수치해석
(2019학년도 1학기)

[11주/1차시 학습내용]:Histogram을 그리는 방법에 대해 알아본다. PDF와 CDF의 의미를 이해한다.

국민대학교 소프트웨어학부

Measures of Spread (확장, 퍼짐 정도의 측정)

- $Unit\ of\ Avg = [v]$
- $Unit\ of\ Variance = [v^2]$
- $in\ oder\ to\ get\ \boldsymbol{the\ same\ unit\ with\ avg}$
- $variance\ has\ been\ used\ as\ \sqrt{variance}$
- $which\ is\ called\ as\ standard\ deviation = \sqrt{variance}$

$80, 70, 60, 90\ [v]: voltage$

$$Avg = \frac{80 + 70 + 60 + 90}{4} = 75\ [v]$$

$$\boldsymbol{variance} = \frac{(80-75)^2 + (70-75)^2 + (60-75)^2 + (90-75)^2}{4} = \frac{500}{4}$$

$$= \boldsymbol{125}[v^2]$$

Mean()

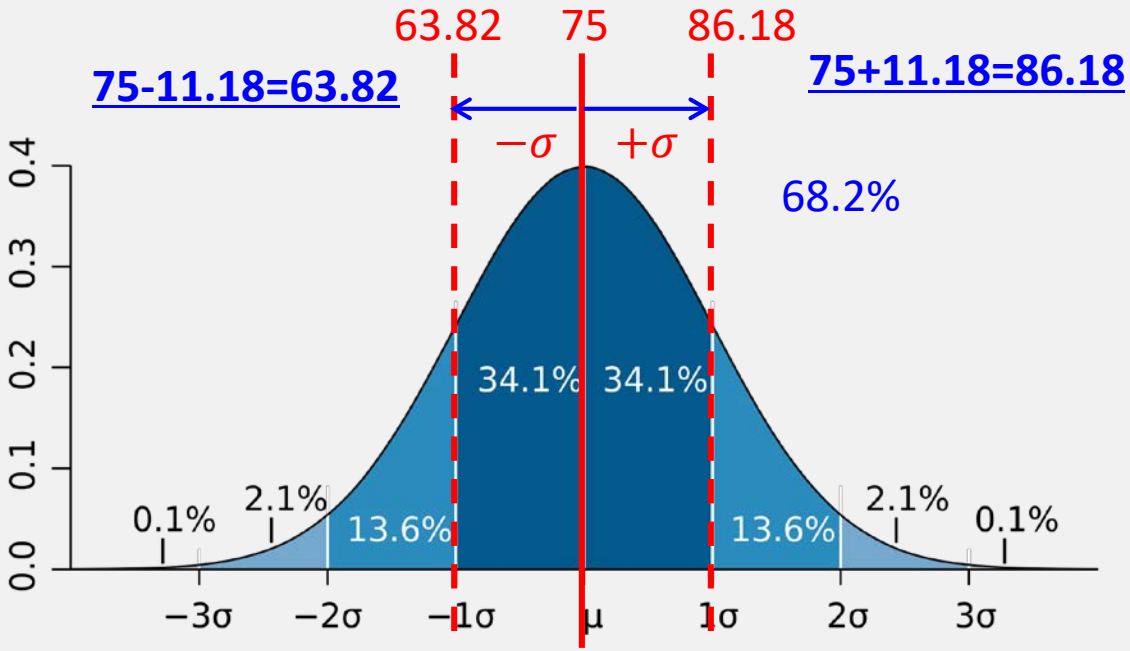
```
import numpy as np
import matplotlib.pyplot as plt
mid=np.array([80, 70, 60, 90])
avg=mid.mean()
var=mid.var()
sd=np.sqrt(var)
var1=((mid[0]-avg)**2+(mid[1]-avg)**2+(mid[2]-avg)**2+(mid[3]-avg)**2)/4
np.sqrt(mid.var())
np.sqrt(var1)
```

Distribution (분포)의 특징은 평균과 편차

$80, 70, 60, 90\ [v]: voltage$

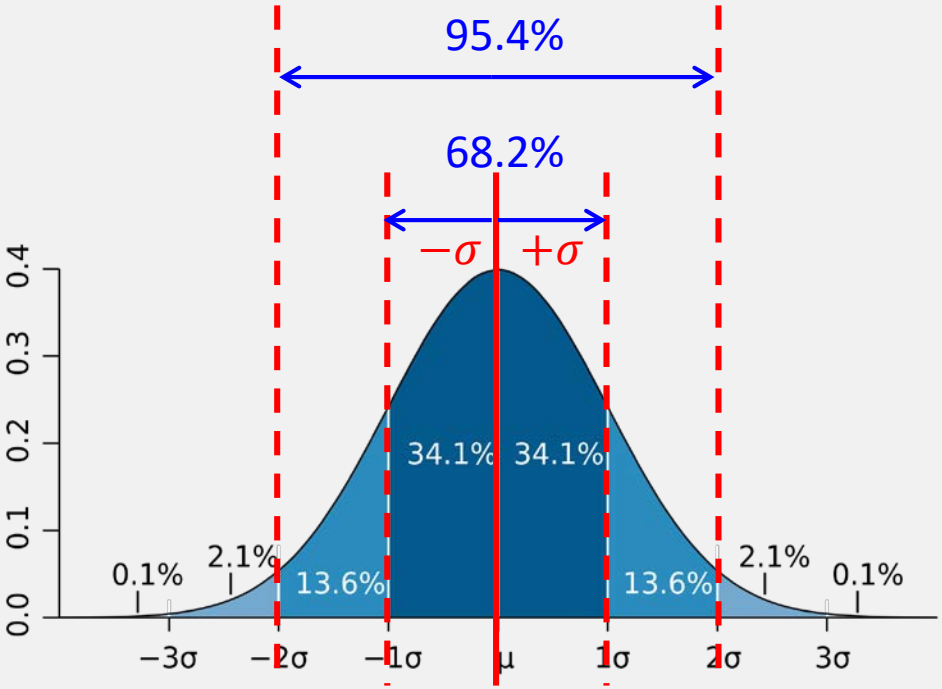
$$Avg = \mu = \frac{80 + 70 + 60 + 90}{4} = \mathbf{75}\ [\boldsymbol{v}]$$

$$deviation = \sigma = \sqrt{variance} = \sqrt{125} = \mathbf{11.18}[\boldsymbol{v}]$$



Normal Distribution (정규 분포): $N(\mu, \sigma)$

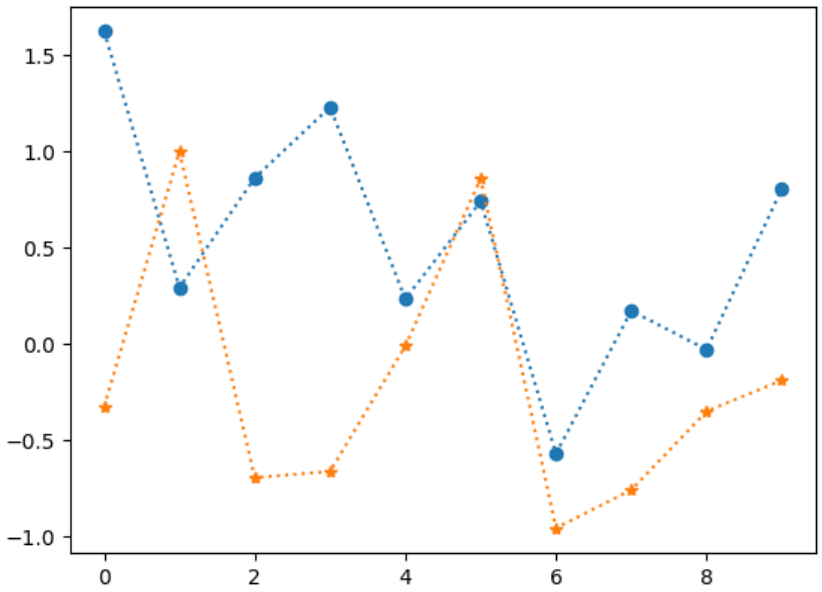
- 정규분포는 2개의 매개 변수 평균 μ 와 표준편차 σ 에 대해 모양이 결정되고, 이때의 분포를 $N(\mu, \sigma^2)$로 표기한다.
- 특히, 평균이 0이고 표준편차가 1인 정규분포 $N(0,1)$ 을 표준 정규 분포(standard normal distribution)라고 한다.



정규분포 (Randn())와 균등분포 (Uniform())

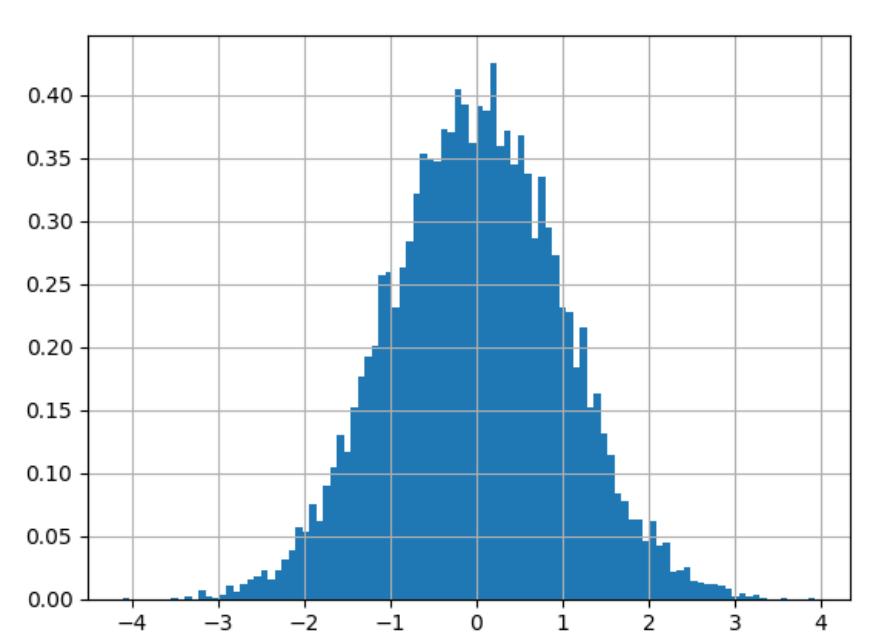
- 정규분포는 1이상의 값과 -1이하의 값이 발생된다.
- 균등분포는 1과 -1 사이에서만 값이 발생된다.

```
plt.figure(1)
n1=np.random.randn(10)
u=np.random.uniform(-1, 1, 10)
plt.plot(n1,'o:')
plt.plot(u, '*:')
```



