EDA 및 시각화

EDA

- **Exploratory Data Analysis (EDA)**
  - 탐색적 자료 분석(Exploratory data analysis)은 미국의 저명한 통계학자 존 투키 (John W. Tukey)가 창안한 자료 분석 방법론
  - 전통적인 통계학이 정보의 추출에서 가설 검정 등에 치우쳐져 있어, 자료가 가지고 있는 본연의 의미를 찾는데 어려움이 있어 이를 보완하고자, 주어진 자료만 가지고도 충분한 정보를 찾을 수 있도록 EDA를 개발

- **vs. 전통적인 분석 방법**
  - 전통적인 분석(Traditional analysis)의 목적은 자료의 성격에 대한 다양한 **추측을 확인**하는 것
  - 탐색적 데이터 분석(EDA)의 목적은 데이터에 대해 발견할 수 있는 정보를 찾기 위해 **데이터를 조사**하는 것

EDA 대표도구

- **데이터 변형** Data transformation
  - 또는 재표현 re-expression
  - 자료분석의 단순화를 위해 원래의 변수를 특정한 척도로 바꾸어 다시 표현하는 것

- **데이터 시각화** Data Visualization
  - 데이터 시각화는 데이터 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달하는 과정을 말함
  - 데이터 시각화의 목적은 도표(graph)라는 수단을 통해 정보를 명확하고 효과적으로 전달하는 것 [Friedman, 2008]

EDA 절차

- **EDA의 자세**
  - EDA 방법은 자료를 어떤 모형에 바로 적용시키기보다는 자료를 있는 그대로 보려는 데에 중점을 둠

- **기초적인 EDA절차**

  기초적인 탐색적 자료 분석의 절차

1. 데이터의 모양을 확인

   head(), tail(), shape, info()

2. (데이터클리닝)

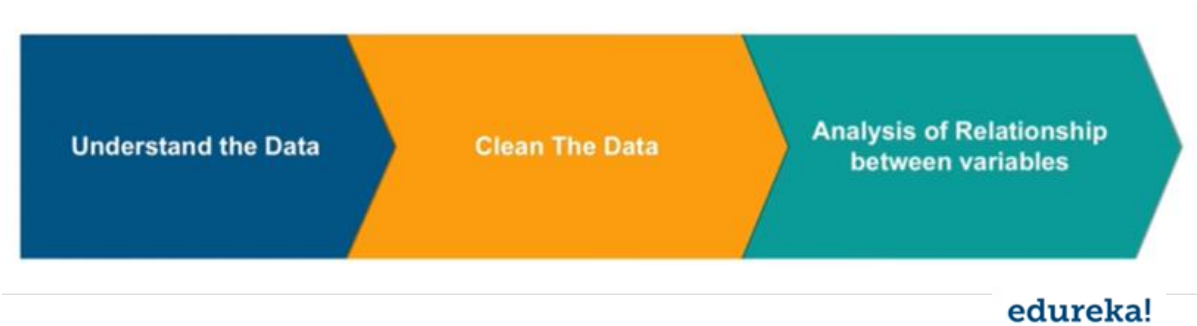
   예) 누락데이터 처리 등

EDA 절차

(이어서)

3. 데이터전처리

   예) 데이터 타입 변환



[이후 시각화 활용]

4. 종속변수의 분포 살펴보기

5. 독립변수 – 1) 범주형 변수의 분포 살펴보기

6. 독립변수 – 2) 연속형 변수의 분포 살펴보기

7. 범주형 변수와 연속형 변수간의 관계 파악하기

실습1.1 데이터세트 로드

- **titanic.csv 사례연구**
  - titanic.csv는 seaborn의 'titanic' 데이터세트와는 일부 다른 변수를 포함하고 있는 데이터세트
  - 교안 다운로드 사이트인 www.eshopping.co.kr에서 다운 가능

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

```
titanic = pd.read_csv('titanic.csv')
```

  - numpy, pandas, matplotlib, seaborn의 4가지 패키지는 파이썬으로 EDA하는데 거의 필수적으로 사용하는 라이브러리

실습1.2 데이터 모양 확인

- 데이터 모양 확인
  - head( ), tail( ) 등으로 데이터 세트 둘러보기
  - .shape로 행과 열의 개수 확인

```
#data overview
titanic.head()
```

	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

```
titanic.shape
```

```
(891, 12)
```

실습1.3 데이터 클리닝

- 예) 누락 데이터 확인
  - seaborn의 titanic에서 없던 Cabin변수의 경우 687행이 누락
  - Embarked는 2개 누락
  - 누락데이터 처리
    1. 삭제
    2. 대치 (Imputation)

```
#data cleaning : missing values
titanic.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
dtype: int64
```

실습1.4 누락 데이터 비율

- 누락 데이터 비율
  - ‘Cabin’과 ‘Embarked’의 누락 데이터 비율을 활용하여 누락데이터 처리 방법을 자동으로 결정하게 할 수도 있음
  - 누락 데이터가 있는 행만 출력

```
#of, ratio
missing_df = titanic.isnull().sum().reset_index()
missing_df.columns=['column','count']
missing_df['ratio']=missing_df['count']/titanic.shape[0]
missing_df.loc[missing_df['ratio']!=0]
```

	column	count	ratio
10	Cabin	687	0.771044
11	Embarked	2	0.002245

실습1.4 데이터타입 확인

- **데이터타입 확인**
  - 데이터 타입을 확인하는 이유는 해당 변수의 타입을 EDA하기 좋은 타입으로 변환하기 위해서

```
titanic.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          891 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
```

