단국대학교 SW중심대학

전공별 AI활용

데이터 살펴보기

정 복 문

Bokmoon.jung@dankook.ac.kr

CONTENTS

▪ UCI 자동차 연비(auto mpg) 데이터셋

- 실린더 수, 배기량, 출력, 차중, 가속능력, 출시년도, 제조국, 모델명에 관한 데이터 398 개로 구성

No.	속성(attributes)		데이터 상세(범위)
1	mpg	연비	연속 값
2	cylinders	실린더 수	이산 값(예시: 3, 4, 6, 8)
3	displacement	배기량	연속 값
4	horsepower	출력	연속 값
5	weight	차중	연속 값
6	acceleration	가속능력	연속 값
7	model_year	출시년도	이산 값(예: 70, 71, 80, 81)
8	origin	제조국	이산 값(예: 1(USA), 2(EU), 3(JPN))
9	name	모델명	문자열

[표 3-1] UCI 데이터셋 – "auto mpg" 상세 항목

- **DataFrame 의 속성과 메소드 도움말 : https://pandas.pydata.org/docs/reference/frame.html**
- **데이터 내용 미리보기**
  - 데이터셋의 내용과 구조를 개략적으로 살펴볼 수 있어 분석방향을 정하는데 필요한 정보를 얻기에 유용함.
  - 데이터프레임이 너무 커서 한 화면에 출력하여 보기 어려운 경우에도 사용
  - head(), tail() 메소드 사용

```
DataFrame.head(n=5) : 처음 n개의 행 반환
DataFrame.tail(n=5) : 마지막 n개의 행 반환
```

    - n : int, default 5, 선택한 행의 수

[예제 3-1]

```
3  import pandas as pd
4
5  # read_csv() 함수로 df 생성
6  df = pd.read_csv('./auto-mpg.csv', header=None)
7
8  # 열 이름 지정
9  df.columns = ['mpg','cylinders','displacement','horsepower','weight',
10                'acceleration','model year','origin','name']
11
12 # 데이터프레임 df의 내용을 일부 확인
13 print(df.head())       # 처음 5개 행
14 print('\n')
15 print(df.tail())       # 마지막 5개 행
```

```
    mpg   cylinders           ...              origin                       name
0  18.0         8             ...                 1    chevrolet chevelle malibu
1  15.0         8             ...                 1            buick skylark 320
2  18.0         8             ...                 1           plymouth satellite
3  16.0         8             ...                 1                amc rebel sst
4  17.0         8             ...                 1                  ford torino

[5 rows x 9 columns]


      mpg   cylinders       ...         origin                name
393  27.0         4         ...              1       ford mustang gl
394  44.0         4         ...              2             vw pickup
395  32.0         4         ...              1         dodge rampage
396  28.0         4         ...              1           ford ranger
397  31.0         4         ...              1             chevy s-10

[5 rows x 9 columns]
```

▪ 데이터 요약 정보 확인하기

- ▪ 데이터프레임의 크기(행, 열)
  - DataFrame.shape 속성 사용
  - 행과 열의 개수를 튜플 형태로 반환
- ▪ 데이터프레임의 기본 정보
  - DataFrame.info( ) 메소드 사용
  - 데이터프레임의 클래스 유형, 행 인덱스의 구성, 열 이름의 종류와 개수, 각 열의 자료형과 개수, 메모리 할당량에 관한 기본 정보를 반환

[예제 3-1]

```
~  ~~~ 생략 ~~~

17  # df의 모양과 크기 확인: (행의 개수, 열의 개수)를 투플로 반환
18  print(df.shape)
```

```
(398, 9)
```

[예제 3-1]

```
~  ~~~ 생략 ~~~

21  # 데이터프레임 df의 내용 확인
22  print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
mpg             398 non-null float64
cylinders       398 non-null int64
displacement    398 non-null float64
horsepower      398 non-null object
weight          398 non-null float64
acceleration    398 non-null float64
model year      398 non-null int64
origin          398 non-null int64
name            398 non-null object
dtypes: float64(4), int64(3), object(2)
memory usage: 24.9+ KB
None
```

▪ 데이터 요약 정보 확인하기

▪ 데이터프레임의 기본 정보

- 각 열의 자료형 확인 : DataFrame.dtypes 속성 사용
- 특정열만 선택하여 자료형 확인 가능: DataFrame.column.dtypes

판다스 자료형	파이썬 자료형	비고
int64	int	정수형 데이터
float64	float	실수형 데이터
object	string	문자열 데이터
datetime64	없음 (datetime 라이브러리 활용)	시간 데이터

[예제 3-1]

```
--- 생략 ---
```

```
# 데이터프레임 df의 자료형 확인
print(df.dtypes)
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight          float64
acceleration    float64
model year        int64
origin            int64
name             object
dtype: object
```

```
# 시리즈(mpg 열)의 자료형 확인
print(df.mpg.dtypes)
print('\n')
```

```
float64
```