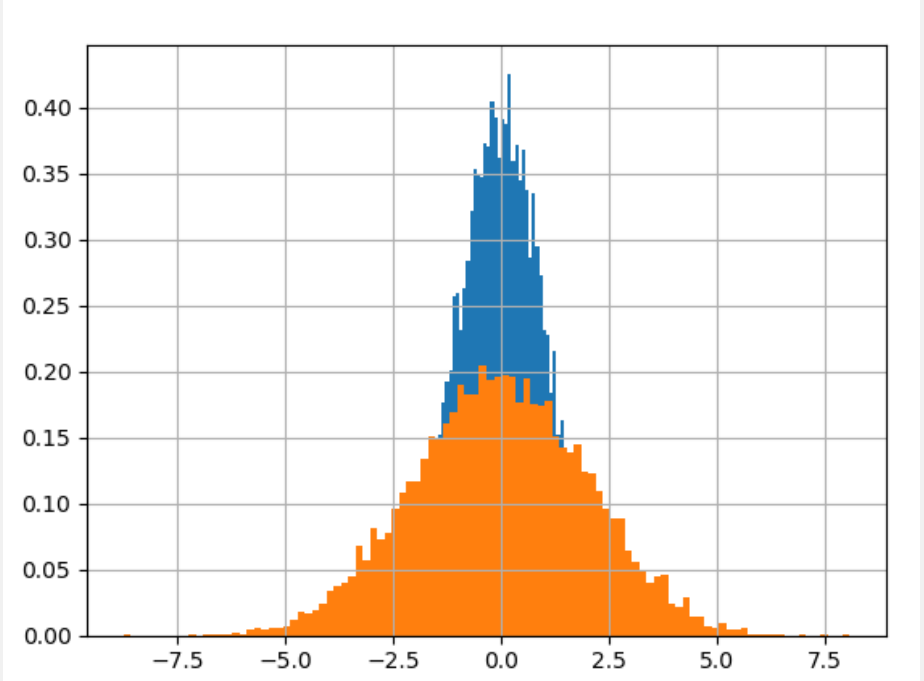
정규분포 ($N(0,1)$) 데이터 시각화

- `n1=np.random.randn(10000)`
  `c1=np.size(np.where((n1>=-1) & (n1<=1) ))`
  - 평균이 0이고, 편차가 1 인 정규 분포
  - [-1, 1]@68%
- plt.hist(n1, 100, normed=1)
  - 빈의 수가 100개인 막대그래프
  - **파란 색 분포**



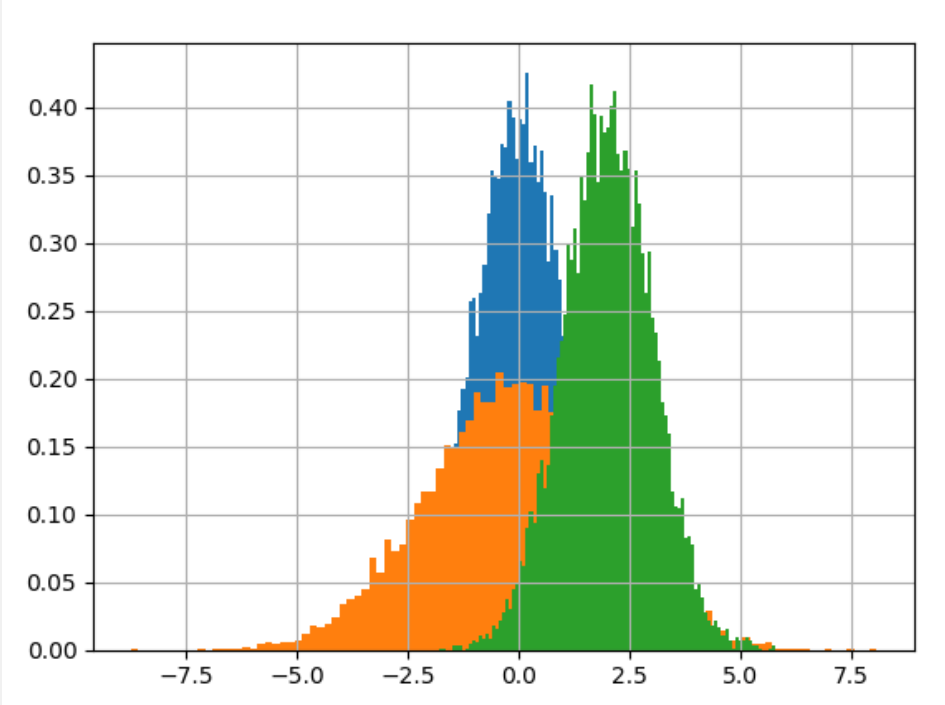
정규분포 ($N(0,2)$) 데이터 시각화

- `n2=np.random.randn(10000)*2`
  `c2=np.size(np.where((n2>=-2) & (n2<=2) ))`
  - 평균이 0이고, 편차가 2인 정규 분포
  - [-2, 2]@68%
- plt.hist(n2, 100, normed=1)
  - 빈의 수가 100개인 막대그래프
  - **주황 색 분포**
- **$N(0,2)$의 폭 비교**
  - **$N(0,1)$의 폭보다 더 넓다.**
- **$N(0,2)$의 높이 비교**
  - **$N(0,1)$의 높이보다 더 낮다.**



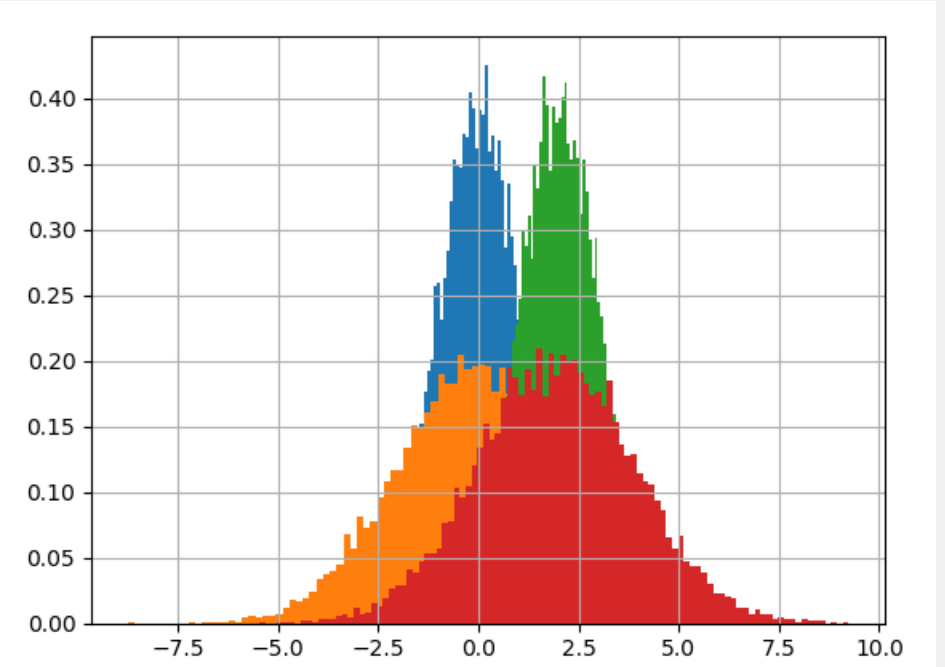
정규분포 $(N(0,1))$+2 데이터 시각화

- `n3=np.random.randn(10000)+2`
  `c3=np.size(np.where((n3>=1) & (n3<=3) ))`
  - 평균이 2이고, 편차가 1인 정규 분포
  - [1, 3]@68%
- plt.hist(n3, 100, normed=1)
  - 빈의 수가 100개인 막대그래프
- **$N(0,1)$ +2 의 평균 이동**
  - **평균 2 만큼 우측 이동 .**
  - **초록 색 분포**



정규분포 2*(N(0,1))+2 데이터 시각화

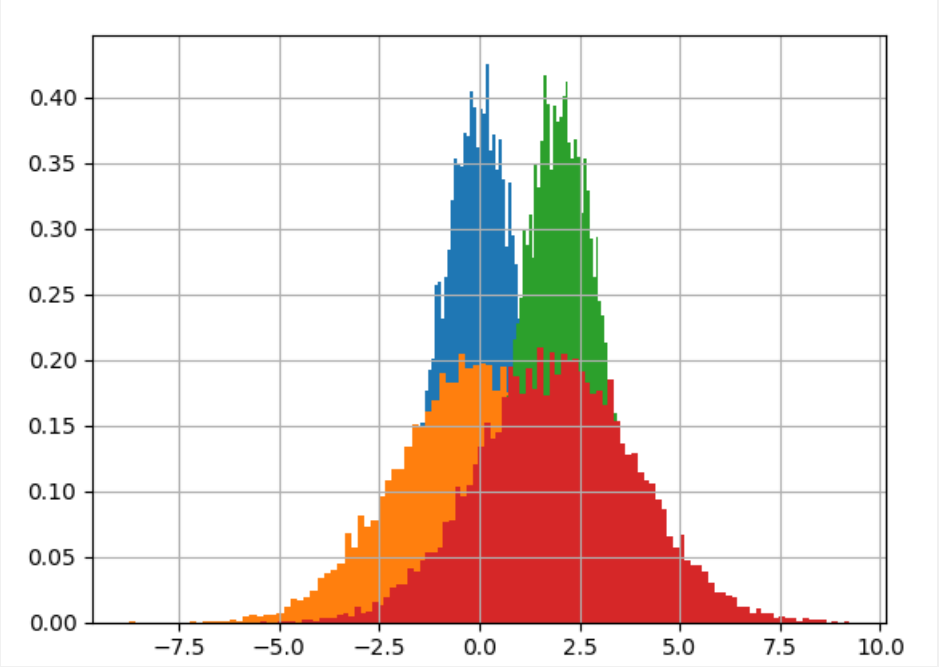
- n4=2*np.random.randn(10000)+2
  c4=np.size(np.where((n4>=0) & (n4<=4)))
  - 평균이 2이고, 편차가 2인 정규 분포 사이인
  - [0, 4]@68%
- plt.hist(n4, 100, normed=1)
  - 빈의 수가 100개인 막대그래프
- **$2*N(0,1)$ +2 의 평균 이동**
  - **평균 2 만큼 우측 이동 .**
  - **편차는 2**
  - **붉은 색 분포**