범주형 변수와 수치형 변수

- 범주형 변수의 경우
  - object 또는 string으로
  - 값이 문자열로 들어가 있으면 자동으로 object타입이 되지만, 만약 숫자로 된 범주형 변수의 경우 int64로 의도하지 않은 타입으로 입력되어있는 경우도 있음
- 수치형 변수의 경우
  - int64 또는 float64로

실습1.5 데이터타입 변환

- 데이터타입 변환
  - ‘Survived’와 ‘Pclass’ 변수는 범주형이지만 int64로 되어 있어 데이터타입 변환이 필요
  - astype(object) 메소드를 사용하여 범주형 데이터타입으로 변환

```
#conversion
titanic['Survived']=titanic['Survived'].astype(object)
titanic['Pclass']=titanic['Pclass'].astype(object)
titanic.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    object
 2   Pclass       891 non-null    object
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 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(3), object(7)
memory usage: 83.7+ KB
```

종속변수와 독립변수

- **독립변수** independent variable
  - 어떤 논문 주제를 정하고 연구가설을 설정하여 "A가 B에게 이러한 변화를 줄 것이다."라고 가정했을 때, 원인이 되는 A를 [독립변수]라고 하며, 독립변수는 원인변수라고도 함

- **종속변수** dependent variable
  - 위 설명에서 원인에 의해서 결과로 나타나는 B는 [종속변수]라고 하며, 종속변수는 결과변수라고도 함
  - 즉, 종속변수는 독립 변수의 변화에 따라 값이 결정되는 변수
  - 예를 들어, 함수 y=f(x)에 있어서 x가 변하는 데에 따라 바뀌는 y가 종속변수임

종속 및 독립변수 예제

- **예제1**
  - 내각 지지율이 오르내리는 원인을 연구할 때 지지율이 종속변수가 되고, 투표율이 오르내리는 원인을 연구할 때는 투표율이 종속변수가 됨

- **예제2**
  - 우리가 '카페인은 수면시간에 영향을 줄 것이다'라고 연구가설을 세웠다고 가정하면, 위 가설에서의 독립변수는 [카페인]이 되고 종속변수는 [수면시간]이 됨
  - 한마디로 독립변수는 '설명하는 변수(원인)'이고 종속변수는 '설명되는 변수(결과)'라고 보면 됨
  - 참고로 독립변수와 종속변수의 관계는 '인과관계'라고 함

프로야구 관중예측 사례

- **국내 프로야구 관중예측 알고리즘 연구**
  - 2018년~2019년 과제
  - 프로야구 구단의 경기장 관중 점유율 향상과 수익 증대와 같은 성과 향상을 위하여, 관중들의 경기장 방문에 영향을 미치는 요인을 분석하여 관중예측모형을 구축

- **종속변수와 독립변수?**
  - 관중수에 영향을 미칠 것으로 예상되는 요인으로는 날씨, 경기요일, 승패, 상대팀 등

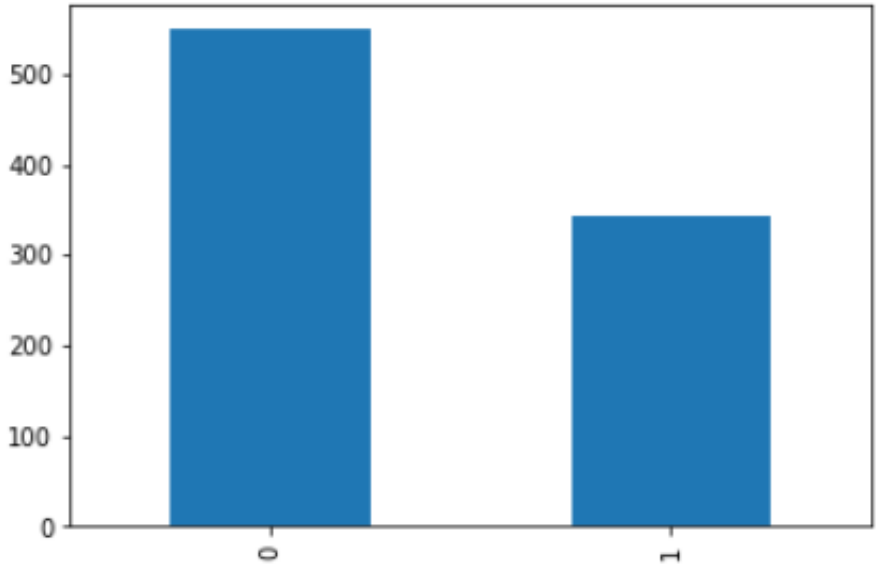
실습1.6 종속변수 확인

- **종속변수는 생존여부?**
  - titanic 데이터세트에서 '성별은 생존여부에 영향을 줄 것이다.'라는 연구를 한다면
  - 종속변수는 생존여부가 됨

```
#dependent variable
titanic['Survived'].value_counts().plot(kind='bar')
plt.show()
```

  - 시각화 도구로 살펴보기

    : 막대그래프(bar plot)



matplotlib

- 기본 그래프 도구
  - matplotlib는 파이썬 표준 시각화 도구라고 부를 수 있을 정도로 2D 평면 그래프에 관한 다양한 포맷과 기능을 지원
  - matplotlib은 파이썬에서 매트랩과 유사한 그래프 표시를 가능케 하는 라이브러리

