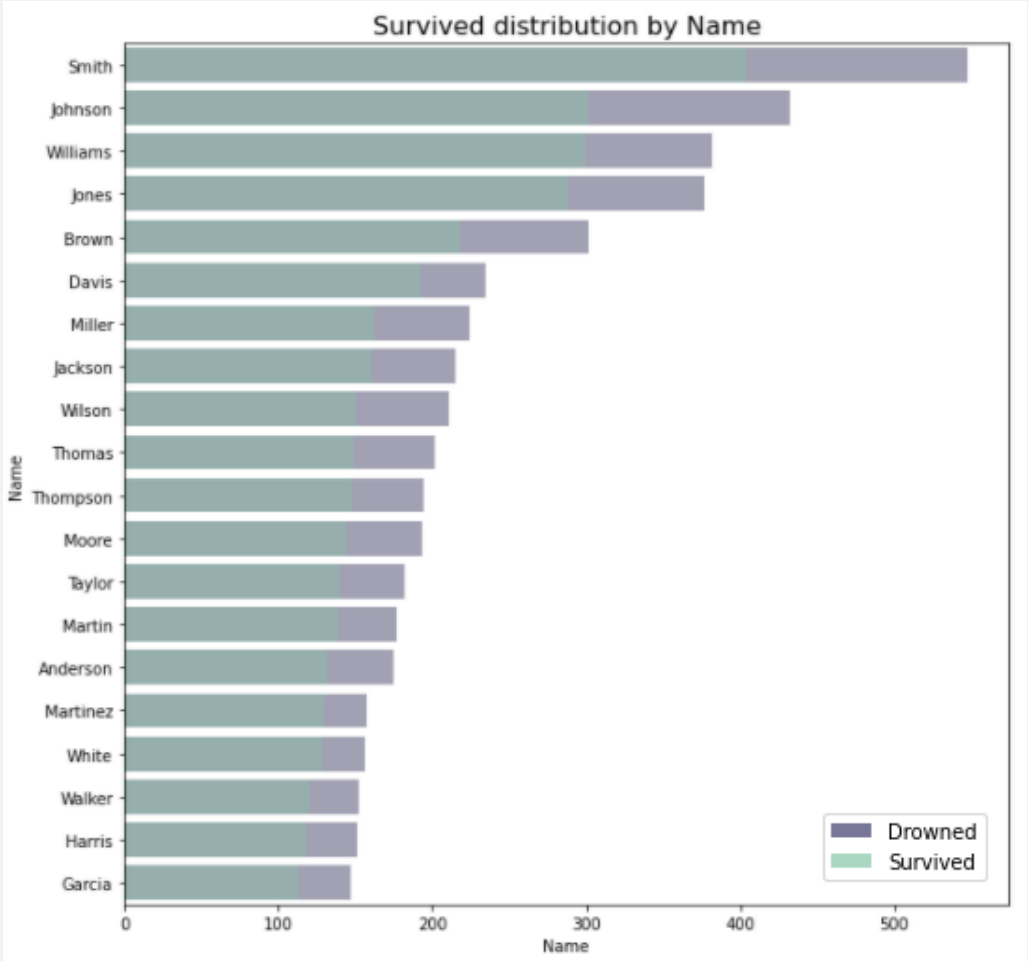
## Name

**Summary** 흔한 First Name일 수록 생존율이 높았지만 그만큼 많은 사람이 탑승했기 때문에 가능한 결과였다.

### First Name에 따른 승객 수




### First Name에 따른 생존율

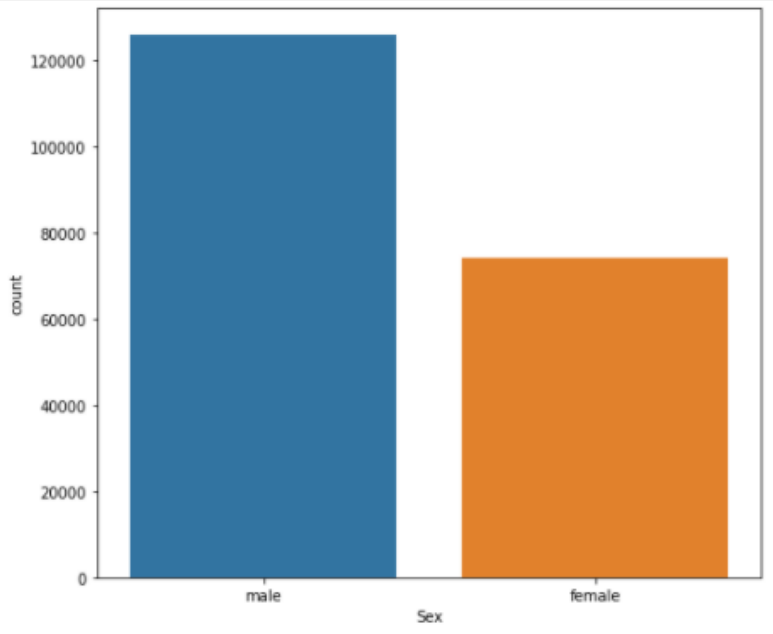


## Sex

Summary

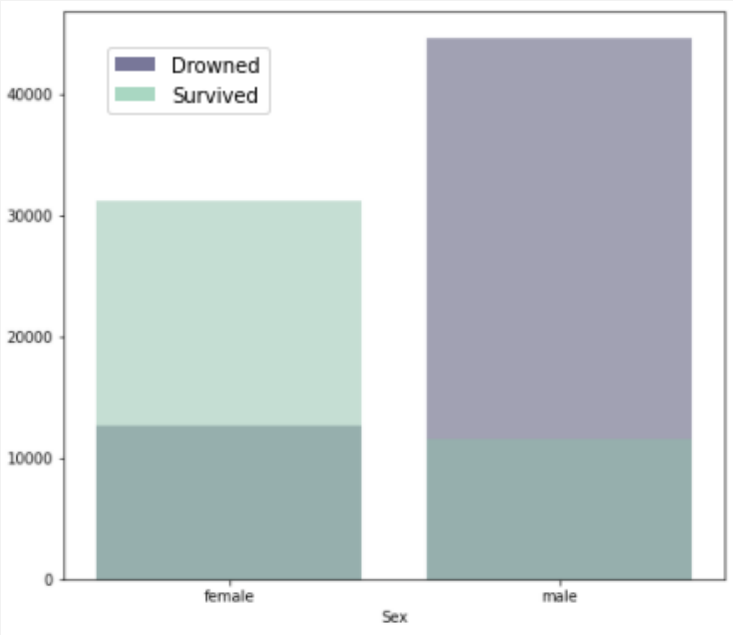
여성의 생존율이 남성의 생존율보다 훨씬 높았다.

### 성별에 따른 승객 수



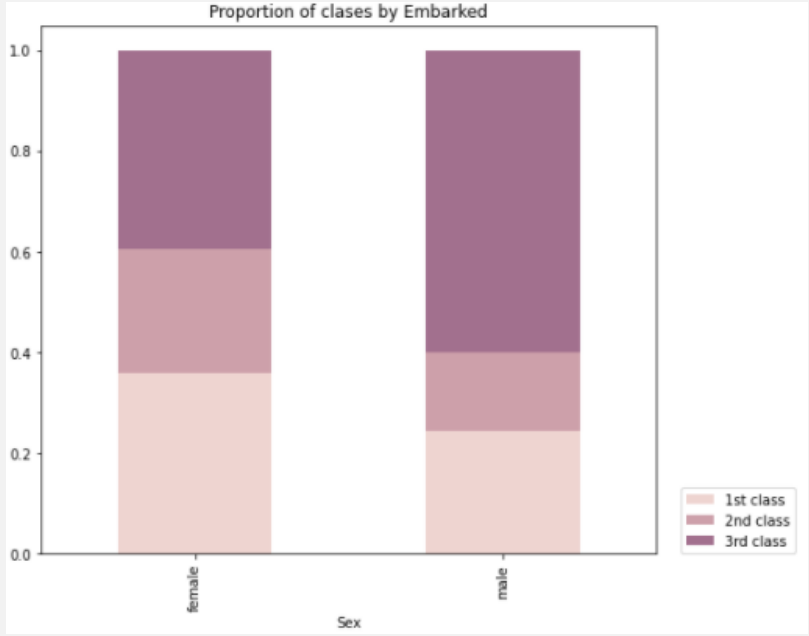

➔ 남성의 수 > 여성의 수

### 성별에 따른 생존율




➔ 남성의 생존율은 여성보다 훨씬 낮음

### 성별에 따른 Pclass 비율



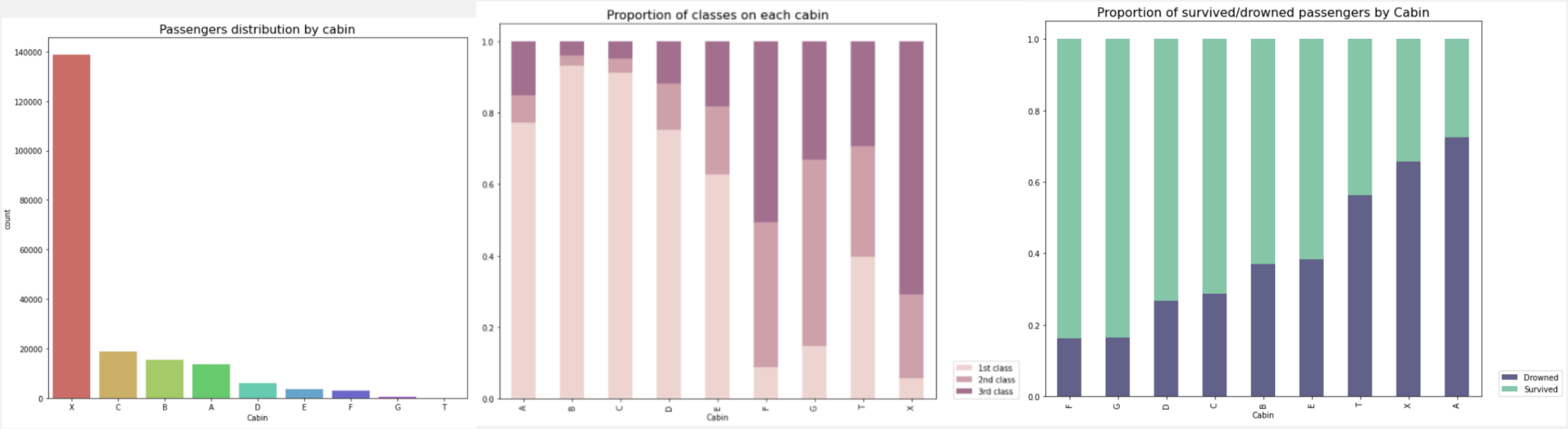

➔ 1등석: 여성 > 남성

➔ 2등석: 여성 > 남성

➔ 3등석: 남성 > 여성

## Cabin

**Summary** 알파벳 순서가 앞쪽인 Deck일 수록 1등급 승객이 많았고, 생존율도 더 높았다.



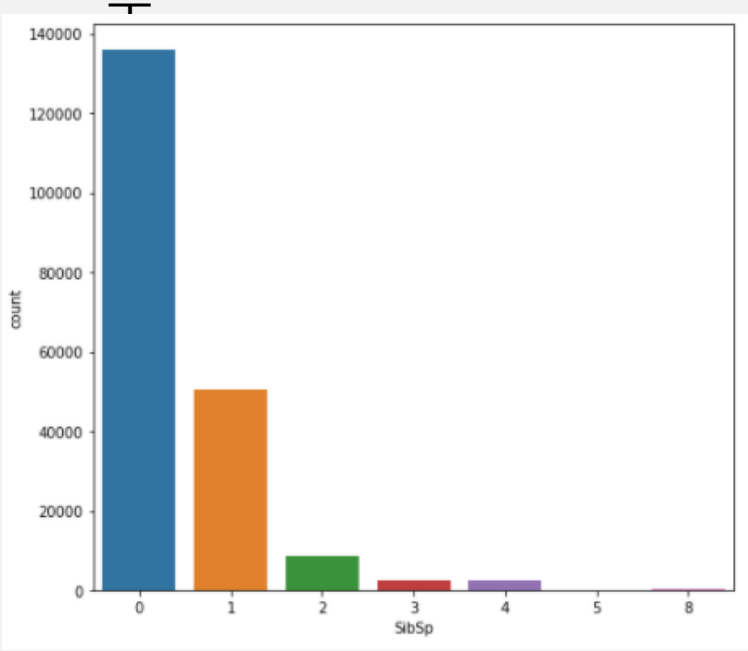

▲생존자 비생존자 색 반대

- 대부분의 승객들은 Cabin code가 없다
- Cabin Code가 있는 승객들 중 가장 많은 수를 차지하는 Deck은 'C'이며 이곳은 1st class 승객이 대부분!

## Sibsp

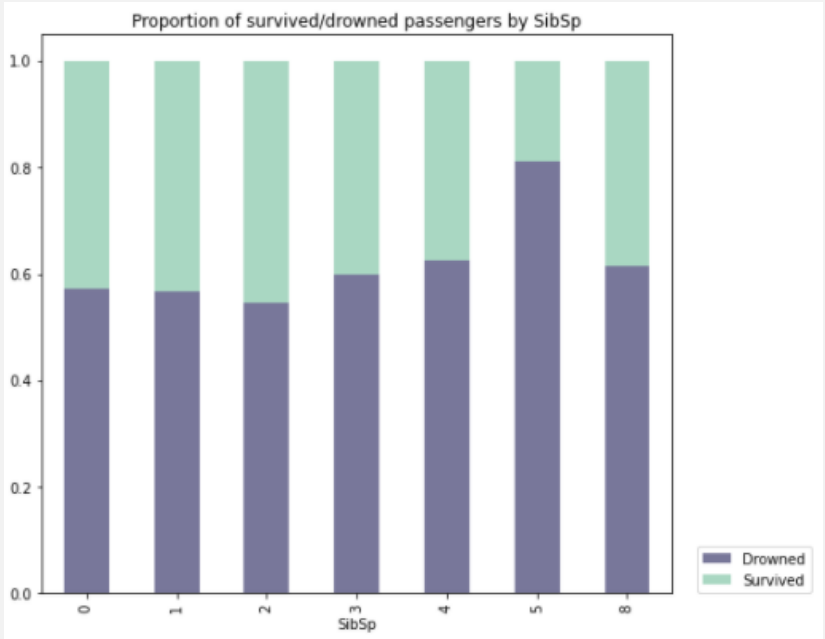
**Summary** 형제 자매/ 배우자 수에 상관없이 대체적으로 생존율은 비슷하였다.