정규분포 데이터 시각화

- [-1, 1] @ 68%
- ([-1, 1] @ 68%)*2= [-2, 2] @ 68%
- ([-1, 1] @ 68%)+2= [-1, 3] @ 68%
- ([-1, 1] @ 68%)*2+2= ([-2, 2] @ 68%)+2
  = ([0, 4] @ 68%)

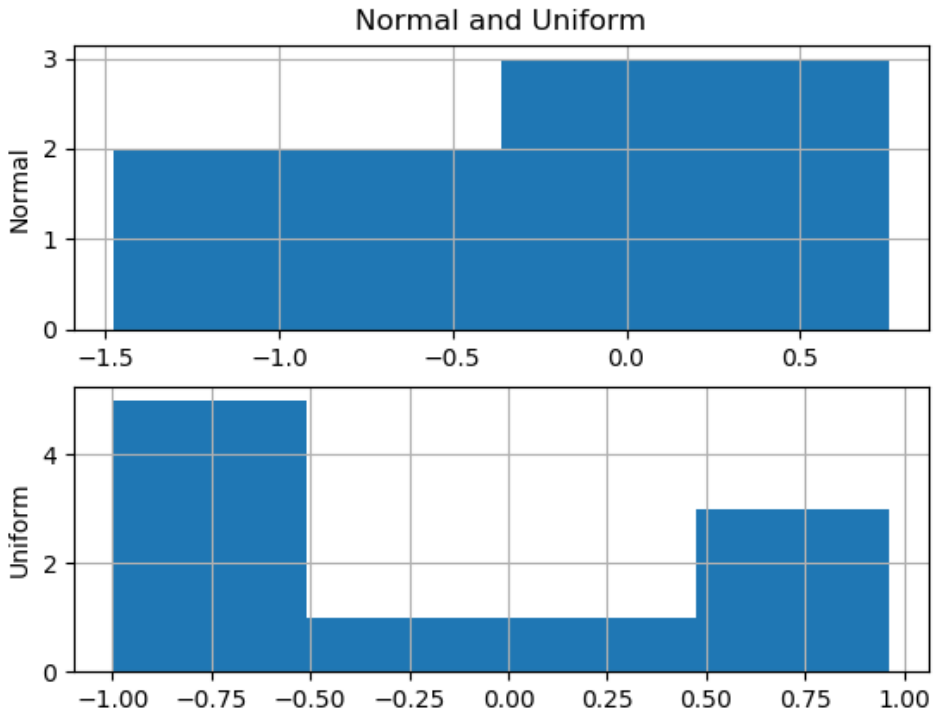
```
n1=np.random.randn(10000)
c1=np.size(np.where((n1>=-1) & (n1<=1) ))
n2=np.random.randn(10000)*2
c2=np.size(np.where((n2>=-2) & (n2<=2) ))
n3=np.random.randn(10000)+2
c3=np.size(np.where((n3>=1) & (n3<=3) ))
n4=2*np.random.randn(10000)+2
c4=np.size(np.where((n4>=0) & (n4<=4) ))

plt.hist(n1, 100, normed=1)
plt.hist(n2, 100, normed=1)
plt.hist(n3, 100, normed=1)
plt.hist(n4, 100, normed=1)
```



히스토그램 데이터 시각화

- plt.hist() : plt에서 제공하는 histogram 메소드
- np.histogram() : np에서 제공하는 histogram 메소드
- 정규분포와 균등분포의 Histogram 그리기

```
import numpy as np
import matplotlib.pyplot as plt
nd=np.random.randn(10)
ud=np.random.uniform(-1, 1, 10)
plt.figure(1)
plt.subplot(2, 1, 1)
plt.hist(nd, 4)
plt.title('Normal and Uniform')
plt.ylabel('Normal')
plt.grid()
plt.subplot(2, 1, 2)
plt.hist(ud, 4)
plt.ylabel('Uniform')
plt.grid()
plt.show()
```



plt.hist() 와 np.histogram()

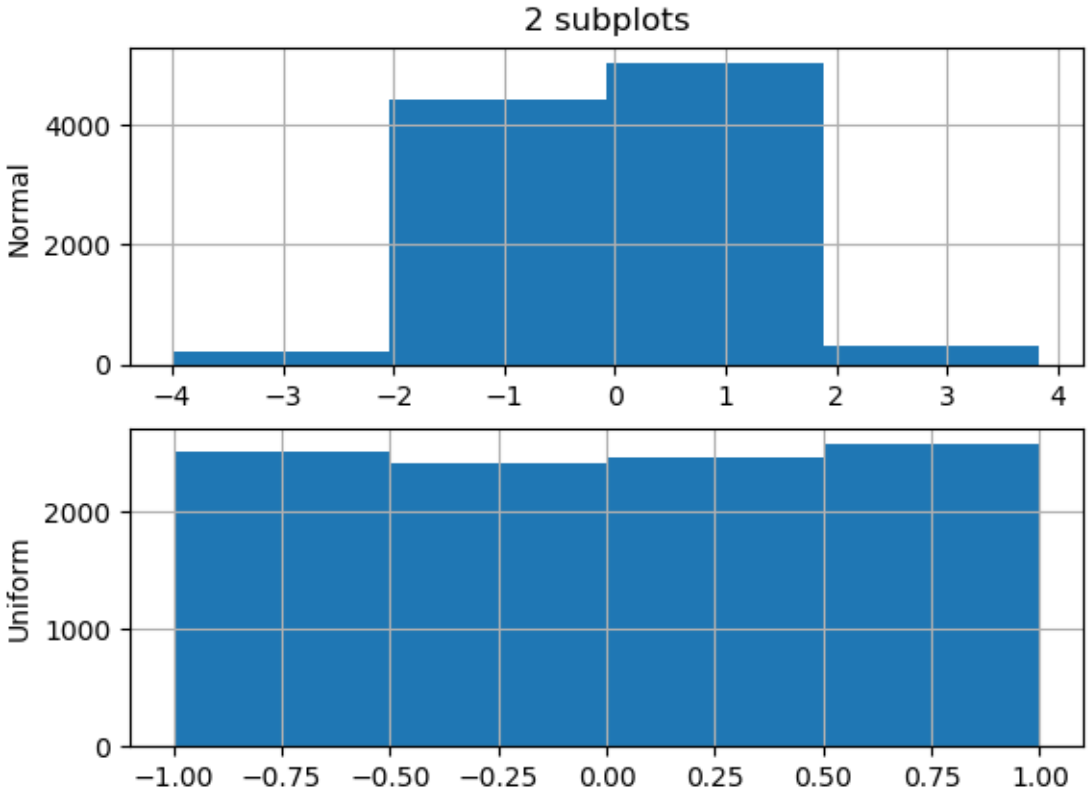
- plt.hist() 와 np.histogram() 의 공통점은 두 메서드 모두 리턴 값으로 hist 높이와 bin_left를 리턴해준다.
- 다른 점은 plt.hist() 는 실제로 히스토그램을 시각화한다.

```
import numpy as np
import matplotlib.pyplot as plt
nd=np.random.randn(10)
ud=np.random.uniform(-1, 1, 10)

count, bin_left=np.histogram(nd, 4)
count1, bin_left1=np.histogram(nd, 4)

count
array([3, 5, 0, 2], dtype=int64)
bin_left
array([-0.93461143, -0.17046795,  0.59367554,  1.35781902,  2.1219625 ])
count1
array([3, 5, 0, 2], dtype=int64)
bin_left1
array([-0.93461143, -0.17046795,  0.59367554,  1.35781902,  2.1219625 ])
```

plt.hist(): Histogram

- plt.hist() 메소드로 histogram 그래프를 시각화할 수 있다

```
nd=np.random.randn(10000)
ud=np.random.uniform(-1, 1, 10000)
plt.figure(1)
plt.subplot(2, 1, 1)
plt.hist(nd, 4)
plt.title('2 subplots')
plt.ylabel('Normal')
plt.grid()
plt.subplot(2, 1, 2)
plt.hist(ud, 4)
plt.ylabel('Uniform')
plt.grid()
plt.show()
```



plt.bar() 메소드를 이용한 히스토그램 시각화

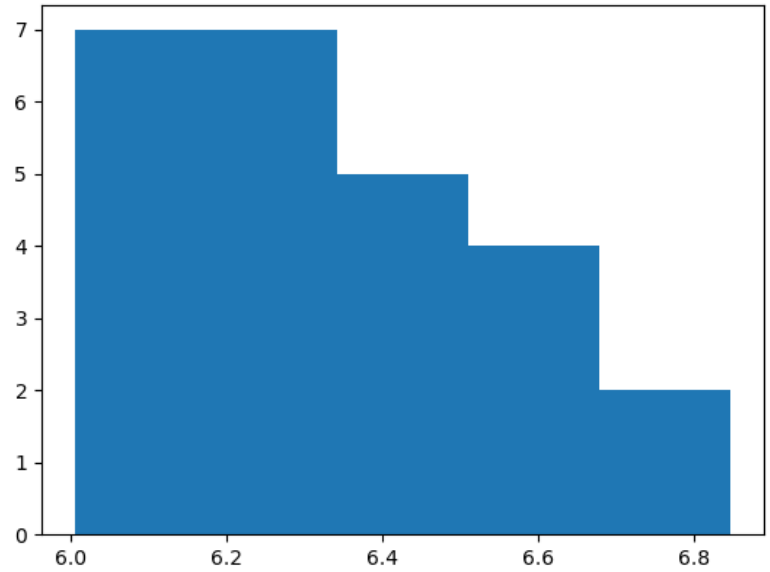
[11주/2차시 학습내용]: plt.bar () 메소드를 이용하여 히스토그램을 시각화하여 보자 .