▪ 데이터 요약 정보 확인하기

▪ 데이터프레임의 기술 통계 정보 요약

- DataFrame.describe( ) 메소스 사용
- 산술 데이터를 갖는 열에 대한 주요 기술 통계정보를 요약하여 반환(결측치를 제외)
- count 값, mean 값(평균), std 값(표준편차), min 값(최소값), max 값(최대값), 50%(중앙값, 2사분위수), 25%(1사분위수), 75%(3사분위수)를 집계

```
DataFrame.describe(percentiles=None, include=None, exclude=None, datetime_is_numeric=False)
```

- percentiles : output에 포함할 백분위수 (기본값은 [.25, .5, .75]이며, 25%,50%,75% 을 반환함)
- include : ‘all’, list-like of dtypes or None (default)
  include=‘all’ : 모든 열이 포함하려는 경우 옵션 추가
  산술 데이터를 가진 열에 대해서 결측치(유효한 값이 없다는 뜻)를 NaN 값으로 표시

1. 데이터프레임 구조

[예제 3-1]

```
~   ~~~ 생략 ~~~

33  # 데이터프레임 df의 기술 통계 정보 확인
34  print(df.describe())
35  print('\n')
36  print(df.describe(include='all'))
```

```
              mpg   cylinders  ...  model year      origin
count  398.000000  398.000000  ...  398.000000  398.000000
mean    23.514573    5.454774  ...   76.010050    1.572864
std      7.815984    1.701004  ...    3.697627    0.802055
min      9.000000    3.000000  ...   70.000000    1.000000
25%     17.500000    4.000000  ...   73.000000    1.000000
50%     23.000000    4.000000  ...   76.000000    1.000000
75%     29.000000    8.000000  ...   79.000000    2.000000
max     46.600000    8.000000  ...   82.000000    3.000000

[8 rows x 7 columns]
```

```
              mpg   cylinders  ...      origin        name
count  398.000000  398.000000  ...  398.000000         398
unique        NaN         NaN  ...         NaN         305
top           NaN         NaN  ...         NaN  ford pinto
freq          NaN         NaN  ...         NaN           6
mean    23.514573    5.454774  ...    1.572864         NaN
std      7.815984    1.701004  ...    0.802055         NaN
min      9.000000    3.000000  ...    1.000000         NaN
25%     17.500000    4.000000  ...    1.000000         NaN
50%     23.000000    4.000000  ...    1.000000         NaN
75%     29.000000    8.000000  ...    2.000000         NaN
max     46.600000    8.000000  ...    3.000000         NaN

[11 rows x 9 columns]
```

▪ 데이터 개수 확인

▪ 각 열의 데이터 개수 : count() 메소드 사용

- 각 열의 데이터 개수를 시리즈 객체로 반환(단, 유효한 값의 개수만을 계산)

```
DataFrame.count(axis=0, level=None, numeric_only=False)
```

- axis: ***{0 or 'index', 1 or 'columns'}, default 0***
- level : axis가 MultiIndex인 경우 level 지정
- numeric_only : True , False (float, int, boolean 데이터만 포함할지 지정)

[예제 3-2]

```
3  import pandas as pd
4
5  # read_csv() 함수로 df 생성
6  df = pd.read_csv('./auto-mpg.csv', header=None)
7
8  # 열 이름 지정
9  df.columns = ['mpg','cylinders','displacement','horsepower','weight',
10                'acceleration','model year','origin','name']
11
12 # 데이터프레임 df의 각 열이 가지고 있는 원소 개수 확인
13 print(df.count())
14 print('\n')
15
16 # df.count()가 반환하는 객체 타입 출력
17 print(type(df.count()))
```

```
mpg             398
cylinders       398
displacement    398
horsepower      398
weight          398
acceleration    398
model year      398
origin          398
name            398
dtype: int64


<class 'pandas.core.series.Series'>
```

▪ 데이터 개수 확인

▪ 각 열의 고유값 개수

- DataFrame.value_counts() 메소드 사용
- 시리즈 객체의 고유값 개수를 계산하는데 사용한다.
- 데이터프레임은 각 열의 고유값의 종류와 개수를 확인할 수 있다.
- 고유값이 행 인덱스가 되고, 고유값의 개수가 데이터 값이 되는 시리즈 객체로 반환

[예제 3-2]

```
~  ~~~ 생략 ~~~

20  # 데이터프레임 df의 특정 열이 가지고 있는 고유값 확인
21  unique_values = df['origin'].value_counts()
22  print(unique_values)
23  print('\n')
24
25  # value_counts 메소드가 반환하는 객체 타입 출력
26  print(type(unique_values))
```

```
1    249
3     79
2     70
Name: origin, dtype: int64


<class 'pandas.core.series.Series'>
```