- 라이브러리 **import**
  - import matplotlib.pyplot as plt 라는 약칭으로 임포트
  - Link : https://matplotlib.org/

matplotlib.pyplot

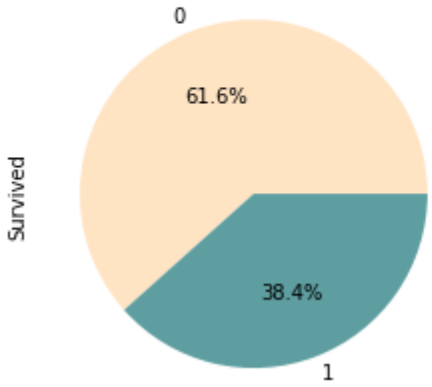
- **기초 활용**
  - 객체에 plot( )메소드를 적용하여 그래프를 출력

    : .plot(kind='bar')
  - .plot( )은 그래프의 형태를
  - .show()는 (코드 실행시) 그래프를 화면에 팝업하는 역할
  - kind옵션은 그래프의 종류 (default: 선그래프 line graph)

실습1.7 pie 차트

- **pie 차트 그리기**
  - 옵션 kind='pie'
  - 옵션 autopct='%1.1f%%' : 퍼센트 표시, 옵션 colors : 색상리스트

```
#dependent variable
titanic['Survived'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['bisque','cadetblue'])
plt.show()
```



독립변수 분포 살펴보기

[독립변수 중 변수의 특성에 따라 분포 살펴보기]

- **범주형 변수** categorical variable
  - ≈ 이산형 변수 discrete variable
  - cf. **비연속형 변수** uncontinuous variable

    : 비연속형 변수는 몇 개의 정수 수치로 표현되며,

    예를 들면 아파트 층수, 퀴즈를 맞춘 개수

- **연속형 변수** continuous variable
  - 연속형 변수는 정수형이나 실수형 수치로 표현됨

변수의 종류 구분법

- **Case1. 데이터 타입으로 구분 (주의. 한가지 방법일 뿐)**
  - 범주형 변수는 object타입으로
  - 비연속형 변수는 int64타입으로
  - 연속형 변수는 float64타입으로
  - 우리는 이 방법을 적용해보자.

- **Case2. 알고리즘으로 구분**
  - 비연속형의 경우 int64이면서, 10개 이하로 조건을 주면,
    예를 들어, 어떤 아파트 층수가 9개이면 비연속형으로 분석하고,
    30층까지 있으면 연속형으로 처리

실습1.8 범주형 변수 선별

- **범주형 변수 선별 작업**
  - 실습1.5에서 범주형 변수의 데이터타입을 object로 모두 변경했기 때문에 아래 코드와 같이 object타입을 가진 열만 선별해서 범주형 변수의 리스트를 생성
  - 독립변수 중에서 범주형 변수만을 선별

    : if titanic[col].dtype==‘object’

    (list comprehension으로 구현)
  - 총 12개의 열 중에서 7개가 범주형

```
#independent variables 1.categorical var.
category_feature=[col for col in titanic.columns if titanic[col].dtype=='object']
category_feature
```

```
['Survived', 'Pclass', 'Name', 'Sex', 'Ticket', 'Cabin', 'Embarked']
```

실습1.9 종속변수 등 제외

- **종속변수 등 제외**
  - 독립변수가 아닌 종속변수 및 데이터의 기본키(인덱스) 등도 제외
  - 집합 메소드를 활용하여 종속변수 'Survived'와 기본키에 해당하는 'Name'을 범주형 변수에서 제외

    : set(category_feature) – set(['Servived', 'Name'])
  - 총 범주형 변수 7개의 열 중에서 5개가 선별

```
category_feat=list(set(category_feature)-set(['Survived','Name']))
category_feat
```

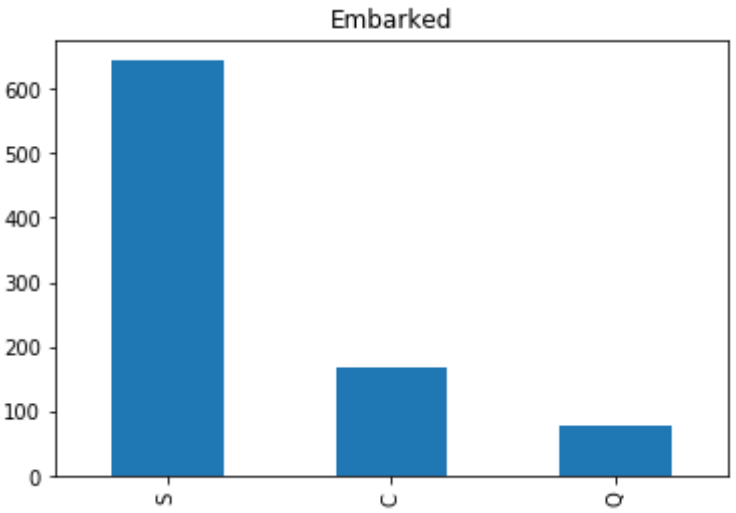
```
['Sex', 'Ticket', 'Pclass', 'Cabin', 'Embarked']
```