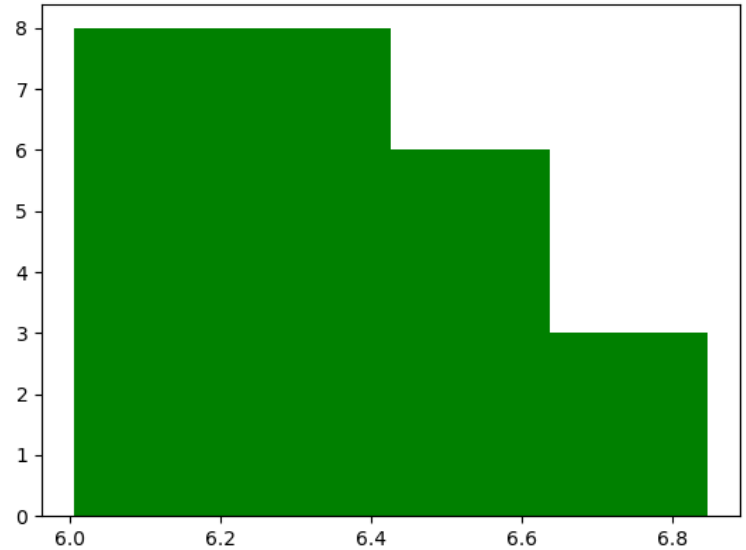
국민대학교 소프트웨어학부

plt.bar() 메소드를 이용한 히스토그램 시각화

- plt.bar() 메소드를 이용한 히스토그램 그리기
- bin 수를 5개, 4개 다르게 해서 히스토그램 그리기

```
s=np.random.uniform(6,7,25)

plt.figure(4), plt.hist(s,5)
plt.figure(5), plt.hist(s,4, color='g')
```





bin Height 계산: np.where()

- 히스토그램은 각 bin의 높이부터 계산하여야 한다
- np.where() 함수를 이용하여 bin 저장소의 높이를 계산할 수 있다

```
smin=s.min()
smax=s.max()
width=smax-smin
bin_width=width/4
first_height=np.size(np.where((s>=smin) & (s< (smin+bin_width)
)))
second_height=np.size(np.where((s>=(smin+bin_width)) & (s<
(smin+bin_width*2) )))
third_height=np.size(np.where((s>=(smin+bin_width*2)) & (s<
(smin+bin_width*3) )))
fourth_height=np.size(np.where((s>=(smin+bin_width*3)) & (s<=
(smin+bin_width*4) )))
```

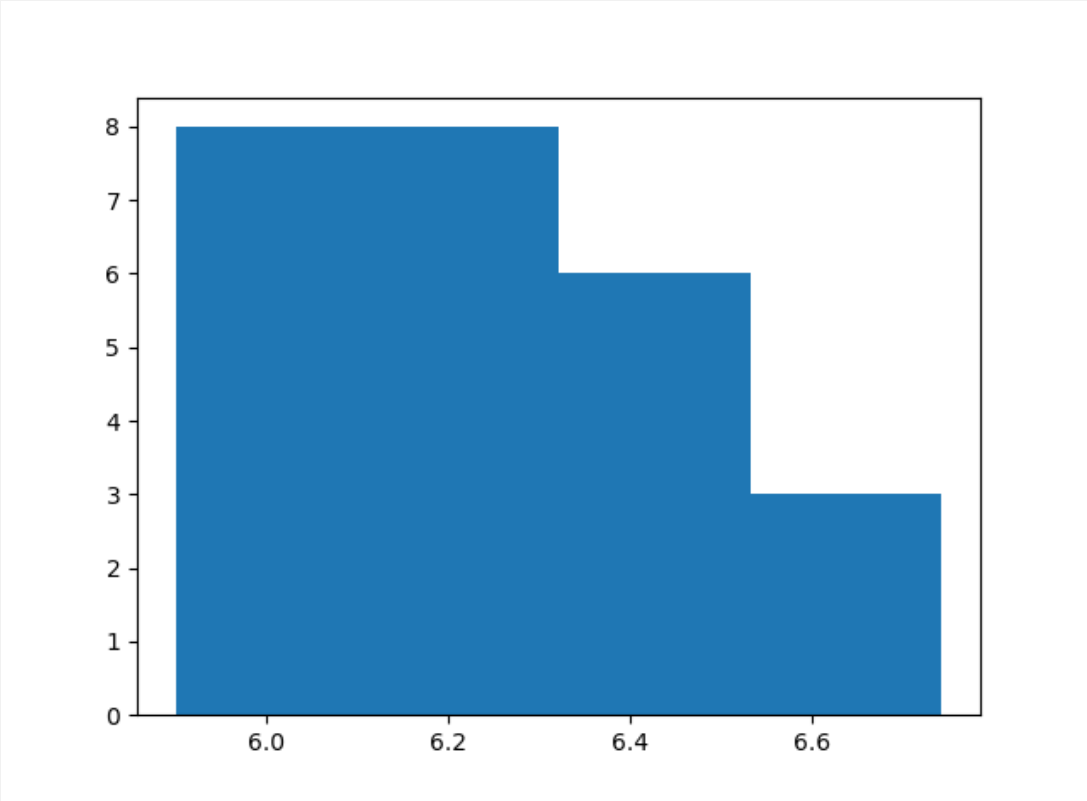
bin 시작점 Left 계산

- 히스토그램은 다음 단계로 각 bin 저장소의 좌측 시작점과 우측 시작점을 파악한다.
- bin 저장소의 좌측 시작점을 계산한다

```
left=np.array([ smin, (smin+bin_width) , (smin+bin_width*2),
(smin+bin_width*3)])
height=np.array([first_height, second_height, third_height,
fourth_height])
plt.figure(6)
plt.plot(left, height, 'ro')
plt.figure(7)
plt.bar(left, height, bin_width)
plt.figure(8)
plt.bar(left, height, bin_width-bin_width/10)
```

bin 너비의 100% 시각화

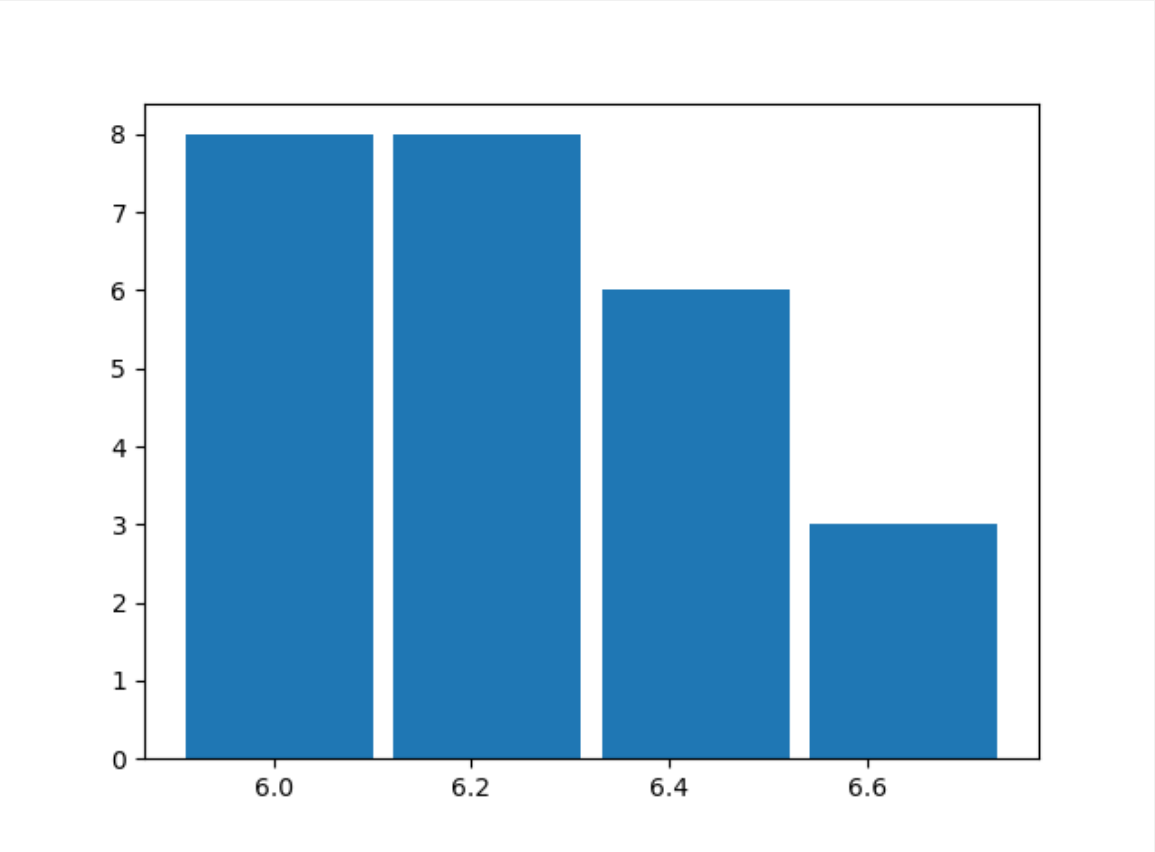
- bin의 left와 bin의 높이를 계산하였으면, plt.bar() 메소드를 이용해서 히스토그램을 시각화할 수 있다.
- bin 너비의 100% 그리기 `plt.bar(left, height, bin_width)`