### 형제자매/배우자 수에 따른 승객 수




➔ 혼자 온 승객 수가 약 68%에 해당

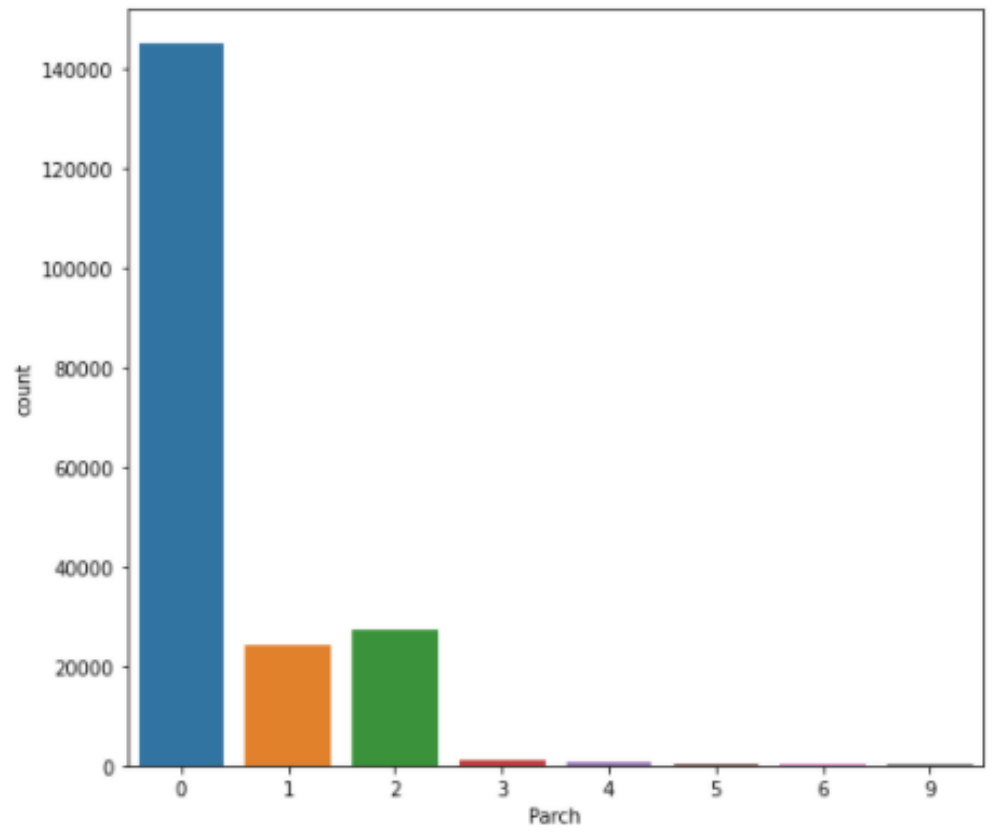
### 형제자매/배우자 수에 따른 생존율



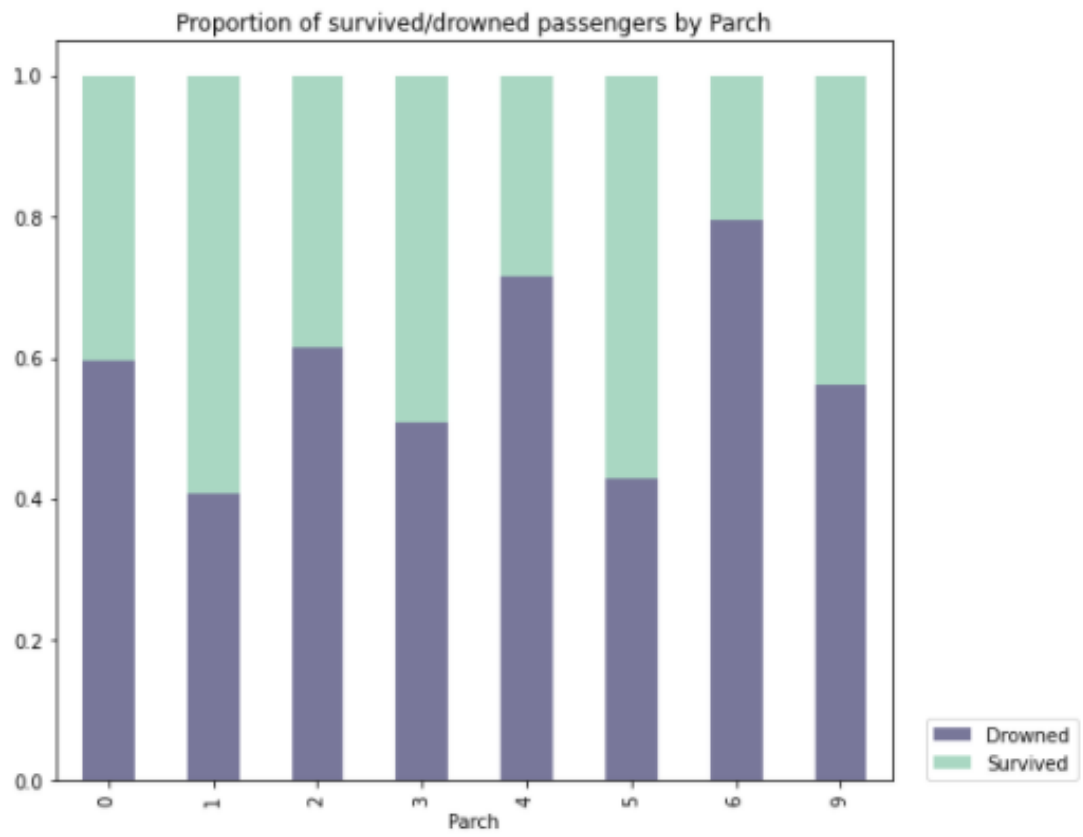
## Parch

**Summary** 부모 자식 수에 따라 생존율이 높거나 낮거나 하지 않았다.

### 부모자식 수에 따른 승객 수



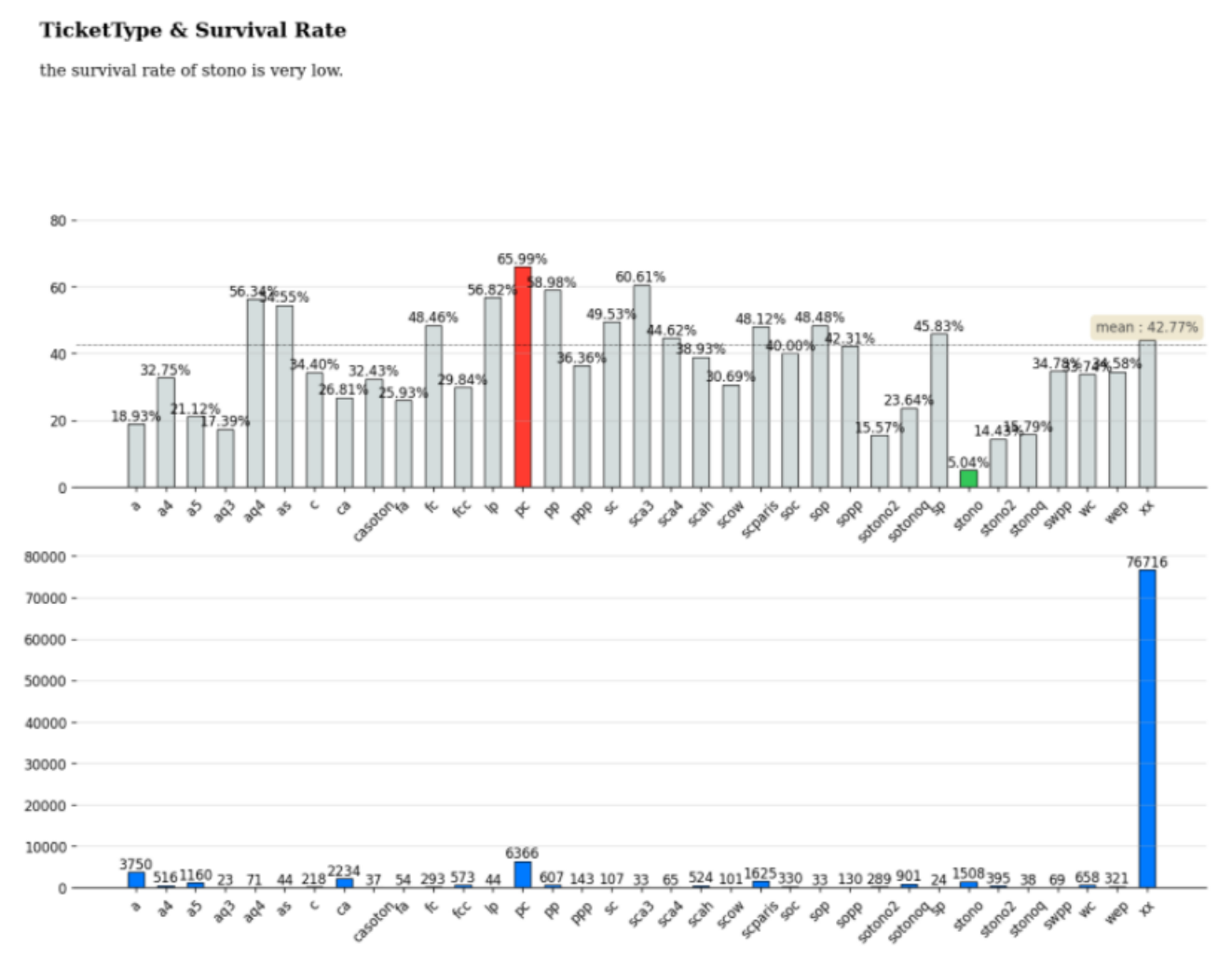

### 부모자식 수에 따른 생존율



## Ticket

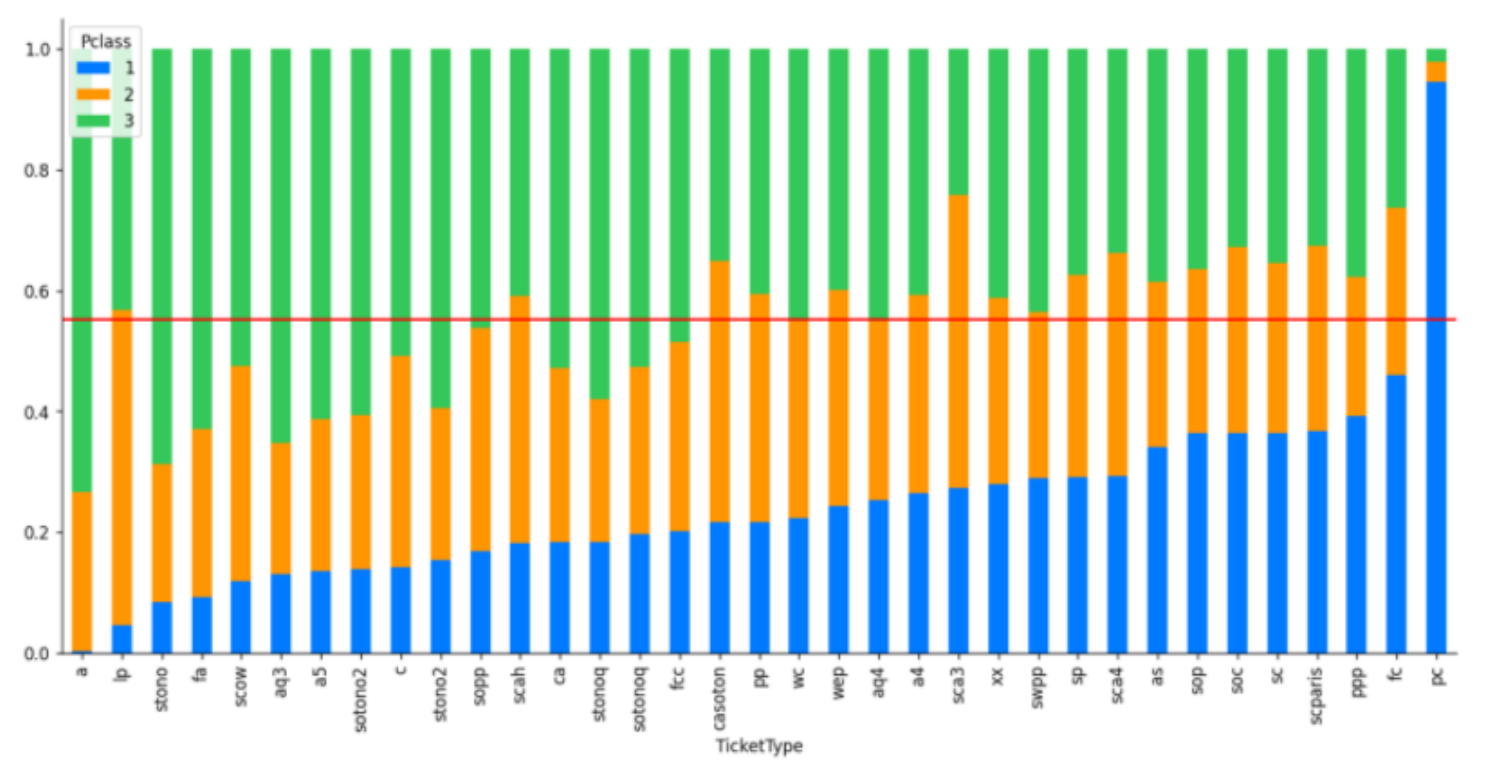
Summary

1등급 클래스 비율이 높은 Ticket Type이 생존율이 더 높았다.




가장 높은 생존율을 보인 Ticket Code는 'PC'이다.

## Pclass & Ticket

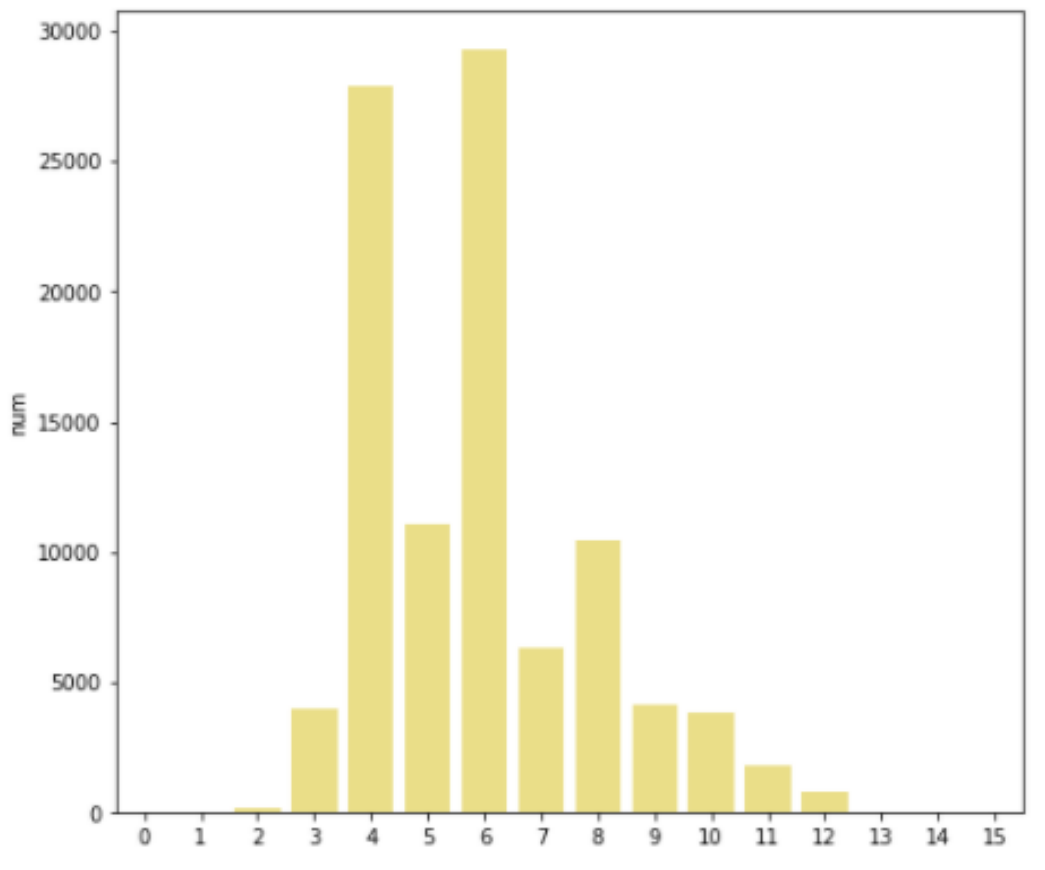



- 1등석 기준: 'PC'가 가장 높은 생존율을 보임 ➔ 1등급의 비율이 높을 수록 생존율은 더 높은 것으로 확인!
- 2등석 기준: lp , sca3 이외의 Ticket은 대체적으로 고른 분포
- 3등석 기준: 'a'가 가장 높은 생존율을 보임

