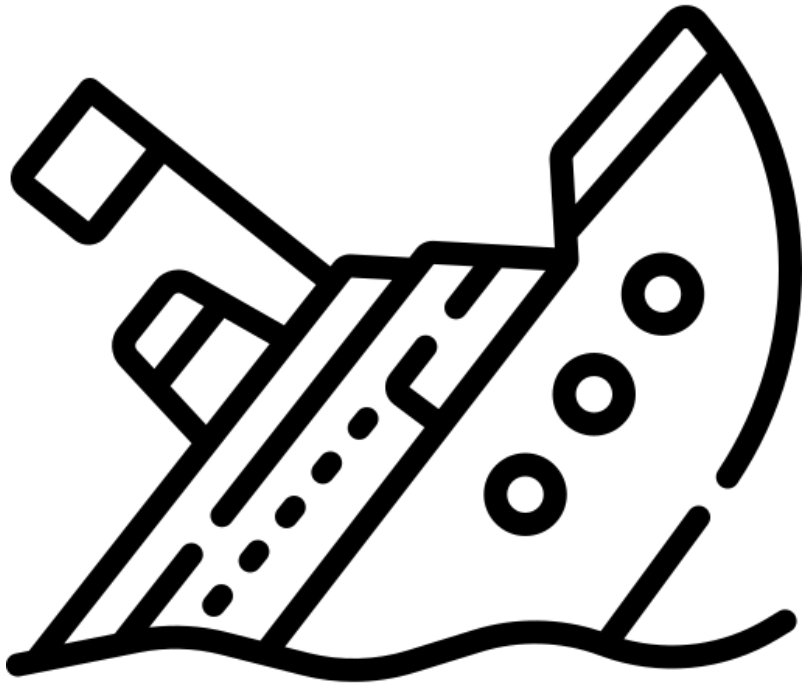


타이타닉 생존율 비교

- 팀 원 : 박윤수 김경태 김혜민 방이현

CONTENTS

01 개요
- 출처 및 목적
- 분석 배경

02 데이터 수집
- 조회
- 데이터 전처리

03 데이터 분석
- 시각화

04 결론

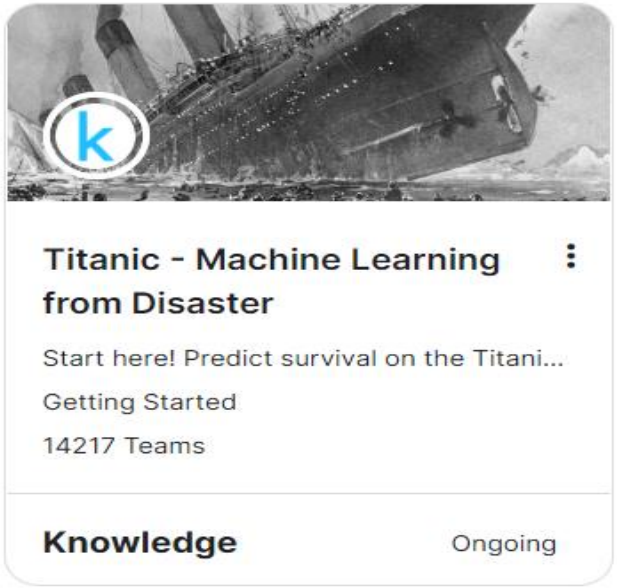
01

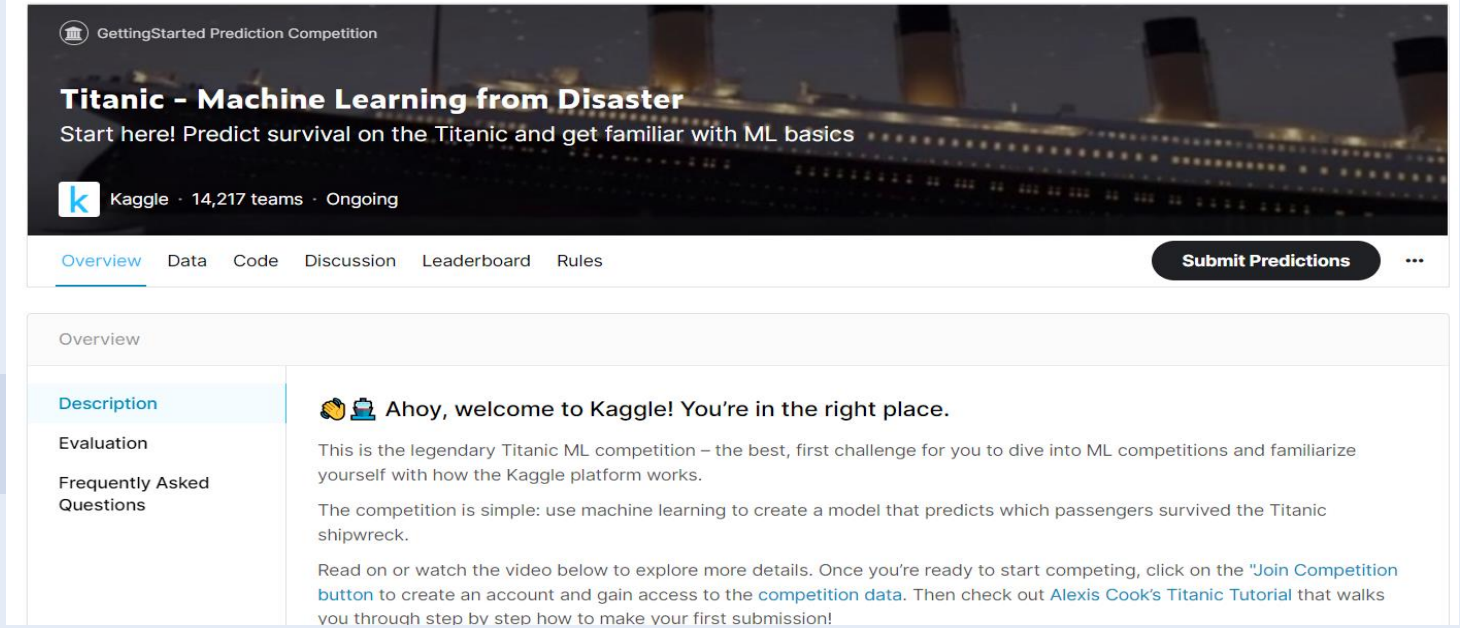
출처 및 목적

kaggle

https://www.kaggle.com/c/titanic

Kaggle은 2010년 설립된 예측모델 및 분석 대회 플랫폼이다.
기업 및 단체에서 데이터와 해결과제를 등록하면, 데이터 과학자들이 이를 해결하는 모델을 개발하고 경쟁한다.

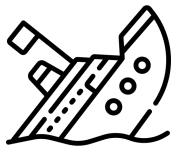




Kaggle에서 주최하는 경진 대회 중 대표적인 것은 **Titanic: Machine Learning from Disaster**이다.

대회의 목표는 **"어떤 사람들이 생존 할 가능성이 더 높은가"**라는 질문에 답하는 예측 모델을 구축하는 것이다.

제공되는 Data Set으로는 학습 데이터(train.csv), 테스트 데이터(test.csv), 제출 양식(gender_submission.csv)가 있다.



01

분석 배경

1912년 4월 10일 오전 9시 30분“타이타닉호“ 승객들은
탑승을 했고, 오전 11시 50분 타이타닉호는 출항을 하였다.
4월 14일, 23시 29분에 항로에 이미 표시 되어있던 빙산을 발견하였다. 충돌하지 않도록 방향을 바꾸도록 명령을 했지만
23시 40분 “타이타닉호 " 는 빙산을 쳤다.
충돌 결과로 여섯 개의 구멍 중 5개가 빙산과의 충돌 과정에서 손상되었다.

“타이타닉호” 의 사고로 적어도
1,496명이 목숨을 잃고 712명정도가 살아 돌아왔다.

1912년 시대적 배경에는
“Lady First”라는 개념이 존재 했다.
어린이와 여자부터
우선적으로 구조 했다고 한다.

“Lady First”
라는 개념이 없었다면

급박한 상황에서 구명정으로 달려들 때
힘센 남성이 구명정을 차지할
가능성이 높아 생존에 유리했을 것이다.