bin 너비의 90%만 시각화

- bin 너비의 90% 시각화

```
plt.bar(left, height, bin_width-bin_width/10)
```

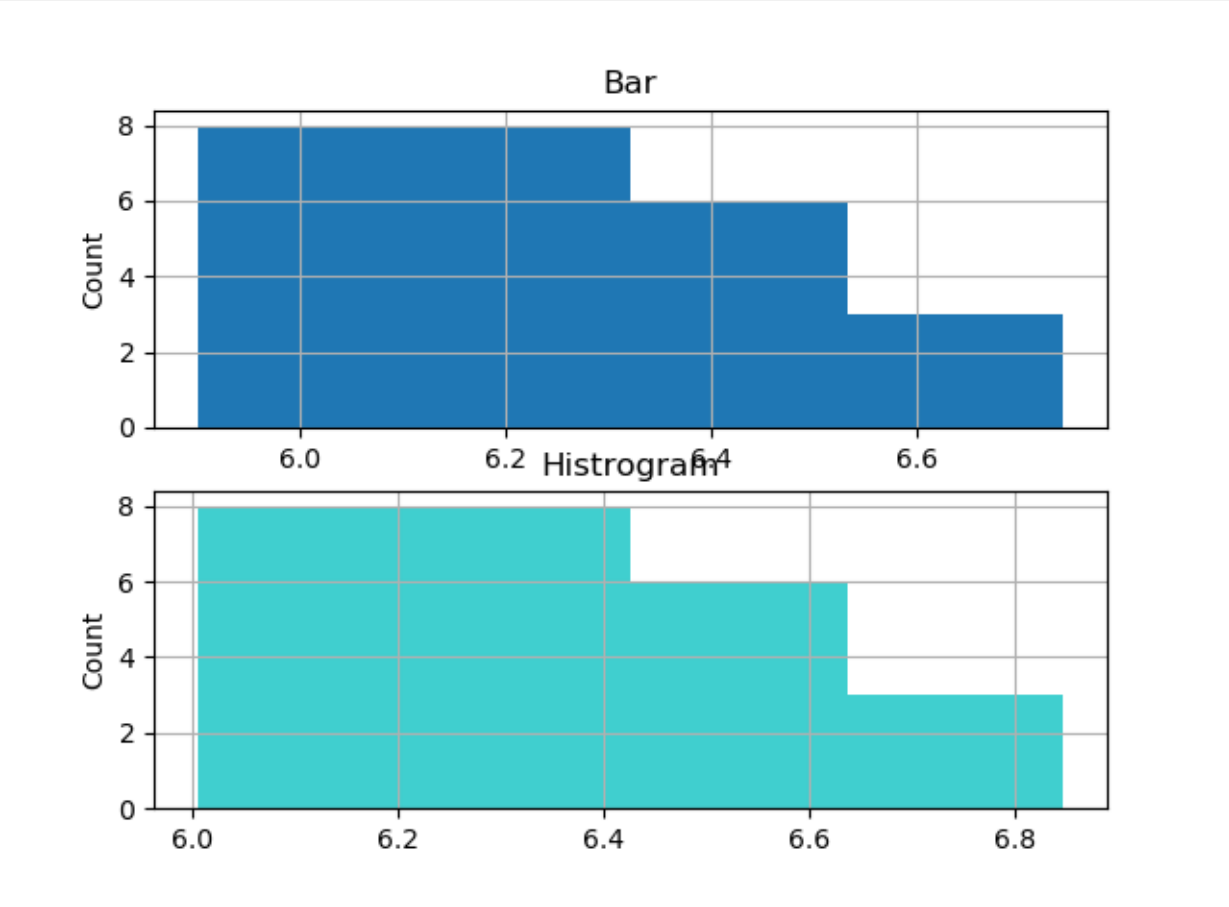


plt.bar()와 plt.hist()를 이용한 히스토그램

- plt.bar()와 plt.hist()를 이용한 히스토그램

```
plt.figure(10)
plt.subplot(2, 1, 1)
plt.bar(left, height, bin_width)
plt.title('Bar')
plt.ylabel('Count')
plt.grid()
plt.subplot(2, 1, 2)
plt.hist(s, bins=4, color='c', alpha=0.75)
plt.title('Histrogram')
plt.ylabel('Count')
plt.grid()
```

plt.bar()와 plt.hist()로 그린 히스토그램



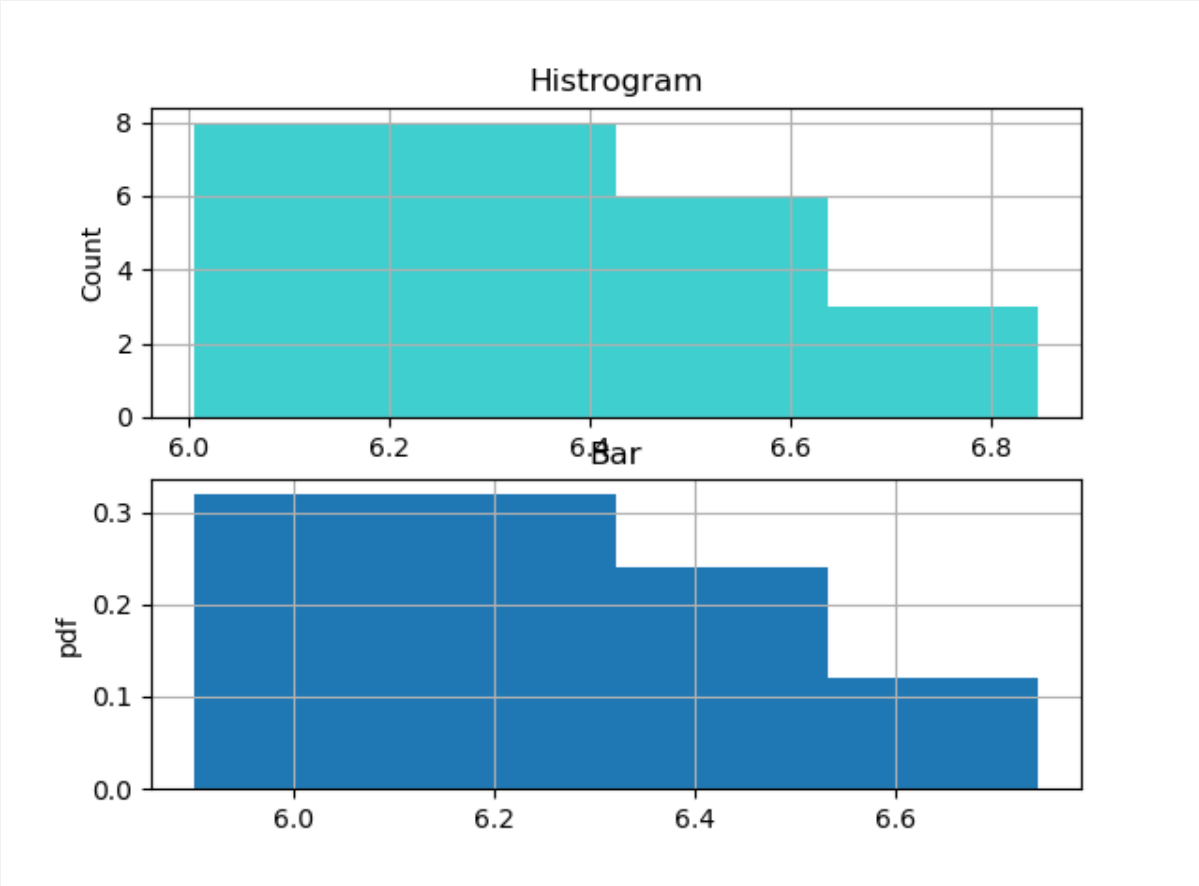
Probability Density Function (확률밀도함수)

- 확률밀도함수는 히스토그램의 높이를 전체 개수로 나눈 값을 의미한다.
- pdf=height/np.size(s)

```
pdf=height/np.size(s)

plt.figure(11)
plt.subplot(2, 1, 1)
plt.hist(s, bins=4, color='c', alpha=0.75)
plt.title('Histrogram')
plt.ylabel('Count')
plt.grid()
plt.subplot(2, 1, 2)
plt.bar(left, pdf, bin_width)
plt.title('Bar')
plt.ylabel('pdf')
plt.grid()
```

Probability Density Function (확률밀도함수)

- 히스토그램의 높이와 확률밀도함수의 확률은 서로 연관되어 있다.