▪ 평균값

- mean() 메소드 사용
- 산술 데이터를 갖는 모든 열이나 특정 열에 대해 평균값을 각각 계산하여 시리즈 객체로 반환

DataFrame.mean() : 모든 열의 평균값 반환

DataFrame['열이름'].mean() : 특정 열의 평균값 반환

[예제 3-3]

```
3  import pandas as pd
4
5  # read_csv() 함수로 df 생성
6  df = pd.read_csv('./auto-mpg.csv', header=None)
7
8  # 열 이름 지정
9  df.columns = ['mpg','cylinders','displacement','horsepower','weight',
10                'acceleration','model year','origin','name']
11
12 # 평균값
13 print(df.mean())
14 print('\n')
15 print(df['mpg'].mean())
16 print(df.mpg.mean())
17 print('\n')
18 print(df[['mpg','weight']].mean())
```

```
mpg             23.514573
cylinders        5.454774
displacement   193.425879
weight        2970.424623
acceleration    15.568090
model year      76.010050
origin           1.572864
dtype: float64


23.514572864321615
23.514572864321615


mpg          23.514573
weight     2970.424623
dtype: float64
```

▪ 중간값

- median() 메소드 사용
- 산술 데이터를 갖는 모든 열이나 특정 열에 대해 중간값을 계산하여 시리즈로 반환

DataFrame.median() : 모든 열의 중간값 반환
DataFrame['열이름'].median() : 특정 열의 중간값 반환

[예제 3-3]

```
~  ~~~ 생략 ~~~

20  # 중간값
21  print(df.median())
22  print('\n')
23  print(df['mpg'].median())
```

```
mpg             23.0
cylinders        4.0
displacement   148.5
weight        2803.5
acceleration    15.5
model year      76.0
origin           1.0
dtype: float64


23.0
```

▪ 최대값

- max() 메소드 사용
- 모든 열 또는 특정 열에서 각 열이 갖는 데이터 값 중에서 최대값을 계산하여 시리즈로 반환

DataFrame.max() : 모든 열의 최대값 반환

DataFrame['열이름'].max() : 특정 열의 최대값 반환

[예제 3-3]

```
~  ~~~ 생략 ~~~

25  # 최대값
26  print(df.max())
27  print('\n')
28  print(df['mpg'].max())
```

```
mpg                          46.6
cylinders                       8
displacement                  455
horsepower                      ?
weight                       5140
acceleration                 24.8
model year                     82
origin                          3
name              vw rabbit custom
dtype: object


46.6
```

▪ 최소값

- min() 메소드 사용
- 모든 열 또는 특정 열에서 각 열이 갖는 데이터 값 중에서 최소값을 계산하여 시리즈로 반환

DataFrame.min() : 모든 열의 최소값 반환

DataFrame['열이름'].min() : 특정 열의 최소값 반환

[예제 3-3]

```
 ~   ~~~ 생략 ~~~


30  # 최소값
31  print(df.min())
32  print('\n')
33  print(df['mpg'].min())
```

```
mpg                                9
cylinders                          3
displacement                      68
horsepower                     100.0
weight                          1613
acceleration                       8
model year                        70
origin                             1
name            amc ambassador brougham
dtype: object



9.0
```

▪ 표준편차

- std() 메소드 사용
- 산술 데이터를 갖는 열 또는 특정 열의 표준편차를 계산하여 시리즈로 반환

DataFrame.std() : 모든 열의 표준편차 반환

DataFrame['열이름'].std() : 특정 열의 표준편차 반환

[예제 3-3]

```
~  ~~~ 생략 ~~~

35  # 표준편차
36  print(df.std())
37  print('\n')
38  print(df['mpg'].std())
```

```
mpg             7.815984
cylinders       1.701004
displacement  104.269838
weight        846.841774
acceleration    2.757689
model year      3.697627
origin          0.802055
dtype: float64


7.815984312565782
```

▪ 상관계수

- corr() 메소드 사용
- 산술 데이터를 갖는 모든 열에 대해 2개씩 서로 짝을 짓고, 각각의 경우에 대하여 상관계수를 계산하여 시리즈로 반환

DataFrame.corr() : 모든 열의 상관계수 반환

DataFrame[열 이름의 리스트].corr() : 특정 열의 상관계수 반환

[예제 3-3]

```
~  ~~~ 생략 ~~~

40  # 상관계수
41  print(df.corr())
42  print('\n')
43  print(df[['mpg','weight']].corr())
```

```
                mpg  cylinders  ...  model year  origin
mpg        1.000000  -0.775396  ...    0.579267  0.563450
cylinders -0.775396   1.000000  ...   -0.348746 -0.562543
displacement -0.804203  0.950721  ...   -0.370164 -0.609409
weight     -0.831741   0.896017  ...   -0.306564 -0.581024
acceleration  0.420289  -0.505419  ...    0.288137  0.205873
model year  0.579267  -0.348746  ...    1.000000  0.180662
origin     0.563450  -0.562543  ...    0.180662  1.000000
[7 rows x 7 columns]


                mpg      weight
mpg        1.000000   -0.831741
weight    -0.831741    1.000000
```