02

조회

1. 필요한 라이브러리 불러오기

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 파일 읽기
train= pd.read_csv('C:/Python_study/Python/Project1/titanic/train.csv')
```

```
# train 데이터 컬럼 보기
print(train.head())
```

```
   PassengerId  Survived  Pclass  ₩
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  ₩
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/02. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```
#정보보기
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

02

데이터 전처리

1. Null 개수 확인

```
#컬럼별 null 개수
train.isnull().sum()
```

```
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
```

2. Null 값 처리

```
#정규 표현식 사용하여 Name에서 Title 추출 전처리
for dataset in train:
    train['Title']=train['Name'].str.extract('([A-Za-z]+)₩.',expand=False)

print(train['Title'].value_counts().sort_index())

print(train.groupby(by='Title')['Age'].mean())


#결측치 Age처리 전처리
train.loc[(train["Age"].isnull())&(train["Title"]=='Mr'),'Age']=32 #Mr의 평균
train.loc[(train.Age.isnull())&(train.Title=='Mrs'),'Age']=36 #Mrs의 평균
train.loc[(train.Age.isnull())&(train.Title=='Miss'),'Age']=22 # Miss의 평균
train.loc[(train.Age.isnull())&(train.Title=='Master'),'Age']=5 # Master의 평균
train.loc[(train.Age.isnull())&(train.Title=='Dr'),'Age']=42 # Dr의 평균
```

```
Capt          1
Col           2
Countess      1
Don           1
Dr            7
Jonkheer      1
Lady          1
Major         2
Master       40
Miss        182
Mlle          2
Mme           1
Mr          517
Mrs         125
Ms            1
Rev           6
Sir           1
Name: Title, dtype: int64
```

■ 타이틀 종류와 개수 확인.

```
Title
Capt        70.000000
Col         58.000000
Countess    33.000000
Don         40.000000
Dr          42.000000
Jonkheer    38.000000
Lady        48.000000
Major       48.500000
Master       4.574167
Miss        21.773973
Mlle        24.000000
Mme         24.000000
Mr          32.368090
Mrs         35.898148
Ms          28.000000
Rev         43.166667
Sir         49.000000
Name: Age, dtype: float64
```

■ 타이틀 평균나이 확인.

```
# Age 결측값 처리 후 결과 확인
train.isnull().sum()
```

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

■ Null 값 처리 확인

02

데이터 전처리

3. 선승지 전처리 str-> int 바꾸기.

```
# 선승지 전처리
train["Embarked"].fillna('S', inplace=True)
dic2 = {"S": 0, "C": 1, "Q": 2}

train.loc[:, "Embarked"] = train.loc[:, "Embarked"].map(dic2)


# print(train.head())
print(train["Embarked"])
```

```
0       0
1       1
2       0
3       0
4       0
       ..
886     0
887     0
888     0
889     1
890     2
Name: Embarked, Length: 891, dtype: int64
```

4. 필요없는 컬럼 버리기.

```
# 테이블 drop

train.drop(labels="PassengerId",axis=1,inplace=True)
train.drop(labels="Name",axis=1,inplace=True)
train.drop(labels="Cabin",axis=1,inplace=True)
train.drop(labels="Ticket",axis=1,inplace=True)
```

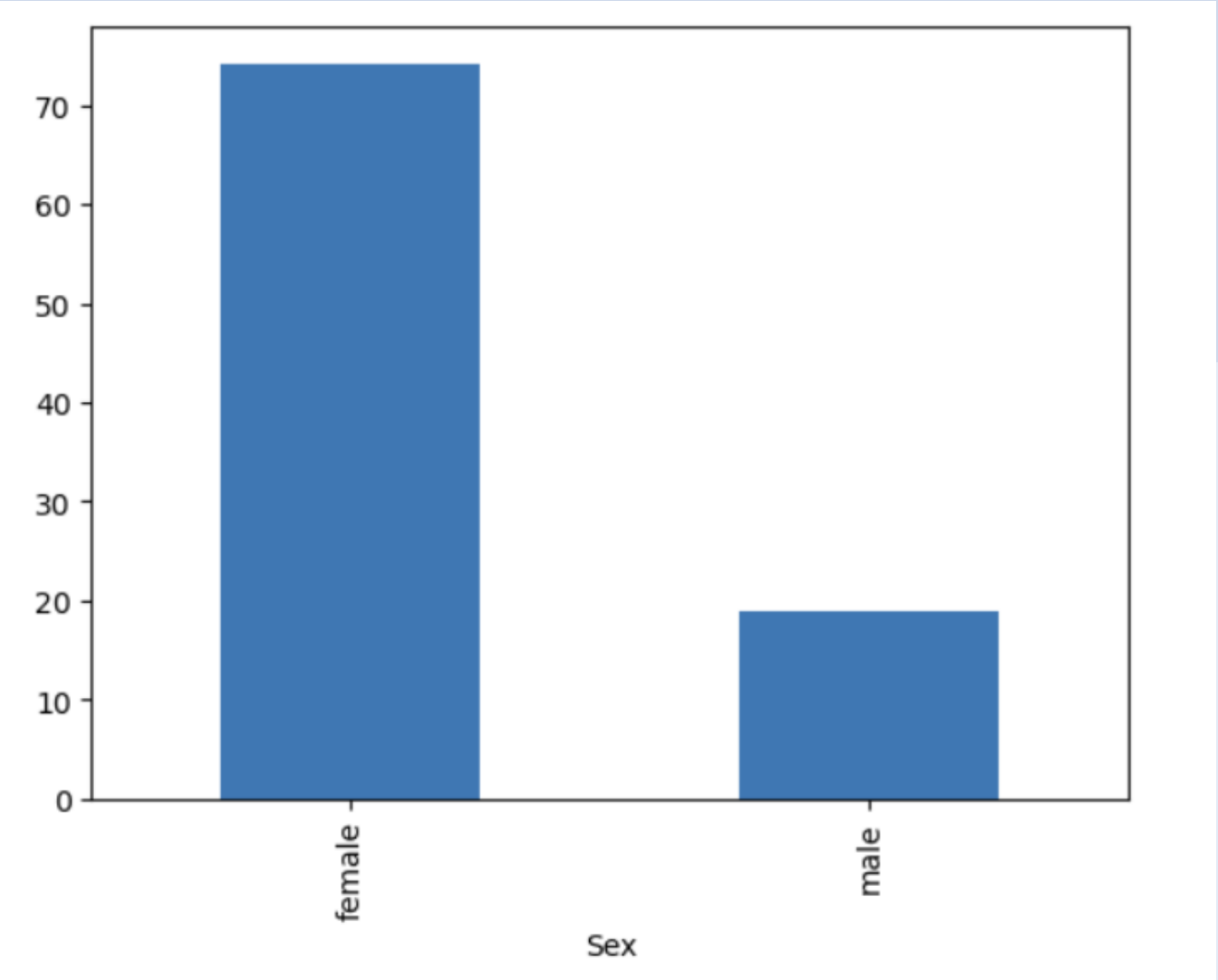
```
# 테이블 drop 후
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    int64
 8   Title     891 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 62.8+ KB
None
```

03

시각화

```
# 성별에 따른 생존율
df_Sex = train.groupby(by='Sex')['Survived'].mean()*100
print(df_Sex)
df_Sex.plot(kind='bar')
```