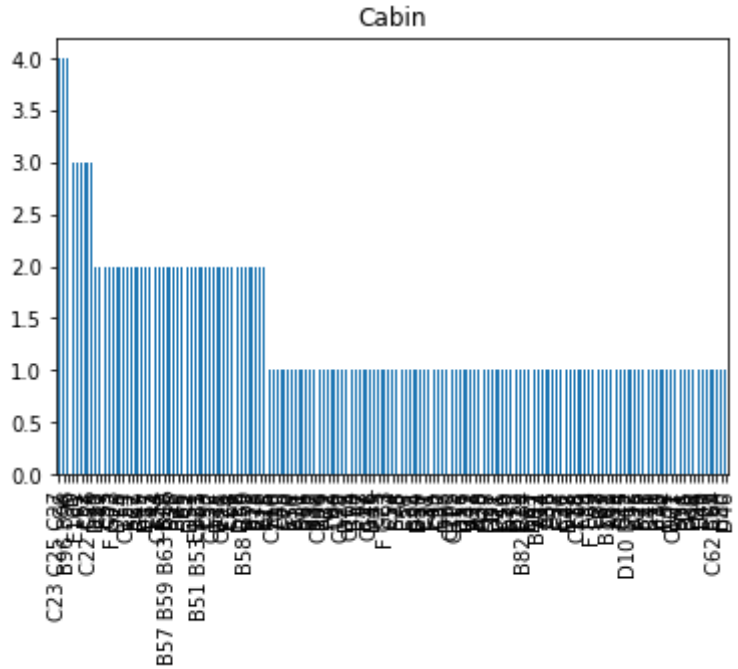
  - cf. 다만, 사주학 연구를 위해서 이름이 생존 여부과 관계가 있는지 알고 싶다면 독립변수로 분석 필요

실습1.10 범주형 변수 시각화

- 시각화 도구로 살펴보기
  - 막대그래프(bar plot)

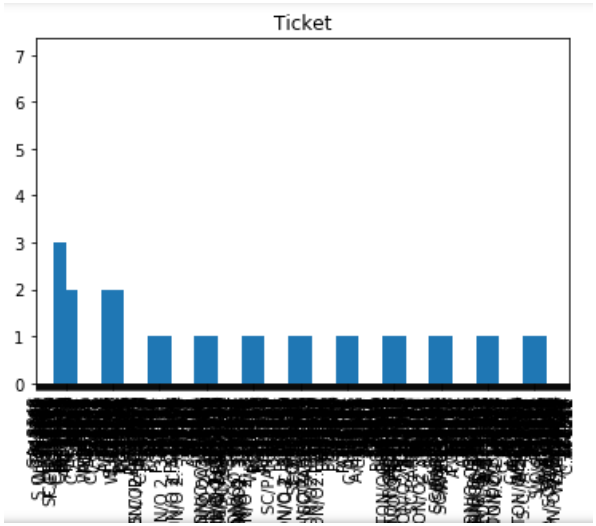
```
for col in category_feat:
    titanic[col].value_counts().plot(kind='bar')
    plt.title(col)
    plt.show()
```

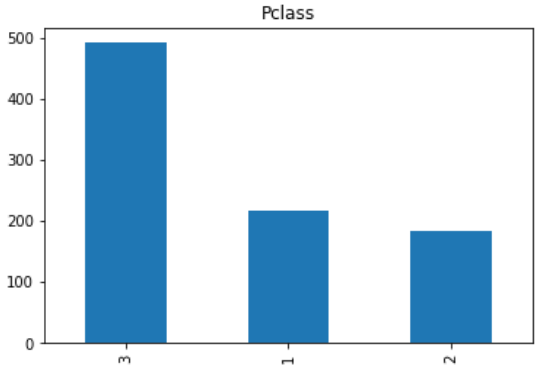


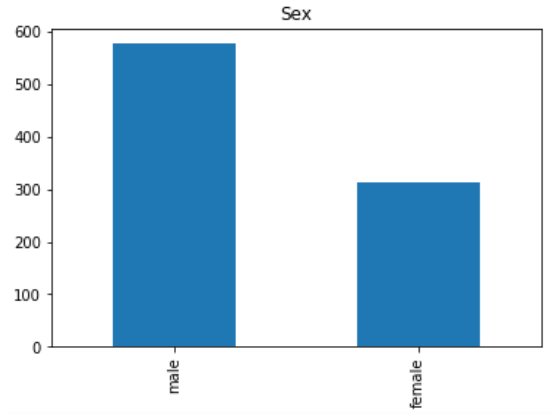


범주형 변수 추가 선별

- 종속변수와 의 상관관계 분석 가능여부
  - 카테고리가 너무 많고, 종속변수와 관련이 없어 보이는 독립변수는 제외







실습1.11 변수간 관계 탐색

- **독립변수와 종속변수의 관계 탐색**
  - 예를 들어, 성별과 생존여부의 관계 파악
  - 여성 생존자 233명, 사망자 81명
  - 남성 생존자 109명, 사망자 488
  - 성별 그룹에 따른 생존여부 그룹의 인원수 카운팅 필요

```
sex_df = titanic.groupby(['Sex', 'Survived'])['Survived'].count().unstack('Survived')
sex_df
```

Survived	0	1
Sex		
female	81	233
male	468	109

[groupby() 기능]

- groupby([‘Sex’, ‘Survived’])
  - 성별 그룹 별 생존여부 그룹
- groupby([‘Sex’, ‘Survived’])[‘Survived’].count()
  - 생존여부 인원수 카운팅

  예) ‘Sex’가 female 이면서 ‘Survived’의 0인 것의 개수 카운팅

```
df = titanic.groupby(['Sex', 'Survived'])['Survived'].count()
df
```

```
Sex     Survived
female  0            81
        1           233
male    0           468
        1           109
Name: Survived, dtype: int64
```

여성 사망자수 카운팅

[unstack() 기능]

- unstack(‘Survived’)
  - 해당 행(row)을 열(column)로
  - 행으로 표현되어 있는 ‘Survived’를 열 표현으로