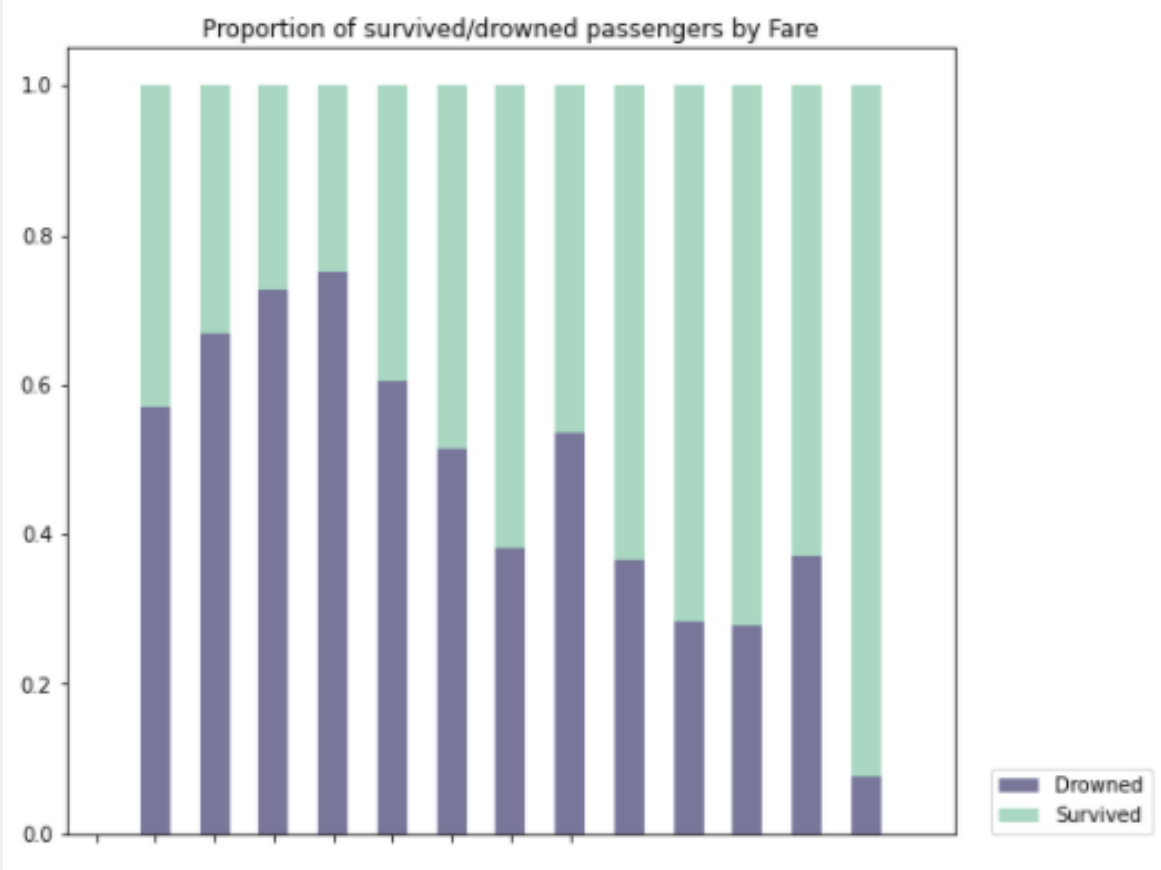
## Fare

**Summary** 높은 요금의 탑승료를 낸 사람들이 더 높은 생존율을 보이고 있다.

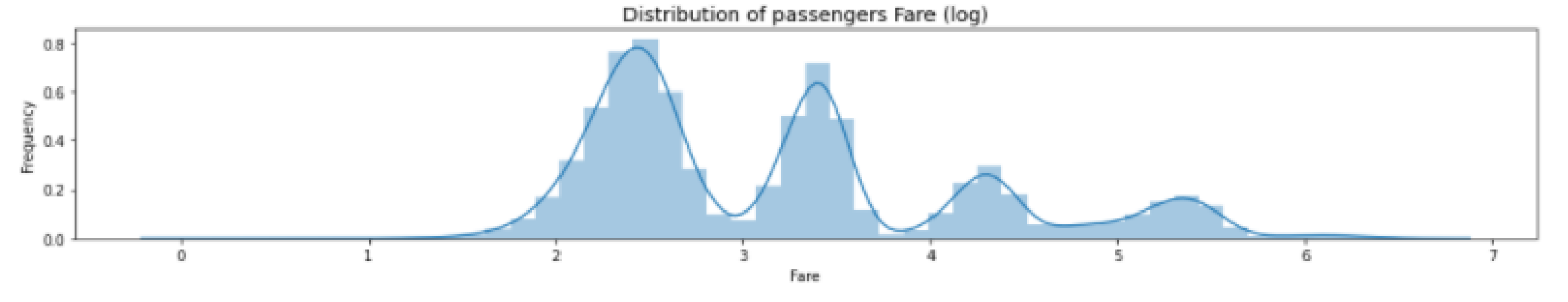
### 탑승료에 따른 승객 수




### 탑승료에 따른 생존율



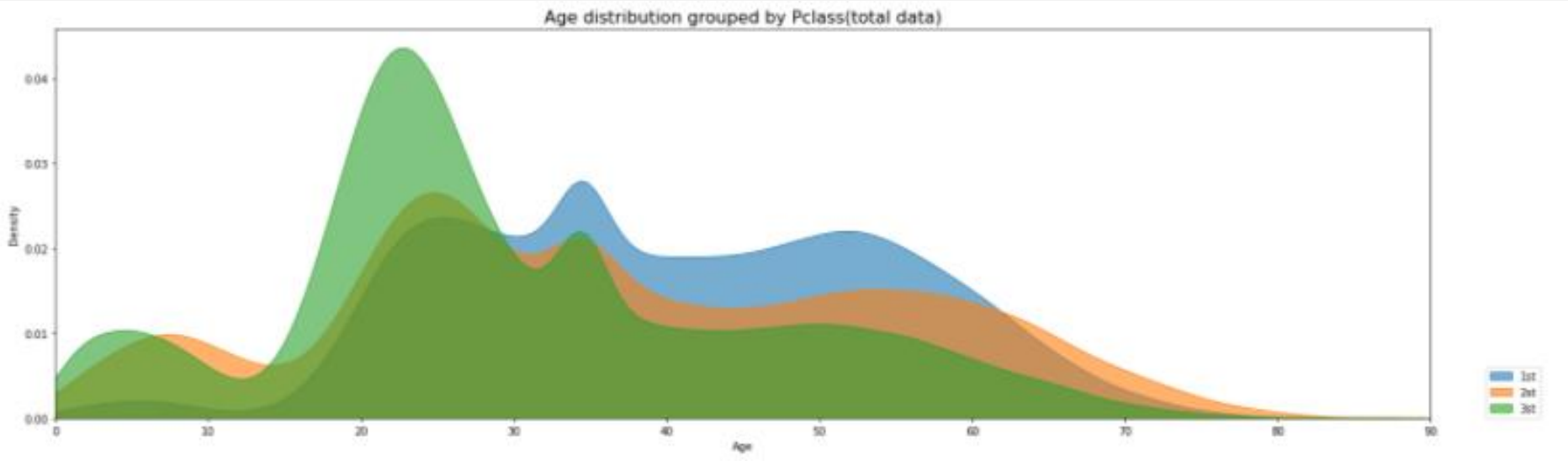
## Fare




➔ Fare의 데이터 값이 불균형하여 값에 Log를 취해서 histogram 생성

## Age

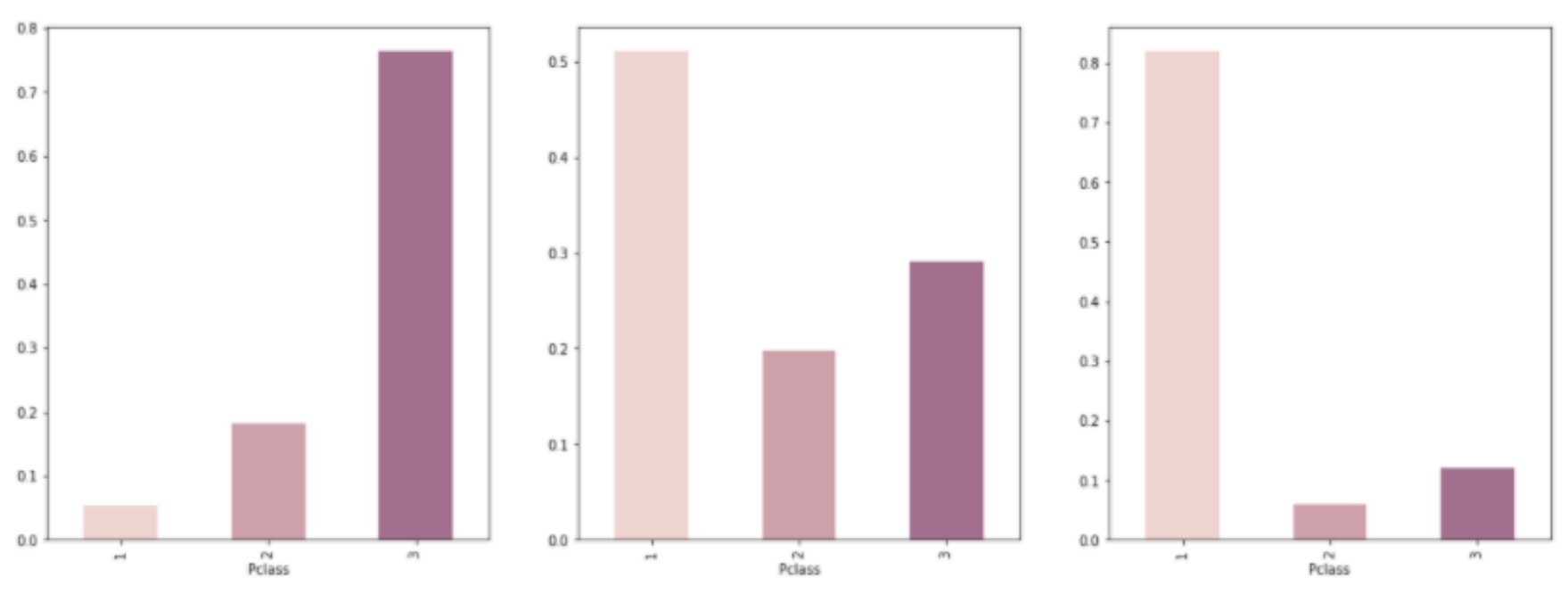
Pclass에 따른 나이 비율




➔ 젊은 층 사람들은 1,2 등석에 비해 3등석에 많이 탑승한 것으로 확인

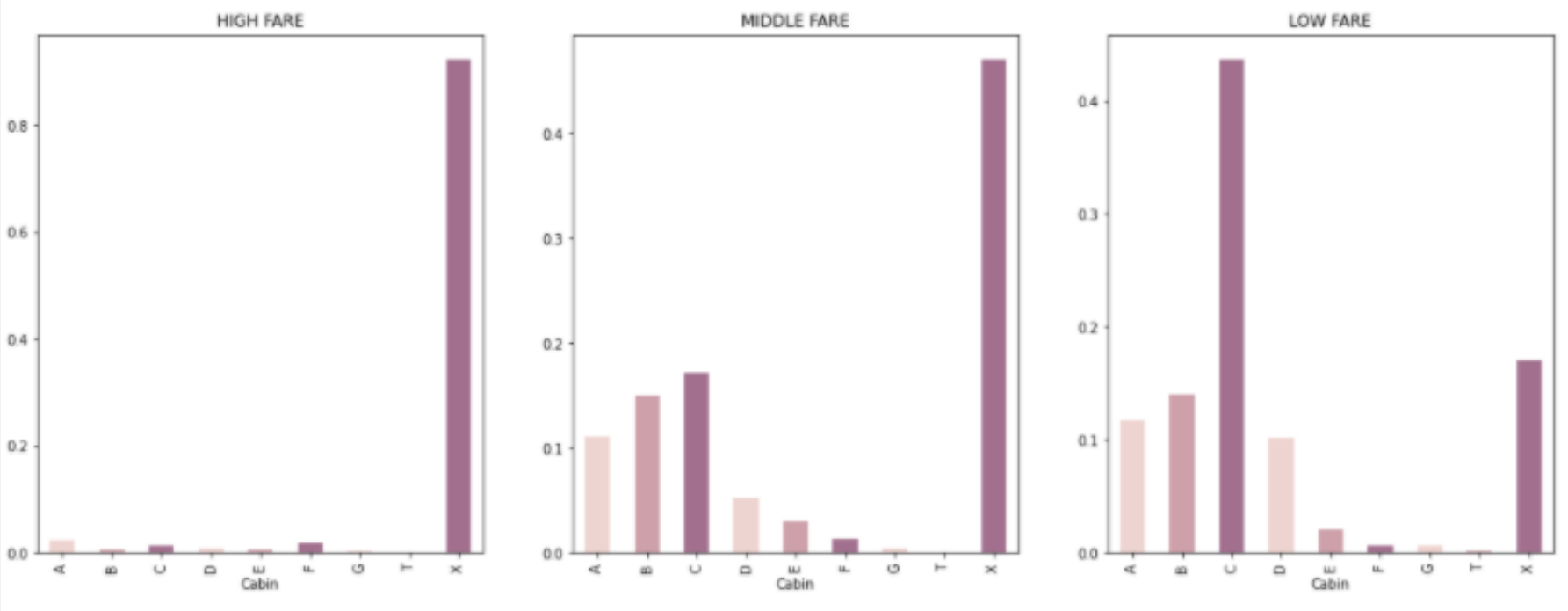
➔ 30대 이상의 사람들은 2, 3등석에 비해 1등석에 많이 탑승함

## Fare & Pclass



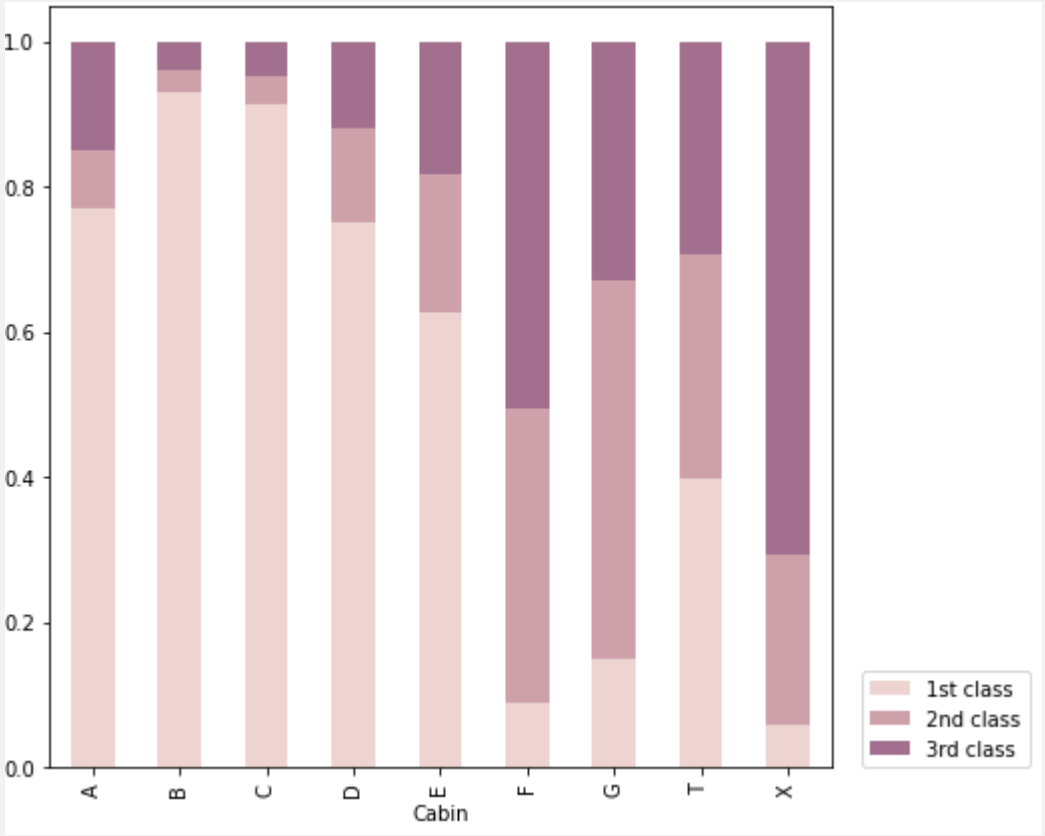

- 첫번째 그래프: LOW Fare일 때 Pclass 에 따른 분포 ➔ 낮은 가격대에는 3등석 승객들이 가장 많이 탑승함
- 두번째 그래프 : MIDDEL Fare 일 때 Pclass 에 따른 분포 ➔ 중간 가격대에서는 1등석 승객들이 가장 많이 탑승함
- 세번째 그래프 : HIGH Fare 일 때 Pclass 에 따른 분포 ➔ 높은 가격대에서는 1등석 승객들이 가장 많이 탑승함

## Fare & Cabin




- 첫번째 그래프: HIGH Fare 일 때 Cabin 에 따른 분포 ➔ 높은 가격대에는 결측치를 제외하곤 다양한 분포를 보임
- 두번째 그래프 : MIDDLE Fare 일 때 Cabin 에 따른 분포 ➔ 중간 가격대에서는 첫번째 그래프에비해 cabin이 골고루 나타남을 보임
- 세번째 그래프 : LOW Fare 일 때 Cabin 에 따른 분포 ➔ 'C'가 가장 많다.

