In [221…
```
data = [1,2,3,4,5]

import pandas as pd
import numpy as np
import seaborn as sns
```

In [222…
```
np.mean(data)
```

Out[222… 3.0

In [223…
```
np.median(data)
```

Out[223… 3.0

In [224…
```
df =sns.load_dataset('tips')
```

In [225…
```
df
```

Out[225…

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [226…
```
np.mean(df['tip'])
```

Out[226… 2.99827868852459

In [227…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

In [228…
```
np.median(data)
```

Out[228… 3.0

In [229…
```
data
```

Out[229… [1, 2, 3, 4, 5]

In [230…
```
import statistics
np.median(data)
```

Out[230… 3.0

In [231…
```
df.describe()
```

Out[231…

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

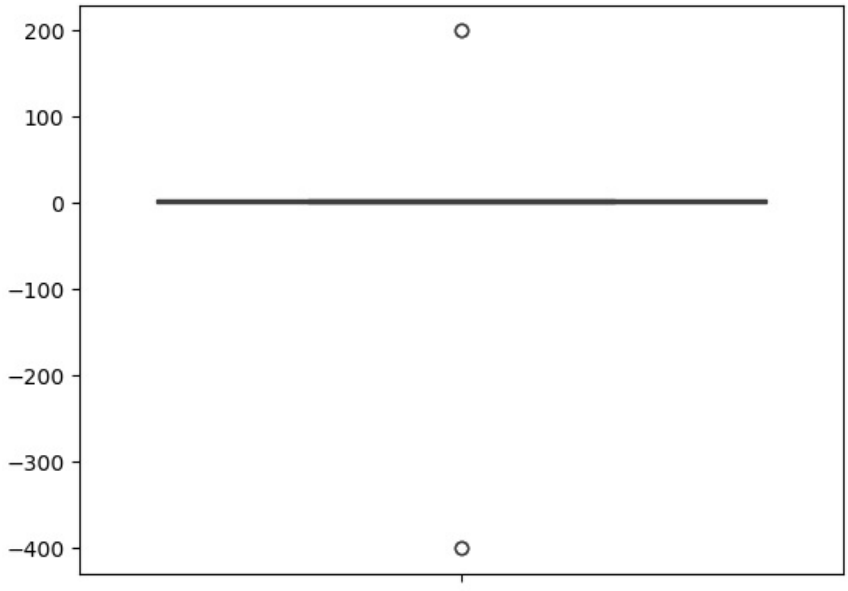
In [232…
```
data
```
Out[232… [1, 2, 3, 4, 5]

In [233…
```
np.percentile(data,[25,23,45])
```
Out[233… array([2. , 1.92, 2.8])

In [236…
```
data.append(-400)
data.append(+200)
```

In [240…
```
data
```
Out[240… [1, 2, 3, 4, 5, -400, 200, -400, 200]

In [241…
```
sns.boxplot(data)
```
Out[241… <Axes: >



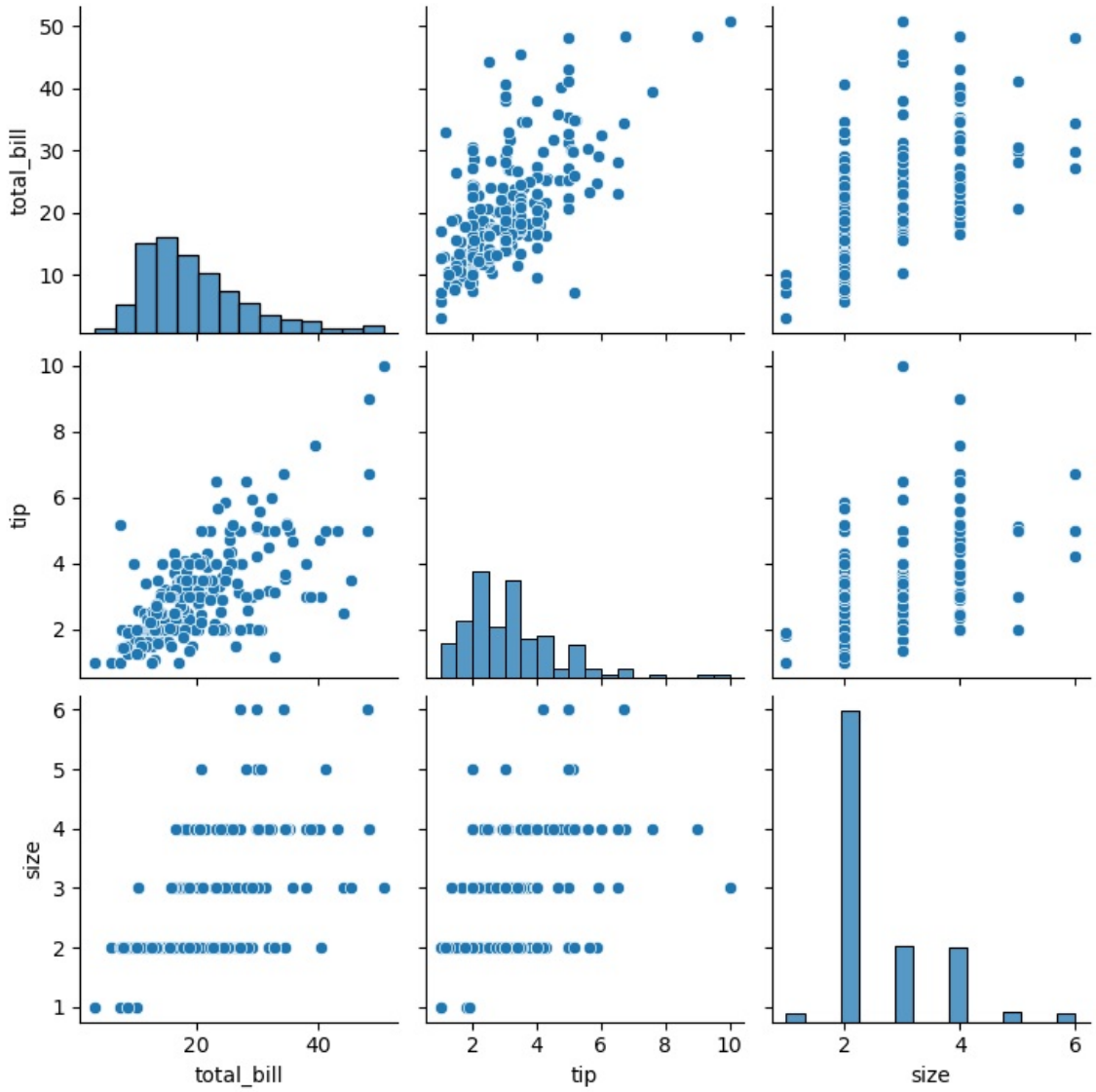
In [242…
```
np.var(data)
```
Out[242… 42620.617283950625

In [255…
```
np.std(data)
```
Out[255… 206.44761389745008

In [256…
```
statistics.variance(data) #sample variance
```
Out[256… 47948.194444444445

In [257…
```
statistics.pvariance(data) #population variance
```
Out[257… 42620.61728395062

In [258…
```
sns.pairplot(df)
```
Out[258… <seaborn.axisgrid.PairGrid at 0x2584085b1d0>

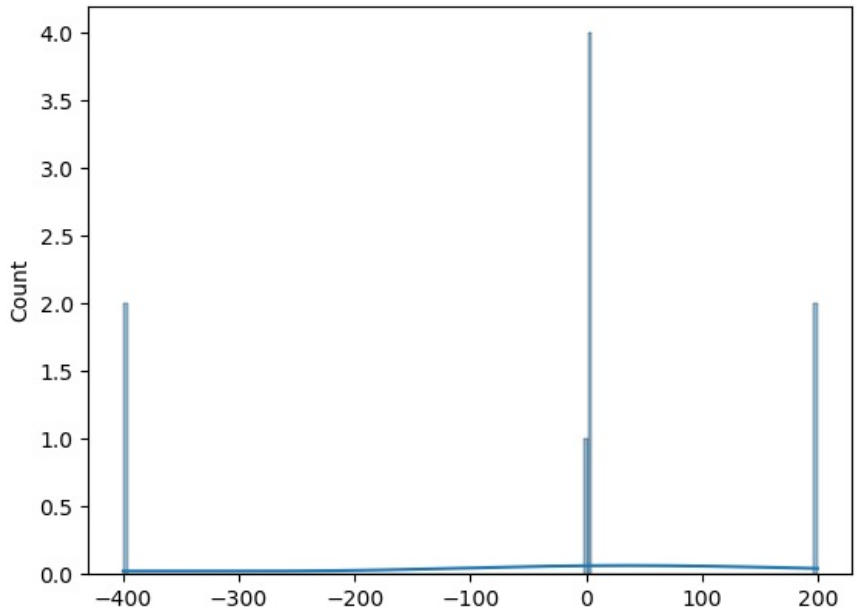


In [259…
```
data
```

Out[259… `[1, 2, 3, 4, 5, -400, 200, -400, 200]`

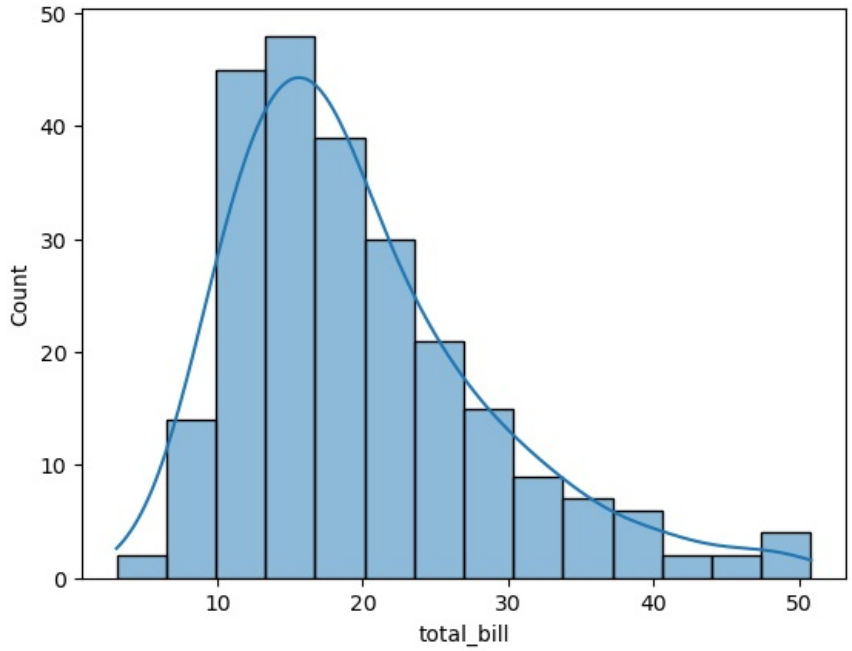
In [260…
```
sns.histplot(data,kde = True)
```