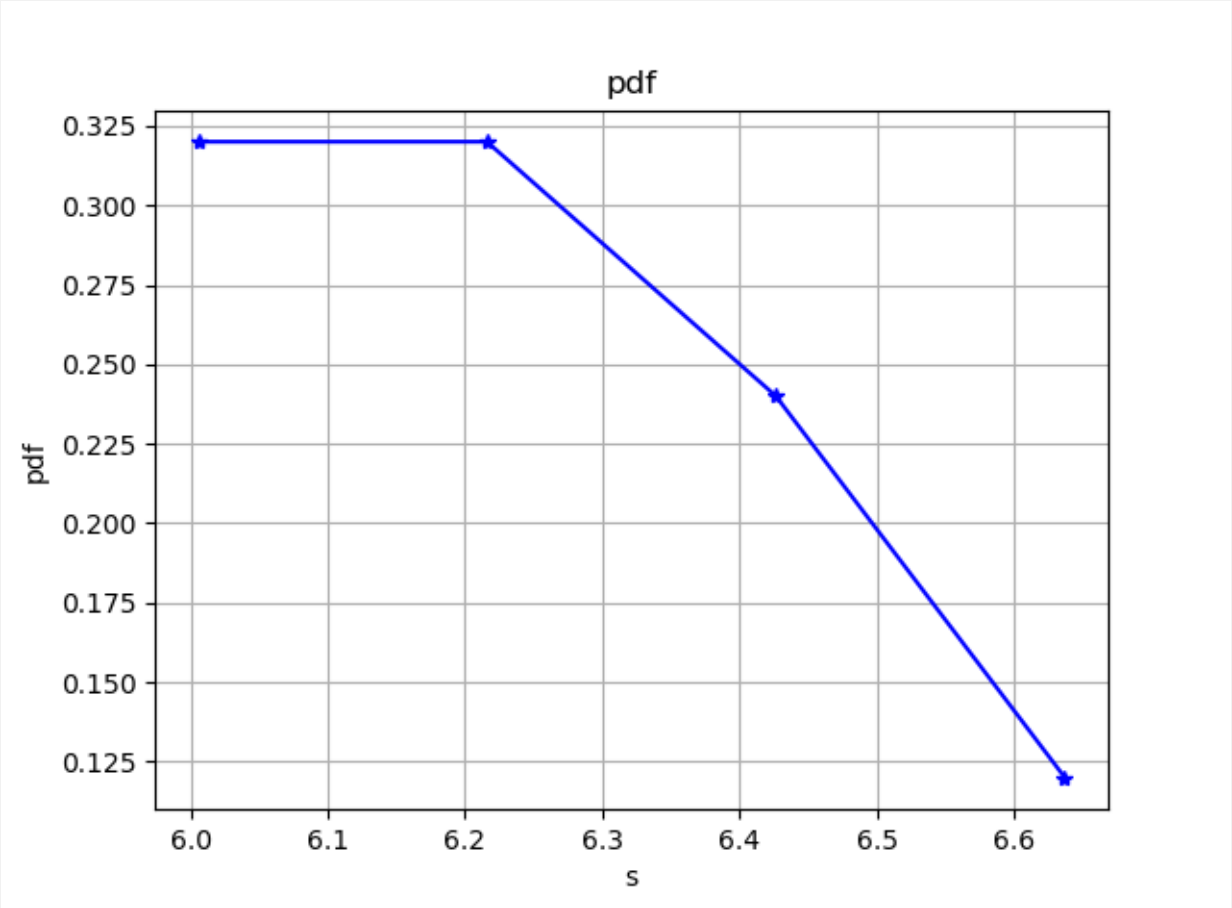


Probability Density Function (확률밀도함수)

- 확률밀도함수를 np.plot으로 그려 보기

```
plt.figure(12)
plt.plot(left, pdf, 'b*-')
plt.title('pdf')
plt.xlabel('s')
plt.ylabel('pdf')
plt.grid()
plt.show()
```

Probability Density Function



Cumulative Density Function (누적 분포함수)

- 누적 분포 함수(cumulative distribution function, cdf)는 어떤 확률 분포에 대해서, 확률 변수가 특정 값보다 작거나 같은 확률을 나타낸다.
- 수식으로 나타내면, 실수 범위를 가지는 확률 변수 X 의 누적 분포 함수는
- $F_X(x) = P(X \leq x)$

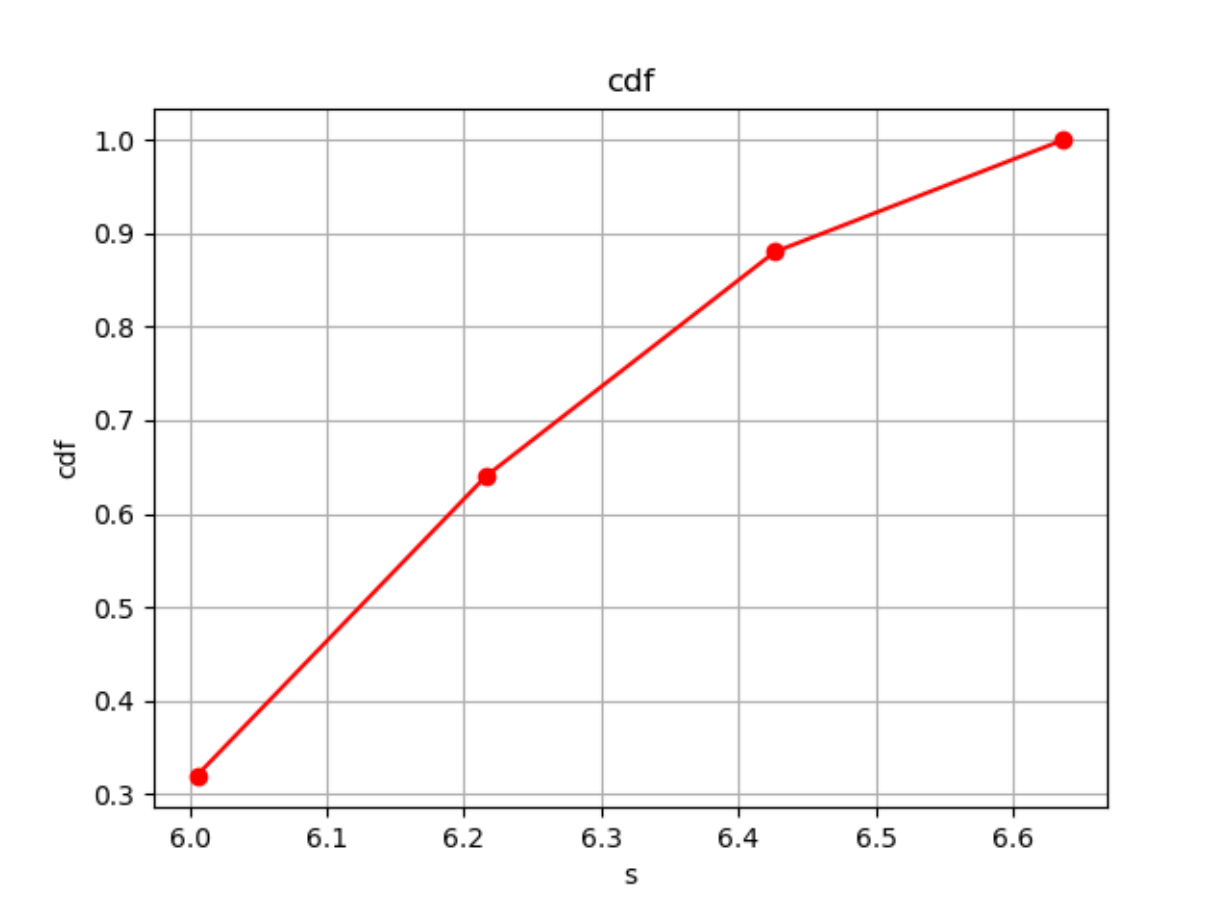
```
pdf=hist/np.size(nd)
cdf=np.cumsum(pdf)
```

Cumulative Density Function (누적 분포함수)

- 누적 분포함수를 np.plot으로 그려 보기

```
cdf=np.cumsum(pdf)
plt.figure(13)
#plt.subplot(1, 2, 2)
plt.plot(left, cdf, 'ro-')
plt.title('cdf')
plt.xlabel('s')
plt.ylabel('cdf')
plt.grid()
plt.show()
```

Cumulative Density Function (누적 분포함수)



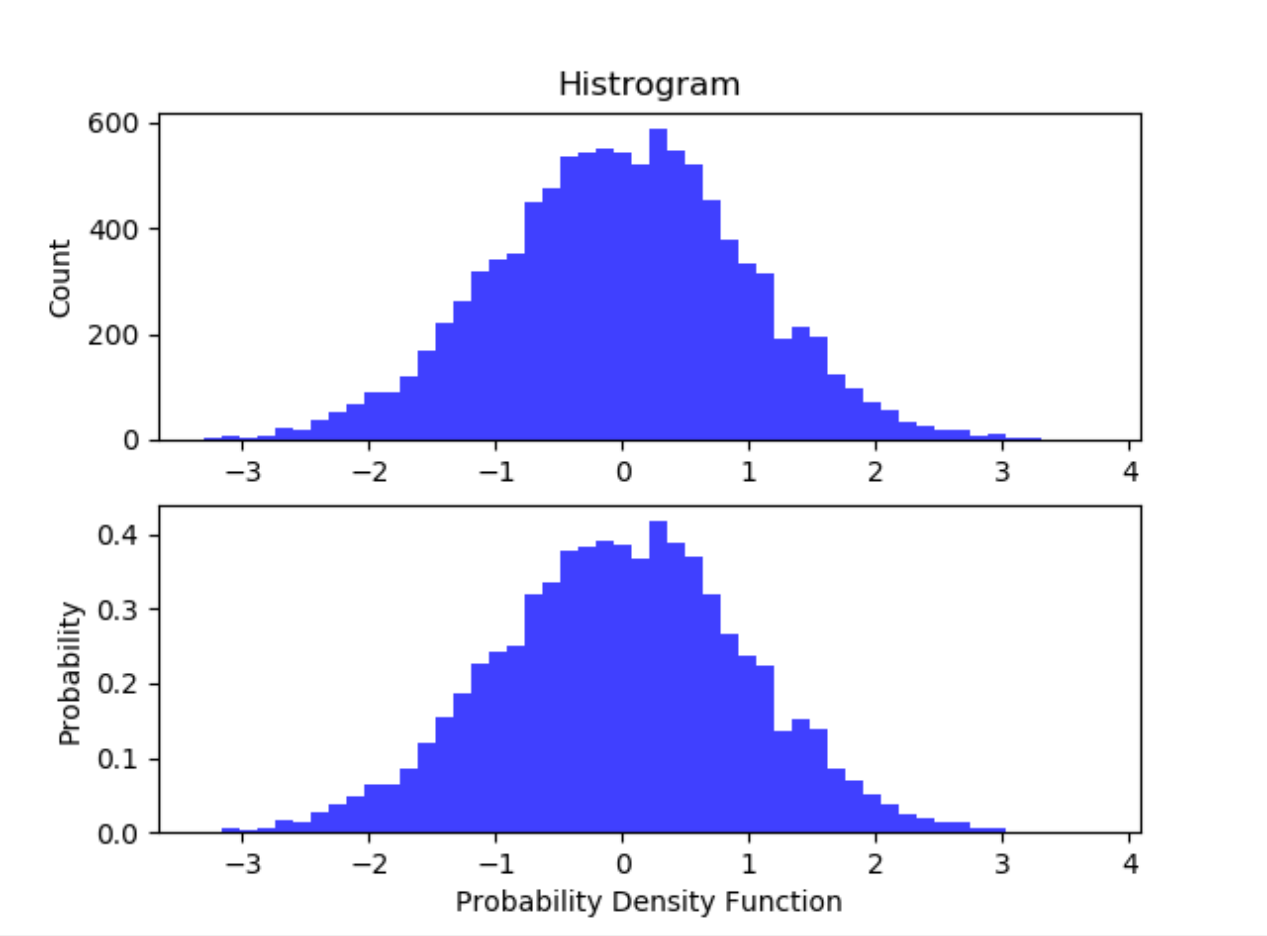
Probability Density Function (확률 밀도 함수)