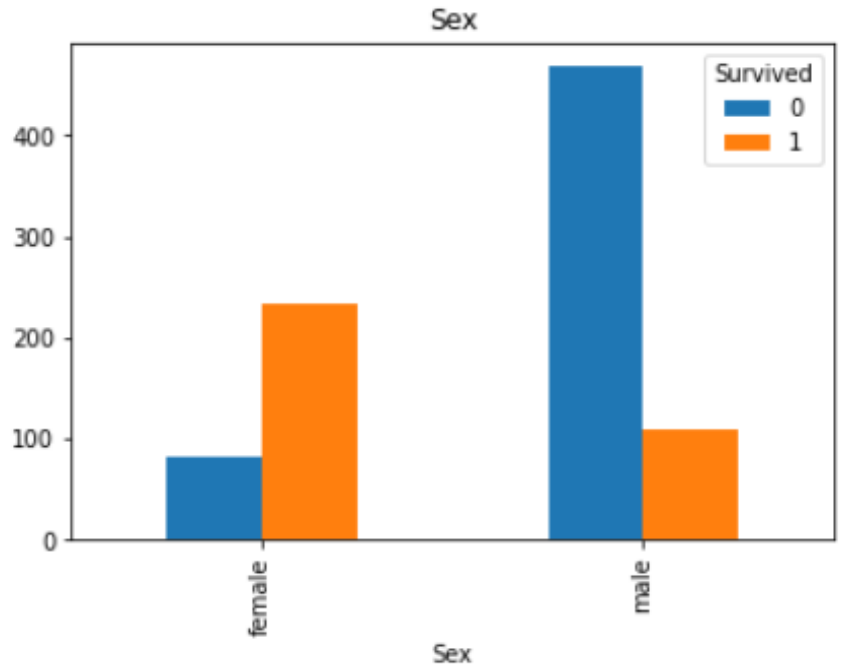
```
sex_df = titanic.groupby(['Sex', 'Survived'])['Survived'].count().unstack('Survived')
sex_df
```

Survived	0	1
Sex		
female	81	233
male	468	109

실습1.12 변수간 관계 시각화

- 독립변수와 종속변수의 관계 시각화
  - 성별에 따른 생존여부
  - 주황색 바 : 생존자
  - 파란색 바 : 사망자

```
df.plot(kind='bar')
plt.title('Sex')
plt.show()
```



실습1.13 수치형 변수 선별

- **수치형 변수 선별 작업**
  - 실습1.8에서 (object타입을 가진 열만 선별한) 범주형 변수의 리스트 category_feature를 아래 코드와 같이 제외
  - 또한, 기타 인덱스 변수('Passnger Id'), 종속변수들을 제외하고 수치형 변수만을 선별
  - 총 12개의 열 중에서 4개가 수치형
  - sort() 메소드로 알파벳순 정렬

```
#independent variables 2.continuous var.
numerical_feature = list(set(titanic.columns)-set(category_feature)-{'PassengerId'})
numerical_feature = np.sort(numerical_feature)
print(numerical_feature)
```

```
['Age' 'Fare' 'Parch' 'SibSp']
```

실습1.14 비연속형 변수 선별

- **비연속형 변수 선별 작업**
  - 실습1.13에서 수치형 변수의 리스트 numerical_feature에서 아래 코드와 같이 int64타입을 가진 열만 선별해서 비연속형 변수의 리스트를 생성
  - 수치형 4개중에서 2개를 비연속형으로 판단

```
titanic['Parch'].value_counts()
```

```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

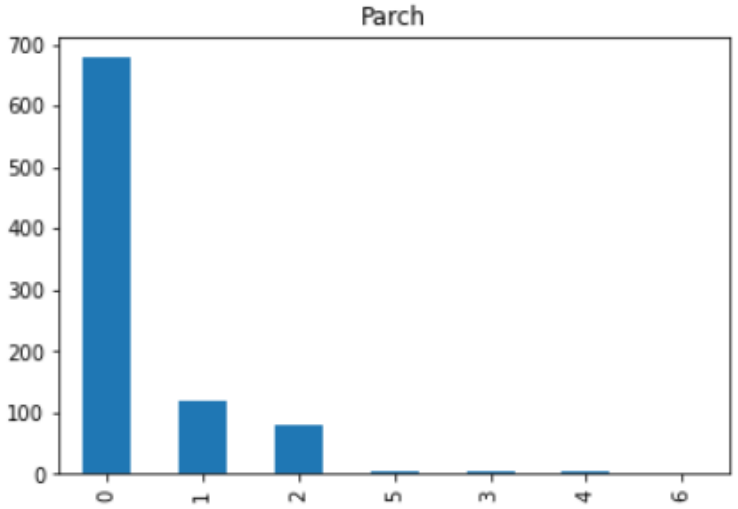
```
uncontinuous_feature=[col for col in numerical_feature if titanic[col].dtype=='int64']
uncontinuous_feature
```

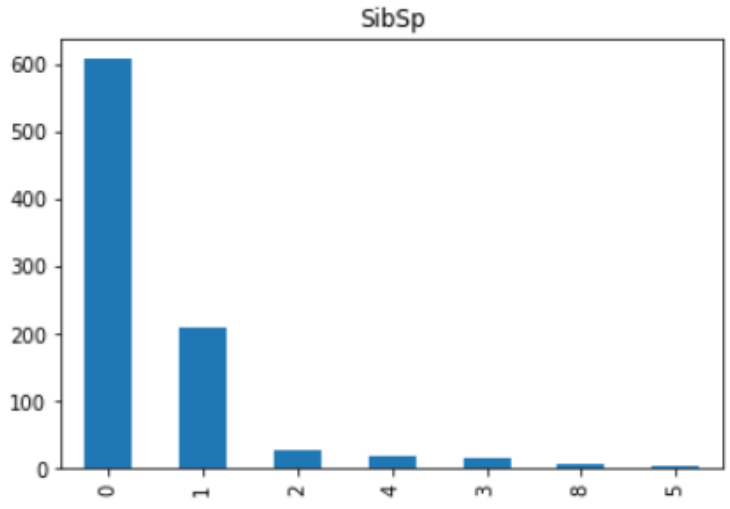
```
['Parch', 'SibSp']
```

실습1.15 비연속형 변수 시각화

- **비연속형 변수는 범주형으로 시각화**
  - 비연속형은 수치형이면서 범주형으로 시각화