- **그래프를 이용한 시각화 방법은 데이터의 분포와 패턴을 파악하는 것에 매우 효과적이다.**
- **판다스 그래프 도구 사용 방법**
  - 시리즈 또는 데이터프레임 객체에 plot() 메소드를 적용하고, kind 옵션으로 그래프의 종류를 선택한다.

kind 옵션	설명	kind 옵션	설명
'line'	선 그래프	'kde'	커널 밀도 그래프
'bar'	수직 막대 그래프	'area'	면적 그래프
'barh'	수평 막대 그래프	'pie'	파이 그래프
'his'	히스토그램	'scatter'	산점도 그래프
'box'	박스 플롯	'hexbin'	고밀도 산점도 그래프

[표 3-3] 판다스 내장 plot( ) 메소드 – 그래프 종류

▪ DataFrame 클래스의 plot() 메서드

```
DataFrame.plot(kind='line', x=column, y=columns, color=color)
```

- kind: 차트 종류
- x: x축 열 이름, y: y축 열 이름들 ➔ DataFrame에 있는 열 이름을 사용해야함
  - 값을 설정하지 않을 경우 DataFrame의 모든 데이터에서 인덱스를 x축으로 하고, 나머지를 y축으로 인식. 숫자가 아닌 경우는 그래프로 나타나지 않음
  - x축으로 설정한 열 이름은 자동으로 x축 제목으로 표시.
    만약 x축에 설정할 열 이름이 DataFrame에 없으면 KeyError 발생

kind : *str*

The kind of plot to produce:

- 'line' : line plot (default)
- 'bar' : vertical bar plot
- 'barh' : horizontal bar plot
- 'hist' : histogram
- 'box' : boxplot
- 'kde' : Kernel Density Estimation plot
- 'density' : same as 'kde'
- 'area' : area plot
- 'pie' : pie plot
- 'scatter' : scatter plot (DataFrame only)
- 'hexbin' : hexbin plot (DataFrame only)

▪ 선 그래프

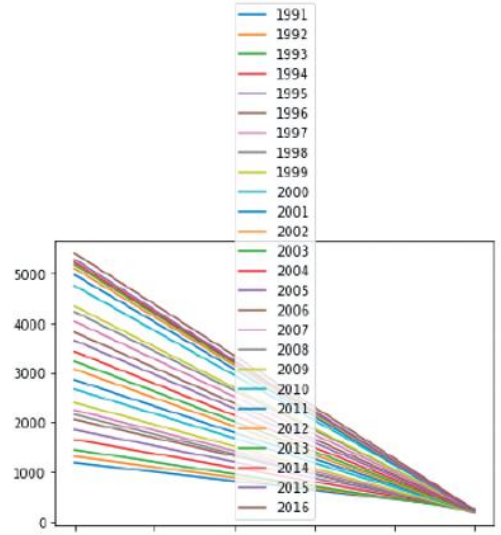
- 데이터 프레임(시리즈) 객체에 plot() 메소드를 적용할 때, 다른 옵션을 추가하지 않으면 가장 기본적인 선 그래프를 그린다.
- **DataFrame.plot()**

[예제 3-4]

```
3  import pandas as pd
4
5  df = pd.read_excel('./남북한발전전력량.xlsx')  # 데이터프레임 변환
6
7  df_ns = df.iloc[[0, 5], 3:]                  # 남한, 북한 발전량 합계 데이터만 추출
8  df_ns.index = ['South','North']              # 행 인덱스 변경
9  df_ns.columns = df_ns.columns.map(int)       # 열 이름의 자료형을 정수형으로 변경
10 print(df_ns.head())
11 print('\n')
12
13 # 선 그래프 그리기
14 df_ns.plot()
```

앞의 그래프 표현은 어색하다. 시간의 흐름에 따른 연도별 발전량 변화 추이를 보기 위해서는 연도 값을 **x**축에 표시해야 하는 것이 적절하다. 다음 예제에서 **x**축, **y** 축 값을 서로 바꿔서 출력한다.