## Cabin Code by Pclass




Pclass에 따른 Cabin Code별 분포

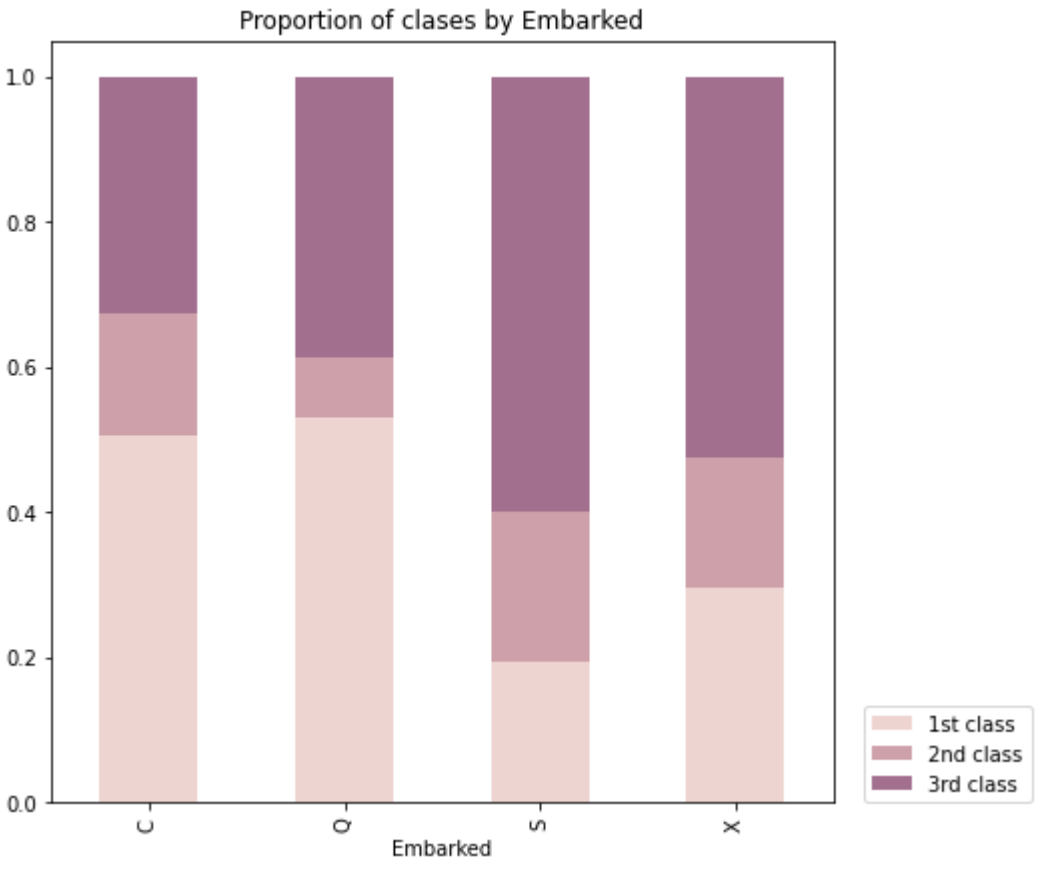
➔ A,B,C,D,E deck에는 1등급 승객들이 가장 많이 분포

➔ F,G에는 2,3등급 승객들이 대부분 분포

➔ T에는 1,2,3 등급 승객들 고르게 분포

➔ 뒷순서의 알파벳일수록 3등급 승객이 더 많아지는 경향을 보인다.

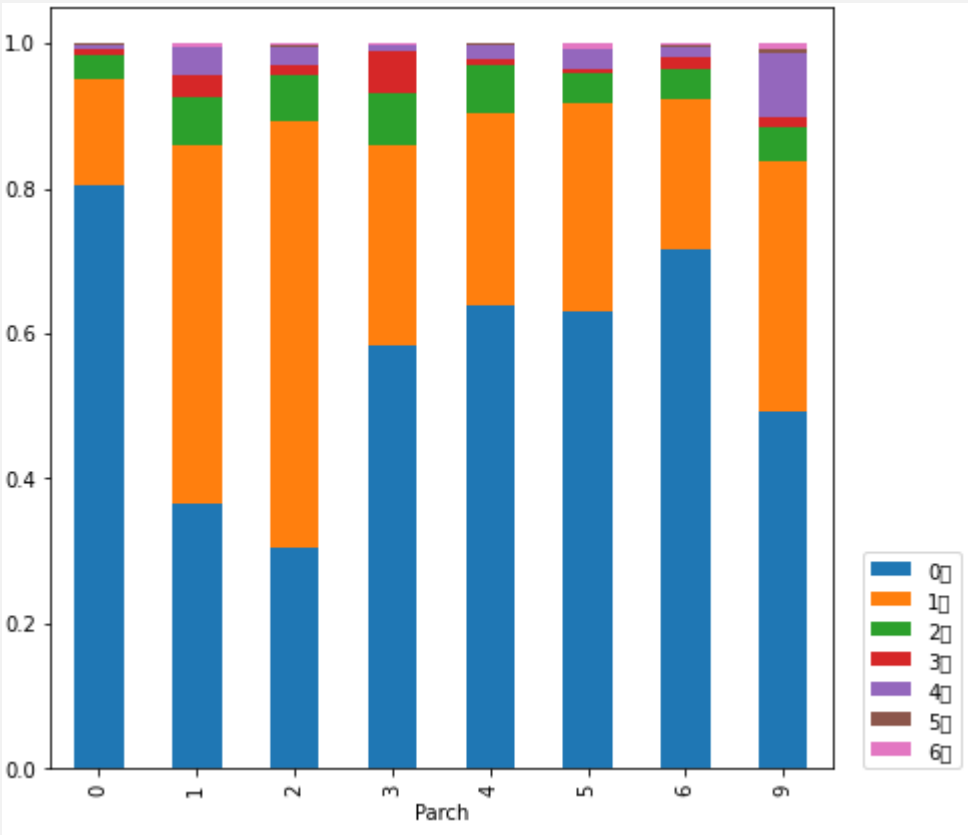
## Pclass by Embarked




Pclass에 따른 탑승항구 분포

➔ S(Southhampton)에는 가장 적은 수의 1등석 승객들, 대부분을 차지하는 3등석 승객들이 분포

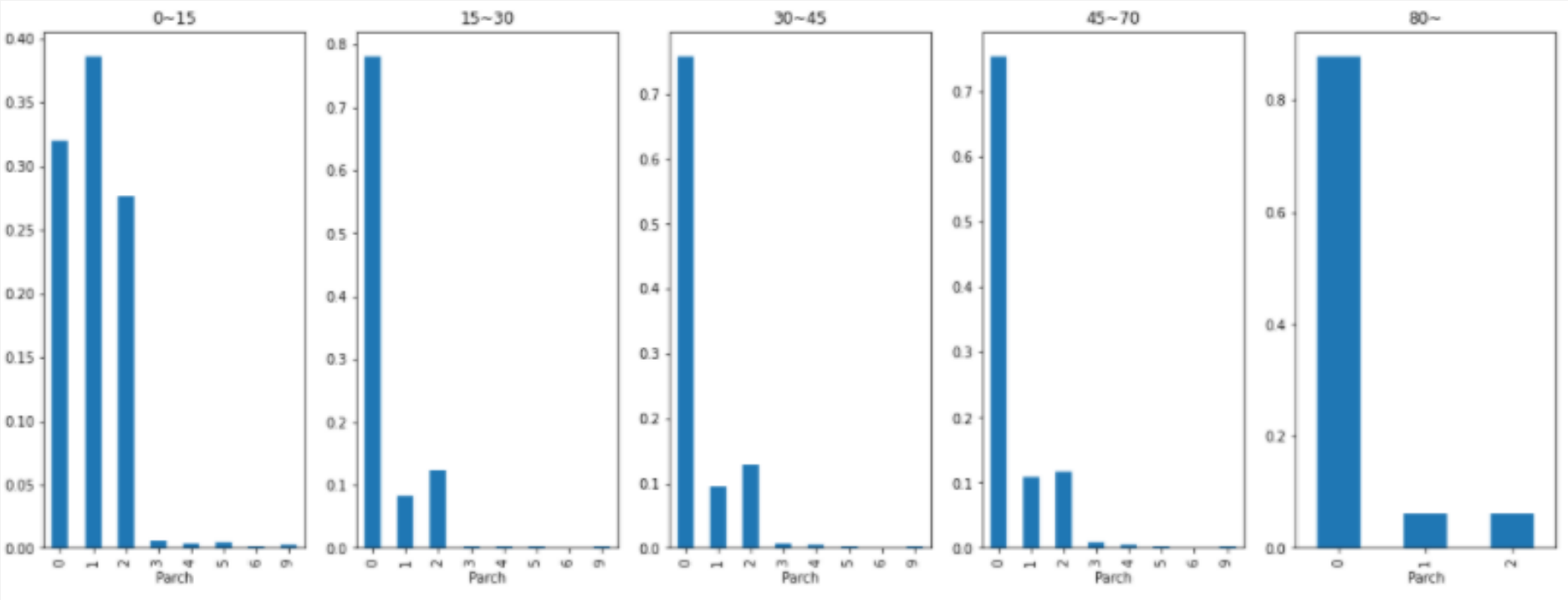
## Parch & Sibsp




➔ 부모와 자식, 형제 자매, 배우자 모두 없이 혼자 온 사람의 비율이 가장 많음

## Parch & Age

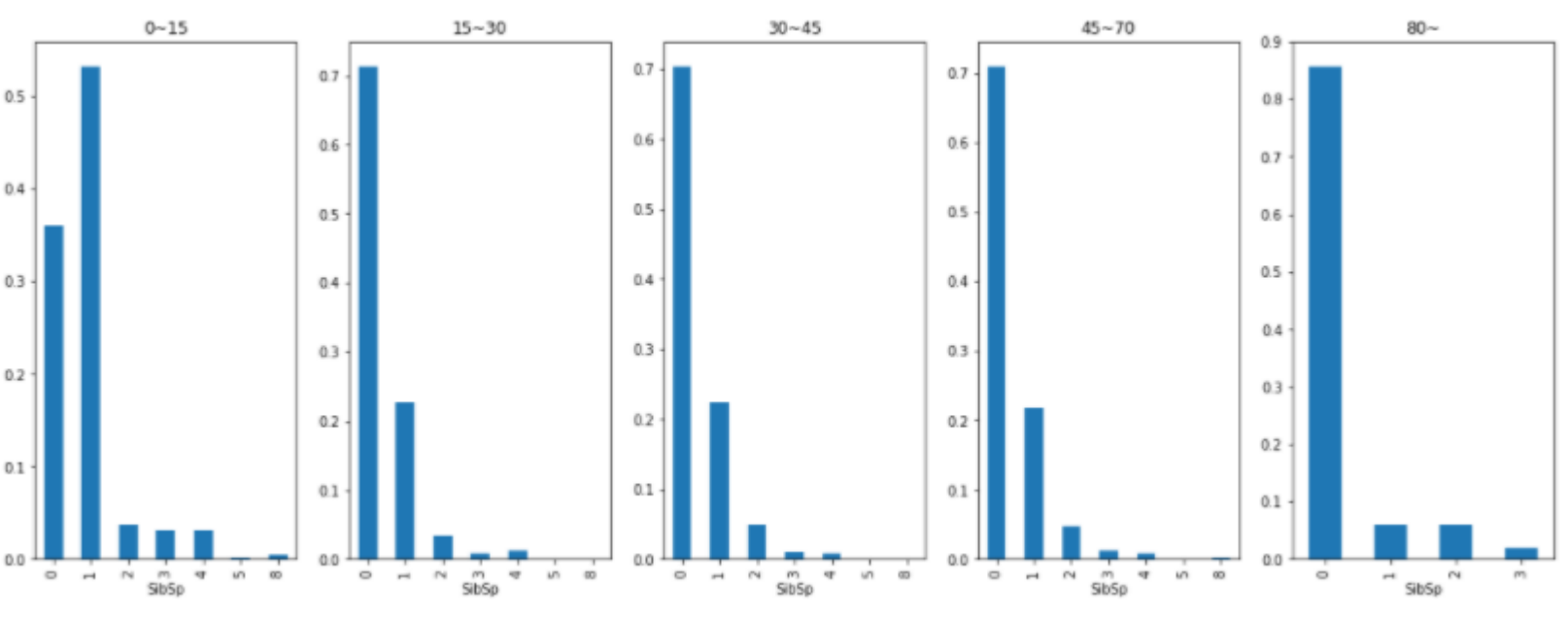
연령대별 부모자식간의 수에 대한 비율




- 80대 이상인 분들을 보면 대부분이 부모 혹은 자식들과 함께 오지 않음
- 연령대별 대체적으로 부모 혹은 자식들과 함께 오지 않음

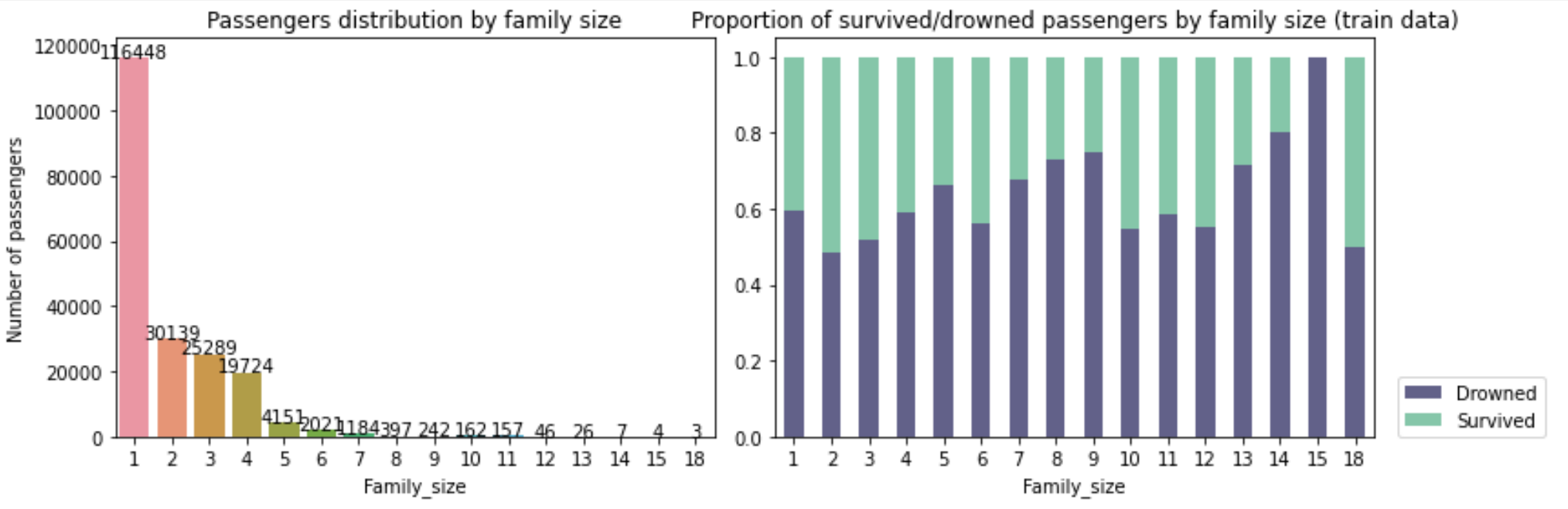
## Sibsp & Age

연령대별 형제자매/배우자 수에 대한 비율



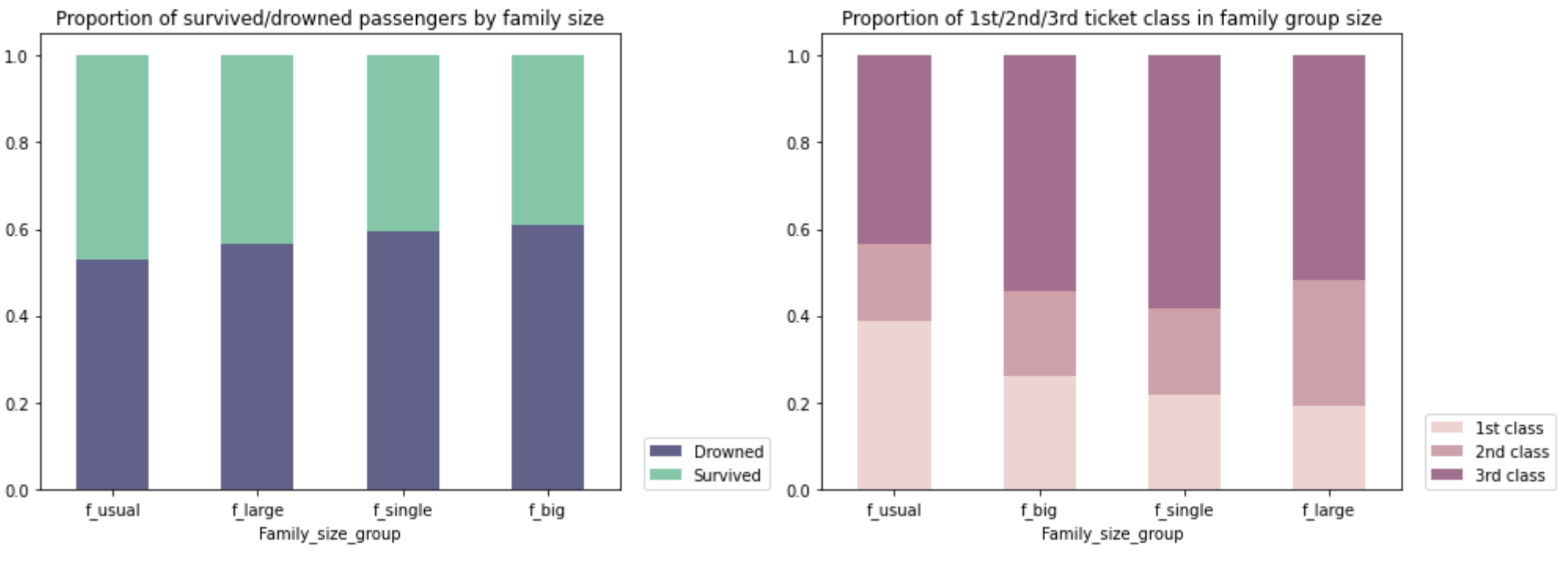

- 가장 어린 연령대에서는 형제 자매 중 자신을 포함하여 2명이 온 경우가 많음
- 가장 높은 연령대에서는 형제 자매, 배우자도 없이 혼자 온 비율이 가장 큼

## Family Size (도출변수) = SibSp + Parch




- Family size 가 15명인 그룹은 모두 살아남지 못하였다.
- 대부분은 혼자 여행하는 사람들이고 생존율은 40% 정도이다.
- 유의미한 의미를 찾지못함.