```
for col in uncontinuous_feature:
    titanic[col].value_counts().plot(kind='bar')
    plt.title(col)
    plt.show()
```





실습1.16 연속형 변수 선별

- **연속형 변수 선별 작업**
  - 실습1.14에서 수치형 변수의 리스트 numerical_feature에서 비연속형 변수 uncontinuous_feature를 아래 코드와 같이 제외
  - 수치형 4개중에서 2개를 연속형으로 판단
  - 연속형은 모두 float64 타입

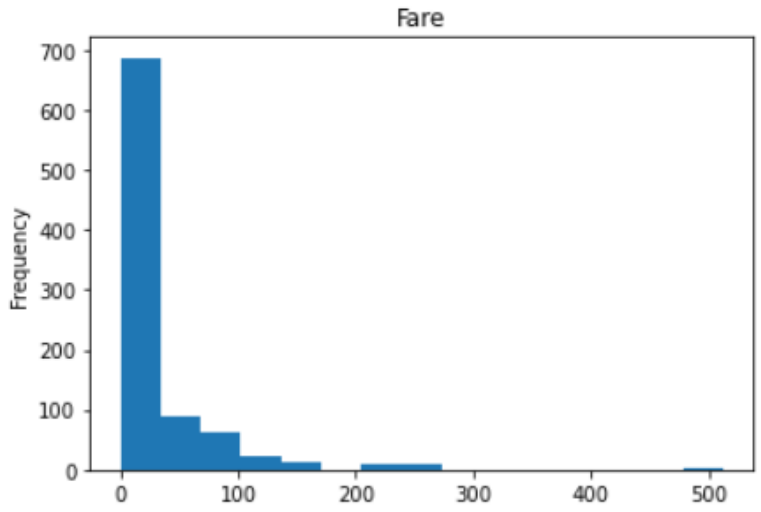
```
#independent variables 2.continuous var.
continuous_feature = list(set(numerical_feature)-set(uncontinuous_feature))
continuous_feature
```

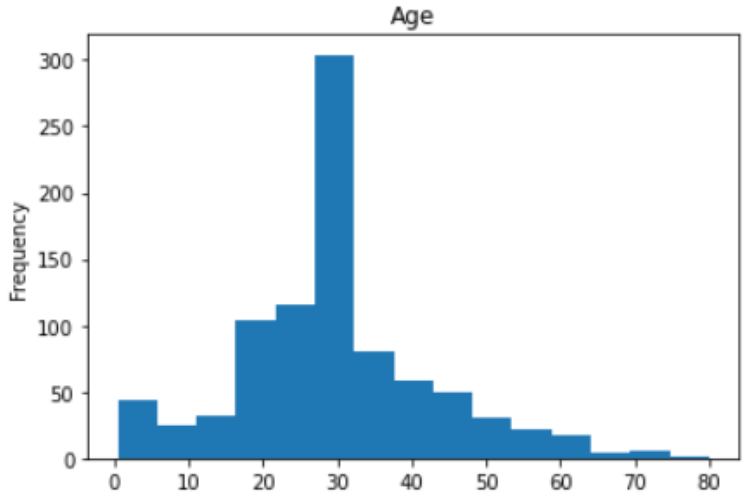
```
['Fare', 'Age']
```

실습1.17 연속형 변수 시각화

- 연속형 변수를 시각화 도구로 살펴보기
  - 히스토그램(histogram)