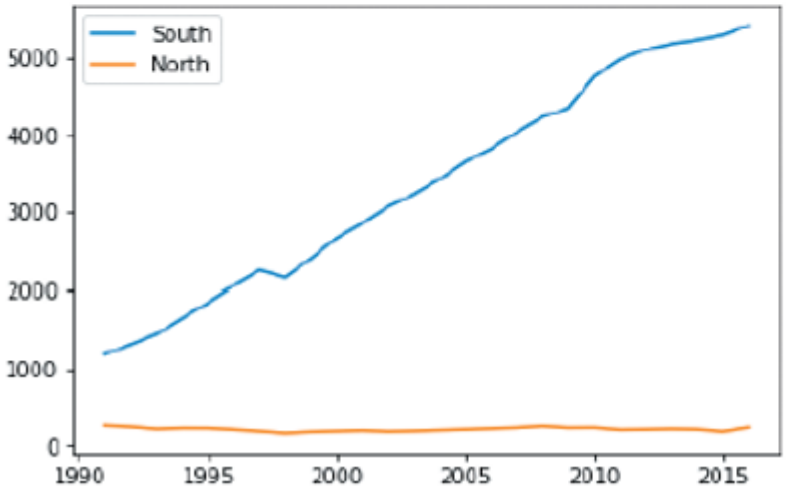
```
       1991  1992  1993  1994  1995 ...  2012  2013  2014  2015  2016
South  1186  1310  1444  1650  1847 ...  5096  5171  5220  5281  5404
North   263   247   221   231   230 ...   215   221   216   190   239

[2 rows x 26 columns]


Out[1]: <matplotlib.axes._subplots.AxesSubplot at 0xa244e50>
```



[예제 3-4]

```
~  ~~~ 생략 ~~~

# 행, 열 전치하여 다시 그리기
tdf_ns = df_ns.T
print(tdf_ns.head())
print('\n')
tdf_ns.plot()
```

```
      South  North
1991   1186    263
1992   1310    247
1993   1444    221
1994   1650    231
1995   1847    230


Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x17e6bb0>
```



▪ 막대 그래프

- 선 그래프와 유사, kind='bar'로 지정, stacked는 하나의 막대에 누적 여부지정

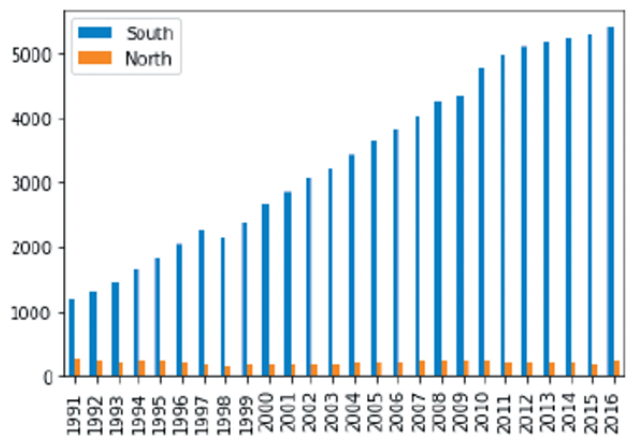
DataFrame.plot(kind='bar', x=column, y=columns, color=color, stacked=False) : 세로 막대 그래프
DataFrame.plot(kind='barh', x=column, y=columns, color=color, stacked=False) : 가로 막대 그래프

- **stacked : *bool, default False in line and bar plots, and True in area plot***

[예제 3-5]

```
3  import pandas as pd
4
5  df = pd.read_excel('./남북한발전전력량.xlsx')  # 데이터프레임 변환
6
7  df_ns = df.iloc[[0, 5], 3:]                  # 남한, 북한 발전량 합계 데이터만 추출
8  df_ns.index = ['South','North']              # 행 인덱스 변경
9  df_ns.columns = df_ns.columns.map(int)       # 열 이름의 자료형을 정수형으로 변경
10
11 # 행, 열 전치하여 막대 그래프 그리기
12 tdf_ns = df_ns.T
13 print(tdf_ns.head())
14 print('\n')
15 tdf_ns.plot(kind='bar')
```

```
      South  North
1991   1186    263
1992   1310    247
1993   1444    221
1994   1650    231
1995   1847    230
```

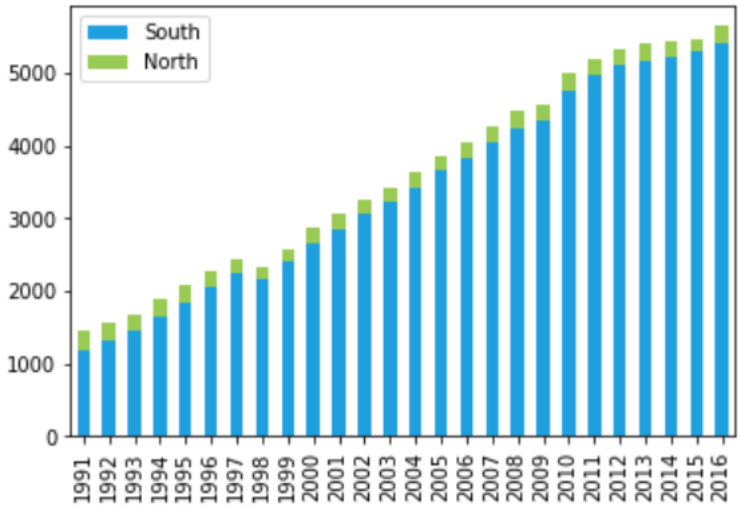


3. 판다스 내장 그래프 도구 활용

[예제 3-5]