성별로 groupby 하여

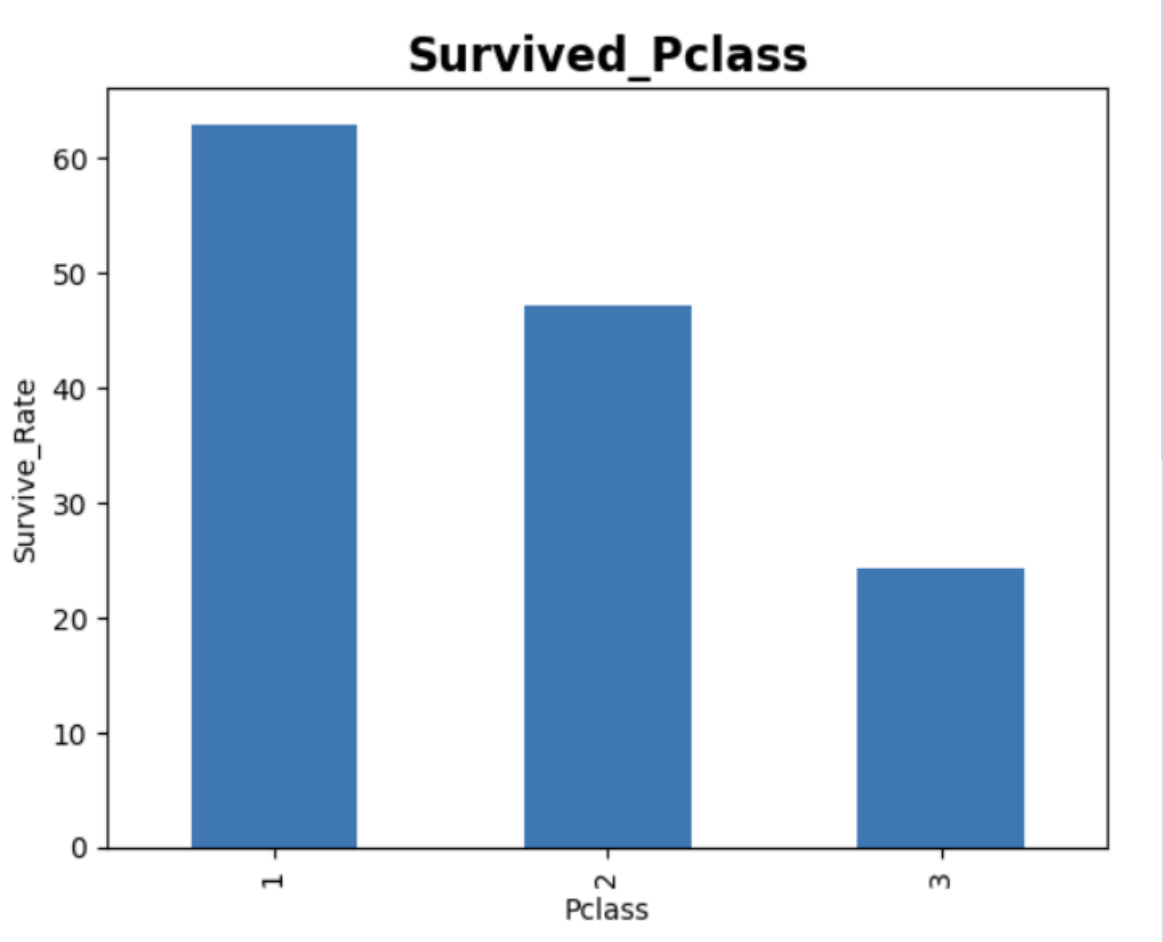
생존자들의 성별 비율을 나타냅니다.

이 그래프를 보아 생존자 중 남성보다
여성이 많은 것을 알 수 있습니다.

03

시각화

```
#객실등급별 생존율
(train.groupby(by='Pclass')['Survived'].mean()*100).plot(kind='bar')
print(train.groupby(by='Pclass')['Survived'].mean()*100)
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Survived_Pclass",fontdict=titiedict)
plt.ylabel("Survive_Rate")
```



객실등급으로 groupby 하고,

백분율을 막대그래프를 사용해

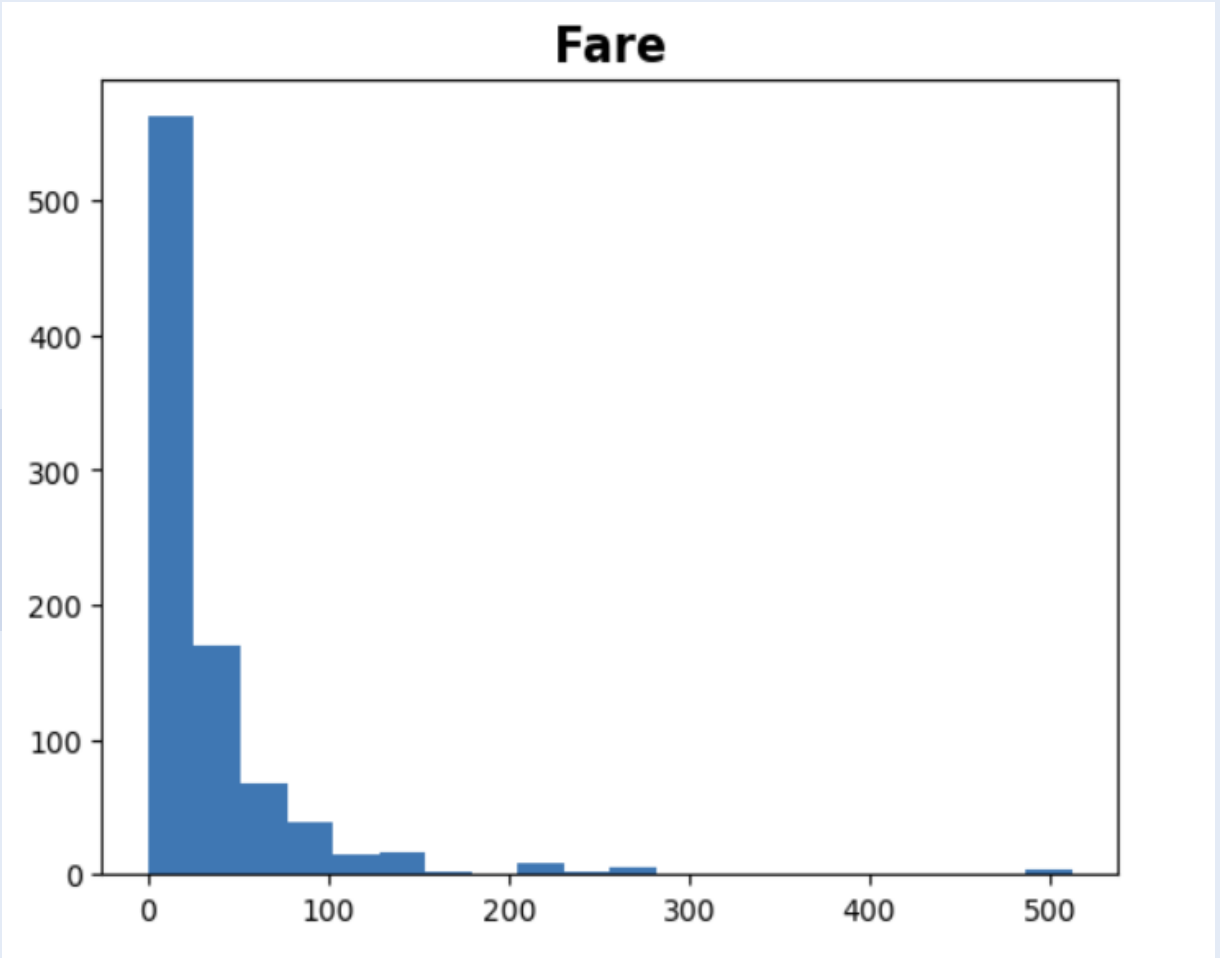
생존자들의 객실등급 비율을 나타냅니다.

이 그래프를 보아 좋은 객실일수록
생존자가 많은 것을 알 수 있습니다.

03

시각화

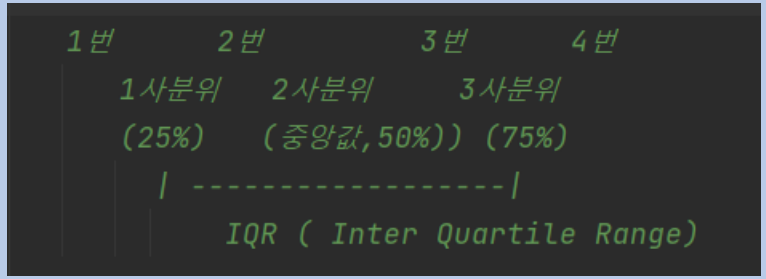
```
# 요금의 분포 확인
fig, ax = plt.subplots(nrows=1, ncols=1)

ax.hist(train['Fare'], bins=20)
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Fare",fontdict=titiedict)
plt.show()
```



이 그래프는 전체 승객의 승선 요금을 막대 그래프 로 표현했습니다.