Out[260… `<Axes: ylabel='Count'>`



In [261…
```
sns.histplot(df['total_bill'],kde=True)
```

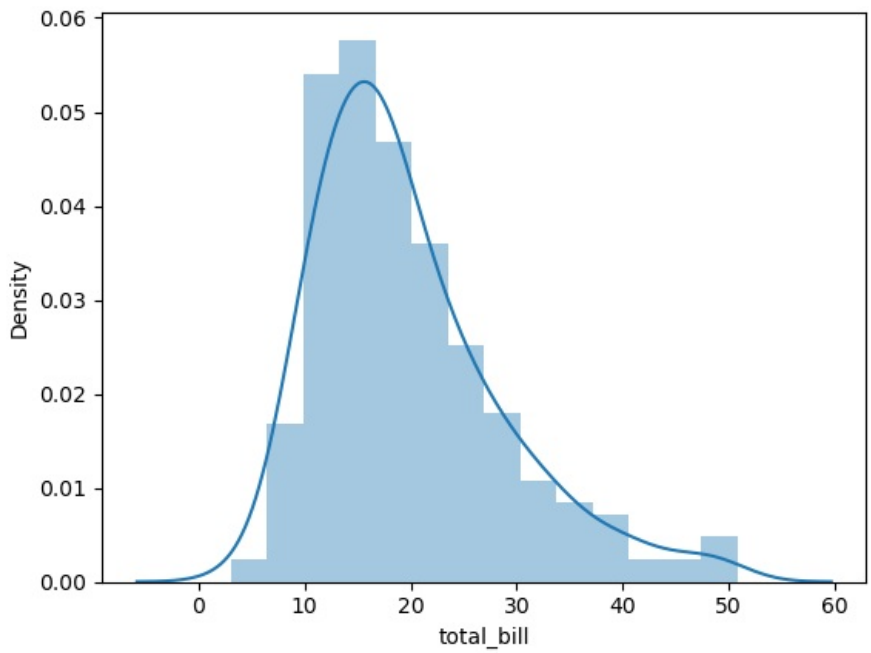
Out[261… <Axes: xlabel='total_bill', ylabel='Count'>



In [262…
```
sns.distplot(df['total_bill'],kde=True)
```

```
C:\Users\mdmaz\AppData\Local\Temp\ipykernel_13772\506577077.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['total_bill'],kde=True)
```

Out[262… <Axes: xlabel='total_bill', ylabel='Density'>

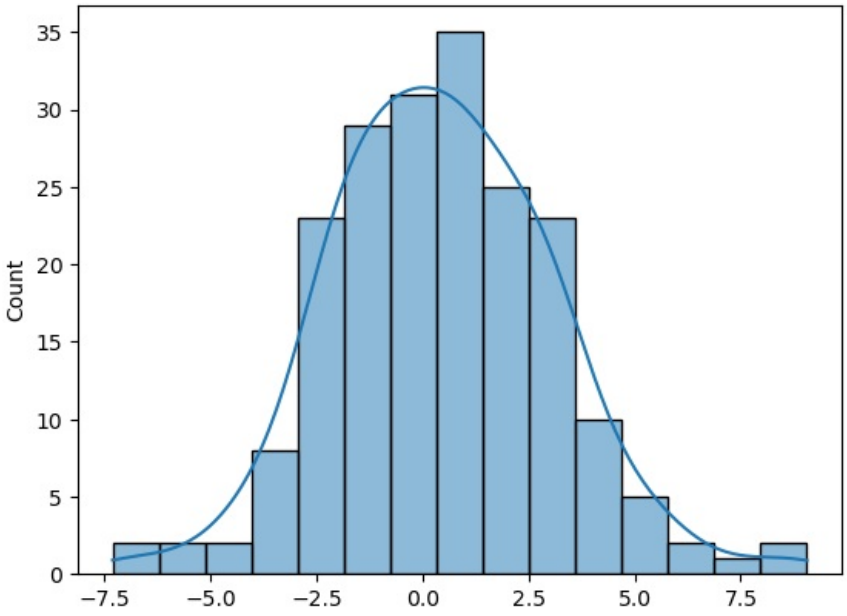


In [263…
```
np.random.normal(0.6,2.4,200)
```

Out[263…
```
array([-2.60232566, -2.51976295, -3.26004418, -2.59660959, -3.36998219,
        0.90081656, -0.04030292,  5.68948367,  5.13029263,  2.94626319,
        0.87858653,  0.73300505,  2.47224234, -2.79949377,  0.8988348 ,
        0.93692402,  2.07324476,  4.73555805, -1.63752303,  1.05161366,
        1.01638268,  2.35698425,  0.94256847,  2.1729262 , -2.09609769,
       -3.20097162, -0.07011747, -2.39890567,  5.93508472,  1.92025704,
        2.14336271, -1.37174426,  0.33784792,  1.32140103,  0.16361795,
        3.29644134,  1.35473277, -2.54006485, -0.13999648, -1.54473928,
       -0.27131875, -0.99774356, -0.24970205, -0.59232397,  1.38699359,
       -4.71496544,  3.98142038, -1.09209535,  0.06777286,  4.62712846,
        3.81522634, -2.12050118, -0.40593382,  2.48176291,  1.38833875,
       -2.32983885,  1.77020415,  3.03322579,  0.93611632,  0.70073189,
       -3.10926751, -2.82099417,  4.85181083, -0.6142576 , -1.60614692,
        2.41634682,  0.96977255, -2.68510669, -1.16204483,  0.72013528,
        3.55152078,  2.89178429,  5.59401974,  1.19041372,  1.33706495,
       -1.95122187, -1.34927117,  3.88973499, -2.0161649 ,  2.90789923,
       -0.78980848,  2.12439263, -0.02445617, -1.21184783,  2.77211331,
        2.13419523,  0.07996377,  0.87402302,  2.56762628,  4.83957005,
       -1.20518464,  4.27495351,  0.78893069,  6.40749615, -3.08363348,
        1.90956134,  1.38731987, -1.60999334,  4.26728662,  4.36176986,
        2.87881999,  1.49032887,  3.09772289,  4.45037684, -0.0625259 ,
        1.23996759, -0.22964787,  1.08817988,  3.57967593,  3.0825113 ,
        4.77001159, -1.35004803,  5.59300237, -4.6690349 ,  0.32328083,
       -0.54181182, -1.63048962, -0.98845463, -0.09756855, -1.50754171,
       -2.04966829,  1.79686312,  1.73575463, -2.79335423,  2.1571447 ,
       -2.48926602, -0.35512301,  2.89103577,  1.67360736,  5.86121386,
       -0.41024656,  2.0802811 , -2.07024064,  3.17408959, -1.00045034,
        0.91203426, -2.38345902,  1.08663493,  4.04725165,  0.23354773,
        5.49526026, -0.23349473,  4.91017649, -0.9427024 ,  0.98952947,
        0.69569595, -1.54744453,  2.23713482,  1.35172852,  1.68325459,
        5.45405659, -1.47888328,  1.6966096 ,  1.3245234 ,  3.13395702,
        0.39713836, -2.50219676,  3.10857214,  0.49969777,  3.5311352 ,
        0.64280175, -1.68774276,  2.64866591,  2.1071225 ,  3.97834887,
       -1.65939063,  0.12551008,  1.21896045,  0.2367624 ,  2.53477469,
       -1.34274584,  0.23961697,  1.38930427, -1.70452344,  4.79721574,
       -6.79326303, -2.68614296,  1.6689285 , -3.12932628, -0.86240373,
       -0.58051216, -3.03054396, -1.46811279, -0.11217034,  0.66452468,
        3.08556587,  3.37606589,  3.3977491 ,  1.29771686, -0.16643246,
       -0.13966926,  1.68586943, -1.0800475 , -0.11237377,  1.96664689,
        4.07628828,  3.82267912,  1.98374603, -0.25609084,  1.0644553 ])
```

In [264…
```
var= np.random.normal(0.6,2.4,200)
sns.histplot(var,kde=True)
```

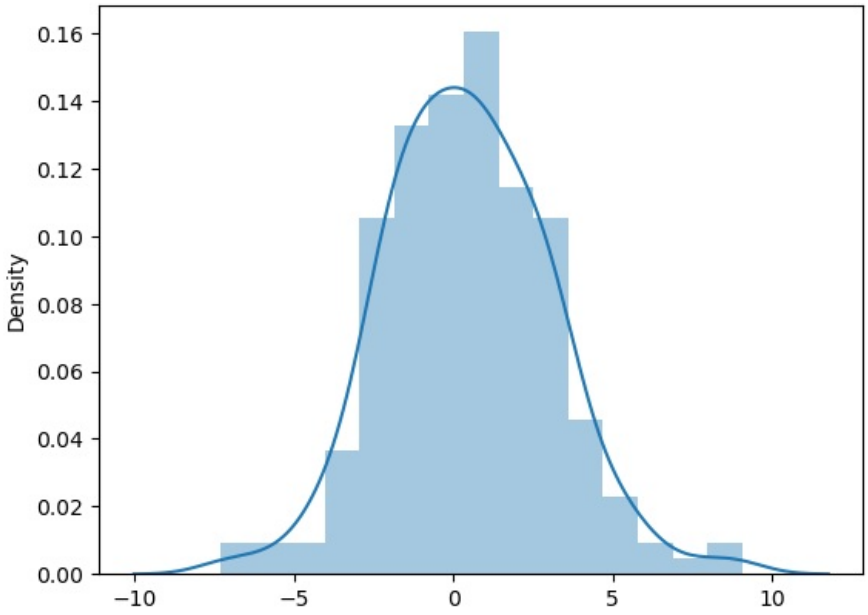
Out[264…
```
<Axes: ylabel='Count'>
```



In [265…
```
sns.distplot(var,kde=True)
```

```
C:\Users\mdmaz\AppData\Local\Temp\ipykernel_13772\1544059795.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(var,kde=True)
```

Out[265… <Axes: ylabel='Density'>



In [266…
```
population = np.random.randint(10,20,40)
```

In [267…
```
population
```

Out[267…
```
array([15, 15, 11, 11, 17, 18, 10, 11, 19, 18, 18, 11, 13, 13, 17, 12, 18,
       16, 10, 18, 18, 13, 18, 19, 18, 14, 16, 14, 17, 12, 18, 16, 16, 15,
       13, 11, 19, 17, 14, 13])
```

In [268…
```
len(population)
```

Out[268… 40

In [269…
```
np.mean(population)
```

Out[269… 15.05

In [270…
```
from statistics import mode
print("mean of the pop",np.mean(population))
print("median of the pop",np.median(population))
print(" the mode pop",mode(population))
```

```
mean of the pop 15.05
median of the pop 15.5
 the mode pop 18
```