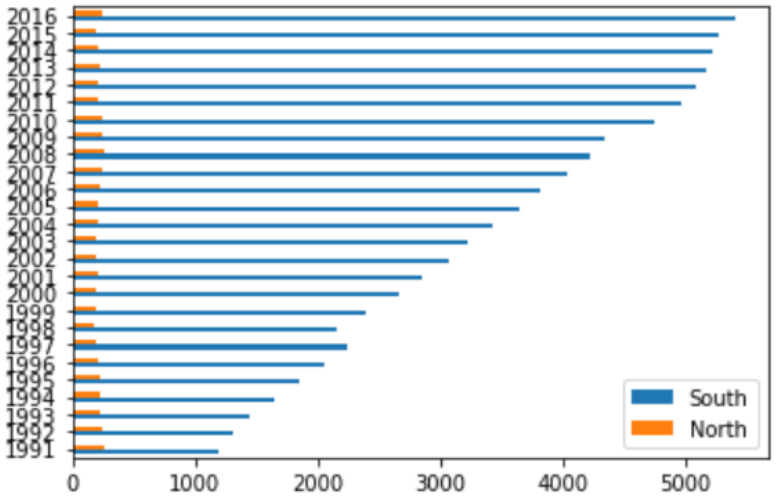
```
#누적 막대그래프
tdf_ns.plot(kind='bar', color=['#209FDF','#99CA53'], stacked=True)
```

```
<AxesSubplot:>
```



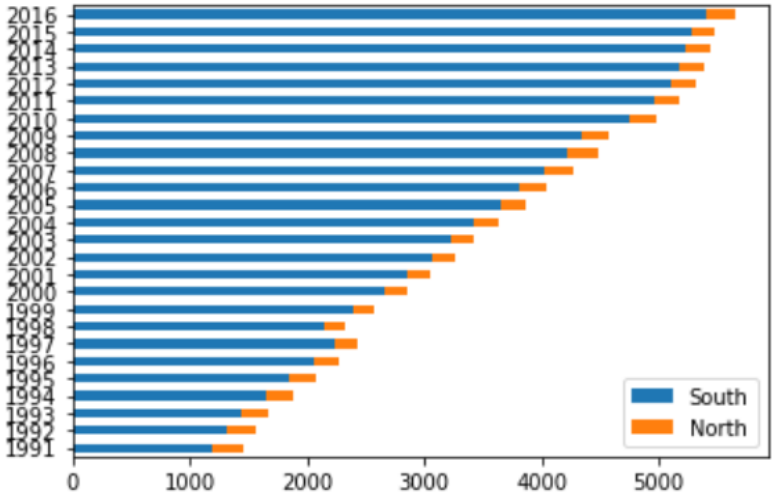

```
#가로 막대 그래프
tdf_ns.plot(kind='barh')
```

```
<AxesSubplot:>
```




```
#가로 누적 막대 그래프
tdf_ns.plot(kind='barh', stacked=True)
```

```
<AxesSubplot:>
```



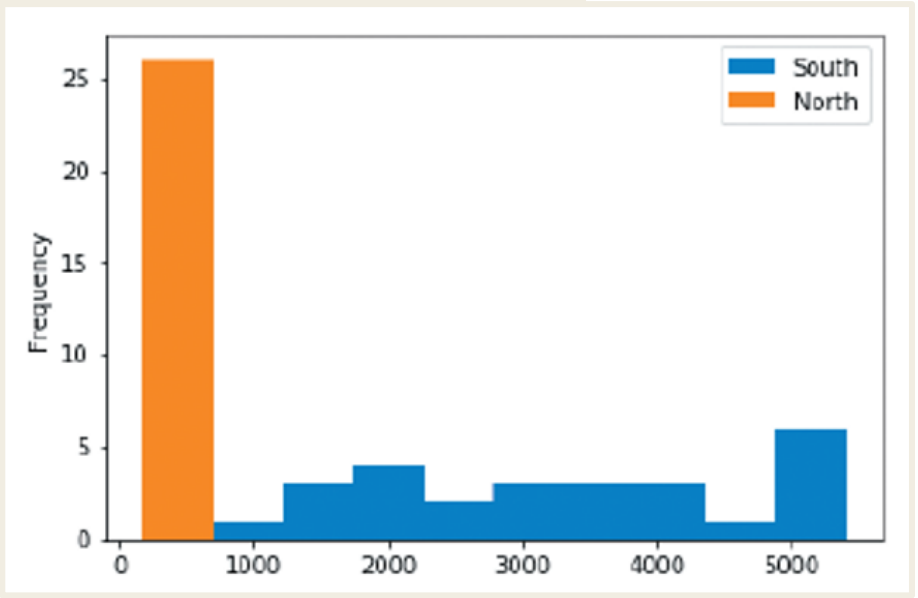
▪ 히스토그램

- kind='hist'로 지정

DataFrame.plot(kind='hist', x=column, y=columns, color=color)