03

시각화




```
print("1 사분위:", np.quantile(train['Fare'], 0.25))
print("2 사분위(중앙값):", np.quantile(train['Fare'], 0.5))
print("3 사분위:", np.quantile(train['Fare'], 0.75))
IQR = np.quantile(train['Fare'], 0.75) - np.quantile(train['Fare'], 0.25)
print("IQR", IQR )

# 요금 실제 최대 최소값
print(train['Fare'].max())
print(train['Fare'].min())
```

```
1 사분위: 7.9104
2 사분위(중앙값): 14.4542
3 사분위: 31.0
IQR 23.0896
512.3292
0.0
```

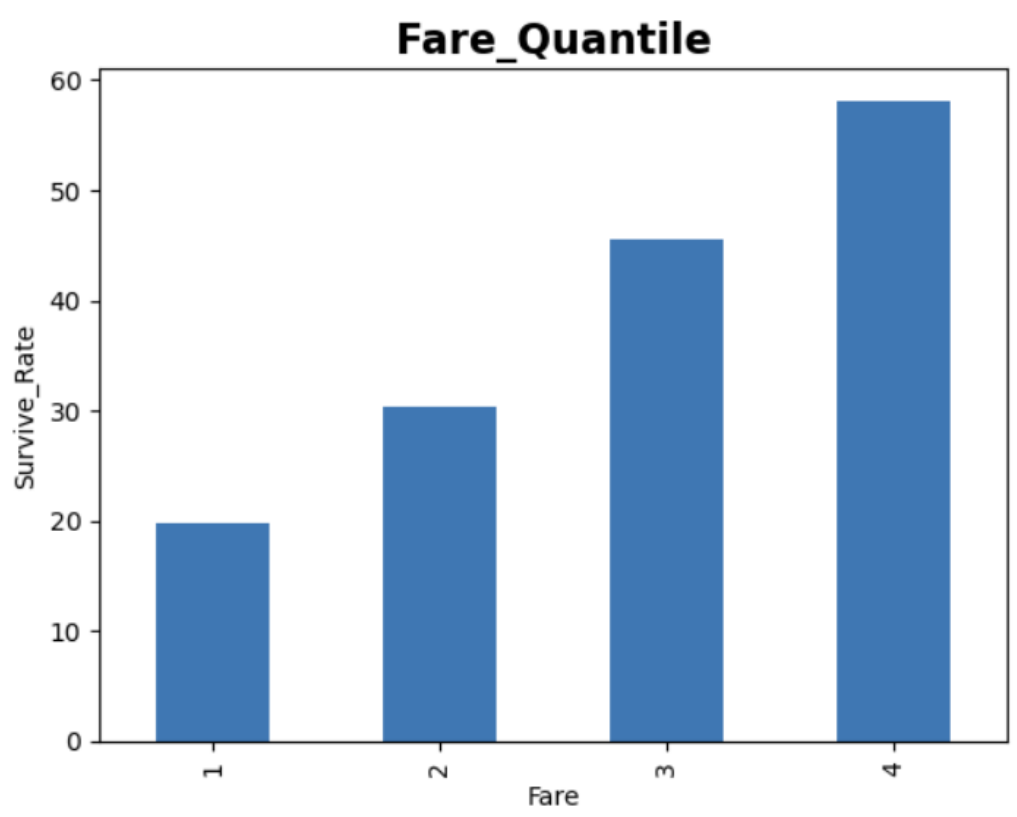
가독성을 위해 **Fare** 컬럼을 크기 순서로 나열한 자료를 4등분하는 관측 값인 **4사분위수로** 표현했습니다.

np.quantile 함수를 사용해
1 사분위는 전체의 25%에 해당되는 7.9104이고,
2 사분위는 전체의 50% 즉 중앙값은 14.4542,
3 사분위는 전체의 75%인 31.0이다.
IQR 은 3사분위의 값에서 1사분위의 값을 뺀 23.0896이 된다.

03

시각화

```
#요금 구간별 생존율 => 구간은 최소, 1/4, 2/4, 3/4, 최대값으로
train['Fare']=pd.cut(train.Fare,
                     bins=[train['Fare'].min()-1, # 설명 필요
                           np.quantile(train['Fare'], 0.25),np.quantile(train['Fare'], 0.5),np.quantile(train['Fare'], 0.75),
                           train['Fare'].max()],
                     labels=['1','2','3','4'])
(train.groupby(by='Fare')['Survived'].mean()*100).plot(kind='bar')
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Fare_Quantile",fontdict=titiedict)
plt.ylabel("Survive_Rate")
print(train.groupby(by='Fare')['Survived'].mean()*100)
```



pd.cut 은 구간을 나눠주는 함수인데

구간의 start 값은 미포함, end 값은 포함하기에

Fare = 1에는 train[“Fare”],0.25인 1사분위 값이,

Fare = 2에는 train[“Fare”],0.50인 2사분위 값이

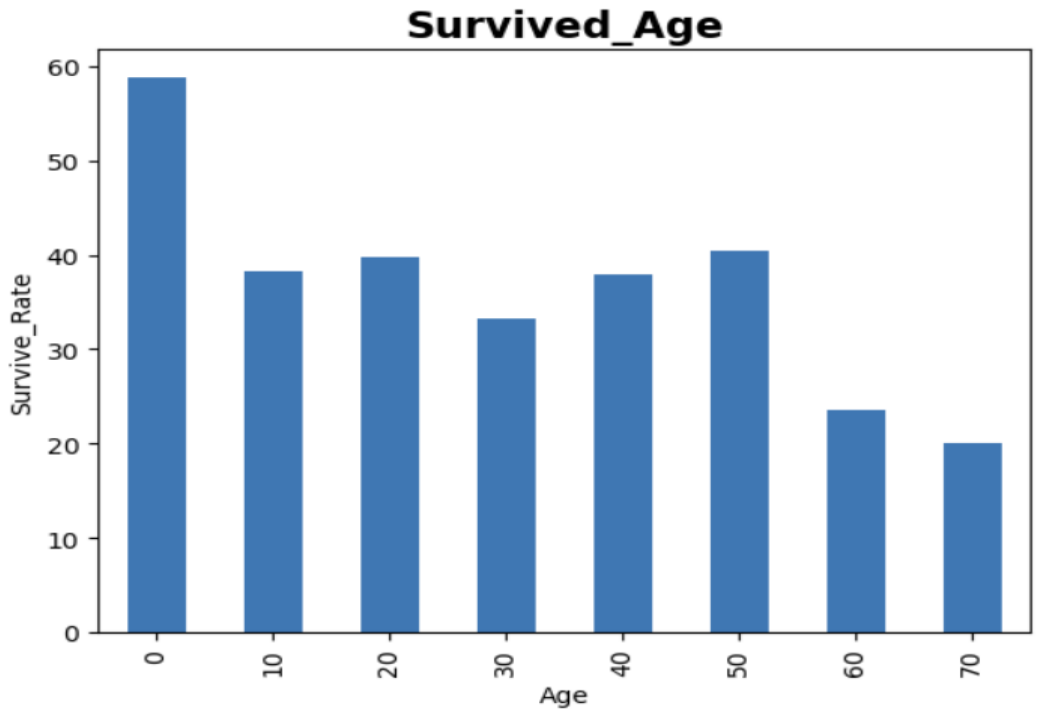
Fare = 3에는 train[“Fare”],0.75인 3사분위 값이

Fare = 4에는 train[“Fare”],Max값이 출력 되었다.

03

시각화

```
#연령대별 생존률 막대 그래프
train['Age']=pd.cut(train.Age,
                    bins=[0,10,20,30,40,50,60,70,80],
                    labels=['0','10','20','30','40','50','60','70'])
(train.groupby(by='Age')['Survived'].mean()*100).plot(kind='bar')
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Survived_Age",fontdict=titiedict)
plt.ylabel("Survive_Rate")
print(train.groupby(by='Age')['Survived'].mean()*100)
```