In [34]:
```
sample = np.random.choice(population,20)
print("mean of the sam",np.mean(sample))
print("median of the sam",np.median(sample))
print(" the mode sam",mode(sample))
```

```
mean of the sam 14.35
median of the sam 14.5
 the mode sam 17
```

In [35]:
```
sample1 = np.random.choice(population,20)
sample2 = np.random.choice(population,20)
sample3 = np.random.choice(population,20)
sample4 = np.random.choice(population,20)
```

In [36]:
```
sample2
```

Out[36]:
```
array([19, 16, 15, 18, 15, 17, 14, 15, 16, 14, 17, 17, 14, 16, 13, 17, 19,
       17, 16, 17])
```

In [37]:
```
sample1
```

Out[37]:
```
array([17, 14, 14, 18, 15, 19, 14, 15, 17, 16, 10, 16, 18, 17, 19, 14, 10,
       17, 13, 12])
```

In [38]:
```
mean_of_sample = []
all_sample =[sample1,sample2,sample3,sample4]
for sample in all_sample:
    mean_of_sample.append(np.mean(sample))
```

In [39]:
```
mean_of_sample
```

Out[39]:
```
[15.25, 16.1, 14.25, 15.1]
```

central limit theorm

In [40]:
```
population = np.random.binomial(10,0.5,100000)
#this is no of trial or experiment #0.5 i the probability of success
#100000 is numb of time the binomial is repeated in the experiment
#10 No time experiment is repeated
```

In [41]:
```
population
```

Out[41]:
```
array([7, 2, 4, ..., 7, 4, 6])
```

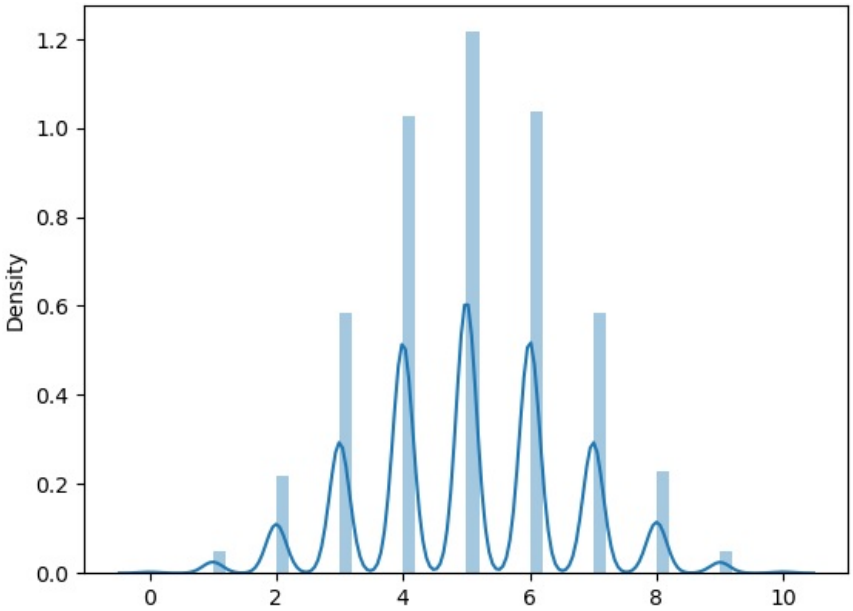
In [42]:
```
len(population)
```

Out[42]:
```
100000
```

In [43]:
```
sns.distplot(population)
```

```
C:\Users\mdmaz\AppData\Local\Temp\ipykernel_13772\3619714928.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(population)
```

Out[43]:
```
<Axes: ylabel='Density'>
```



In [44]:
```
sample_size = int(len(population)*0.03)
```

In [45]:
```
sample_size
```

Out[45]:
```
3000
```

In [77]:
```
mean_of_sample= []
for i in range(1,1000):
    sample = np.random.choice(population,size = sample_size)
    mean_of_sample.append(np.mean(sample))
```

In [78]:
```
mean_of_sample
```

Out[78]:
```
[-0.046812515929637394,
 -0.04114047398464917,
 0.032345985486096934,
 -0.2772571672585719,
 0.13148991048180586,
 -0.0684070265659026,
 -0.0816142717600603,
 -0.09883668247610304,
 0.06138455280367417,
 -0.16675257272856112,
 0.04929942670075118,
 -0.161934900768684,
 0.10596754405614778,
 0.006576243965986462,
 -0.19536833664611336,
 -0.07021674930124462,
 0.07778950767101864,
 0.04701163181943885,
 -0.13059088457307172,
 -0.09779435133340178,
 -0.11595981784220964,
 -0.03909123408588446,
 -0.18494417474542846,
 0.1584921919583373,
 -0.011800293827763552,
 -0.051422171832004794,
 -0.2576387065065155,
 -0.06523445647443292,
 -0.06941781517823947,
 -0.136418768195592,
 -0.0164881253626562,
 0.04028363847743664,
 -0.03831087017704576,
 0.03152130825740846,
 -0.12213735576874037,
 -0.20369264317585223,
 -0.08836669120113667,
 -0.23015879647448734,
 0.08692729236819031,
 -0.04606915419512507,
 0.075237986478965,
 -0.022214341946601,
 -0.11485823873108884,
 0.17728683299063244,
 -0.2021516845769313,
 0.031896460943343356,
 -0.006722531216764893,
 0.09175406012672119,
 -0.036047563090005064,
 -0.20310491954866425,
 0.13369575131555186,
 0.09617786804444287,
 -0.09739368760388416,
 -0.112417000020334,
 0.19659201831947873,
 0.0621714784890195,
 -0.22856135994690865,
 -0.02344655430954114,
 -0.07512847379000423,
 -0.21506149340066458,
 -0.037623036022508896,
 -0.12721463094062604,
 -0.13167171019003768,
 -0.0027713972480809795,
 -0.05869500723399881,
 -0.06527044956382548,
 0.046657207612417755,
 -0.18490647401229435,
 -0.13558391643116183,
 0.12964659546431256,
 -0.10128397378004406,
 -0.0905906946934982,
 -0.1179782427800741,
 -0.16835759227263153,
 -0.04298583082903967,
 -0.3398377118823773,
 0.08142686296900084,
 -0.055787122440773385,
 -0.019969572670772227,
 -0.13778860008745328,
 -0.06383744181357098,
```

-0.18926201757356226,
-0.1097753527583904,
-0.08640784516632107,
0.2316689190803713,
0.1112125310638641,
-0.003952790533279924,
-0.015043272105869693,
0.036358382676271554,
0.03512728162631668,
-0.1440620952749493,
0.027288904055480644,
0.09006084184926266,
0.054544531409176165,
0.08668466890345172,
0.04897504773961486,
-0.0017643351427631881,
-0.10021204852713404,
-0.11991898830061536,
0.11800746857949135,
-0.30206419068658763,
-0.03581922292222846,
0.054247319627582995,
0.17463496422834912,
-0.06582452623252312,
-0.2225479609621744,
-0.25140103372084527,
-0.15952801340055595,
-0.007448817552857815,
-0.18394659104470484,
-0.13537887029502418,
-0.06237849450082621,
-0.10860700957819099,
0.08918794067178533,
0.012947337657334392,
-0.10677730592436284,
0.16744815063503388,
-0.013294454461363476,
0.0426576339776545,
-0.07419243354460199,
0.03237326557035733,
0.09334883105162914,
0.05832855810852388,
-0.044134359587644126,
0.07652899246751967,
-0.01709159468710623,
-0.027178341447961538,
0.06435939477180196,
0.12258075039537969,
0.20690529400752372,
0.061712048053507856,
-0.26935402352027144,
-0.2697076582994342,
0.04295248882378513,
-0.39586131378359724,
-0.19560589951047258,
0.19099133405810498,
-0.16483939985159235,
0.09397182853167699,
0.060518985712752806,
-0.05236248565602807,
0.05594794663239412,
0.041910356090902975,
-0.061365481884430466,
-0.12524599661439692,
-0.12378822199095349,
-0.18672952006242027,
-0.01269013794401745,
0.010604651089522278,
-0.15733401736766703,
-0.08154777266188397,
-0.19283280789330204,
0.14664188818387128,
-0.07746850397825727,
-0.10718054225913423,
0.006765275096833627,
0.0804790476309884,
-0.16799269186464794,
0.12018581393270811,
-0.13943528402465444,
-0.07086537770059492,
-0.05626929469642359,
0.010256389097816393,
0.04002539489604692,

0.12021152407424854,
-0.2533137250601186,
-0.03785963599714097,
-0.026136339293640735,
-0.28183117872193997,
-0.05388292910301045,
-0.11376968490810306,
0.08482016503357834,
0.02469220229964716,
0.1640753649365857,
-0.010919258773142149,
-0.05808345691901901,
0.11632730706792213,
-0.06685998727847096,
-0.15654152727537643,
0.019664127694579366,
-0.16505853441098964,
0.02761571884870724,
-0.22238751320197803,
0.24432918565848583,
-0.3858754985976396,
-0.037973699172807344,
0.17524817634563675,
-0.18651868087812712,
-0.18584199739053114,
0.06934701455043099,
-0.28558903749458847,
0.1370030436041497,
0.14948943607679316,
0.022501100044453853,
0.33209004112104856,
-0.06900986473122005,
-0.0834219490332126,
0.08947332293258324,
0.11285744160611025,
-0.009867507167484786,
-0.050487339987088146,
0.09881628047064311,
-0.024741566422422857,
-0.19165466533716002,
-0.09635562615251109,
-0.06000911068111616,
0.15295064209958803,
0.01191011634755589,
-0.11430946791157609,
-0.02516637627405181,
0.024165048506090067,
-0.08044694497349864,
-0.0693329735976361,
-0.13377535329296245,
0.007893972746734628,
-0.05924147282425077,
0.14242960890733947,
0.05346927677268888,
0.11083662373975937,
0.042532437933171734,
0.07280843163075891,
0.16091029485041827,
-0.08129891946328105,
-0.08503271146876162,
-0.1530640313817426,
-0.05270954896389041,
-0.08168794997865997,
0.0007107127538966479,
0.14931357218203195,
0.10887945064782029,
-0.1536041136972757,
-0.12019931766106268,
-0.05573576632657117,
-0.06387724421938615,
0.058472538223229904,
-0.09158622147955747,
0.10277687886522037,
0.01262583162836501,
0.013514817406811597,
0.016483936304772805,
-0.17797390562505583,
0.051812280289597036,
-0.11500597603086274,
-0.16301700480369122,
-0.0401164350627713,
-0.1815895963107934,
-0.00872574588674039,