## Family Size = SibSp + Parch




- single
- usual(sizes 2,3,4,5)
- big(6,7,8,9)
- large

(all bigger then 10)

- 왼쪽 범위에 따라 Family Size를 나눈 결과 ➔ family size에 따른 생존율은 큰 차이가 없음
- 오른쪽 그래프 ➔ family size 그룹에 따른 Pclass 에 대한 승객 수 또한 큰 차이를 보이지 않음

# PART 3 FEATURE ENGINEERING

## Age

Age에 대한 Feature Engineering

생존자의 나이
Age의 Null값에는 전체 나이의 평균을 넣었다. 나이에는 어떠한 경향이 없어보였기 때문이다.
나이별 생존자 비율을 살펴보면, 나이가 많을수록 높은 생존비율을 보인다. 이것의 이유는 다양한 이유로 설명할 수 있을것이다. 시각화 자료를 보면 20대의 3등실 비율이 높았기때문이라고도 설명할 수 있다. 하지만 그렇게만본다면 20대가아닌 다른 연령에대해선 설명이 부족하기때문에 다른 변수와의 상관관계를 고려하지않는 접근을 해보고자 한다. 타이타닉 침몰 당시에 선장의 지시로 "어린" 사람을 먼저 태웠던 것을 근거로 "이 데이터에선" 나이가 많은 사람을 먼저 구조했을수도있다고 추론할 수 있다. 여러 근거로 나이가 많을수록 생존률이 올라가는 경향을 보이기에 이 변수는 유의미하다고 판단했다.

2등 항해사 라이톨러는 선장에게 여자와 어린이를 먼저 태울 것을 건의하고, 선장은 승인했다. 1등 항해사 머독은 더 이상 여성과 아이들이 보이지 않으면 남자를 태우는 것을 허용하는 등 비교적 남성에게도 관대한 대응을 했지만 2등 항해사 라이톨러는 "여자와 어린이 먼저"를 "여자와 어린이만"으로 받아들여 혼란을 막기 위해 여성과 어린이 우선의 관습을 철저히 실시했다.

## Pclass

Pclass에 대한 Feature Engineering

생존자들의 객실. 1에서 3등급 객실이 존재한다.
Pclass에는 결측치가 없어 결측치 처리는 생략한다.
1부터 3으로 이루어진 범주형 데이터이며 항목별 순위가 존재하기때문에 라벨인코딩을 하였다.
객실의 급이다. Pclass별 생존률을 살펴보면 1등실일수록 생존률이 높았다. 그 이유는 여러가지가 있지만, 가장 현실적인 이유는 1,2,3등석 별로 해당되는 갑판에서 구명보트에 탑승했는데 배의 3등석 갑판에는 구명보트가 없었다는 기록으로 보아 pclass 별로 구명보트를 탈수있는 인원이 제한적이었기 때문으로 보인다. 그래서 이 변수는 유의미하다고 판단하였다.

이민법 때문에 이민자들은 건강검사를 하고 이민절차를 밟을 때까지 분리됐다"고 전했다. 또 모든 탑승객들은 1·2·3등석 별로 해당 갑판에서 구명보트에 탑승했다. 다만, 결정적으로 배의 3등석 갑판에는 구명보트가 없었다. 따라서 3등석 탑승객들은 1·2등석 탑승객들과 달리 미로같은 복도와 계단을 통해 살 길을 찾아야 했다.

## Embarked

**Embarked**에 대한 Feature Engineering

배를 탄 항구. 결측치를 제외하면 세가지 문자로 이루어져있다.
결측치는 굉장히 낮은 비율이었고, X로 표시했다.
범주형 데이터이며, 종류는 별로 없었지만 항구별로 순위가 있어보였기때문에 라벨인코딩을 하였다.
. S에서 탄 사람이 가장 많았고, 그다음이 C,Q 순이다. 시각화 자료를 봤을때, 항구별로 생존률이 확연하게 차이가 났다. (C>Q>S) 이 차이는 탑승항구별로 빈부격차가 존재했기때문으로 추측했고, 항구별 pclass를 시각화하니 지켜볼만한 결과가 나왔다. 3등실에 묵지않은 사람의 비율이 (C>Q>S)였기 때문이다. 이를 통해 의견이 타당하다고 생각했고, 이 변수는 생존여부에 유의미한 변수라고 생각했다.

## Name

**Name**에 대한 Feature Engineering

승객의 이름. 결측치는 없지만, 성이 없는 데이터가 존재한다.
이름 보다는 성이 중요할것같아, 성만 분리하여 데이터 처리를 하였고, 성이없는 값은 X로 표시했다.
문자로 이루어진 범주형 데이터. 종류가 굉장히 많아 라벨인코딩으로 진행하였다.
성에 따라서는 그 시대에 흔한 성(박,김,이와 같은) 일수도 있고, 귀족가문의 성이 있을수도있다. 시각화한 자료에따르면, 가장 많은 수의 성을 가진사람이 많이 살아남았다. 하지만 이것은 그 성을 가진 사람의 수 자체가 많기때문이라고 판단하였다. 그래서 생존여부에 큰 영향을 주는 변수가 아니라고 판단하여 평가지표를 통해 확인하였고 Name 변수를 제외하자 조금의 차이로 낮게 나와 조금의 영향을 가진 변수라고 판단하였다.

## Sex

**Sex**에 대한 Feature Engineering

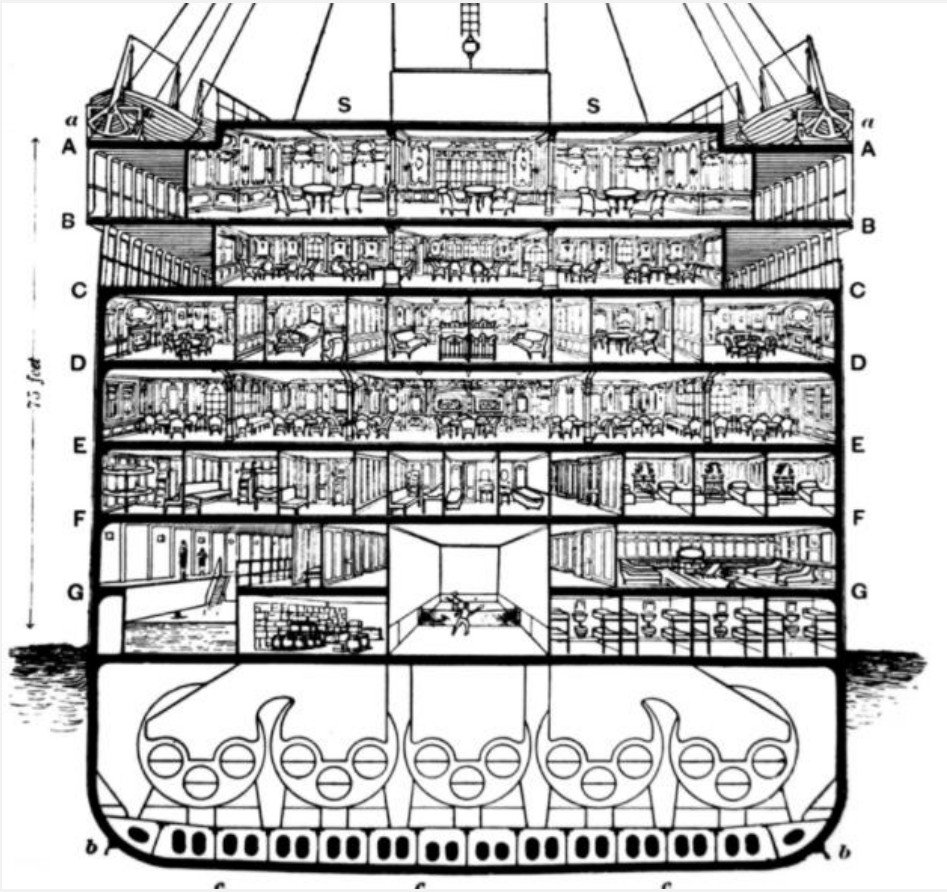
승객의 성별. 결측치는 없다. 여성과 남성으로 구분되어있다.
0과 1로 이루어진 범주형 데이터이다. 여성의 생존비율이 더 높아 우선순위가 있는것같아 라벨인코딩.
시각화 자료에 따르면 여성의 생존비율이 더 높은 것을 확인 할 수 있었다. 타이타닉호의 사고 당시,
선장은 "여자와 어린이를 먼저" 태우라고 지시했다. 그리고 항해사마다 그 지시를 받아들인 형태가
달랐는데, 1등 항해사 머독은 남자들도 태운 반면, 2등항해사 라이톨러는 지시를 철저히 실시했기에
여성의 생존률이 높았다고 추정된다. 그래서 성별에 따른 생존여부가 유의미하다고 판단하였다.

2등 항해사 라이톨러는 선장에게 여자와 어린이를 먼저 태울 것을 건의하고, 선장은 승인했다. 1등 항해사 머독은 더 이상 여성과 아이들이 보이지 않으면 남자를 태우는 것을 허용하는 등 비교적 남성에게도 관대한 대응을 했지만 2등 항해사 라이톨러는 "여자와 어린이 먼저"를 "여자와 어린이만"으로 받아들여 혼란을 막기 위해 여성과 어린이 우선의 관습을 철저히 실시했다.

## Cabin

Cabin에 대한 Feature Engineering

승객이 있었던 갑판. A~부터 알파벳으로 이루어져있다.
결측치가 총 데이터의 70%결측치는 X로 표시했다.
Cabin의 갑판 종류 하나하나에 대해 분석하면 좋을것같아 one-hot 인코딩으로 진행하였다.
보면 A갑판일수록 위쪽에 위치한다. 배가 침몰할때에는 아래쪽에서부터 물이 차오르기 때문에 아래쪽 갑판일수록 생존률이 낮아질수밖에 없다고 생각된다. 그래서 이 독립변수는 생존에 유의미한 변수라 판단하였다.



## SibSp

SibSp에 대한 Feature Engineering

형제자매의 수. 0부터 8까지 다양하다.
처음에 형제자매의 수가 많을수록 생존률이 낮아질거라 추론하였지만 시각화 자료를 보면 5명 가족을 제외하곤 생존률이 거의 동일하다. (이마저도 5명의 형제자매를 가진 생존자가 표본이 매우적어 편향되었을 가능성이 높다고 판단했다) 그래서 이 독립변수는 생존에 그렇게 큰 영향을 주는것도 아니지만, 생존여부의 객관성을 떨어트리는 지표는 아니라고 판단하였다. 그래서 평가지표를 통해 확인하였고 sibsp 변수를 제외하자 조금의 차이로 낮게 나와 조금의 영향을 가진 변수라고 판단하였다.

## Parch

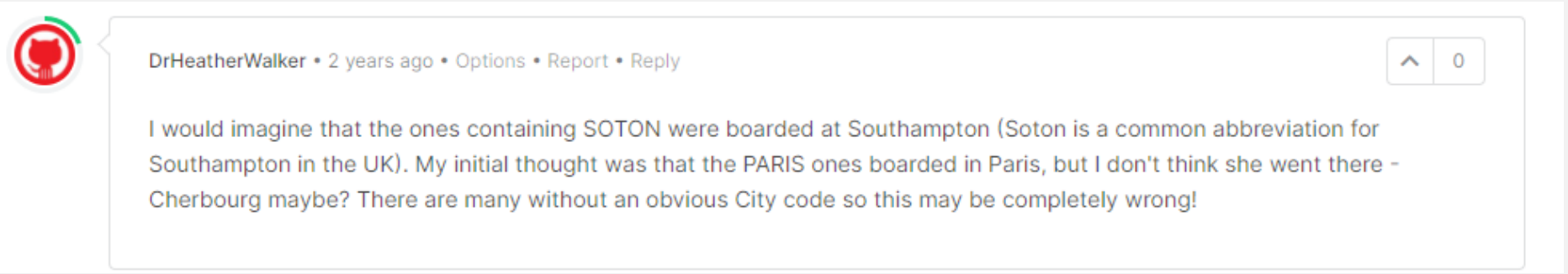
**Parch**에 대한 Feature Engineering

승객의 배우자와 자식의 수.
이것또한 0명이 가장 많았다. 하지만 시각화를 하면 sibsp 와 다르게, 이 독립변수는 회귀모형의 중요한 가정 중 하나인 선형성을 크게 위반하는 양상을 띈다. 그래서 이 독립변수는 회귀모형에 적합하지 않은 데이터라고 판단하였다. 그래서 이 독립변수는 유의미하지않다고 판단하여 회귀모형에서 제외시킨다.

## Ticket

### **Name**에 대한 Feature Engineering

숫자와 문자로 구성되어있다. 숫자는 티켓의 번호, 문자는 티켓의 발행지로 추정된다
문자와 숫자를 구분하여 문자만 따왔고, 결측치와 문자가 없는 데이터는 X로 기입했다.
티켓의 번호는 의미가 없고, 티켓의 발행지는 지역에따라 묶을 수 있으므로 유의미하다고 판단했다. 우리는 티켓의 발행지에 따라 빈부격차가 존재한다고 생각하였고 pclass와 함께 시각화를 해보았다. 그 결과 높은 pclass의 비율이 큰 발행지에 따라 생존율이 높아지는 것을 확인하였다. 그래서 생존여부에 상관이 있을거라 판단하여 넣기로 했다.



## Fare

**Fare**에 대한 Feature Engineering

승객별 지불 요금이다.
상식적으로 지불요금은 타이타닉의 객실에 지불한 요금이 지배적일거라 판단하여 pclass별 요금 평균을 결측치에 넣었다.
Fare의 시각화 자료를 보면, 지불한 금액이 많을수록 높은 생존율을 보이는 경향을 확인 할 수 있다. 이것은 cabin 또는 pclass의 영향이 있어보였다. Fare는 다른 독립변수들에 비해 pclass와 cabin과의 상관관계 지수가 높은 것을 근거로 들 수 있다. Fare에 따라서 높은 생존율을 보이는 것을 pclass와 연관지어 설명하면, 1등객실 사람들은 A갑판부터 E갑판까지 이동이 자유로웠지만, 2등실부터는 이동이 제한적이었다. 그래서 높은 요금을 지불한 사람일수록 생존율이 높아질 수 밖에 없다고 판단하였다. 그래서 이 독립변수는 유의미하다고 판단하였다.