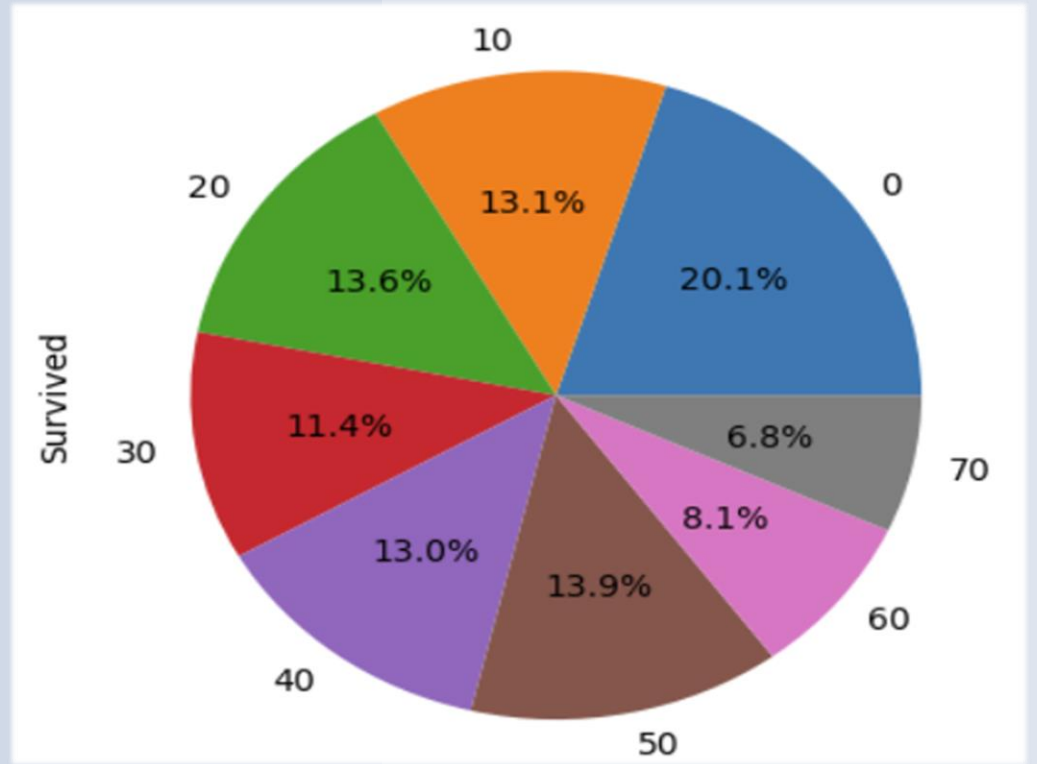


Age를 **groupby** 하여 생존률을 보여주는 그래프이다

왼쪽은 **kind=bar** 로 막대그래프로,

오른쪽은 **kind=pie** 로 파이차트로 표현했으며,

비율을 나타내기 위해 **autoptc="%.1f%%"** 로 설정했다.




```
# 연령대별 생존률 파이차트
(train.groupby(by='Age')['Survived'].mean()*100).plot(kind='pie',autopct="%.1f%%")
```

AGE

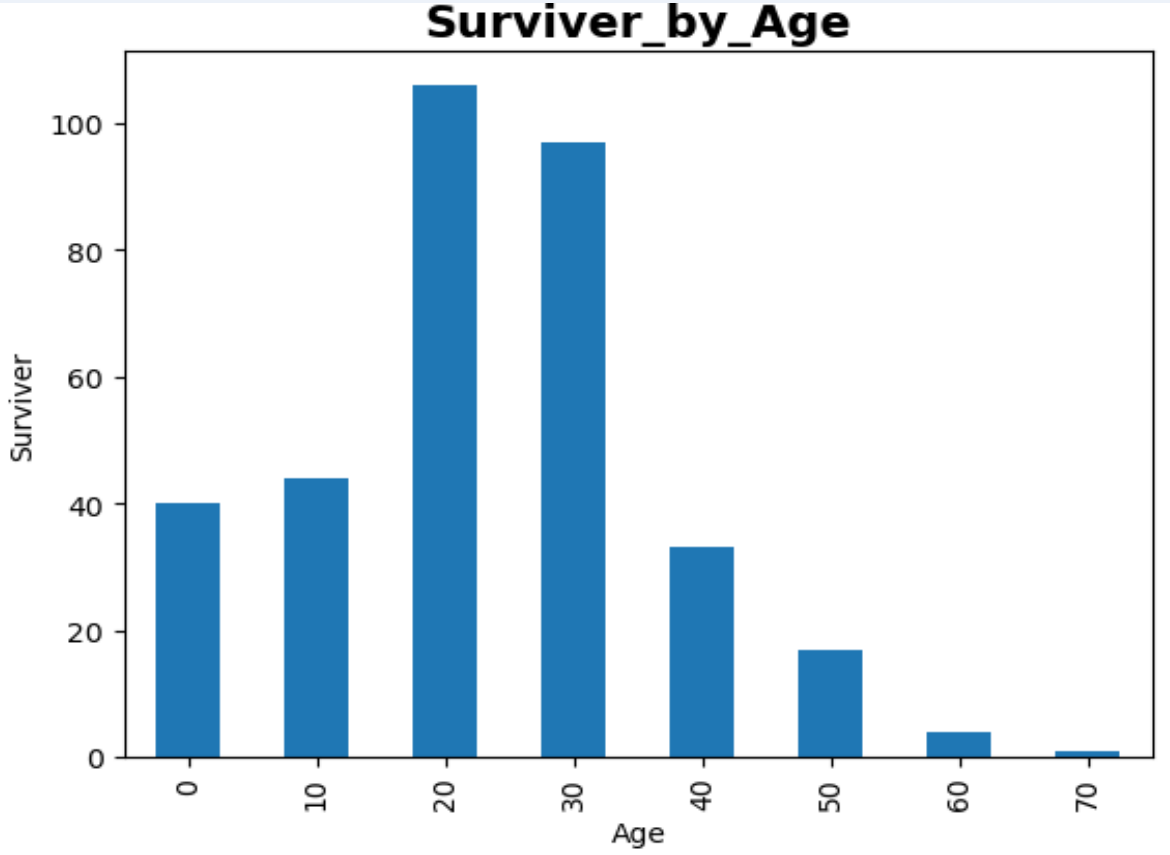
03

시각화

```
# 생존자 중 객실별 나이 분포
d_df=train[train['Survived']==0]
s_df=train[train['Survived']==1]
```

```
# 나이 별 생존자 수
s_df['Age'].value_counts().sort_index().plot(kind='bar')
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Surviver_by_Age",fontdict=titiedict)
```

```
plt.xlabel("Age")
plt.ylabel("Surviver")
```



d_df 에는 Survived가 0 인 사망자들을,
S_df 에는 Survived가 1 인 생존자들의 데이터를 나눠
생존자들의 나이를 표현했습니다.