- Probability Density Function

```
x3=np.random.randn(10000)

plt.figure(20)
plt.subplot(2, 1, 1)
n3, bins3, patches3 = plt.hist(x3, bins=50, color='b', alpha=0.75)
plt.title('Histrogram')
plt.ylabel('Count')

plt.subplot(2, 1, 2)
n4, bins4, patches4 = plt.hist(x3, 50, normed=1, facecolor='b',
alpha=0.75)
plt.xlabel('Probability Density Function')
plt.ylabel('Probability')
```

Probability Density Function

- plt.hist() with normed=1

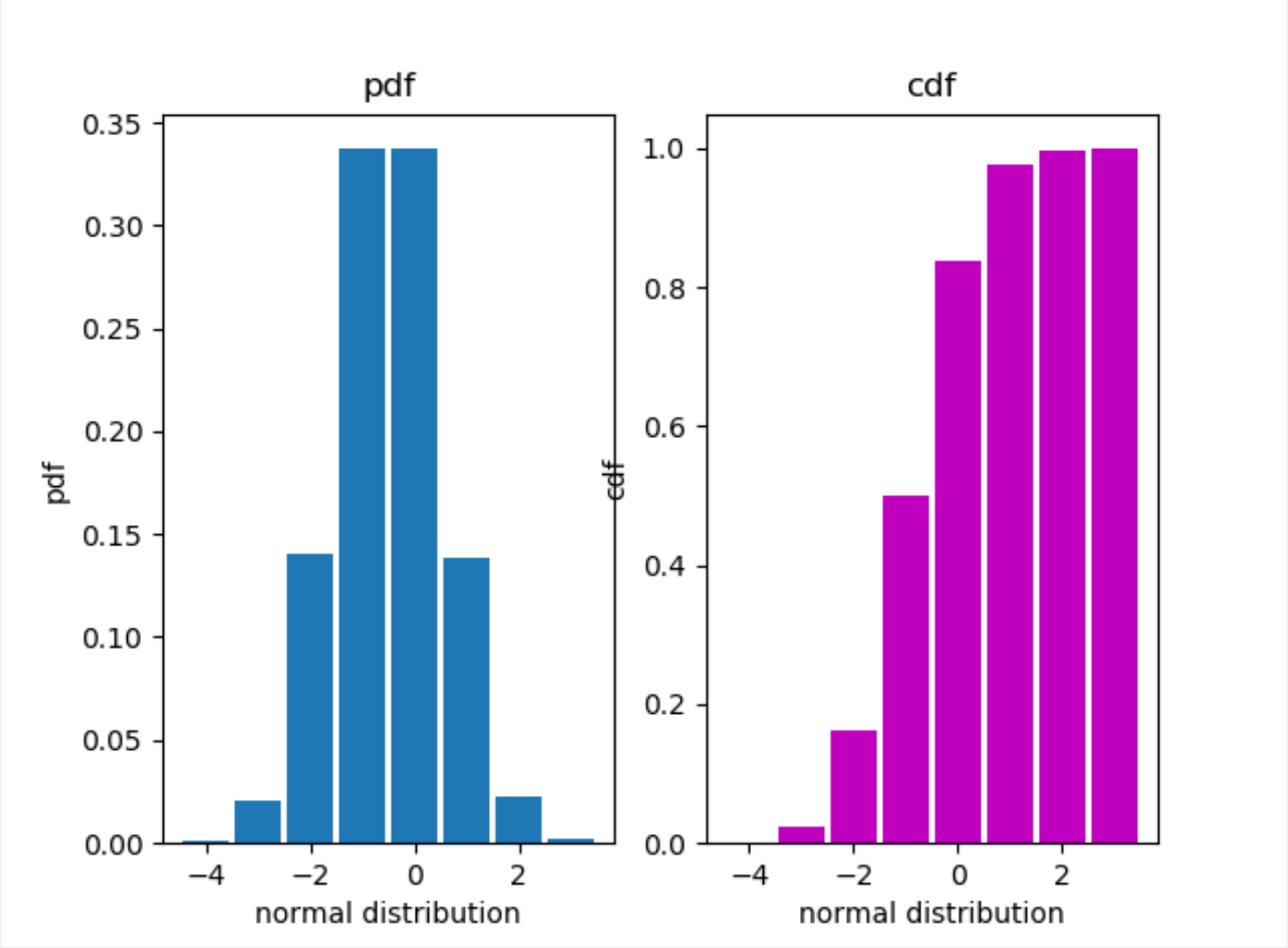


pdf와 cdf 그리기

```
nd=np.random.randn(10000)
ra=np.int(np.floor(np.min(nd)))
rb=np.int(np.ceil(np.max(nd)))

hist, bin_left=np.histogram(nd, bins=np.arange(ra, rb+1, 1))
bin_width=(rb-ra)/np.size(bin_left)

pdf=hist/np.size(nd)
cdf=np.cumsum(pdf)
plt.figure(100)
plt.subplot(1, 2, 1)
plt.bar(bin_edges[:-1], pdf, bin_width)
plt.title('pdf')
plt.xlabel('normal distribution')
plt.ylabel('pdf')
plt.show()
plt.subplot(1, 2, 2)
plt.bar(bin_edges[:-1], cdf, bin_width, facecolor='m')
plt.title('cdf')
plt.xlabel('normal distribution')
plt.ylabel('cdf')
plt.show()
```

Bar()특성상 n개-1 까지만

pdf와 cdf 그리기

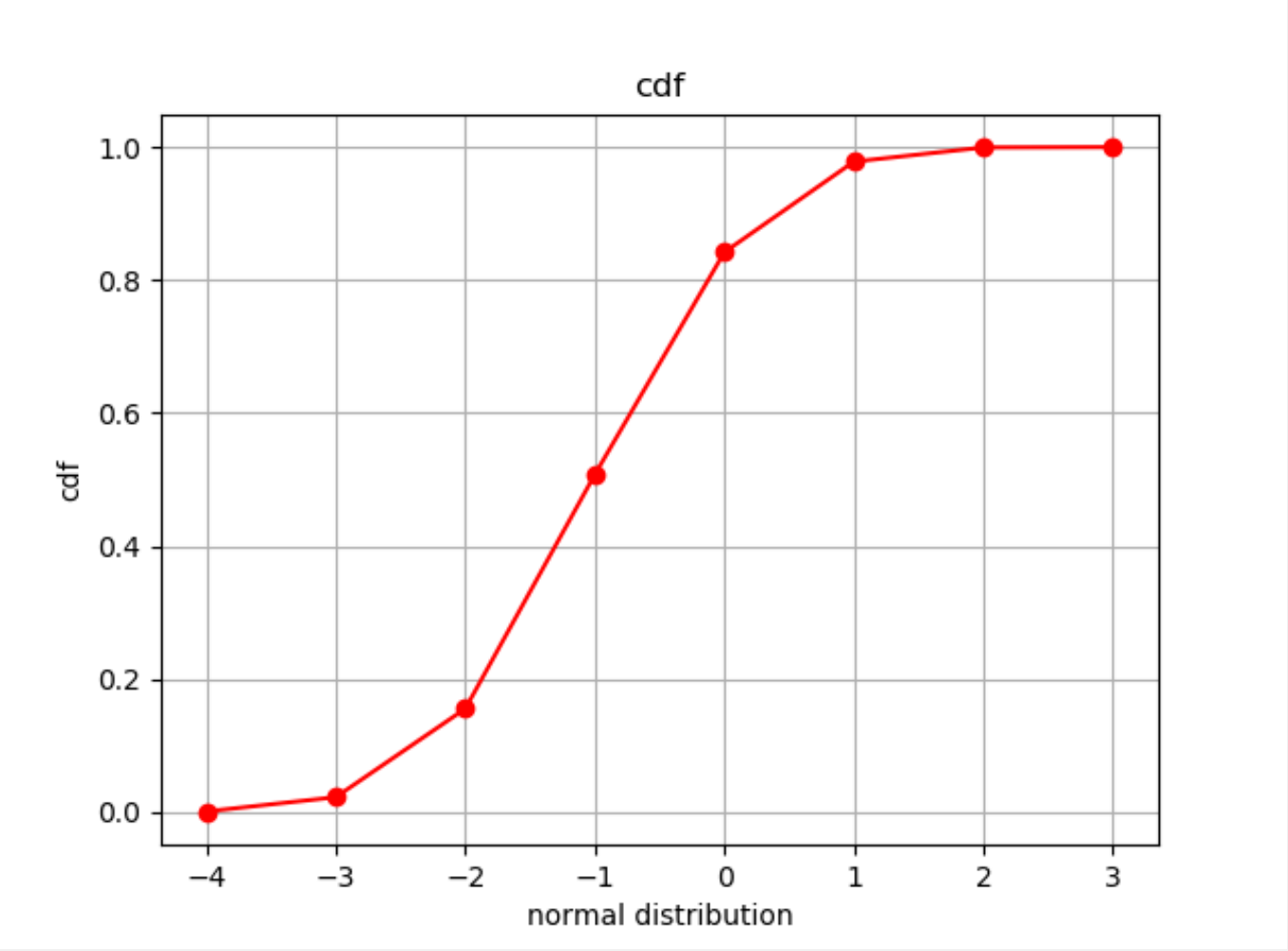

pdf
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
pdf
−4
−2
0
2
normal distribution
cdf
1.0
0.8
0.6
0.4
0.2
0.0
cdf
−4
−2
0
2
normal distribution