**2등실** [편집]

2등실에는 총 285명의 중산층 승객이 타고 있었다. 전체적으로 봤을 때 1등실만큼 좋지는 않았지만 그래도 비교적 편리한 시설이 설치되어 있었다. 객실은 갑판 D부터 갑판 F까지 설치되어 있었다. 흡연실(갑판 B), 레스토랑(갑판 B), 도서관(갑판 C), 상점 등이 있었다.

**3등실** [편집]

3등실에는 총 710명의 가난한 승객들이 타고 있었다. 주로 아메리칸 드림으로 미국에서 새로운 보금자리를 얻기 위해 승선한 승객들이었다. 객실은 2등실과 마찬가지로 갑판 D부터 갑판 G까지 설치되어 있었다.시설은 1등실과 2등실만 못하고 엔진이 가동되는 소리가 울려퍼졌으나 다른 배들에 비해서 비교적 좋은 대우를 해주었다. 배에 탑승하기 전에는 검역을 걸쳐서 전염병이나 이/벼룩을 확인했고 여자와 남자는 배의 앞머리와 뒷머리에 각각 따로 떨어져 승선했으나 가족 단위일 경우 같이 승선할 수 있었다.

# PART 4 MODELING

## Summary

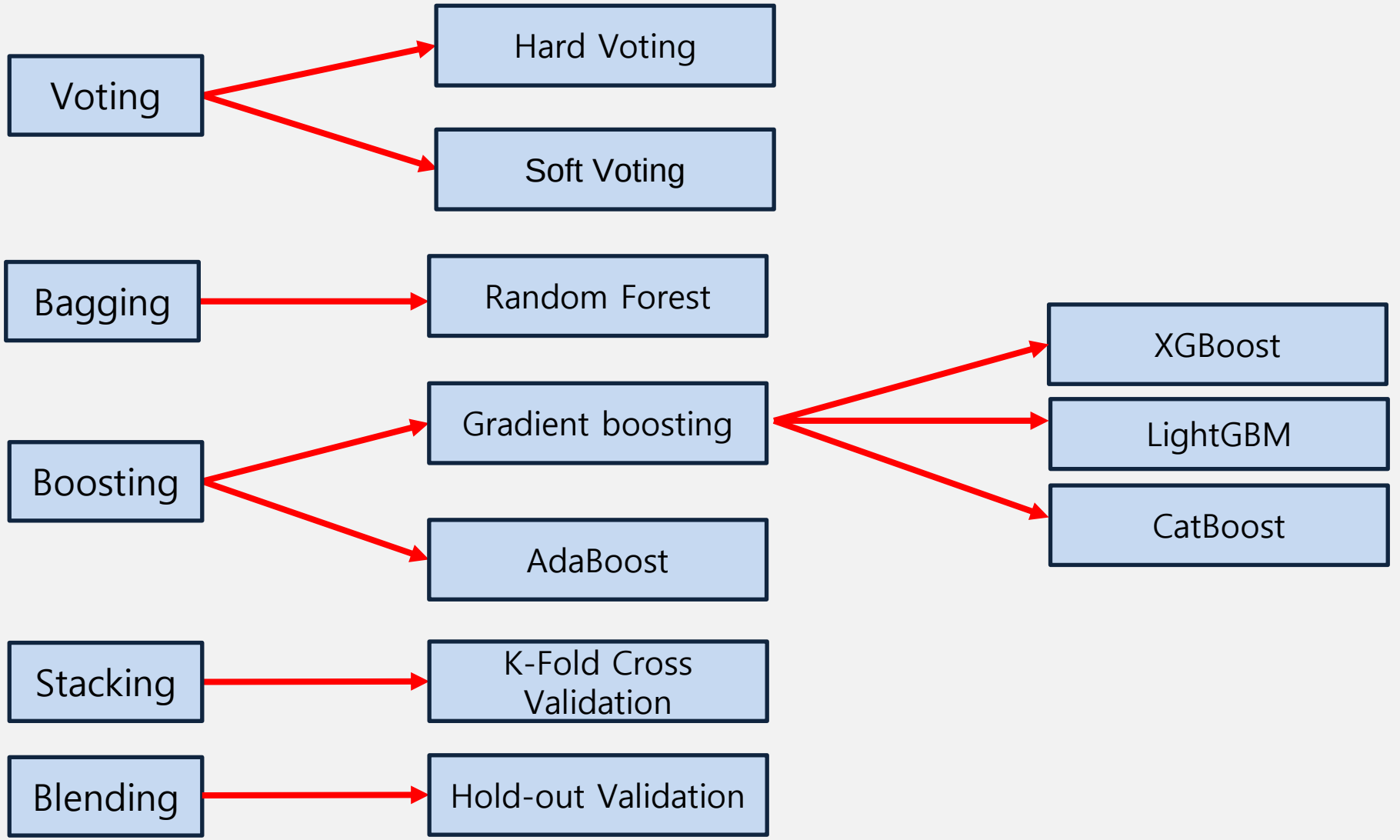
**목적**

모델링 과정에 대한 전반적인 이해에 초점을 맞추고 EDA에 맞춰 가장 좋은 학습 모델을 만드는 것

**결론**

- Beginner 에겐 Pycaret이 편리한 Framework 이다.
- Pycaret을 통해 데이터 셋에 가장 적합한 모델을 찾았고, 그것은 LightGBM과 Catboost의 앙상블 모델이다.
- Stacking과 Blending이 모든 데이터에 적합한 것이 아니다.

## 앙상블 학습 유형


Voting
Hard Voting
Soft Voting
Bagging
Random Forest
Boosting
Gradient boosting
AdaBoost
XGBoost
LightGBM
CatBoost
Stacking
K-Fold Cross Validation
Blending
Hold-out Validation

## 개념 정리 – ensemble 학습 방법

Hard / Soft Voting

Stacking

Blending

## Hard Voting / Soft Voting 과정

Hard Voting

```
0        0
1        2
2        2
3        0
4        2
        ..
99995    2
99996    0
99997    0
99998    2
99999    2
Name: Label, Length: 100000, dtype: int64
0        0.0
1        1.0
2        1.0
3        0.0
4        1.0
        ...
99995    1.0
99996    0.0
99997    0.0
99998    1.0
99999    1.0
Name: Label, Length: 100000, dtype: float64
```

Soft Voting

```
[0.23772708 1.03853512 1.71419914 ... 0.17386636 1.51608978 1.81941758]
[0. 1. 1. ... 0. 1. 1.]
```

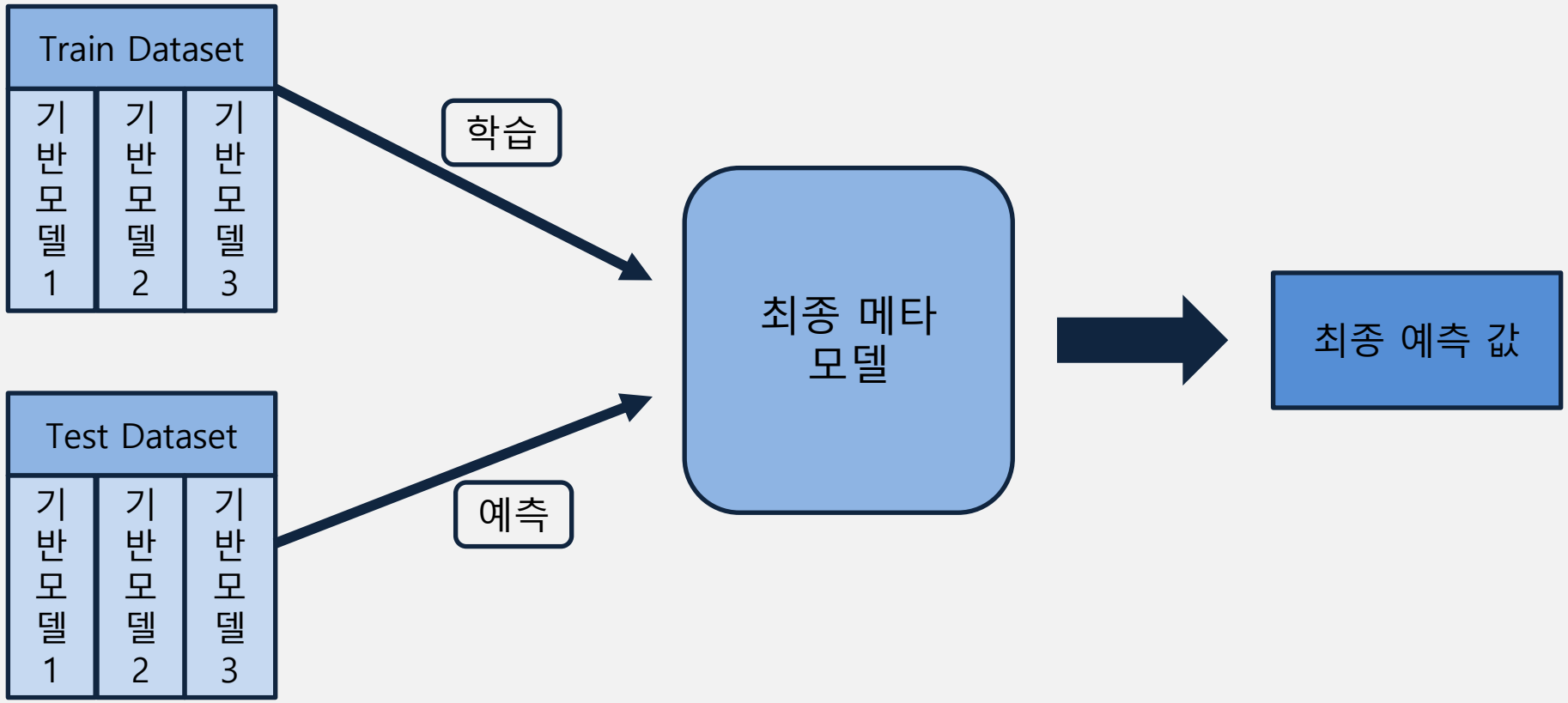
- Hard Voting보다 Soft Voting을 채택하는 이유

➔ 단순히 투표를 통해 정하게 되면, 좋지 않은 판단 모델이 많을 수록 잘못된 판단을 할 확률이 높아진다.

➔ Soft Voting은 확률이기 때문에 다수결의 오류를 범할 가능성이 적다.

➔ 따라서 Soft Voting을 채택한다.

## Stacking


Train Dataset
기반모델1
기반모델2
기반모델3
학습
Test Dataset
기반모델1
기반모델2
기반모델3
예측
최종 메타 모델
최종 예측 값

## 개념 정리 – Hyper Parameter 탐색 방법

### Grid Search

- 그리드 서치란?

➔ 격자탐색이라 하며, 모델의 하이퍼 파라미터에 넣을 수 있는 값들을 순차적으로 입력

➔ 가장 높은 성능을 보이는 하이퍼 파라미터를 찾는 탐색 방법

### Random Search

- 랜덤 서치란?

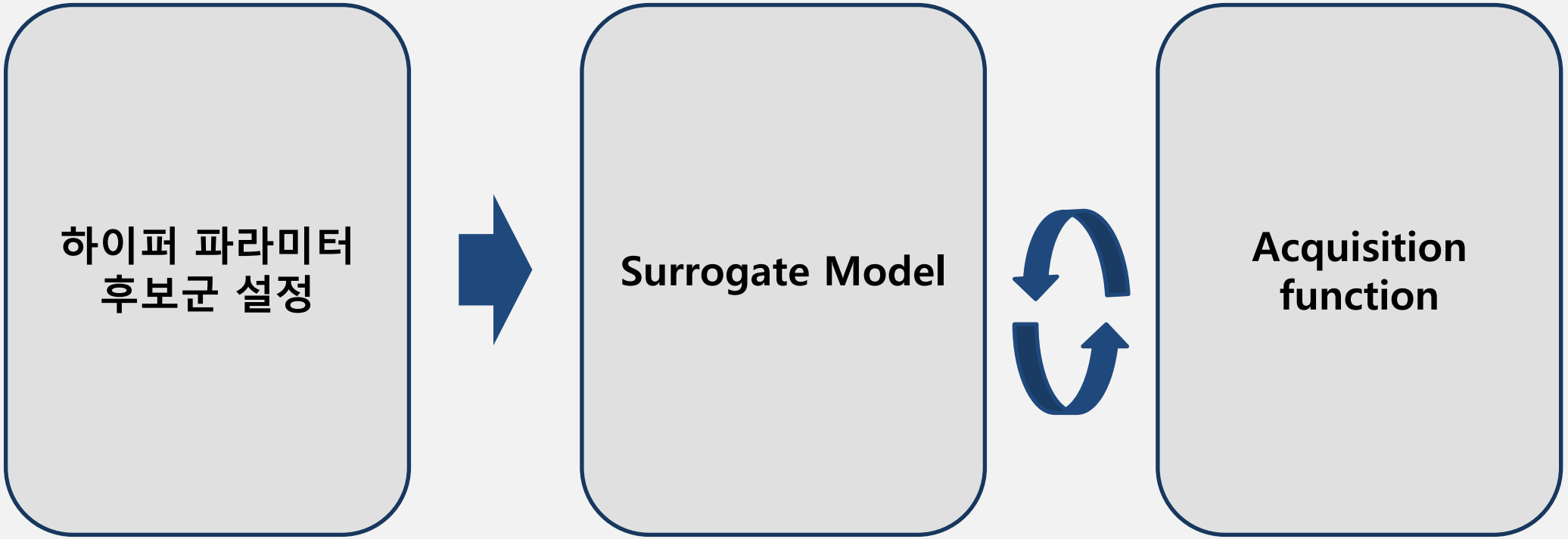
➔ 그리드 서치는 말 그대로 모든 경우를 테이블로 만든뒤 격자로 탐색하는 방식!

➔ 그러나 랜덤 서치는 하이퍼 파라미터를 랜덤하게 넣는다

➔ 그 중 우수한 값을 모인 하이퍼 파라미터를 활용!