-0.16127574543029816,
0.0008201481195628496,
-0.2725889034680401,
-0.004448041625904278,
-0.022488186790866456,
-0.23499812176683274,
0.05441604977539948,
-0.011262313725586993,
0.03200243166801449,
0.016613863915623174,
-0.40777391657458173,
0.11546542601337433,
0.19800269009480434,
-0.19657276974561708,
-0.0830513581725553,
0.028323946395266714,
0.037039280626669537,
-0.10702736554538393,
-0.017943690531272052,
-0.09786647643608948,
-0.12856556969898533,
-0.03368376280415269,
0.10748321236841482,
-0.18368062724835243,
0.01868486698859606,
-0.18005033549329488,
-0.40816008912806245,
0.0441262987850793,
-0.28734801655799386,
0.2561550484106116,
0.046098106454192586,
0.2435819130965539,
-0.009811427321356064,
-0.08785639086140487,
0.06590754087933705,
0.34160544437575785,
-0.059064822269227005,
-0.3297656836881229,
0.1900206068778439,
0.017361213338964964,
0.14516367738156605,
0.1401109534317258,
-0.04064191418063831,
0.2842163104829654,
0.23010732963359168,
0.11872487848927386,
-0.4085597842421671,
0.0623308010785777,
0.0689674088780294,
0.17314690636526897,
0.018268201707916574,
0.03413051279739699,
-0.10694500950126687,
0.13903445956953459,
0.0493631563058607,
-0.10366408741338981,
-0.11815800897378077,
-0.05466542564457405,
0.12289808242217486,
0.11655057686529681,
-0.0727947539542387,
0.08564810226913497,
-0.05553313074421115,
-0.061547099954079164,
0.12269034786603776,
0.0571798393746461,
0.14486311282310335,
0.10008147162690743,
-0.1585693900337786,
-0.1958288234568543,
-0.0216091175303119,
0.060350996196234824,
-0.3119214940671591,
-0.22962487518075414,
-0.0166253395159505,
-0.1565632792015509,
0.062250511329718906,
0.022482678367416357,
-0.10553840949251629,
-0.10653423307933008,
0.004919815424290914,
0.049122462515647405,
-0.11509084935583187,

0.010730594997515004,
0.00025857231417575566,
-0.0862470339508481,
0.18871443955182546,
-0.35516994314318157,
0.05004630686919407,
-0.136627497644726,
-0.02922008821957921,
-0.026871776293150143,
0.17935875751171365,
0.23739814896968942,
0.06699167708040861,
-0.038710742027978334,
-0.12147521780110847,
-0.13675631363929433,
0.04867175643677009,
0.05624145477466625,
0.232999495821573,
-0.056130046483373094,
0.055324353200934394,
-0.13870246152936477,
-0.0069926827185574055,
-0.30504837734843565,
0.024918005592577147,
-0.16543558799752717,
-0.028694143003876748,
-0.028265292377988187,
-0.05184644666156073,
0.08124616529308522,
-0.3081057165471595,
-0.20404397721344847,
0.00516228074183303,
0.03539034081871458,
0.05628832960331948,
-0.06930039302246657,
-0.26509962522896713,
-0.23500304774571404,
-0.11319409770994639,
0.10751164388038846,
-0.06640529868974981,
-0.1480737224225316,
-0.3960420143355609,
-0.0899056280282118,
-0.051207607174910796,
0.16143951513033167,
-0.0953664187282816,
0.142693894480708,
-0.2411292793359873,
0.25339266144243405,
-0.08159103330082125,
-0.10951842495193238,
-0.15444634558988704,
-0.18591607196171012,
-0.042590681546313916,
-0.1755664740109737,
-0.0919663739923127,
-0.07430907813393581,
-0.19465129439293188,
-0.11659816875951277,
0.027364127561682067,
-0.10499601145580874,
-0.08104271137642312,
-0.03636475193286058,
-0.010091733603667503,
-0.20027141676720583,
0.09786374382037093,
0.04460179648364363,
0.024094950199281363,
0.05407910886165295,
-0.06436850078487914,
0.2375434684321789,
-0.14178419454146352,
-0.019869007136873066,
0.01398043221784321,
-0.049691190643077615,
-0.19106393195548982,
-0.11350409755720255,
-0.11883187782549767,
-0.06062338887611595,
0.0007184834431018583,
-0.1390881565051354,
-0.02269595300426498,
-0.23639648099400673,

0.009530866330058379,
-0.042373617594877594,
0.10195317784750799,
0.002660383608899348,
-0.3286627086650806,
0.05836095510462281,
0.2993066330216978,
-0.2563499484264268,
0.04788688623006518,
0.06847496225180877,
-0.049435622162085725,
0.010299356084182018,
-0.33916214826340574,
-0.12653090378128304,
-0.17256275117696002,
-0.0360673654022825,
-0.15083904156351713,
0.22639799962421672,
-0.05611964820052183,
0.04768240257686837,
-0.13009370288148006,
-0.2421446418677059,
0.0484866173544126,
0.036089197170631776,
0.07849750766336756,
-0.05119643216922601,
-0.03261941183545073,
-0.2569445763827498,
0.06900462344960809,
-0.2995812185484538,
0.024444957215746847,
-0.08409082141921649,
0.09369586672049898,
0.017697218976960402,
0.0560971594044727,
-0.10486913188283127,
-0.3337705940054,
-0.11377117951136625,
-0.012434155912993097,
0.4098585872350961,
0.2159082108128256,
0.02809467913520447,
-0.14910715386727463,
-0.140241486416206,
0.20173619388717237,
-0.23566989021159201,
-0.12877110361244104,
-0.10365691787210482,
-0.10278217715521949,
-0.06452483990865526,
-0.10987402688526934,
-0.2759672328665134,
0.09563723033119045,
-0.014672248745915249,
0.113021621471313 95,
-0.04687560733792501,
-0.19293037205637603,
0.002970047067698879,
-0.5116577969715809,
-0.11562189426629418,
0.11242607132009301,
-0.29781141238916825,
-0.10138211346473154,
0.2082263702277755,
-0.20113570336161263,
-0.1104343205647847,
-0.10490805896236982,
0.022609972753507058,
0.06896677878224709,
-0.1759067260936867,
0.20001030059328198,
-0.02264841557581624,
-0.13956678887722082,
-0.1309950688002693,
0.1958269657997767,
0.06486517222756116,
0.024198556800990475,
-0.012230416622704769,
-0.16818663002957795,
-0.1817969245015557,
-0.173372944054216,
-0.11425633393054824,
-0.3640291725054821,

-0.2028672527586086,
0.2445154612528492,
-0.26173574991517445,
-0.05153383132884422,
-0.1779751709399288,
-0.035126189255891706,
0.1463746036908648,
-0.38911710077374445,
0.05850184515695385,
0.1992264187142239,
0.05895936812967097,
-0.08737783984500773,
-0.17548283988220004,
0.0015227119190218729,
-0.33523414297676396,
0.04741791472404233,
-0.12618884305415254,
0.030005824599633017,
-0.05779516040169975,
-0.4007933649249853,
0.03781585244222196,
-0.2470338304207377,
0.3122589721282004,
-0.0221243463987275,
-0.0687602266613089,
0.10960479789917463,
0.13434335206527562,
-0.11439315382665816,
-0.11452273439975157,
-0.01699114135556785,
0.043693414652139405,
0.03904357963685743,
0.1073231607905847,
-0.06485578152635232,
0.13149838599233196,
-0.03244557604688572,
0.02589270467417501,
-0.09378973308726585,
0.2675331207586502,
-0.03962324491705155,
0.017514174731461166,
0.005164937617248482,
0.2824842518813062,
-0.11700366174673363,
-0.019125576039179252,
-0.05735895781296918,
-0.010943837306284598,
-0.21683345636506307,
-0.20730749825876013,
-0.24587112484061993,
-0.04221387441412434,
0.21066885529906712,
-0.04454417669938822,
0.16973249149100628,
-0.10850126052758209,
-0.025616893803991812,
-0.0787525933368963,
-0.07322931212711206,
0.10776931005344928,
-0.10261685494177251,
-0.000867507849125162,
-0.11354976307483974,
0.02822694908716223,
-0.22793475896222418,
-0.08340118931806588,
-0.3106410216307577,
-0.12077168367412781,
-0.12686518674783492,
-0.45901329689394926,
-0.2137270838443347,
-0.019158544649930675,
-0.035931912948140174,
0.10060849285451912,
-0.21719564465935812,
0.08959923272555281,
0.1732885177134246,
0.027616336566024278,
-0.39272079236753044,
-0.1555485120438213,
0.028039062594527674,
-0.3480886432220188,
-0.30509283215198074,
-0.23173411356261867,

-0.2786131957554254,
-0.018618352408318044,
-0.14478778935159278,
0.12650079700785324,
-0.031795495705672125,
0.08991684587816039,
-0.1575403094424994,
0.31278308551702816,
0.10444022524960428,
0.15626256569128744,
0.0990751201575274,
0.12434701034494593,
0.4145071132273006,
-0.00866313600244013,
-0.10373676801879624,
-0.07058785267891554,
0.1081706625797797,
0.010430949390025993,
-0.11781833516929784,
-0.006732749384327464,
0.008199415828623983,
0.026306191501012665,
-0.219912061803316,
0.1960433373315226,
0.0818739083043757,
-0.0765172845504419,
-0.2734962487617154,
-0.03291702634395682,
0.3154771435661635,
-0.10843699603682957,
-0.18565809497692112,
-0.03169954428270024,
0.021831614924904038,
-0.1457503999107123,
-0.09321373620482219,
-0.06778409028821464,
0.2462636785060166,
0.13587150030610962,
0.01555100114325514,
-0.07702313288683381,
0.18579039899148447,
-0.2274342154091754,
-0.1122002519730248,
0.17715305225746492,
0.04843859797605978,
0.3509970993514686,
0.05948355662331297,
0.07398314197196326,
-0.24103598231471357,
-0.1857838669077913,
-0.09910491830665996,
-0.1360502777444465,
0.19246979086794266,
0.016625074896190875,
0.026687725299423554,
0.3629931974610936,
0.14854425455653075,
-0.1043821084556318,
-0.06453558623168947,
-0.045115317497131276,
-0.051640412729051184,
-0.1583679459446109,
0.08125180819832334,
-0.10136538074329537,
0.07470321555726096,
0.12355331790614088,
-0.3996545803583804,
0.15720069749240206,
-0.04163087028913615,
0.028452541289540126,
-0.221614109871994,
0.023402972391441484,
-0.058987551227871915,
0.03630086499359034,
-0.07008348676217285,
0.0540965670334745,
-0.11597104416843072,
0.2550137923962923,
-0.03927122982696887,
0.1158200732523518,
0.04589857787393686,
0.1350395336671813,
-0.06936290476616185,