```
for col in continuous_feature:
    titanic[col].plot(kind='hist', bins=15)
    plt.title(col)
    plt.show()
```

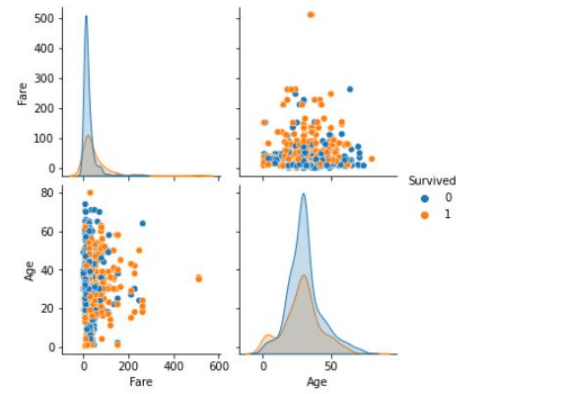




다변수 탐색

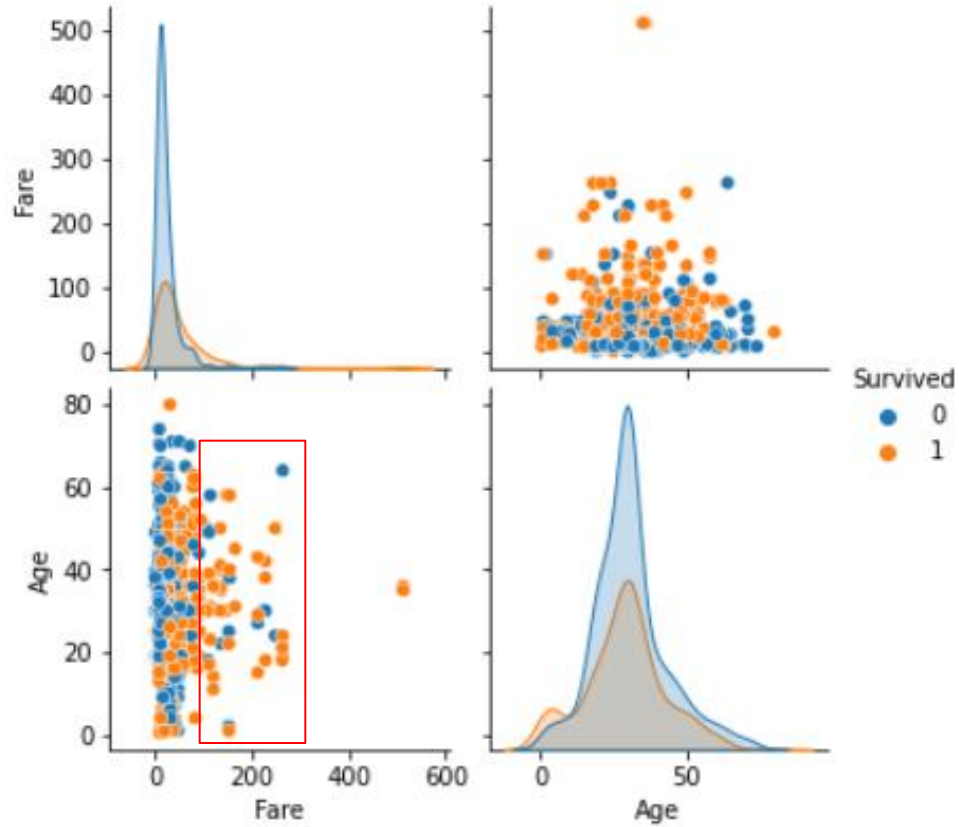
- 다변수를 한번에 시각화 도구로 살펴보기
  - seaborn패키지의 pairplot()을 통해 종속변수를 포함한 3개의 변수를 한 번에 볼 수 있도록 플로팅해보자
  - pariplot()은 여러 변수의 관계를 한 번에 파악할 수 있으며,
  - hue옵션을 통해 종속변수를 지정함으로써 세 변수의 관계를 파악
  - 산점도를 보았을 때, 요금과 생존여부는 상관관계가 있는가?

```
sns.pairplot(titanic[continuous_feature+['Survived']], hue='Survived')
plt.show()
```



실습1.18 다변수 탐색