## Bayesian Optimization

불필요한 하이퍼 파라미터 반복 탐색을 줄여 빠르게 최적의 하이퍼 파라미터를 찾을 수 있는 방법

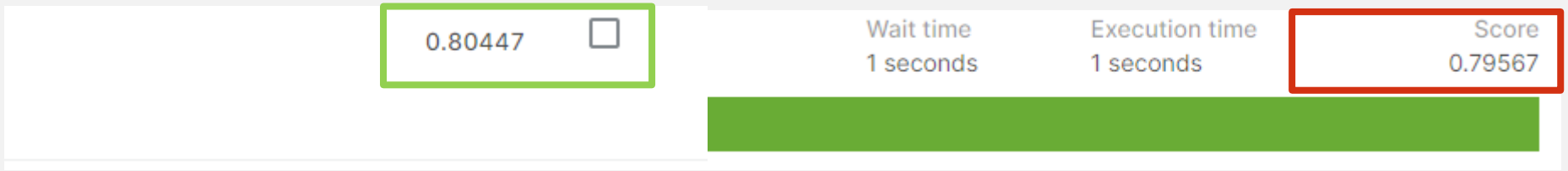


## Bayesian Optimization

**사용하지 않는 이유**

- 파라미터를 제대로 이해 못한 상태에서 임의로 파라미터 많은 앙상블 모델을 사용하는데 모델 하나하나의 파라미터를 전부 파악할 수 없다.
- Sklearn 하이퍼 파라미터 튜닝을 하면 임의의 범위이기 때문에 좋은 모델 튜닝이 되었다고 확신 할 수 없음

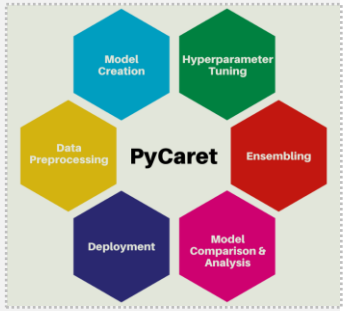
소모하는 시간에 비해 적절한 튜닝이 이루어지지 않음



## Scikit Learn


- 머신러닝 관련한 기술들을 통일되고 쉬운 인터페이스로 사용할수 있게 해주는 라이브러리
- 라이브러리 외적으로 scikit 스택을 사용하고 있기 때문에 다른 라이브러리와의 호환성이 좋다는 것이 가장 큰 장점
- 딥러닝이나 강화 학습을 다루지 않고, 그래픽 모델과 시퀀스 예측(Sequence Prediction) 기능을 지원하지 않음
- 파이썬 이외의 언어에서는 사용할 수 없고, 파이썬 JIT(Just-in-Time) 컴파일러인 파이파이(PyPy)나 GPU를 지원하지 않음
- 딥러닝이나 강화 학습을 지원하지 않아 정확한 이미지 분류와 신뢰성 있는 실시간 언어 분석, 번역 같은 문제를 해결하는 데는 적절치 않음

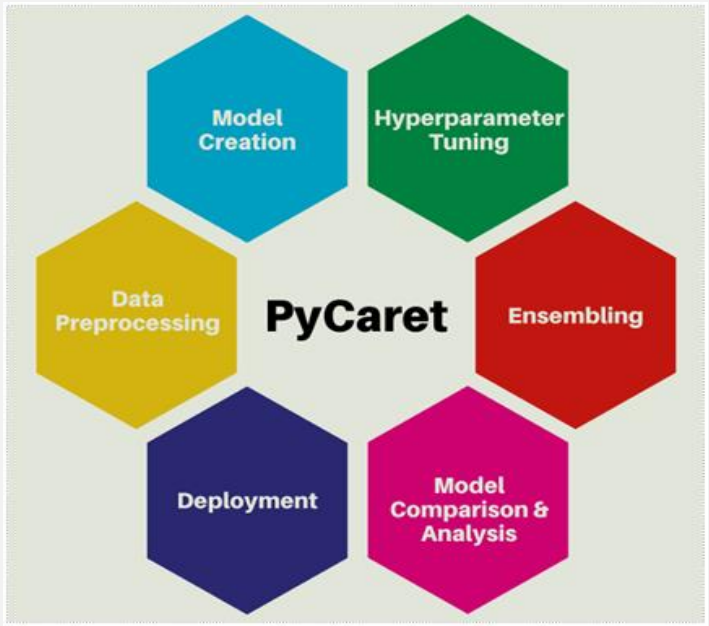
## Pycaret




- 수백 줄의 코드를 몇 단어로만 대체하는 데 사용할 수 있는 대체 로우 코드 라이브러리
- 매우 적은 코드 라인과 적은 수동 구성으로 많은 것을 얻을 수 있다
- 패키지의 목표는 분류 및 회귀를 위한 기계 학습 알고리즘을 평가하고 비교하는 주요 단계를 자동화하는 것
- 코딩에 대한 배경 지식이 거의 또는 전혀없는 데이터 과학을 처음 접하는 사람들에게 적합
- Scikit-Learn , XGBoost , LightGBM 등과 같은 여러 기계 학습 라이브러리 및 프레임 워크를 둘러싼 Python 라이브러리
- Pycaret을 사용하면 직관적이며, 빠르고 효율적이다.

## Sklearn vs Pycaret

VS




- 결론 : Pycaret을 사용하여 자동으로 하이퍼파라미터 튜닝을 하고, 예상치까지 계산하는 것이 편리하다고 생각하기 때문에 채택함

## Sklearn을 채택하지 않은 이유

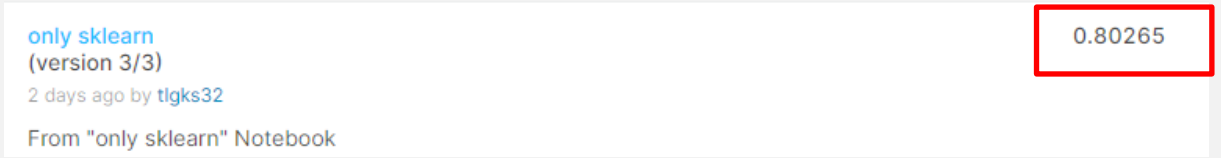
- 사이킷 런에서 하이퍼 파라미터 튜닝을 위한 서치를 진행.

모든 모델에 대한 파라미터에대한 이해가 부족하기때문에 한정된 파라미터 내에서 한정된 수치를 지정할 수 밖에 없는 한계가 있다.

```
[ ]: random_grid={'n_estimators':n_estimators,
                  'max_depth':max_depth,
                  'min_data_in_leaf':min_data_in_leaf
                 }

[ ]: lgbm_model=lgb.LGBMClassifier()

▶ rs_lgbm=RandomizedSearchCV(lgbm_model,param_distributions=random_grid)
  rs_lgbm.fit(X,y)
```

```
[ ]: after_lgbm=lgb.LGBMClassifier(n_estimators= 56, min_data_in_leaf=118, max_depth=6)
     after_gbc=GradientBoostingClassifier(min_samples_split= 2, min_samples_leaf= 12, max_depth= 3,random_state=1)
     after_ada=AdaBoostClassifier()

+ Code + Markdown

[ ]: after_lgbm.fit(X,y)
     after_gbc.fit(X,y)
     after_ada.fit(X,y)
```

파이캐럿과 동일한 방식으로 summit을 해보았으나 더 낮은 점수가 나왔다.

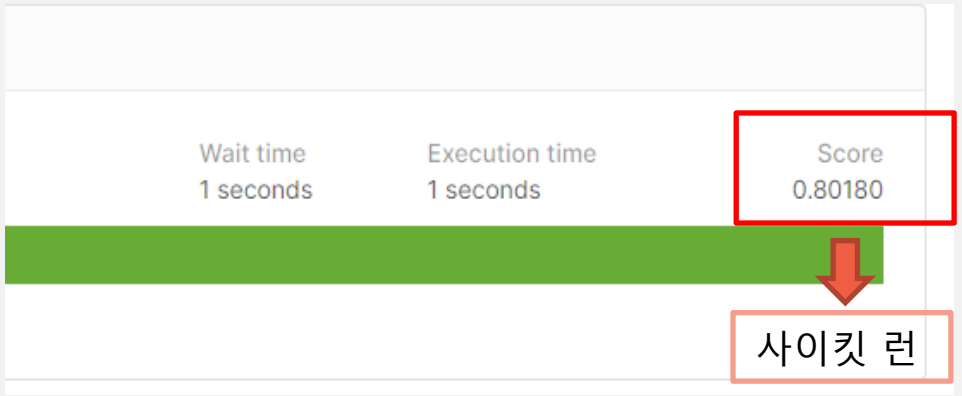


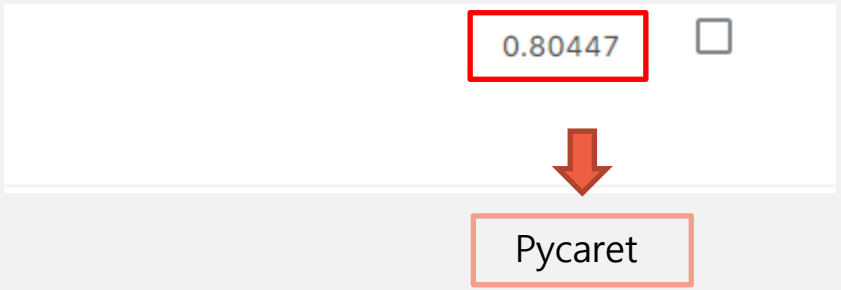

Pycaret에서 자동으로 tune한것과 모델의 파라미터에 대한 이해가 부족한 채, 일일히 파라미터 튜닝을 한 것의 차이로 보인다.

## Sklearn의 프레임워크와 Pycaret의 프레임 워크는 다르다.

- 사이킷 런에서 Pycaret과 동일한 파라미터를 넣어 모델링 진행.

Pycaret에서 tune_model 후, 튜닝이 완료된 파라미터를 sklearn 모델에 튜닝했다.




Pycaret의 파라미터로 sklearn lgbm, catboost 에서 predict 까지 돌려보았다.
0.80180으로 다른 결과를 확인할 수 있었다.
비슷한 프레임워크라고 생각했었지만, 모델링과 predict를 하는 과정에서 차이가 발생하고있다고 판단할 수 있다.

Pycaret과 sklearn의 프레임워크 구조는 동일하지 않다는 것을 알 수 있다.

# PYCARET – Compare model

```
best_model = compare_models(sort = 'Accuracy', n_select = 4)
```

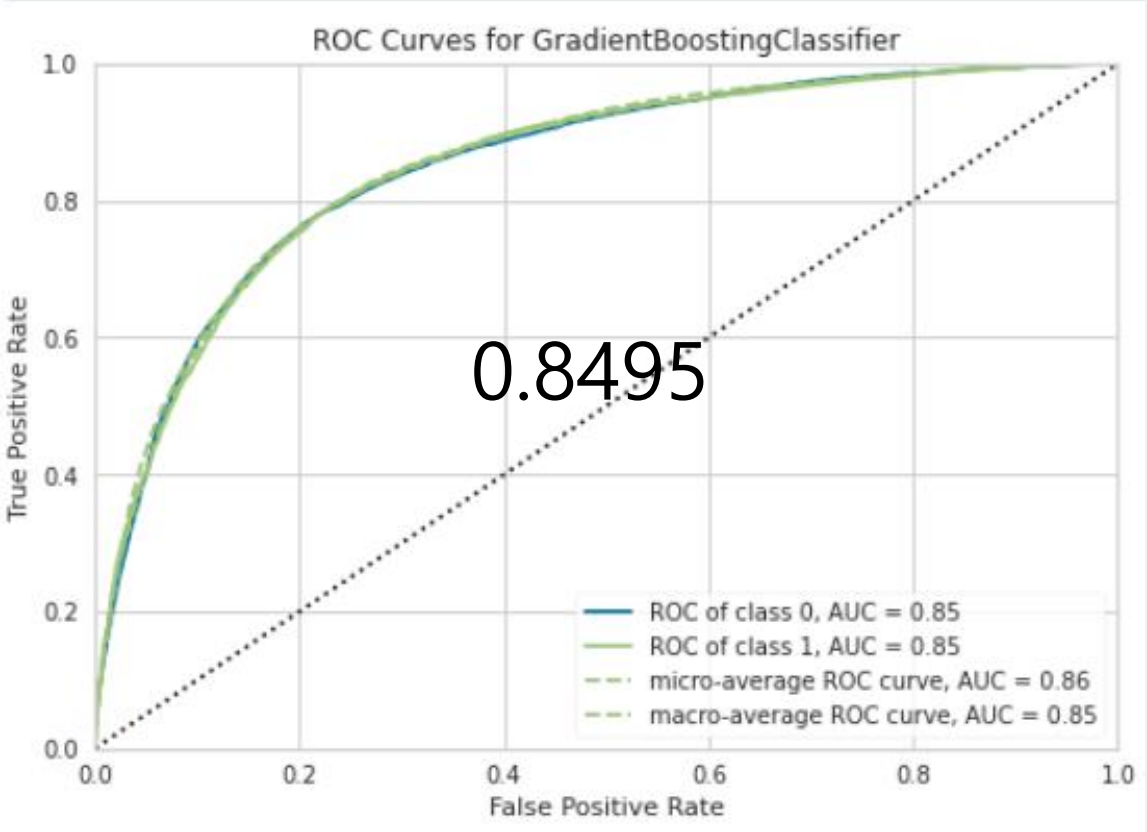
| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| lightgbm | Light Gradient Boosting Machine | 0.7824 | 0.8505 | 0.7426 | 0.7477 | 0.7451 | 0.5552 | 0.5553 | 0.7380 |
| gbc | Gradient Boosting Classifier | 0.7803 | 0.8495 | 0.7479 | 0.7416 | 0.7447 | 0.5519 | 0.5519 | 4.0600 |
| catboost | CatBoost Classifier | 0.7796 | 0.8492 | 0.7373 | 0.7456 | 0.7414 | 0.5494 | 0.5495 | 21.5960 |
| xgboost | Extreme Gradient Boosting | 0.7767 | 0.8451 | 0.7308 | 0.7435 | 0.7371 | 0.5431 | 0.5432 | 21.1440 |
| ada | Ada Boost Classifier | 0.7751 | 0.8432 | 0.7047 | 0.7542 | 0.7286 | 0.5369 | 0.5379 | 1.2580 |
| lda | Linear Discriminant Analysis | 0.7723 | 0.8351 | 0.7548 | 0.7249 | 0.7396 | 0.5374 | 0.5377 | 0.1640 |
| ridge | Ridge Classifier | 0.7722 | 0.0000 | 0.7543 | 0.7250 | 0.7394 | 0.5372 | 0.5375 | 0.0960 |
| rf | Random Forest Classifier | 0.7647 | 0.8289 | 0.7073 | 0.7338 | 0.7203 | 0.5174 | 0.5176 | 4.4400 |
| et | Extra Trees Classifier | 0.7571 | 0.8216 | 0.6982 | 0.7247 | 0.7112 | 0.5017 | 0.5019 | 3.5980 |
| lr | Logistic Regression | 0.7419 | 0.8090 | 0.6410 | 0.7242 | 0.6786 | 0.4648 | 0.4683 | 1.1340 |
| nb | Naive Bayes | 0.7169 | 0.8068 | 0.5643 | 0.7148 | 0.6307 | 0.4063 | 0.4137 | 0.0900 |
| dt | Decision Tree Classifier | 0.6883 | 0.6820 | 0.6384 | 0.6356 | 0.6370 | 0.3639 | 0.3639 | 0.3020 |
| knn | K Neighbors Classifier | 0.6146 | 0.6341 | 0.4695 | 0.5598 | 0.5107 | 0.1968 | 0.1988 | 0.6460 |
| svm | SVM - Linear Kernel | 0.6026 | 0.0000 | 0.1294 | 0.7396 | 0.1651 | 0.0912 | 0.1352 | 2.0460 |
| qda | Quadratic Discriminant Analysis | 0.5674 | 0.5083 | 0.1288 | 0.5144 | 0.1874 | 0.0275 | 0.0460 | 0.1160 |

Best Model
TOP3
- LightGBM
- GBC
- Catboost

## PYCARET – ROC Curve

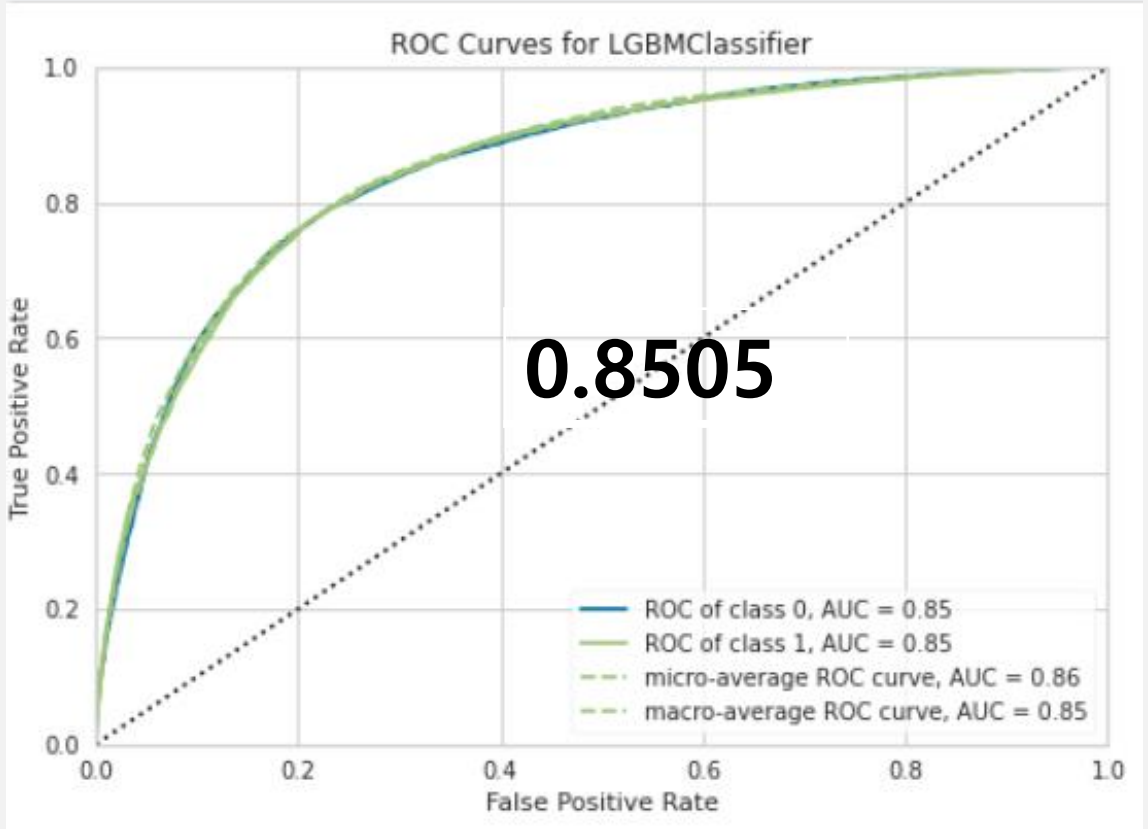
### Gradient Boosting

```
plot_model(gbc)
```