-0.17460376878848016,
0.09681812434271249,
0.09768432601306053,
-0.3545159156334688,
0.015872293929278445,
0.06711717675146935,
0.03530371473262527,
0.010989853539293375,
-0.05378120181412511,
0.0715277544122741,
-0.00015041707282946072,
0.1642250569975278,
-0.23352739542734646,
-0.14950592229667684,
-0.15487810396853585,
-0.2662839731198765,
-0.09190472152502778,
0.07783740892927976,
0.0668423917686105,
0.2369656432068841,
-0.23195694403074557,
-0.023764781353002185,
-0.012351593751500911,
0.09818296210330495,
0.18279307089042368,
-0.1179316128979137,
-0.21429210814883337,
0.13866473838011878,
0.05302151875598932,
-0.011993912463417277,
0.11943947573694307,
-0.1261137078902342,
-0.18182449377262505,
0.17058898495452432,
0.06365841619760254,
0.1086862087680374,
0.0025661193002350967,
-0.28572676200736147,
-0.019113192822923116,
-0.16569305793200395,
-0.032215222198972195,
-0.14251582561938236,
-0.004441310672507344,
-0.03976147142188159,
-0.15965503595237623,
0.14276419305091795,
-0.09946142583249136,
0.23867603191269218,
-0.015459794534824165,
0.11907833395568193,
-0.01701612740545173,
-0.006482943568240817,
0.0445337920847148,
0.029758412958329542,
0.1303199403097349,
-0.11640137588735232,
0.04503018296049068,
0.0065363819740884895,
0.24733816658268867,
0.003396627760579727,
-0.12662726825917742,
-0.019816697566021217,
-0.20703260551012337,
-0.09197134752078728,
-0.1959606615561401,
0.1844824518017036,
0.01749248314588148,
0.07156198141430165,
-0.07716589370788354,
0.04989027736546375,
-0.11949441230941067,
0.008986848600546138,
-0.03403240405183534,
0.09095867772863084,
0.16583521532300918,
-0.12362214990098709,
0.07857253250251449,
0.010792898165329377,
-0.31238904099165177,
-0.005914390927368517,
-0.12447358839303496,
0.005727451529870962,
-0.19112186394014516,

-0.10739350726095014,
-0.07371734452260377,
-0.25864477547709774,
0.027058573089162284,
-0.011278541274927858,
-0.14039842257475132,
0.010677680211550156,
-0.28106928383494556,
0.0099392065153169,
-0.03109085330463905,
-0.09030740373943276,
-0.11064299977141107,
-0.23253864406493832,
0.039033419316398865,
-0.1954236989246447,
0.02711889854019592,
0.025137048961550078,
-0.14811356195186934,
-0.03423477828368981,
-0.08094155309377965,
-0.17306002151872363,
0.04461457507748998,
0.18396840479979557,
-0.029762000329263297,
-0.026035439067276088,
-0.15413386849887314,
0.06658258587011998,
0.10647517807041615,
-0.23153166966394584,
0.10353288188780532,
0.06459902774411021,
-0.02003321294239571,
0.09525191511107199,
0.02722920282196734,
-0.050991574659925834,
0.06702206560516835,
-0.015204726852147988,
-0.08724924641574709,
-0.280243905582517,
0.06870829257856928,
0.022024300743888996,
0.05864639363862903,
0.03830317105238613,
-0.07003370579269431,
-0.18663199917533485,
-0.07646470304413454,
-0.10740787521152344,
-0.20574447493216028,
-0.0872083737796541,
-0.03334588054199627,
0.06700723817359515,
0.20191322937537529,
-0.1939565589996909,
-0.1401091042311663,
-0.21051341909668866,
-0.11316664839526659,
-0.11741425342431736,
0.22221615039349307,
-0.09003268630063864,
0.10066803425807916,
-0.3693282057740406,
0.025071297004767137,
-0.11929405153902486,
-0.1483087605262247,
0.17391084464753795,
-0.18152691745290306,
-0.25911424345774864,
-0.03389098424504951,
-0.22190288167277067,
-0.09924366714184521,
-0.2033942659455355,
0.08339767372720061,
-0.21992177870507798,
-0.16161579927722783,
0.020788424135463574,
0.09701943970022936,
-0.2281132089726919,
-0.045619239682740176,
-0.27224176821657364,
-0.14680967829035158,
-0.10776238900827682,
-0.4230819680049978,
-0.38620351789766416,

0.04432647735502664,
0.11163514128981776,
0.1245095772325773,
-0.10672182249541871,
-0.08161448009228799,
0.17737370779890466,
-0.1382106619056207,
-0.09961631653094852,
-0.04986809419320318,
0.0842352586355399,
0.1208956490000687,
-0.12429038287338813,
-0.10623091947083513,
0.18649779568546918,
0.09071948232284198,
-0.21521846751422494,
0.06311682823058047,
0.0818748799011542,
-0.22911139602052305,
-0.251742016965368,
0.06958442648648083,
0.14601321582906596,
0.022480540911895615,
-0.2230476083649029,
0.035512021794276344,
-0.13567915474658135,
-0.22267747799313628,
-0.2108183437363328,
-0.04616543104041097,
0.30335882280686916,
0.04117417332480617,
0.00018844222845518121,
-0.17092982557628786,
-0.052741067719446165,
-0.07228300333775156,
-0.04317834989800502,
0.16471190831664623,
-0.12846061535458966,
-0.27309754561728666,
0.07995853341352391,
-0.0860812506612789,
-0.011414953787544601,
0.12664163117550176,
-0.13630425805621027,
0.05654508062402905,
-0.03178988709589588,
0.050651653127765754,
-0.0857923852369491,
-0.049301696159325725,
-0.3432861675586934,
-0.01992002460274707,
-0.15896103686386118,
-0.38966722544252624,
0.24046194329697207,
-0.3620197943750317,
-0.13723044622008668,
-0.047305744404069525,
-0.0374499616889197,
-0.09208783962131781,
0.23658488786704449,
-0.15091810149679963,
0.0036437868541051043,
-0.0688965206168207,
-0.055474009577836256,
0.12657285442816868,
-0.013046574658529635,
0.11363743477702565,
-0.0988046724582259,
-0.1752773916617266,
-0.17304638613578724,
-0.1879275159689159,
0.06752315490253637,
-0.1725061580072054,
-0.1952322339619565,
0.15931851723475132,
0.006684734191787247,
0.007035853163422723,
-0.008807229673512834,
-0.0920906572834768,
-0.017044633692563683,
-0.015548379706396775,
-0.31164054835609345,
0.009277633733061089,

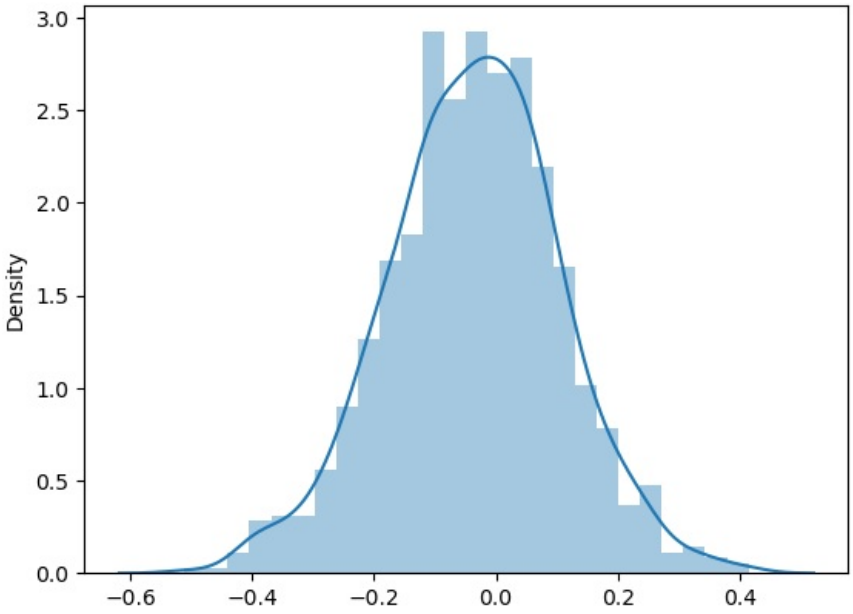
0.038883933821122714,
0.07647104930421925,
-0.015166226510934,
-0.2027105052252756,
-0.10290534615540295,
-0.039269849228157044,
-0.17576707017884266,
0.009929721537991312,
-0.0419566410501732,
0.14472834603391804,
-0.01591763004269575,
-0.1141794221940721,
-0.26701268147507795,
-0.016849970414475453,
-0.09559497595257124,
-0.09386031949829123,
-0.012352594713332015,
-0.1742192762006628,
0.04141409310330488,
-0.13845801764756802,
-0.017722284752863752,
-0.17149082072012478,
0.04163481679244436,
0.07797024154674827,
-0.008506427426346153,
0.09403495612404544,
0.015540302021597095,
0.06385263312768401,
-0.19105290902784566,
0.0965582305587651,
-0.2384201255268209,
-0.04289876134526468,
-0.08769477893666272,
0.0475915997867183,
-0.07516798765510639,
0.007394650134933123,
-0.10497087167723099,
-0.016324905477810146,
-0.24233739255555548,
-0.041251924788616 86,
-0.3453686460135482,
0.018365302388378125,
0.07834053930205533,
-0.15581043351777932,
0.023118684783853937,
0.14091164427606218,
0.04561836392082341,
-0.13224137111403753,
0.06738344122884417,
-0.1851096590479332,
-0.09334163105690782,
0.05139031477907976,
0.24523296315271392,
-0.06412684311095154,
0.2621580816606128,
-0.06343043951945267,
0.0004510552227502407,
-0.1230358931055432,
0.09569483001059682,
-0.22310389179752468,
0.09301340913731021,
0.093694822634817,
0.06607692068658523,
-0.04752917112090332,
-0.139109153000316,
-0.10448745257054182,
-0.21577970861324922,
0.05578254384194146,
0.11623492892368846,
-0.15294714534725526,
-0.029332298635609853,
-0.10125150251209603,
-0.27510848233063806,
0.01752519684223274,
0.062289774408659716,
-0.01736839348465058,
-0.04052717807243431,
0.37221299893293747,
-0.18686686594288213,
-0.20849732399861243,
-0.02980481044417411,
0.0659747429290975,
0.0496558244116906,

```
0.2121266774343136,
0.21142698919095554,
-0.016373418062697473,
0.15284321287306207,
-0.08733367835663458]
```

In [79]:
```
sns.distplot(mean_of_sample,kde = True)
```

```
C:\Users\mdmaz\AppData\Local\Temp\ipykernel_13772\352863661.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mean_of_sample,kde = True)
```