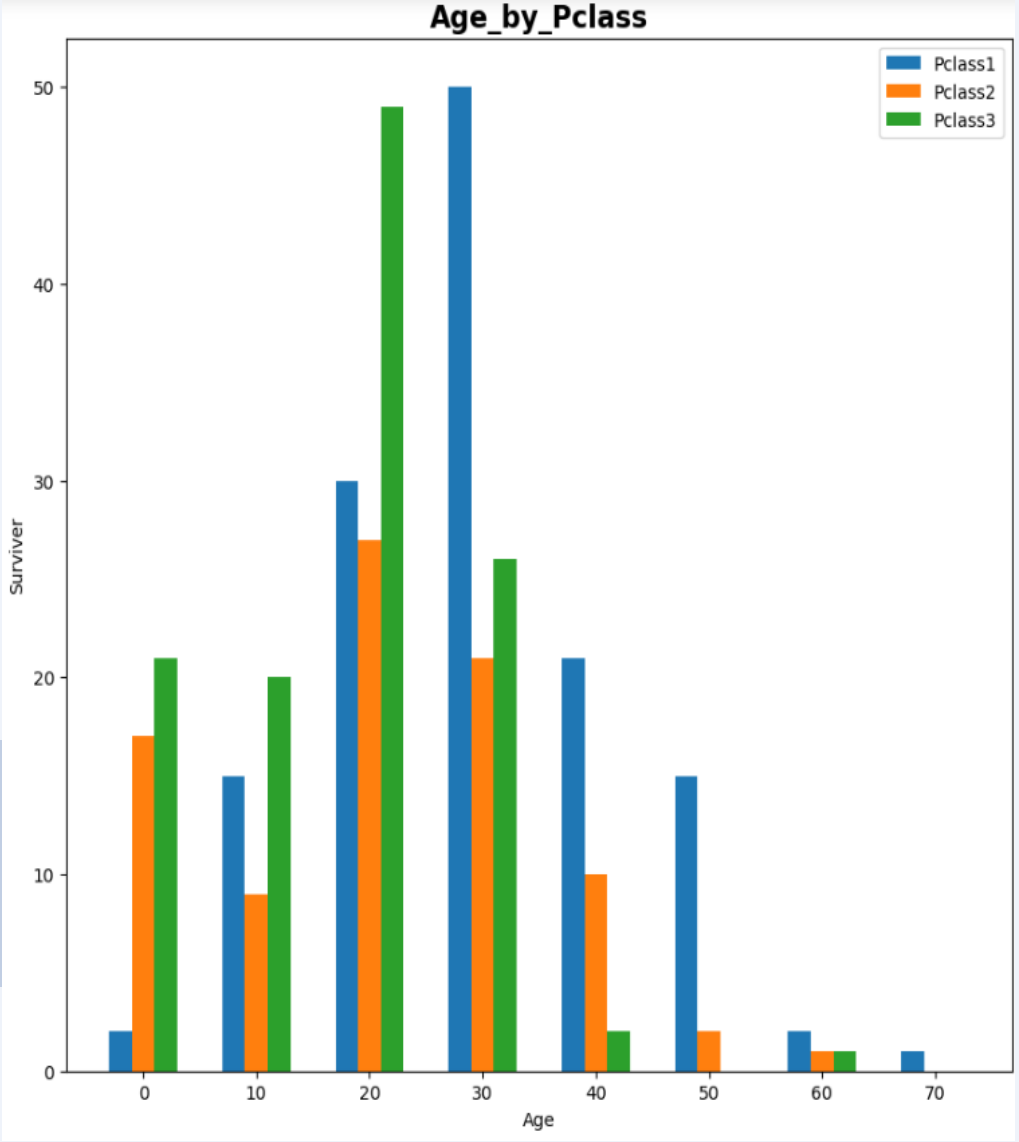
AGE

03

시각화

```
#나이별 클래스에 따른 생존자 분포
fig = plt.figure(figsize=(10,10))
index=np.arange(8)
w=0.2
plt.bar(index-w,s_df[s_df['Pclass']==1]['Age'].value_counts().sort_index(),width=w)
plt.bar(index,s_df[s_df['Pclass']==2]['Age'].value_counts().sort_index(),width=w)
plt.bar(index+w,s_df[s_df['Pclass']==3]['Age'].value_counts().sort_index(),width=w)
```

```
plt.xticks(index,[0,10,20,30,40,50,60,70])

plt.legend(['Pclass1','Pclass2','Pclass3'])
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Age_by_Pclass",fontdict=titiedict)
plt.xlabel("Age")
plt.ylabel("Surviver")
plt.show()
```

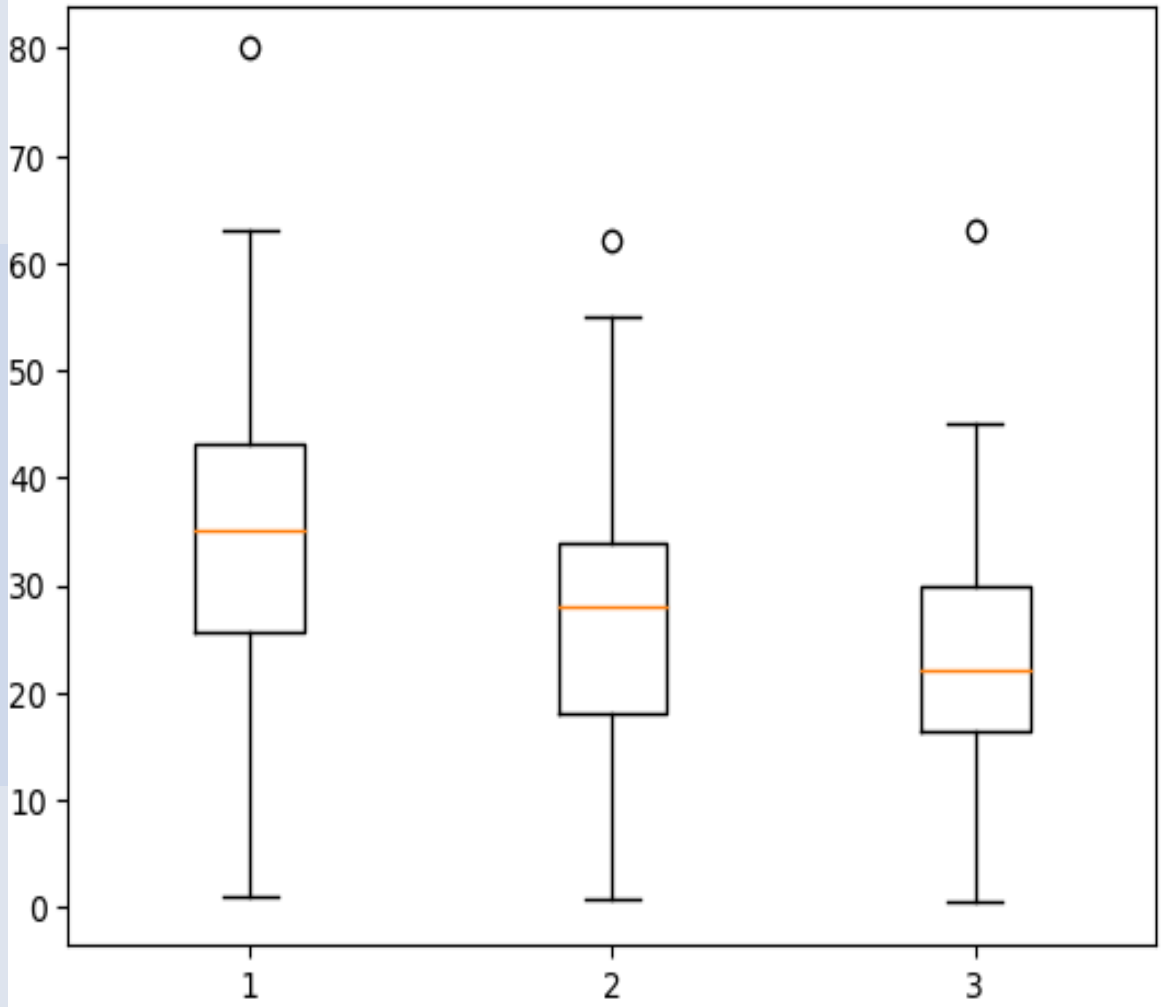


각각 Pclass=1, Pclass=2, Pclass=3일 때 **value_counts** 를 사용해 객실 등급 별 나이를 보여주고 있다.

03

시각화

```
fig = plt.figure()

plt.boxplot([s_df[s_df['Pclass']==1]['Age'],
             s_df[s_df['Pclass']==2]['Age'],s_df[s_df['Pclass']==3]['Age']])

plt.show()
```



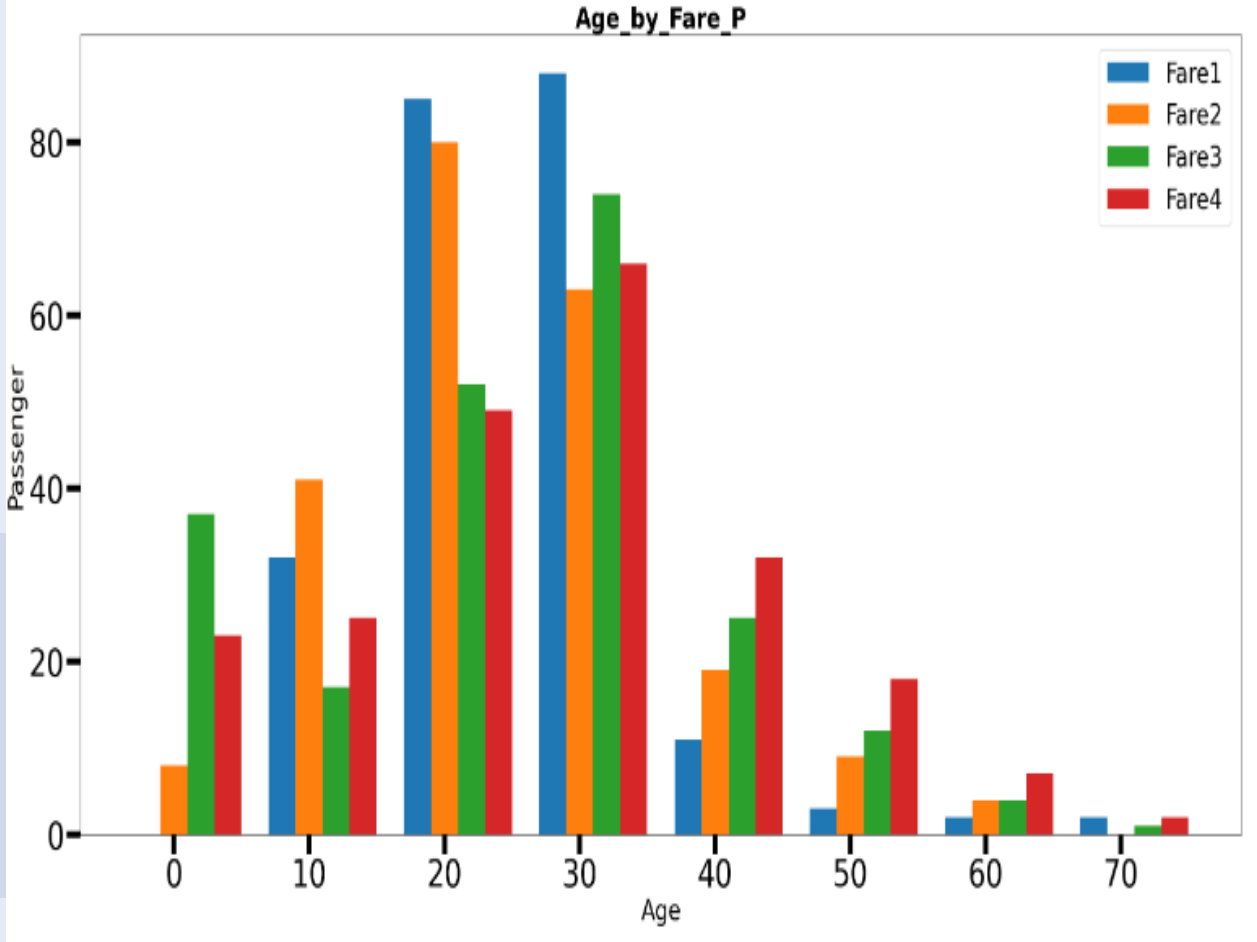
생존자들의 데이터가 담겨 있는 s_df 에서 각각 Pclass = 1인 1등석 승객들의 나이 분포, Pclass = 2인 2등석 승객들의 나이 분포, Pclass = 3인 3등석 승객들의 나이 분포를 나타냅니다.