### LightGBM

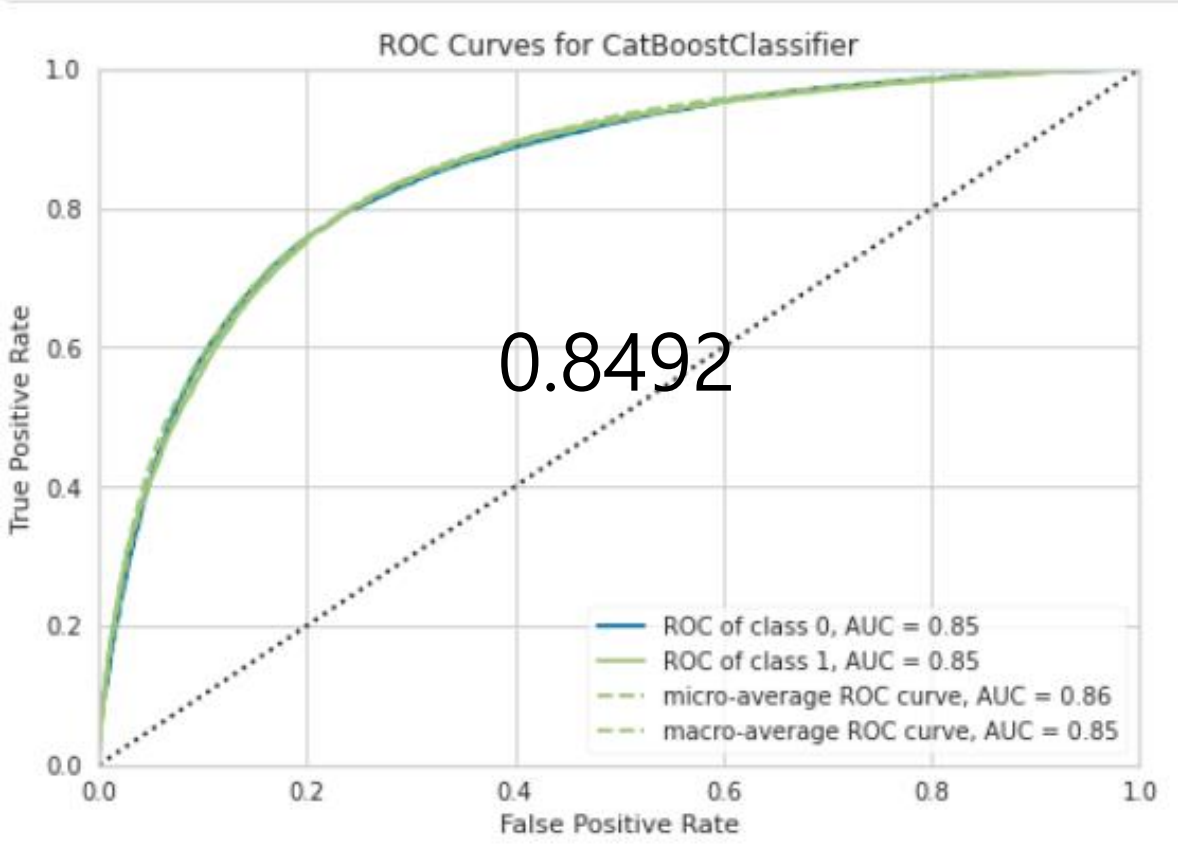
```
plot_model(lgbm)
```



## PYCARET – ROC Curve

Catboost

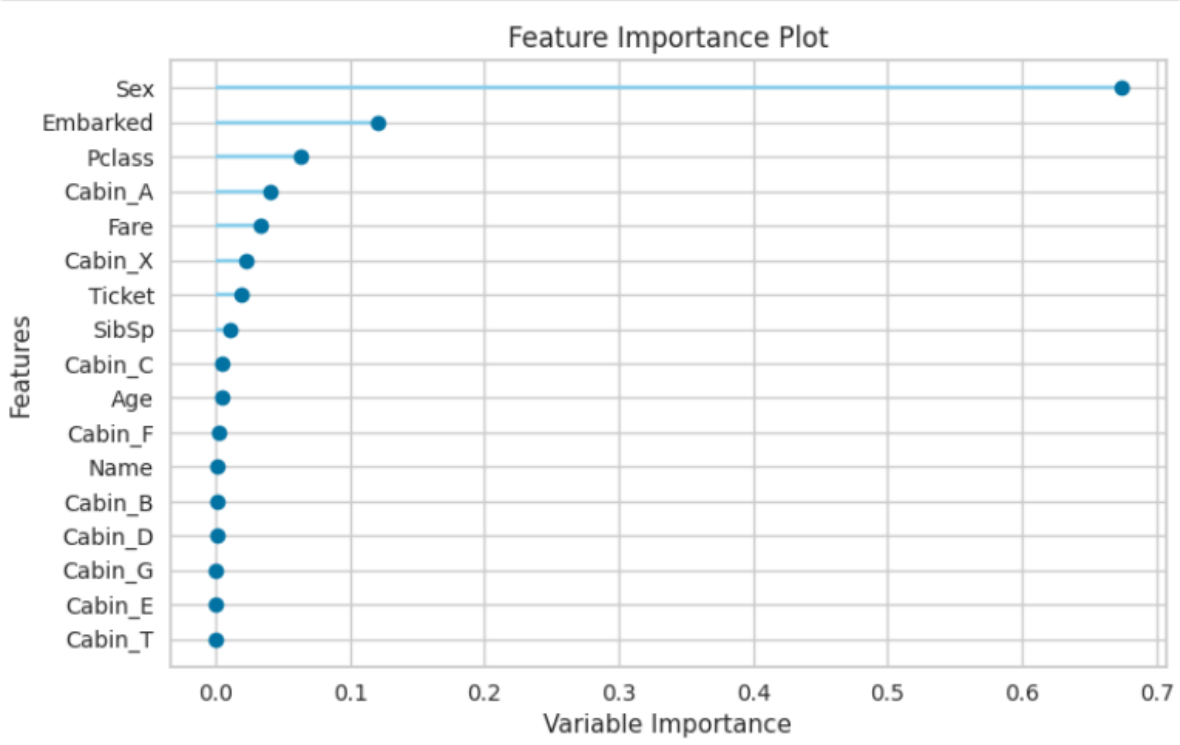
```
plot_model(cb)
```



# PYCARET – Feature Importance

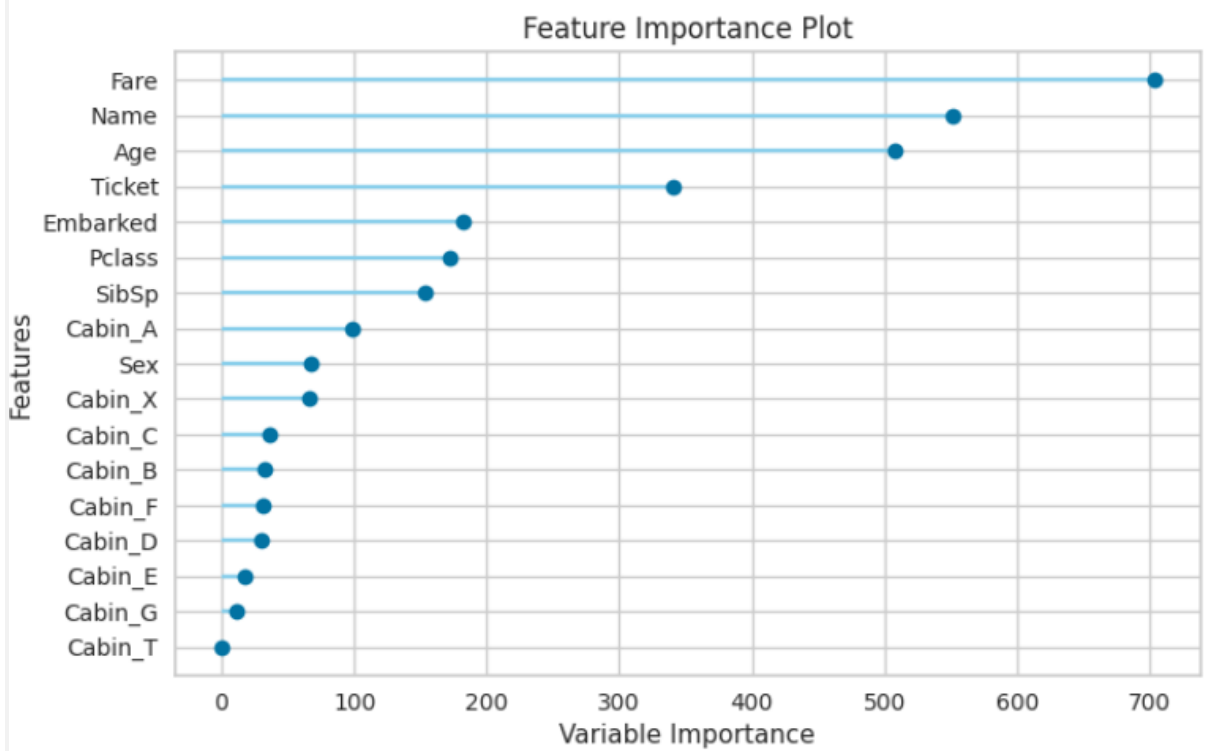
## Gradient Boosting

```
plot_model(gbc, plot = 'feature_all')
```




## LightGBM

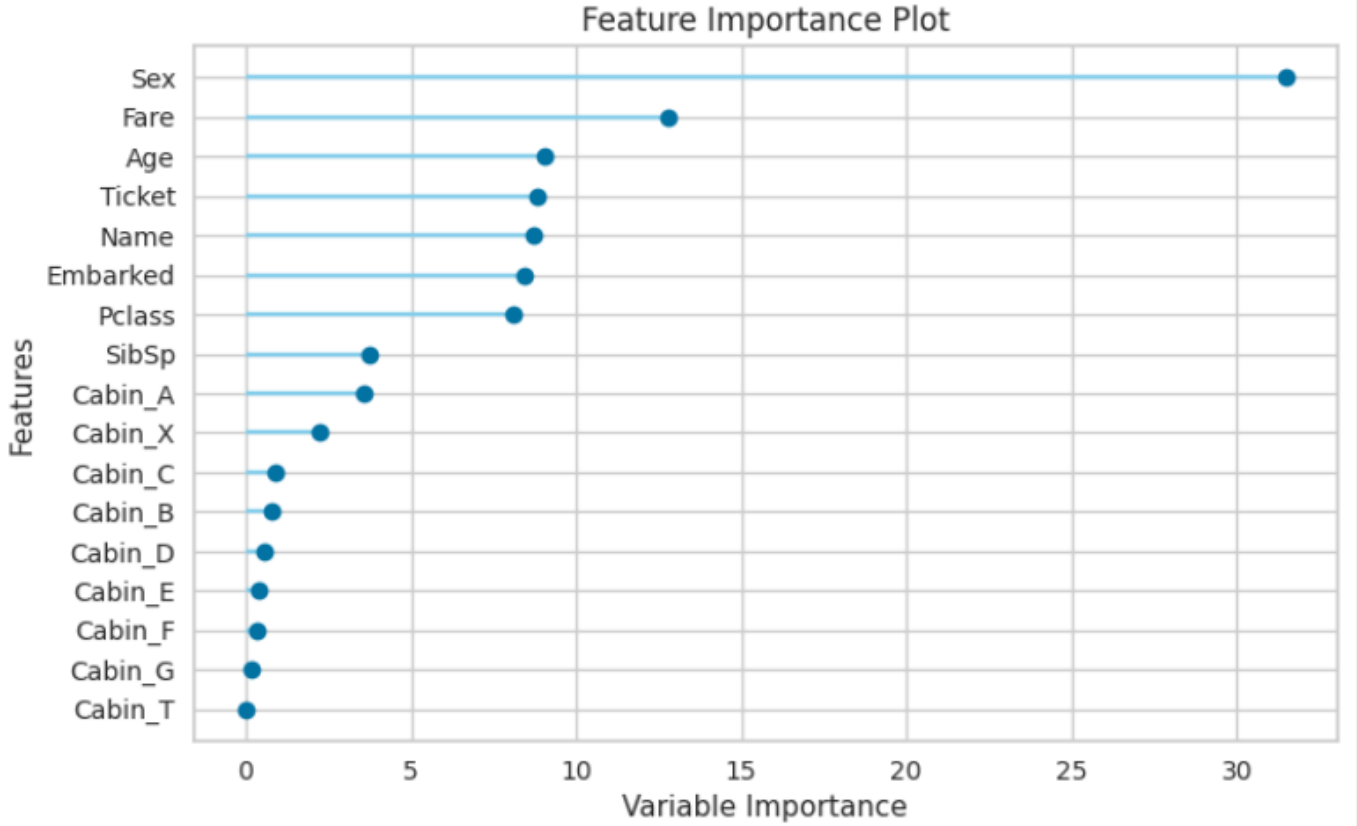
```
plot_model(lgbm, plot = 'feature_all')
```



## PYCARET – Feature Importance

Catboost

```
plot_model(cb, plot = 'feature_all')
```



## PYCARET – Best model

Summary

AUC가 가장 높은 모델: Tuned_catboost
Accuracy가 가장 높은 모델: LightGBM
F1 Score가 가장 높은 모델: Tuned_gbc

| | AUC | Accuracy | F1 Score |
|---|---|---|---|
| lightgbm | 0.8505 | 0.7824 | 0.7451 |
| Tuned_lightgbm | 0.8497 | 0.7812 | 0.7420 |

| | AUC | Accuracy | F1 Score |
|---|---|---|---|
| gbc | 0.8495 | 0.7803 | 0.7447 |
| Tuned_gbc | 0.8507 | 0.7812 | 0.7456 |

| | AUC | Accuracy | F1 Score |
|---|---|---|---|
| catboost | 0.8492 | 0.7796 | 0.7414 |
| Tuned_catboost | 0.8512 | 0.7809 | 0.7434 |

## PYCARET – 최종 모델들 결정

Summary

가장 높은 kaggle score가 나온 모델은 lgbm 과 catboost 이다.

| Models | Kaggle Score |
|---|---|
| Gbc + lgbm+ catboost | 0.80378 |
| Gbc + lgbm | 0.8037 |
| Gbc + catboost | 0.80305 |
| lgbm + catboost | 0.80406 |

## PYCARET – Kaggle Score

Summary 이를 통해 Stacking과 Blending이 모든 데이터에 적합한 것이 아님을 알 수 있었다.

| | | Stacking | Blending |
|---|---|---|---|
| Pycaret | Top4 | 0.80249 | 0.80241 |
| | Lgbm+cat | 0.80079 | 0.8035 |
| Sklearn | Top4 | 0.80314 | 0.8031 |
| | Lgbm+cat | 0.80293 | 0.8031 |

- stacking과 blending이 정확도를 올리는 모델링 방법이라고 적용해 보았지만, LightGBM+Catboost 단순 voting 방식이 더 적합함을 알 수 있었다.

## Kaggle Final Score


| | | | | |
|---|---|---|---|---|
| 348 | KWON JUNG MIN | | 0.80418 | 20 |

상위 약 27.9%

# 참고자료

## References


- Lee, H. J. (2021, April 24). *0423Project*. Kaggle. https://www.kaggle.com/tlgks32/0423project
- Jun, E. J. (2021, April 25). *TPS April EDA & Feature Engineering* ✨. Kaggle. https://www.kaggle.com/eunijnjun/tps-april-eda-feature-engineering
- Seo, J. H. (2021, April 25). *titanic_visualization_baysianoptimization*. Kaggle. https://www.kaggle.com/elon4773/titanic-visualization-baysianoptimization
- Kwon, J. M. (2021, April 25). *notebook5df7ae9696*. Kaggle. https://www.kaggle.com/kwonjungmin/notebook5df7ae9696

  Bayesian-Optimization Git. https://kjm94.github.io/2021/04/29/Bayesian-Optimization/
-