Out[79]:
```
<Axes: ylabel='Density'>
```



In [80]:
```
#ztest
```

In [81]:
```
from numpy.random import randn
population = randn(100)
```

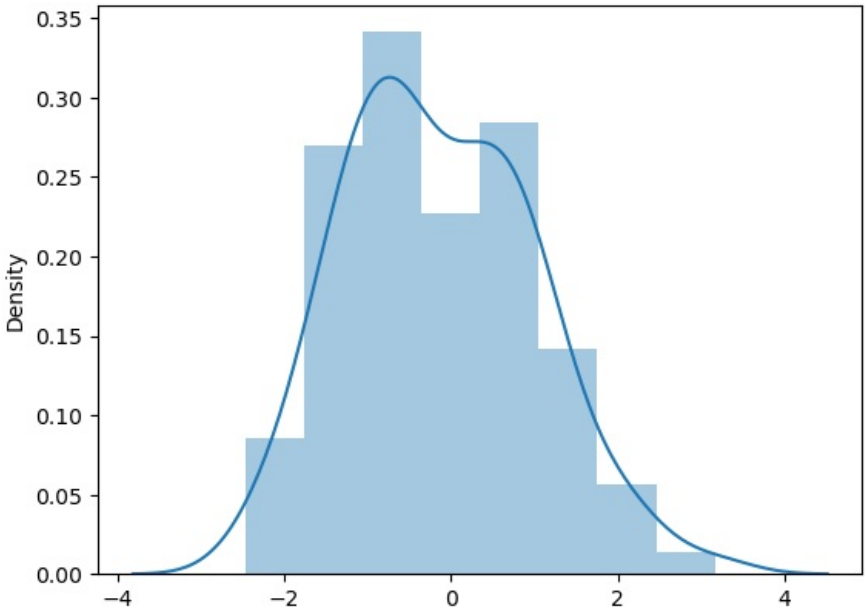
In [82]:
```
population
```

Out[82]:
```
array([-1.10106026,  0.87507247, -0.43842807, -1.36277282, -0.93298624,
        1.07481495, -0.35508688, -0.15849545, -2.46311287, -0.92140255,
        0.66425589,  2.27005077, -0.01676983, -2.00227207, -0.43445505,
       -0.83471054,  1.07958168, -0.35316214,  0.5963354 ,  1.15880977,
        0.48985498, -1.06553438,  1.00797229, -0.35826956, -0.65993933,
        0.32766707, -1.09734235, -1.47366418,  0.14906553,  0.05871534,
       -1.68646318, -1.29528417, -0.77333709,  0.50781745, -2.07496011,
       -0.18884388, -0.70469543,  2.06730925,  3.16456888,  1.06011677,
       -0.30386334, -0.63650131,  0.92066027,  1.53724439,  0.35279654,
       -0.73101992, -1.1395406 , -0.29656453, -0.60567695, -1.25372677,
        0.73207251, -1.40316683,  0.60715524,  0.61808341, -0.55034943,
        1.30763383,  1.27296699,  0.260015  ,  0.74813482,  0.88950904,
       -0.87491516,  1.00985368,  1.33458489,  0.27421521, -1.2907097 ,
        0.42083677,  0.3021707 , -1.33028475, -0.88811794, -0.22777571,
       -1.94164885, -1.6950097 ,  1.69945439, -1.12943639,  1.02002174,
       -0.7627204 ,  0.73934273, -1.06462113, -0.6431417 ,  1.90035519,
       -0.67463983, -1.35674904,  0.44313994,  0.28417489, -1.31345768,
        0.20161162,  2.35396675, -0.37758367, -0.76584419, -2.1976929 ,
       -1.78874224, -0.66143941,  1.6043075 ,  0.69440623,  0.42835914,
        0.03090215, -0.67808978, -1.41125429, -1.5420616 ,  0.09463864])
```

In [83]:
```
sns.distplot(population)
```

```
C:\Users\mdmaz\AppData\Local\Temp\ipykernel_13772\2958836585.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(population)
```

Out[83]: <Axes: ylabel='Density'>



In [84]:
```
np.mean(population),np.std(population)
```

Out[84]: (-0.13654771485126274, 1.1238328103637583)

In [85]:
```
null_mean = 0.09
```

In [86]:
```
null_mean
```

Out[86]: 0.09

In [87]:
```
from statsmodels.stats.weightstats import ztest
```

In [88]:
```
ztest(population,value = null_mean,alternative = 'larger')
```

Out[88]: (-2.0057443429619983, 0.9775582342267461)

In [95]:
```
zscore, p_value = ztest(population,value = null_mean,alternative = 'larger')
```

In [96]:
```
zscore
```

Out[96]: -2.0057443429619983

In [97]:
```
p_value
```

Out[97]: 0.9775582342267461

In [98]:
```
if p_value<0.05:
    print("reject the null hypothesis")
else:
    print("fail to  reject the hypothesis")
```

```
fail to  reject the hypothesis
```

In [99]:
```
#school calculated its IQ score for 50 students, average turned out to be 50
#std is16
#that iq increase if student study more than the average student with a 5%signficance level

#null hypothesuis:mean_iq = 90
#alternative:mean_is>90
```

In [100…
```
import scipy.stats as stats
sample_mean = 100
population_mean = 90
```

```
population_std = 16
sample_size = 50
alpha = 0.05
```

In [101…
```
zscore = (sample_mean-population_mean)/(population_std/np.sqrt(50))
print("zscore is ",zscore)
```

```
zscore is  4.419417382415922
```

In [102…
```
#stats>>statics module is scipy
#norm>>normal distribution
#ppf>>person point function>>inverse of cumalitive distribution function
#alpha is 0.05
zcritical = stats.norm.ppf(1-alpha)
```

In [103…
```
zcritical
```

Out[103… 1.6448536269514722

if zscore > zcritical: print("Reject the null hypothesis") else: print("fail to reject the null hypothesis")

In [ ]:

In [104…
```
p_value = 1-stats.norm.cdf(zscore)

if p_value > alpha:
   print("reject the null hypothesis")
else:
   print("fail  to reject the null hypothesis")
```

```
fail  to reject the null hypothesis
```

In [105…
```
p_value
```

Out[105… 4.948367312573865e-06

In [106…
```
#T_test  sapmle_size >30
#one sample t test(with the respect to one independent sample)
#twp sample t-test (with respect to two independent samples)
#paired sample(two sample from population on different time interval)
```

In [107…
```
#one population
population = np.random.randint(10,50,50)
```

In [108…
```
population
```

Out[108…
```
array([39, 28, 29, 42, 37, 20, 22, 11, 46, 43, 29, 45, 40, 11, 22, 22, 27,
       30, 21, 40, 10, 46, 18, 22, 24, 47, 35, 28, 28, 10, 17, 38, 47, 32,
       29, 19, 30, 25, 48, 14, 44, 35, 18, 43, 12, 29, 36, 20, 40, 15])
```

In [109…
```
sample = np.random.choice(population,20)
```

In [110…
```
sample
```

Out[110…
```
array([46, 45, 30, 11, 29, 25, 46, 40, 17, 32, 38, 40, 12, 17, 30, 20, 44,
       47, 11, 21])
```

In [111…
```
np.mean(sample)
```

Out[111… 30.05

In [112…
```
null_value = 25
import scipy
scipy.stats.ttest_1samp(sample,null_value)
```

Out[112… TtestResult(statistic=1.773539442071365, pvalue=0.09216911876346907, df=19)

In [113…
```
virat_kholi_score = np.random.choice(population,50)
```

In [ ]:

In [114…
```
virat_kholi_score
```

Out[114…
```
array([25, 28, 18, 20, 22, 40, 46, 43, 22, 42, 25, 22, 18, 22, 40, 40, 35,
       22, 12, 25, 28, 27, 25, 36, 40, 44, 27, 44, 11, 43, 21, 38, 20, 35,
       19, 39, 22, 18, 10, 28, 37, 28, 14, 28, 22, 29, 47, 30, 27, 30])
```

In [115…
```
rohit_sharma_score = ([97, 41, 36, 16, 36, 42, 17, 98, 18, 30, 36, 49, 38, 47, 44, 26, 18,
       12, 16, 39, 10, 18, 22, 64, 30, 33, 23, 23, 35, 17, 18, 42, 30, 23,
       76, 33, 14, 18, 16, 47, 12, 42, 37,50, 133, 10, 18, 19, 41, 37])
np.mean(rohit_sharma_score),np.mean(virat_kholi_score)
```

Out[115… (34.94, 28.68)

In [116…
```
scipy.stats.ttest_ind(virat_kholi_score,rohit_sharma_score)
```

Out[116… TtestResult(statistic=-1.705647253885332, pvalue=0.09124144754960584, df=98.0)

In [117…
```
#paired ttest
rohit_sharma_first_inning = [97, 41, 36, 16, 36, 42, 17, 98, 18, 30, 36, 49, 38, 47, 44, 26, 18,
       12, 16, 39, 10, 18, 22, 64, 30, 33, 23, 23, 35, 17, 18, 42, 30, 23]
rohit_sharma_second_inning = [44, 10, 15, 30, 17, 23, 18, 18, 33, 16, 44, 23, 33, 14, 18, 17, 12,
       21, 17, 23, 37, 24, 10, 44, 39, 44, 10, 19, 36, 30, 10, 12, 19,80]
```

In [118…
```
np.mean(rohit_sharma_first_inning),np.mean(rohit_sharma_second_inning)
```

Out[118… (33.64705882352941, 25.294117647058822)

In [119…
```
  scipy.stats.ttest_rel(rohit_sharma_first_inning,rohit_sharma_second_inning)
```

Out[119… TtestResult(statistic=2.04256488916791, pvalue=0.049151322458030285, df=33)

In [120…
```
#chi_square
data = sns.load_dataset("tips")
```

In [121…
```
data = sns.load_dataset("tips")
```

In [122…
```
data
```

Out[122…

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [123…
```
data_table = pd.crosstab(data['sex'],data['smoker'])
```

In [124…
```
data_table
```

Out[124…

smoker	Yes	No
sex		
Male	60	97
Female	33	54

In [125…
```
 observed_values = data_table.values
```

In [126…
```
observed_values
```

Out[126…
```
array([[60, 97],
       [33, 54]], dtype=int64)
```

In [127…
```
 import scipy.stats as stats
stats_test,dof,p,expected_values = stats.chi2_contingency(observed_values)
```

In [128…
```
expected_values
```

Out[128…
```
array([[59.84016393, 97.15983607],
       [33.15983607, 53.84016393]])
```

In [129…
```
observed_values
```

Out[129…
```
array([[60, 97],
       [33, 54]], dtype=int64)
```

In [130…
```
#dof=(no of row-1 )*(no of coloumn-1)
dof = 1
```

In [131…
```
from scipy.stats import chi2
```

In [132…
```
for i in zip(observed_values,expected_values):
    print(i)
```

```
(array([60, 97], dtype=int64), array([59.84016393, 97.15983607]))
(array([33, 54], dtype=int64), array([33.15983607, 53.84016393]))
```

In [133…
```
chisquare_test  = sum([(o-e)**2/e for e,o in zip(observed_values,expected_values)])
```

In [134…
```
chisquare_test
```

Out[134…
```
array([0.00119996, 0.00073648])
```

In [135…
```
chisquare_stats  = chisquare_test  [0]+ chisquare_test[1]
chisquare_stats
```