03

시각화

```
#나이별 요금 분포도
fig = plt.figure(figsize=(30,15))
index=np.arange(8)
w=0.2
plt.bar(index-w,train[train['Fare']=='1']['Age'].value_counts().sort_index(),width=w)
plt.bar(index,train[train['Fare']=='2']['Age'].value_counts().sort_index(),width=w)
plt.bar(index+w,train[train['Fare']=='3']['Age'].value_counts().sort_index(),width=w)
plt.bar(index+2*w,train[train['Fare']=='4']['Age'].value_counts().sort_index(),width=w)
```

```
plt.xticks(index,[0,10,20,30,40,50,60,70],size=40)
plt.yticks(size=40)
plt.tick_params(axis='both',length=20,width=7)

titiedict={
    "fontsize":30,
    "fontweight":"bold"
}
plt.title("Age_by_Fare_P",fontdict=titiedict)
plt.xlabel("Age",fontsize=30)
plt.ylabel("Passenger",fontsize=30)
plt.legend(['Fare1','Fare2','Fare3','Fare4'],fontsize=30)
plt.show()
```



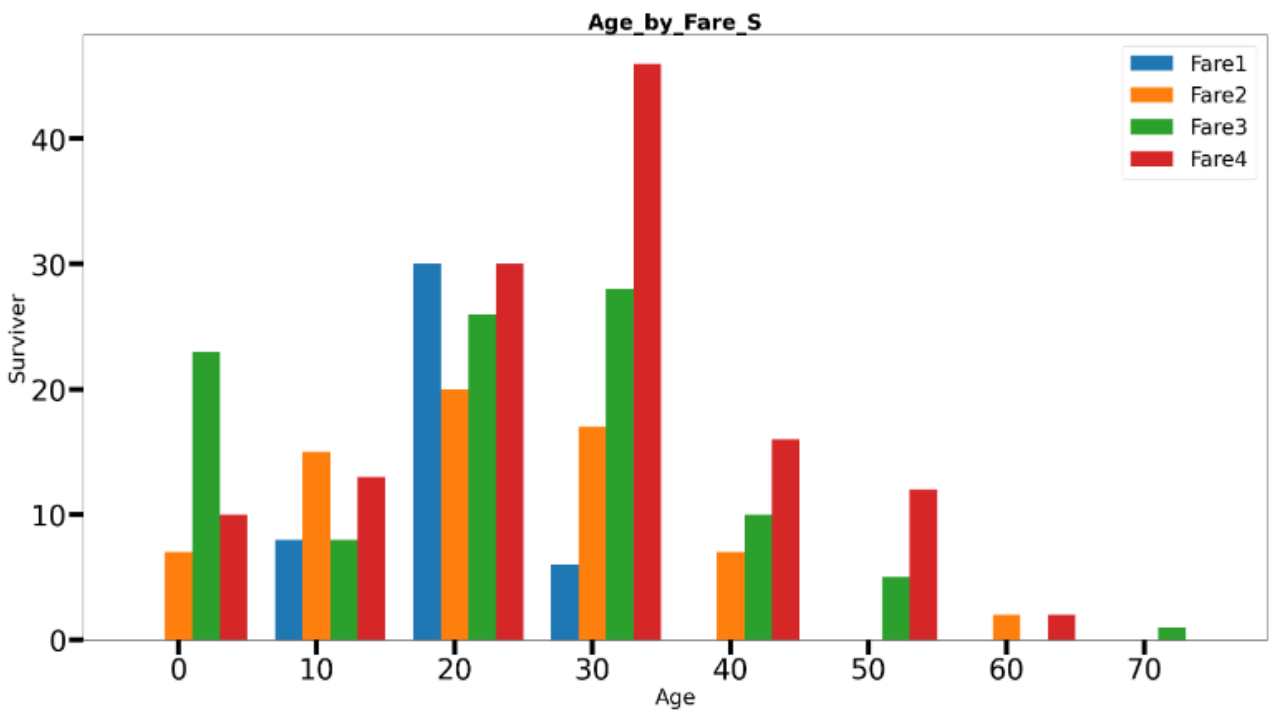
조절한 **Figsize** 에 맞게끔, **ticks** 의 size 를 조절했다.

이전에 4사분위로 나눈 **Fare** 를 이용해
전체 승객의 Fare =1 , Fare =2, Fare =3, Fare =4 의
연령대를 보여준다.

03

시각화

```
#생존자중 나이별 요금 분포도
fig = plt.figure(figsize=(30,15))
index=np.arange(8)
w=0.2
plt.bar(index-w,s_df[s_df['Fare']=='1']['Age'].value_counts().sort_index(),width=w)
plt.bar(index,s_df[s_df['Fare']=='2']['Age'].value_counts().sort_index(),width=w)
plt.bar(index+w,s_df[s_df['Fare']=='3']['Age'].value_counts().sort_index(),width=w)
plt.bar(index+2*w,s_df[s_df['Fare']=='4']['Age'].value_counts().sort_index(),width=w)

plt.xticks(index,[0,10,20,30,40,50,60,70],size=40)
plt.yticks(size=40)
plt.tick_params(axis='both',length=20,width=7)

titiedict={
    "fontsize":30,
    "fontweight":"bold"
}
plt.title("Age_by_Fare_S",fontdict=titiedict)
plt.xlabel("Age",fontsize=30)
plt.ylabel("Surviver",fontsize=30)
plt.legend(['Fare1','Fare2','Fare3','Fare4'],fontsize=30)
plt.show()
```