[예제 3-6]

```
1  # -*- coding: utf-8 -*-
2
3  import pandas as pd
4
5  df = pd.read_excel('./남북한발전전력량.xlsx')  # 데이터프레임 변환
6
7  df_ns = df.iloc[[0, 5], 3:]                  # 남한, 북한 발전량 합계 데이터만 추출
8  df_ns.index = ['South','North']              # 행 인덱스 변경
9  df_ns.columns = df_ns.columns.map(int)       # 열 이름의 자료형을 정수형으로 변경
10
11 # 행, 열 전치하여 히스토그램 그리기
12 tdf_ns = df_ns.T
13 tdf_ns.plot(kind='hist')
```



▪ 산점도(Scatter Chart)

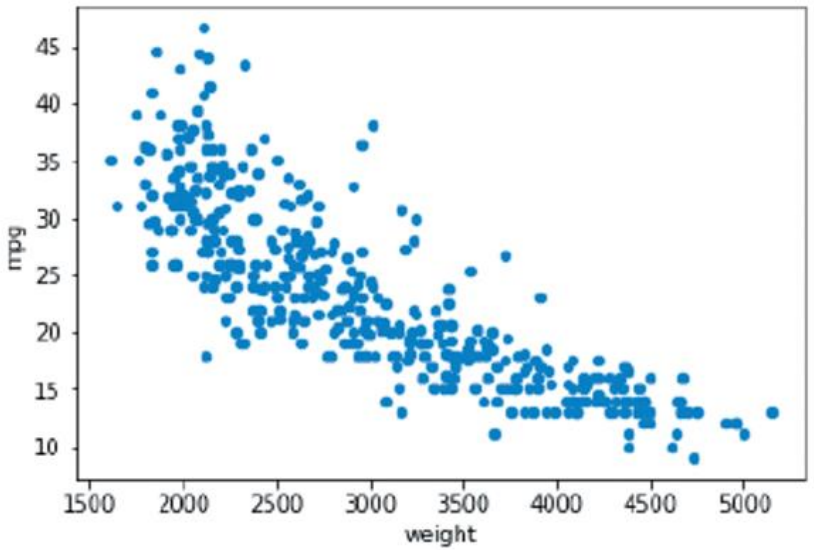
```
DataFrame.plot(kind='scatter', x=column, y=columns, color=color, s=area)
```

- x축, y축 좌표의 값을 점으로 표현
- 기본 형식에서 x, y값은 필수이며, x축의 size와 y축의 size가 같아야 한다.
- x축 1개와 y축 1개를 하나씩 매치시켜서 그래프로 표현하는 방법을 사용
- x와 y의 값이 같은 경우 Error는 발생하지 않으나, 그래프의 모양이 직선으로 나타나 분석의 의미가 없어짐
- s인자는 점의 크기를 설정

[예제 3-7]

UCI 자동차 연비 데이터셋을 이용하여 x축에 차량의 무게 데이터를 갖는 'weight'열을 지정하고, y축에는 연비를 나타내는 'mpg' 열 지정하여 두 변수의 관계를 나타내는 산점도를 그려보자

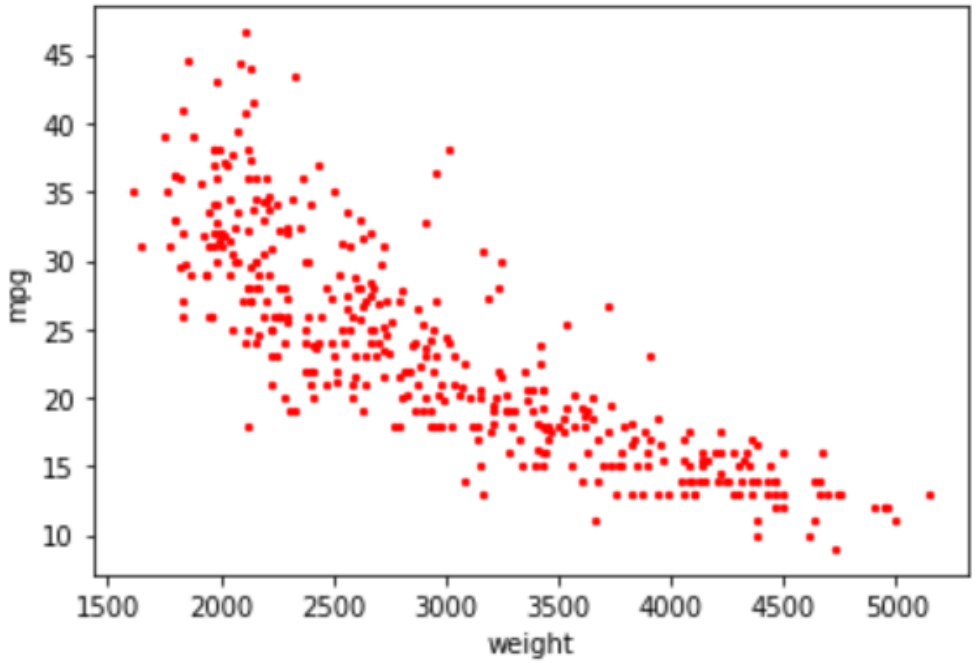
```
import pandas as pd

# read_csv() 함수로 df 생성
df = pd.read_csv('./auto-mpg.csv', header=None)

# 열 이름 지정
df.columns = ['mpg','cylinders','displacement','horsepower','weight',
              'acceleration','model year','origin','name']

# 2개의 열을 선택하여 산점도 그리기
df.plot(x='weight',y='mpg', kind='scatter')
```