Out[135…
```
0.001936441617319103
```

In [136…
```
dof
```

Out[136…
```
1
```

In [137…
```
alpha
```

Out[137…
```
0.05
```

In [138…
```
chi2_critical = chi2.ppf(1-alpha,df = dof)
```

In [139…
```
chi2_critical
```

Out[139…
```
3.841458820694124
```

In [140…
```
if chisquare_stats>=chi2_critical:
    print("reject the null hypotheis,there is a realtionship between two categorical variable")
else:
    print("fail to reject the null hypotheis,no relationship")
```

```
fail to reject the null hypotheis,no relationship
```

In [141…
```
#no of hours study by student
observed_data = [9,8,5,4,3,1]
expected_data = [1,3,4,8,5,9]
sum(expected_data),sum(observed_data)
```

Out[141…
```
(30, 30)
```

In [142…
```
chisquare_stats,pvalue = stats.chisquare(observed_data,expected_data)
```

In [143…
```
chisquare_stats,pvalue
```

Out[143…
```
(82.49444444444444, 2.521799643395917e-16)
```

In [144…
```
# find the critical value
alpha = 0.05
dof = len(observed_data)-1
critical_values = stats.chi2.ppf(1-alpha,dof)
```

In [145…
```
critical_values
```

Out[145…
```
11.070497693516351
```

In [146…
```
if chisquare_stats>critical_values:
    print("reject the null hypothesis")
else:
    print("fail to reject the null hypothesis")
```

```
reject the null hypothesis
```

In [147…
```
# Two worker one is more effective then the other
```

In [148…
```
worker1 = [12,22,32,45,65,54,31,59,90,81]
worker2 = [12,21,31,23,43,13,41,23,56,40]
```

In [149…
```
#F statistics
```

In [150…
```
fstats = np.var(worker1)/ np.var(worker2)
```

In [151…
```
fstats
```

Out[151… 3.1284107421559657

In [152…
```
df1 = len(worker1)-1
```

In [153…
```
dof
```

Out[153… 5

In [154…
```
df2= len(worker2)-1
```

In [155…
```
dof
```

Out[155… 5

In [156…
```
alpha = 0.05
alpha
```

Out[156… 0.05

In [157…
```
critical_value = stats.f.ppf(q = alpha,dfn = df1,dfd = df2)
```

In [158…
```
critical_value
```

Out[158… 0.31457490615130795

In [159…
```
fstats = np.var(worker1)/np.var(worker2)
```

In [160…
```
fstats
```

Out[160… 3.1284107421559657

In [161…
```
critical_value = stats.f.ppf(q=alpha,dfn=df1,dfd=df2)
```

In [162…
```
if fstats>critical_value:
   print("reject the null hypothesis")
else:
   print("fail to reject the null hypothesis")
```

```
reject the null hypothesis
```

In [163…
```
df
```

Out[163…

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [205…
```
#create  a sample dataset with the missing value
data = {
    'A': [1,2,np.nan,4,5],
    'b': [3,np.nan,7,8,9],
    'c': [np.nan,12,13,14,15],
    'd': [16,17,18,np.nan,20]
}
#convery dictionary to pandas dataframe
```

```
df = pd.DataFrame(data)
print('orginal dataframe')
print(df)
```

```
orginal dataframe
     A    b     c     d
0  1.0  3.0   NaN  16.0
1  2.0  NaN  12.0  17.0
2  NaN  7.0  13.0  18.0
3  4.0  8.0  14.0   NaN
4  5.0  9.0  15.0  20.0
```

In [206…
```
df
```

Out[206…

	A	b	c	d
0	1.0	3.0	NaN	16.0
1	2.0	NaN	12.0	17.0
2	NaN	7.0	13.0	18.0
3	4.0	8.0	14.0	NaN
4	5.0	9.0	15.0	20.0

In [207…
```
df.isnull()
```

Out[207…

	A	b	c	d
0	False	False	True	False
1	False	True	False	False
2	True	False	False	False
3	False	False	False	True
4	False	False	False	False

In [208…
```
#to see how much null value in coloumn
df.isnull().sum()
```

Out[208…
```
A    1
b    1
c    1
d    1
dtype: int64
```

In [209…
```
data = {
    'A': [1,2,100,4,5],
    'b': [3,np.nan,7,8,9],
    'c': [np.nan,12,13,14,15],
    'd': [16,17,18,np.nan,20]
}
```

In [210…
```
df.dropna() #row which does not have null value
```

Out[210…

	A	b	c	d
4	5.0	9.0	15.0	20.0

In [212…
```
df.dropna(axis = 1)#to drop colomn who has null valuue
```

Out[212…

0
1
2
3
4

In [213…
```
df['b']
```

Out[213…
```
0    3.0
1    NaN
2    7.0
3    8.0
4    9.0
Name: b, dtype: float64
```

In [214…
```
df['b'].fillna(df['b'].mean())
```

Out[214…
```
0    3.00
1    6.75
2    7.00
3    8.00
4    9.00
Name: b, dtype: float64
```

In [215…
```
df['b'].mean()
```

Out[215… 6.75

In [216…
```
df['c']
```

Out[216…
```
0     NaN
1    12.0
2    13.0
3    14.0
4    15.0
Name: c, dtype: float64
```

In [217…
```
df['c'].fillna(100)
```

Out[217…
```
0    100.0
1     12.0
2     13.0
3     14.0
4     15.0
Name: c, dtype: float64
```

In [218…
```
df['d'].median()
```

Out[218… 17.5

In [219…
```
df['d'].fillna(df['d'].median())
```

Out[219…
```
0    16.0
1    17.0
2    18.0
3    17.5
4    20.0
Name: d, dtype: float64
```

In [220…
```
df
```

Out[220…

	A	b	c	d
0	1.0	3.0	NaN	16.0
1	2.0	NaN	12.0	17.0
2	NaN	7.0	13.0	18.0
3	4.0	8.0	14.0	NaN
4	5.0	9.0	15.0	20.0

In [179…
```
import seaborn as sns
df = sns.load_dataset('titanic')
df.head()
```

Out[179…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [180…
```
df.isnull().sum()
```

Out[180…
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [181…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 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          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [182…
```
df.shape
```

Out[182…
```
(891, 15)
```

In [183…
```
df.dropna()
```

Out[183…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton	yes	False
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton	yes	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False	D	Southampton	yes	False
872	0	1	male	33.0	0	0	5.0000	S	First	man	True	B	Southampton	no	True
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	C	Cherbourg	yes	False
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True

182 rows × 15 columns

In [184…
```
#after droping null value now shape change
df.shape
```

Out[184…
```
(891, 15)
```

In [185…
```
#coloumn wise drop

df.dropna(axis = 1)
```

Out[185…

	survived	pclass	sex	sibsp	parch	fare	class	who	adult_male	alive	alone
0	0	3	male	1	0	7.2500	Third	man	True	no	False
1	1	1	female	1	0	71.2833	First	woman	False	yes	False
2	1	3	female	0	0	7.9250	Third	woman	False	yes	True
3	1	1	female	1	0	53.1000	First	woman	False	yes	False
4	0	3	male	0	0	8.0500	Third	man	True	no	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	0	0	13.0000	Second	man	True	no	True
887	1	1	female	0	0	30.0000	First	woman	False	yes	True
888	0	3	female	1	2	23.4500	Third	woman	False	no	False
889	1	1	male	0	0	30.0000	First	man	True	yes	True
890	0	3	male	0	0	7.7500	Third	man	True	no	True

891 rows × 11 columns

In [186…
```
df_updated1 = df.dropna(axis =1)
df_updated1.shape
```

Out[186… (891, 11)

In [187…
```
#imputation of missing value
df
```

Out[187…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alo
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fal
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fal
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tr
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fal
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tr
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	Tr
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	Tr
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	Fal
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	Tr
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	Tr

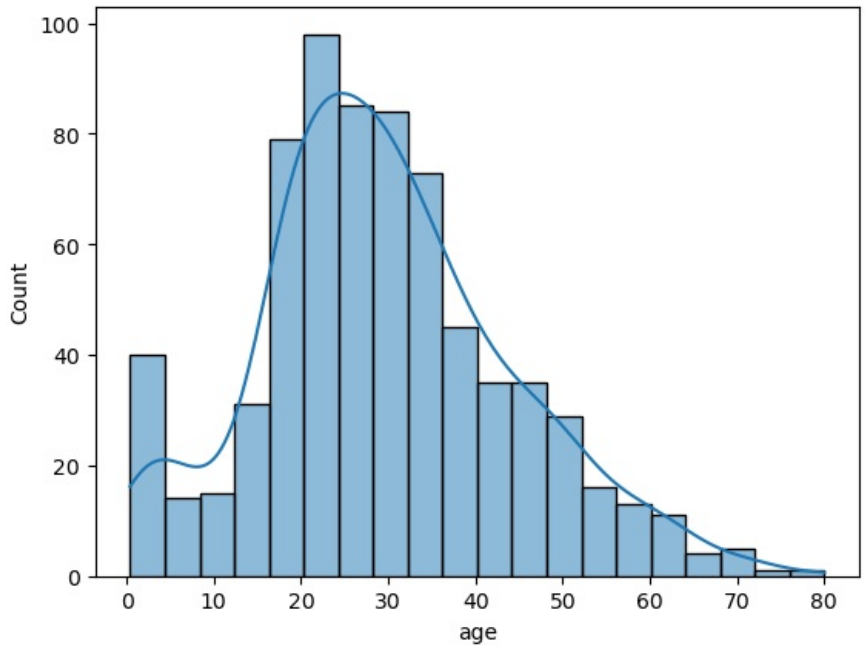
891 rows × 15 columns

In [188…
```
df.isnull().sum() #age is missing at random
```

Out[188…
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [189…
```
sns.histplot(df.age,kde = True)
```

Out[189… <Axes: xlabel='age', ylabel='Count'>



In [193…
```
#whenever we have nornmal distribution data  we should impute with mean
 df['mean_imputation'] = df['age'].fillna(df['age'].mean())
```

```
  Cell In[193], line 2
    df['mean_imputation'] = df['age'].fillna(df['age'].mean())
    ^
IndentationError: unexpected indent
```

In [191…
```
df[['age','mean_imputation']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[191], line 1
----> 1 df[['age','mean_imputation']]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:4108, in DataFrame.__getitem__(self, key)
   4106     if is_iterator(key):
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6200, in Index._get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6252, in Index._raise_if_missing(self, key, indexer, axis_name)
   6249     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6251 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 6252 raise KeyError(f"{not_found} not in index")

KeyError: "['mean_imputation'] not in index"
```

In [ ]:
```
#imputatiion with mode #when categrical data
```

In [ ]:
```
df.columns
```

In [ ]:
```
df.isnull().sum()
```

In [ ]:
```
df['embarked'].isnull()
```

In [194…
```
# to see row wise
df[df['embarked'].isnull()]
```

Out[194…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
61	1	1	female	38.0	0	0	80.0	NaN	First	woman	False	B	NaN	yes	True
829	1	1	female	62.0	0	0	80.0	NaN	First	woman	False	B	NaN	yes	True