```
sns.pairplot(titanic[continuous_feature+['Survived']], hue='Survived')
plt.show()
```

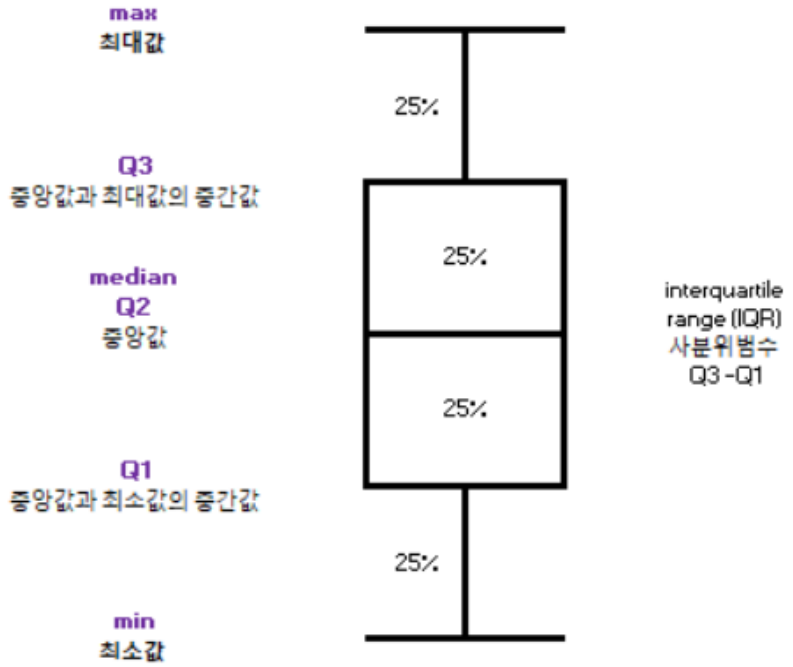


범주형, 연속형 변수간 관계

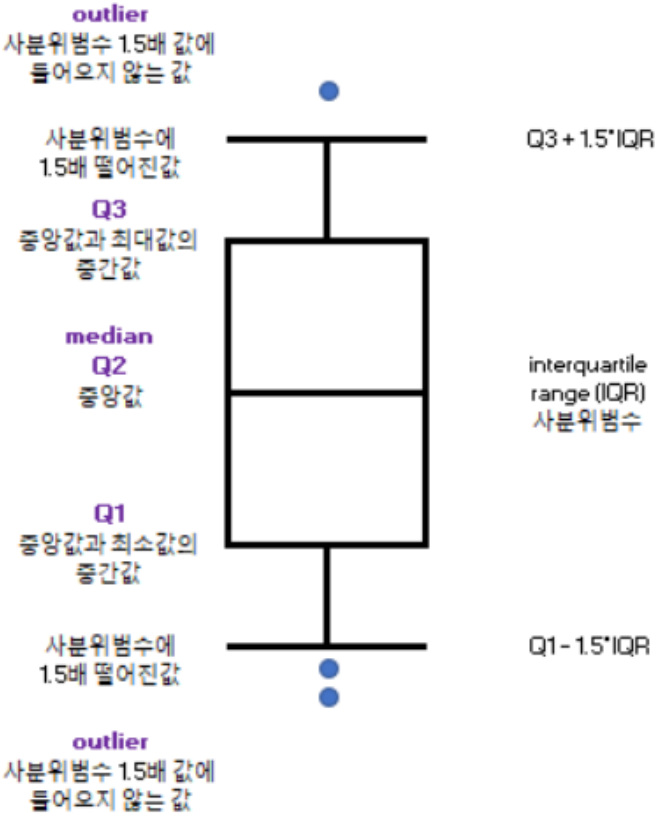
- **범주형과 연속형 변수간의 관계를 한번에 시각화 도구로 살펴보기**
  - 예를 들어, 성별, 나이, 생존여부 3개의 변수를 동시에 탐색하는 경우
  - 종속변수는 생존여부, 독립변수는 성별과 나이
  - 성별은 대표적인 범주형 변수, 나이는 연속형 변수
  - 성별에 따라 나이의 boxplot을 플로팅하여 변수간의 관계를 탐색

- **상자 도표(Box Plot) 시각화 도구**
  - 최대값, 최소값 및 사분편차(Q1, Q2, Q3)를 사용하여 자료의 측정값들이 어떤 모양으로 분포되어 있으며, 극단값 또는 이상치(outliers)들은 어떠한지 등을 쉽게 알 수 있도록 보여주는 도표
  - seaborn의 boxplot()메소드로 실습

boxplot 개요



<이상치 예측하지 않았을 때 상자그림: 상자수염의 끝은 최대값과 최소값>

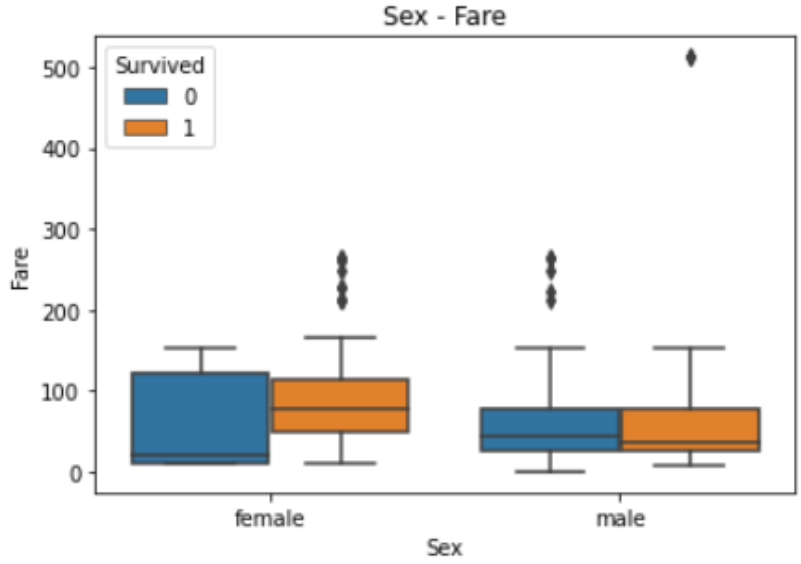


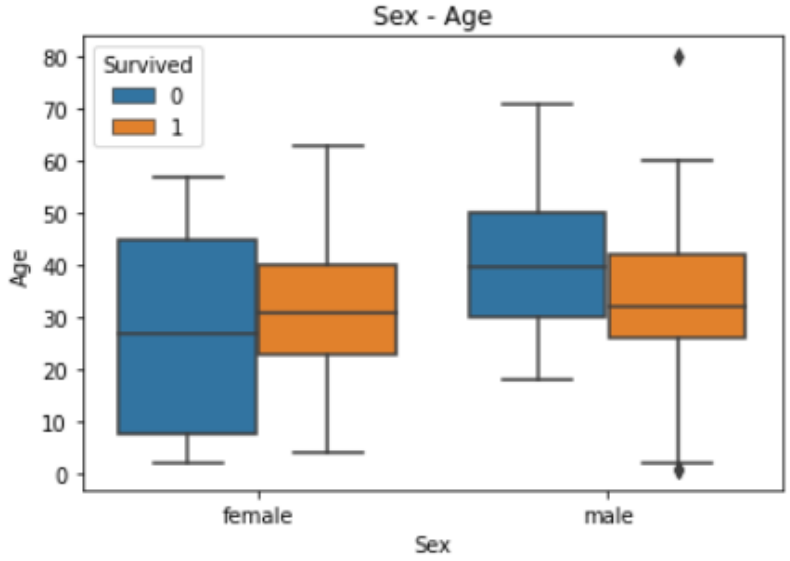
<이상치 예측할 때 상자그림: 상자수염의 끝은 사분위범수에 1.5배 떨어진 값>

실습1.19 boxpot 시각화

- **botplot으로 데이터의 형태 확인**
  - 예) 극단치 확인 : 남성 생존자중 요금 500대 -> 이상치?

```
for col in continuous_feature:
    sns.boxplot(x='Sex', y=col, hue='Survived', data=titanic.dropna())
    plt.title('Sex - {}'.format(col))
    plt.show()
```





참고도서

➢ reference

[1] 파이썬을 활용한 데이터길들이기

- 프로그래밍인사이트

[2] 모두의 데이터분석

- 길벗

[3] 파이썬머신러닝 판다스데이터분석

- 정보문화사

End