[예제 3-7]

```
# 2개의 열을 선택하여 산점도 그리기
# 점의 색과 크기 설정
df.plot(x='weight',y='mpg', kind='scatter', color='red', s=5)
```

```
<AxesSubplot:xlabel='weight', ylabel='mpg'>
```



▪ 박스 플롯

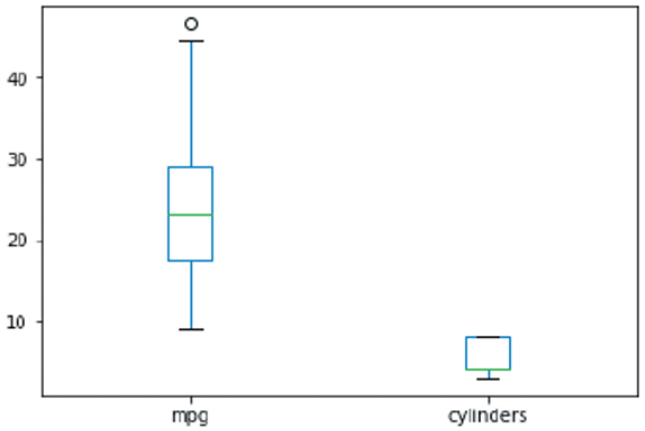
- 특정 변수의 데이터 분포와 분산 정도에 대한 정보를 제공
- kind='box'

DataFrame.plot(kind='box', x=column, y=columns, color=color)

[예제 3-8]

- **연비(mpg열) 데이터는 10 ~ 45 범위에 넓게 분표되어 있고 'o' 표시의 이상값(outlier)도 확인된다.**
- **실린더 개수('cylinders' 열)는 10미만의 좁은 범위에 몰려있다는 것을 확인할 수 있다.**

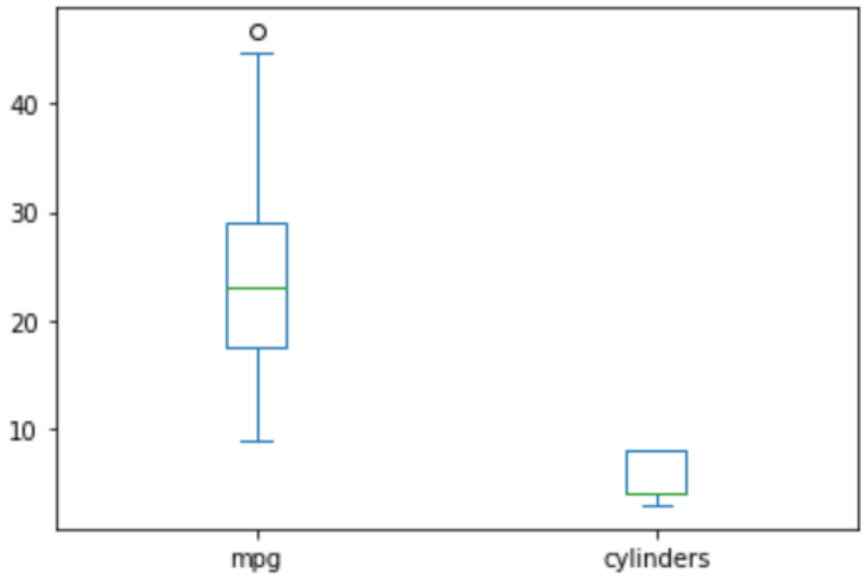
```
3  import pandas as pd
4
5  # read_csv() 함수로 df 생성
6  df = pd.read_csv('./auto-mpg.csv', header=None)
7
8  # 열 이름 지정
9  df.columns = ['mpg','cylinders','displacement','horsepower','weight',
10                'acceleration','model year','origin','name']
11
12 # 열을 선택하여 박스 플롯 그리기
13 df[['mpg','cylinders']].plot(kind='box')
```



[예제 3-8]

```
df.plot(kind='box', y=['mpg','cylinders'])
```

```
<AxesSubplot:>
```



감사합니다

「전공별 AI활용」