np.plot()으로 cdf 그리기

- np.plot()으로 cdf 그리기

```
plt.figure(103)
#plt.subplot(1, 2, 2)
plt.plot(bin_edges[:-1], cdf, 'ro-')
plt.title('cdf')
plt.xlabel('normal distribution')
plt.ylabel('cdf')
plt.grid()
plt.show()
```

np.plot()으로 cdf 그리기



np.semilopy()으로 cdf 그리기

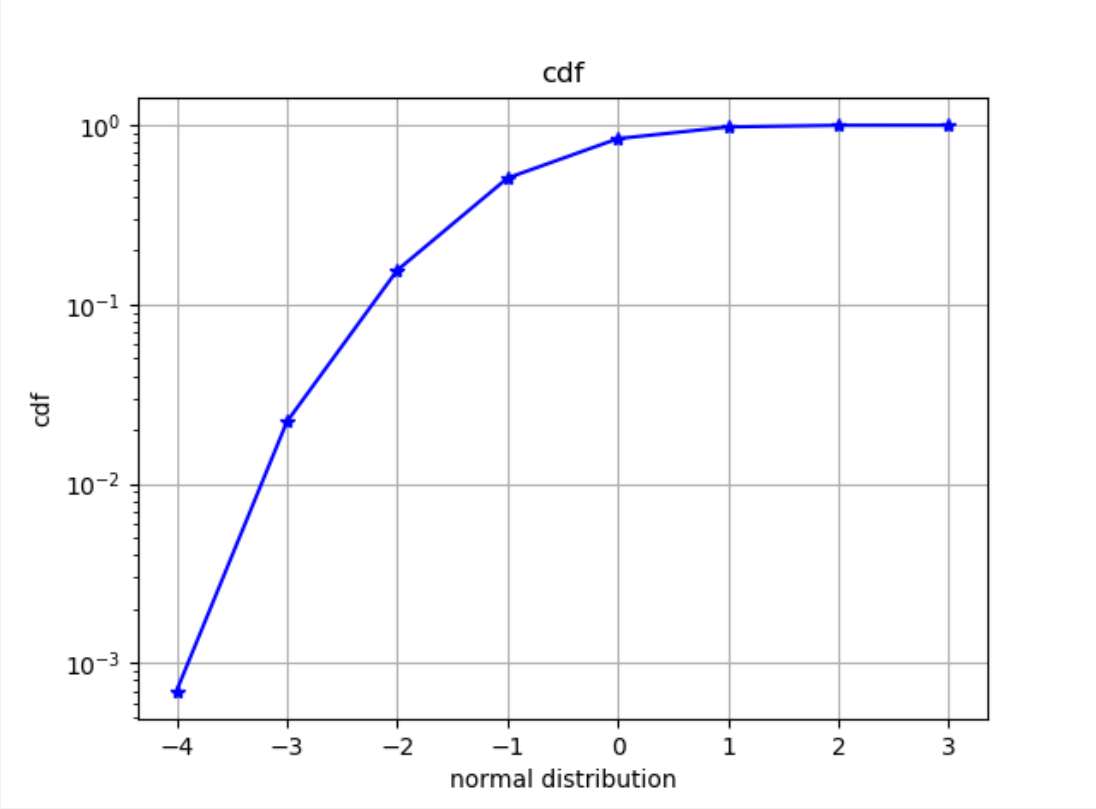
- np.semilogy()으로 pdf 그리기
- cdf 값은 0.1 (10^{-1}) 부터 0.01 (10^{-2}) , 0.001 (10^{-3}) , 0.0001 (10^{-4}) 이하의 아주 작은 수들을 가지고 있다.
- 이러한 아주 작은 숫자들을 그래프에 명확히 나타내고 싶을 때, y축 스케일을 로그 스케일로 나타낸다.

```
cdf
array([7.000e-04, 2.220e-02, 1.553e-01, 5.076e-01, 8.419e-01, 9.782e-01,
       9.995e-01, 1.000e+00])
```

```
plt.figure(104)
#plt.subplot(1, 2, 2)
plt.semilogy(bin_edges[:-1], cdf, 'b*-')
plt.title('cdf')
plt.xlabel('normal distribution')
plt.ylabel('cdf')
plt.grid()
plt.show()
```

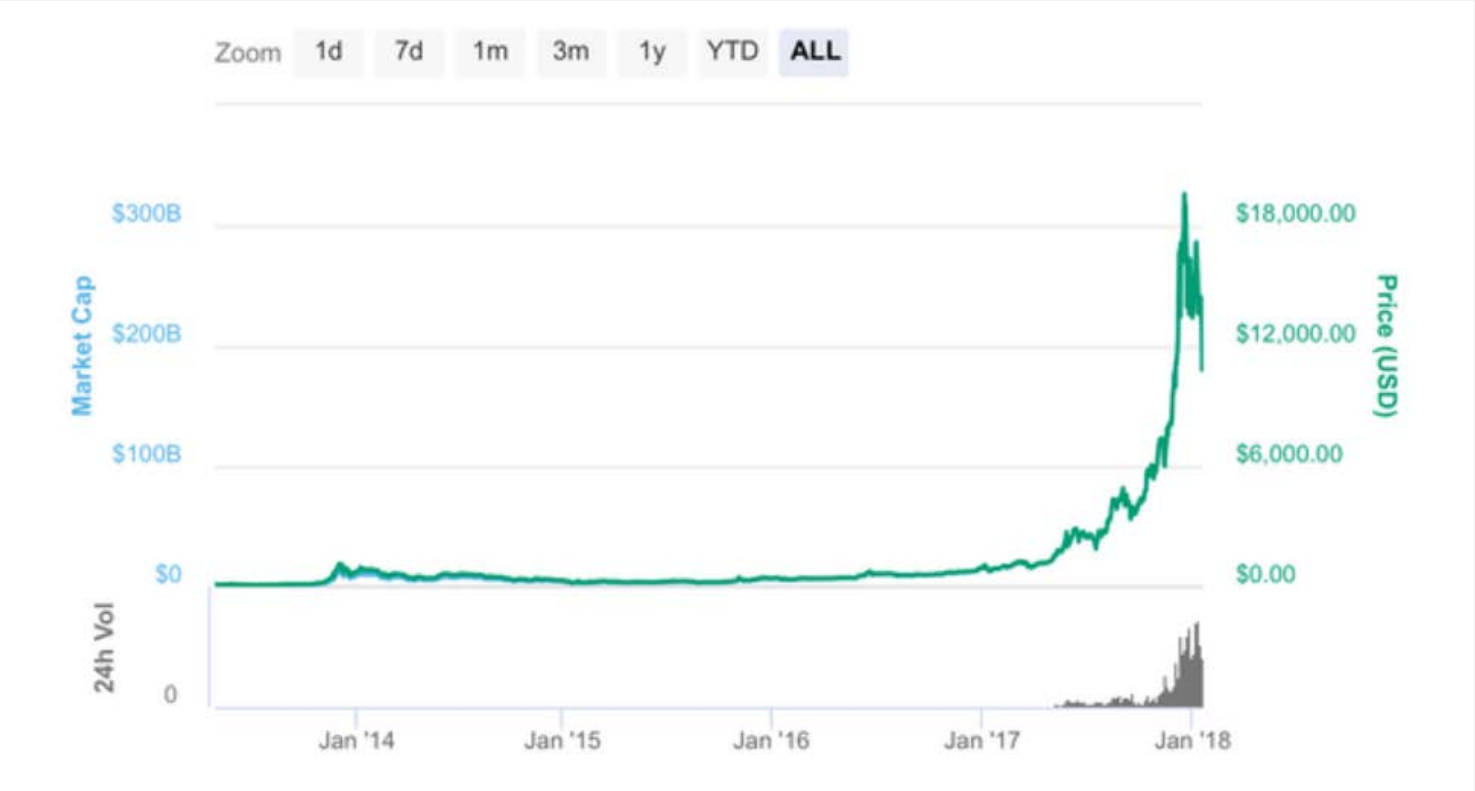
np.semilopy()으로 cdf 그리기

- semilogy는 y축에 대해 로그 스케일을 사용하여 데이터를 플로팅한다. semilogy(Y)는 y축에 대해 밑이 10인 로그 스케일을 사용하고 x축에 대해 선형 스케일을 사용하여 플롯을 만든다.



비트코인의 일반 스케일 그래프

- 장점: 2018년 1월에 300B$ 급등한 것을 알 수 있다.
- 단점: 2018년 1월까지 작은 1B$에서 10B$까지의 **세심한 변화를 보기 어렵다.**



비트코인의 로그 스케일 그래프

- 장점: 2018년 1월까지 작은 1B$에서 10B$까지의 세심한 변화를 볼 수 있다.
- 단점: 2018년 1월에 300B$ 급등한 부분을 쉽게 찾을 수 없다.

로그 스케일 그래프

- 일반 스케일은 10, 100, 1000의 간격으로 시각화를 하기 때문에 10과 1000사이의 차이가 두드러져 나게 된다.
- 로그 스케일은 10, 20, 30의 간격으로 시각화를 하기 때문에, 일반 스케일에서 나타나는 10과 1000사이의 차이만큼, 10과 30의 차이가 두드러져 나지 않는다.