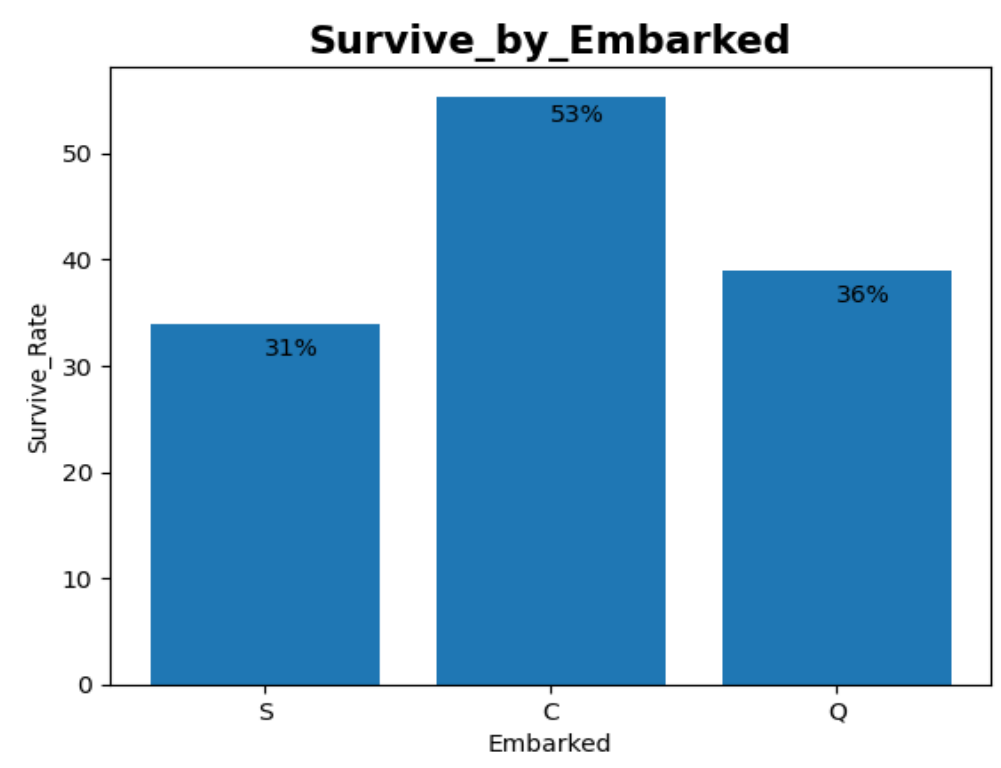
이전에 4사분위로 나눈 Fare 를 이용해 생존자

Fare =1 , Fare =2, Fare =3, Fare =4 연령대를 보여준다.

03

시각화

```
# 선승지별 생존율
x=['S','C','Q']
plt.bar(x,train.groupby(by='Embarked')['Survived'].mean()*100)
plt.text('C',53,'53%')
plt.text('Q',36,'36%')
plt.text('S',31,'31%')
titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Survive_by_Embarked",fontdict=titiedict)
plt.xlabel("Embarked")
plt.ylabel("Survive_Rate")

plt.show()
```



선승지를 나타내는 컬럼인

Embarked 를 **groupby** 하여

각각 S,Q,C의 선승지 별 생존율을 나타낸다.

Plt.text 를 통해 수치를 그래프안에 표현했다.

03

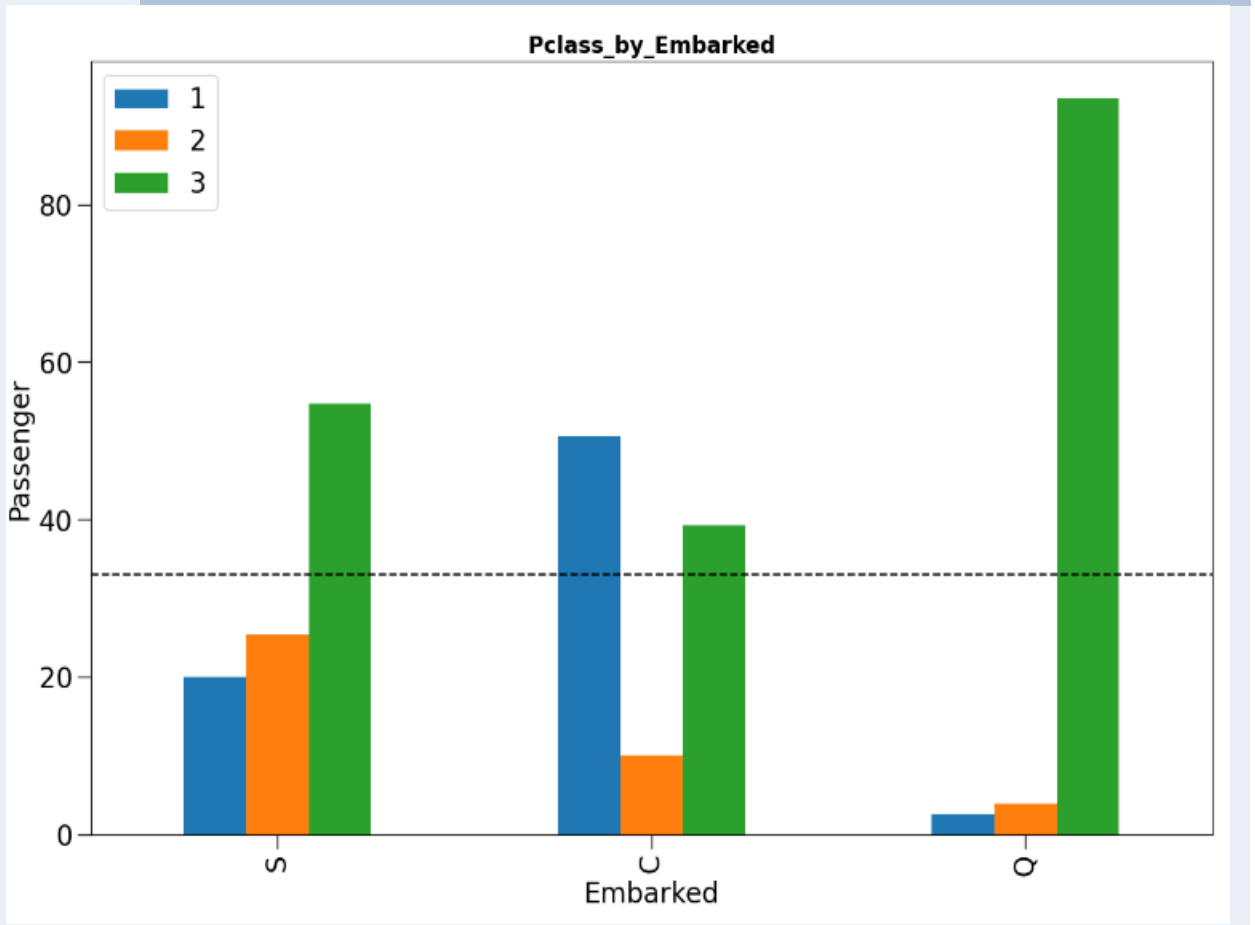
시각화

이전에 S는 0, C는 1, Q는 2 전처리 한 데이터 프레임을 이용하여
선승지 별 객실 비율에 관한 데이터 프레임을 생성했다.

생성된 데이터 프레임의 열을 합한 후 각 행에 **div** 하여
백분율로 나타냈다.

```
#선승지별 객실 비율 DF생성
S=train[train['Embarked']==0]['Pclass'].value_counts().sort_index()
C=train[train['Embarked']==1]['Pclass'].value_counts().sort_index()
Q=train[train['Embarked']==2]['Pclass'].value_counts().sort_index()
```

```
df=pd.DataFrame([S,C,Q],index=['S','C','Q'])
print(df)
print(df.div(df.sum(axis=1),axis=0)*100)

# 선승지별 객실클래스 비율
(df.div(df.sum(axis=1),axis=0)*100).plot(kind='bar',
                                         figsize=(15,10))


plt.xticks(size=20)
plt.yticks(size=20)
plt.tick_params(axis='both',length=10,width=1)
plt.legend(fontsize=20)
plt.axhline(33, color='black', linestyle='--')

titiedict={
    "fontsize":16,
    "fontweight":"bold"
}
plt.title("Pclass_by_Embarked",fontdict=titiedict)
plt.xlabel("Embarked",fontsize=20)
plt.ylabel("Passenger",fontsize=20)
plt.show()
```



04

결론

데이터 분석을 통해 얻은 결론

1. 시대적 배경에 따라 남성보다는 여성이, 그리고 어린아이들의 생존률이 높았다.

2. 좋은 객실, 높은 요금을 선택할수록 생존률이 높았다.

3. 신체능력이 비교적 낮은 연령대일수록 여러 요소의 영향을 더 많이 받았다.

향후 목표

저희가 분석한 것처럼 많은 사람들이 데이터를 분석하고,
이를 토대로 생존자 예측 모델을 제작하여 Kaggle 경진대회에 참여 하고 있습니다.
이후 배우게 되는 머신러닝과 딥러닝을 통해 모델을 직접 설계해보고 Kaggle에
제출하여 좋은 점수를 얻는 것이 팀의 최종 목표입니다.

05

끝