In [195…
```
df['embarked'].unique()
```

Out[195…
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [196…
```
# to see how many value are  null value
df.notna().sum()
```

Out[196…
```
survived       891
pclass         891
sex            891
age            714
sibsp          891
parch          891
fare           891
embarked       889
class          891
who            891
adult_male     891
deck           203
embark_town    889
alive          891
alone          891
dtype: int64
```

In [197…
```
df[df['embarked'].notna()]['embarked'].mode()
```

Out[197…
```
0    S
Name: embarked, dtype: object
```

In [198…
```
df[df['embarked'].notna()]['embarked'].mode()[0]
```

Out[198…
```
'S'
```

In [202…
```
mode_value = df[df['embarked'].notna()]['embarked'].mode()[0]
```

In [203…
```
df ['udated_embarked'] = df['embarked'].fillna(mode_value)
```

In [204…
```
df ['udated_embarked','embarked']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('udated_embarked', 'embarked')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[204], line 1
----> 1 df ['udated_embarked','embarked']

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: ('udated_embarked', 'embarked')
```

In [150…
```
#class imbalance>> when class has a higher percentage.er percentage
```

In [151…
```
#handle class imbalance probelm
#undersampling
#oversampling
#smote
np.random.seed(1)#it will generate same number again and again
```

In [152…
```
np.random.seed(1)
no_sample = 1000
class_0_ratio = 0.9  #1000*90 #90 percent of 1000
no_class_0 = 900     #1000*0.9
no_class_1 = 100      #1000*%10 #10 perent of 1000
```

In [153…
```
no_class_0
```

Out[153… 900

In [154…
```
no_class_1
```

Out[154… 100

In [155…
```
class_0 = {'feature1': np.random.normal(0,1,no_class_0),   #mean value 0 ,standard deviation value = 1,size = n
           'feature2': np.random.normal(0,1,no_class_0),
           'target': [0]*no_class_0}
```

In [156…
```
class_0 = pd.DataFrame(class_0)
```

In [157…
```
class_0
```

Out[157…

	feature1	feature2	target
0	1.624345	-0.446699	0
1	-0.611756	0.204377	0
2	-0.528172	0.612233	0
3	-1.072969	0.744885	0
4	0.865408	-0.036281	0
...	...	...	...
895	0.578464	0.833679	0
896	-0.961264	2.160456	0
897	-1.458324	1.998992	0
898	0.494342	0.764041	0
899	-1.494194	1.687255	0

900 rows × 3 columns

In [158…
```
class_1 = {'feature1': np.random.normal(3,1,no_class_1),   #mean value 0 ,standard deviation value = 1,size = n
           'feature1': np.random.normal(3,1,no_class_1),
           'target': [1]*no_class_1}
```

In [159…
```
class_1 = pd.DataFrame(class_1)
```

In [160…
```
class_1
```

Out[160…

	feature1	target
0	3.933630	1
1	1.236225	1
2	2.589783	1
3	2.546333	1
4	2.410570	1
...	...	...
95	3.188583	1
96	3.560918	1
97	2.078341	1
98	3.647375	1
99	4.386826	1

100 rows × 2 columns

In [161…
```
df = pd.concat([class_0,class_1])
```

In [162…
```
df
```

Out[162…

	feature1	feature2	target
0	1.624345	-0.446699	0
1	-0.611756	0.204377	0
2	-0.528172	0.612233	0
3	-1.072969	0.744885	0
4	0.865408	-0.036281	0
...	...	...	...
95	3.188583	NaN	1
96	3.560918	NaN	1
97	2.078341	NaN	1
98	3.647375	NaN	1
99	4.386826	NaN	1

1000 rows × 3 columns

In [163…
```
df = pd.concat([class_0,class_1]).reset_index(drop = True)
df
```

Out[163…

	feature1	feature2	target
0	1.624345	-0.446699	0
1	-0.611756	0.204377	0
2	-0.528172	0.612233	0
3	-1.072969	0.744885	0
4	0.865408	-0.036281	0
...	...	...	...
995	3.188583	NaN	1
996	3.560918	NaN	1
997	2.078341	NaN	1
998	3.647375	NaN	1
999	4.386826	NaN	1

1000 rows × 3 columns

In [164…
```
df.target.value_counts()
```

Out[164…
```
target
0    900
1    100
Name: count, dtype: int64
```

In [175…
```
#upsampling
df_majority = df[df.target == 0]
df_minority = df[df.target == 1]
```

In [176…
```
df_majority
```

Out[176…

	feature1	feature2	target
0	1.624345	-0.446699	0
1	-0.611756	0.204377	0
2	-0.528172	0.612233	0
3	-1.072969	0.744885	0
4	0.865408	-0.036281	0
...	...	...	...
895	0.578464	0.833679	0
896	-0.961264	2.160456	0
897	-1.458324	1.998992	0
898	0.494342	0.764041	0
899	-1.494194	1.687255	0

900 rows × 3 columns

In [177…
```
df_minority_upsampled = resample(df_miniority,replace = True,n_samples = len(df_majority),random_state = 1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[177], line 1
----> 1 df_minority_upsampled = resample(df_miniority,replace = True,n_samples = len(df_majority),random_state =
1)

NameError: name 'resample' is not defined
```

In [178…
```
df_minority_upsampled
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[178], line 1
----> 1 df_minority_upsampled

NameError: name 'df_minority_upsampled' is not defined
```

In [179…
```
df_minority_upsampled .shape
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[179], line 1
----> 1 df_minority_upsampled .shape

NameError: name 'df_minority_upsampled' is not defined
```

In [180…
```
df_minority_upsampled .head
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[180], line 1
----> 1 df_minority_upsampled .head

NameError: name 'df_minority_upsampled' is not defined
```

In [181…
```
#down_sample
```

In [182…
```
df_majority_downsample = resample(df_majority,replace = False,n_samples = len(df_minority),random_state = 1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[182], line 1
----> 1 df_majority_downsample = resample(df_majority,replace = False,n_samples = len(df_minority),random_state
= 1)

NameError: name 'resample' is not defined
```

In [183…
```
df_majority_downsample
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[183], line 1
----> 1 df_majority_downsample

NameError: name 'df_majority_downsample' is not defined
```

In [184…
```
df_downsample = pd.concat([df_minority,df_majority_downsample])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[184], line 1
----> 1 df_downsample = pd.concat([df_minority,df_majority_downsample])

NameError: name 'df_majority_downsample' is not defined
```

In [185…
```
df_downsample
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[185], line 1
----> 1 df_downsample

NameError: name 'df_downsample' is not defined
```

In [186…
```
df_downsample.target.value_counts()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[186], line 1
----> 1 df_downsample.target.value_counts()

NameError: name 'df_downsample' is not defined
```

In [187…
```
#smote
from sklearn.datasets import make_classification
```

In [188…
```
x,y = make_classification(n_samples = 1000, n_redundant=0, n_features=2, n_clusters_per_class = 1,weights =[0.9
```

In [189…
```
x
```

Out[189…
```
array([[ 1.53682958, -1.39869399],
       [ 1.55110839,  1.81032905],
       [ 1.29361936,  1.01094607],
       ...,
       [-0.55662536, -0.15983725],
       [ 1.00499902,  0.93628981],
       [ 1.46210987,  1.14497791]])
```

In [190…
```
y
```

Out[190…
```
array([1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [191…
```
len(y[y==0])
```

Out[191… 894

In [192…
```
df1 = pd.DataFrame(x,columns = ['f1','f2'])
df2 = pd.DataFrame(y,columns = ['target'])
final_df = pd.concat([df1,df2],axis = 1)
```

In [193…
```
final_df
```

Out[193…

	f1	f2	target
0	1.536830	-1.398694	1
1	1.551108	1.810329	0
2	1.293619	1.010946	0
3	1.119889	1.632518	0
4	1.042356	1.121529	0
...	...	...	...
995	2.210439	2.006772	0
996	1.910941	2.011860	0
997	-0.556625	-0.159837	0
998	1.004999	0.936290	0
999	1.462110	1.144978	0

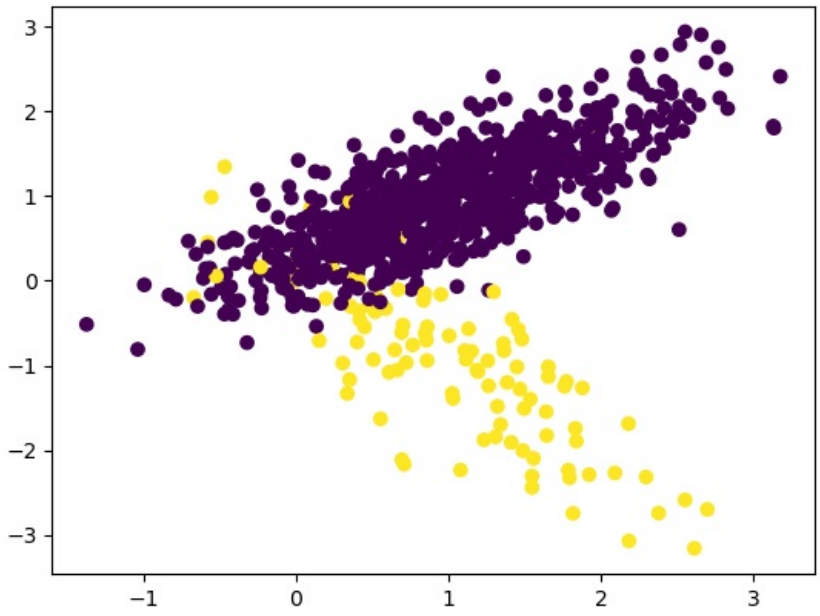
1000 rows × 3 columns

In [194…
```
final_df.target.value_counts()
```

Out[194…
```
target
0    894
1    106
Name: count, dtype: int64
```

In [195…
```
import matplotlib.pyplot as plt
plt.scatter(final_df['f1'],final_df['f2'],c = final_df['target'])
```

Out[195…
```
<matplotlib.collections.PathCollection at 0x1ce46217bc0>
```



In [196…
```
!pip install imblearn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: imblearn in c:\users\mdmaz\appdata\roaming\python\python312\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\programdata\anaconda3\lib\site-packages (from imblearn) (0
.12.3)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-lear
n->imblearn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn
->imblearn) (1.13.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from imbalance
d-learn->imblearn) (1.5.1)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-lear
n->imblearn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanc
ed-learn->imblearn) (3.5.0)
```

In [197…
```
from imblearn.over_sampling import SMOTE
```

In [198…
```
oversample = SMOTE()
```

In [199…
```
x,y =  oversample.fit_resample(final_df[['f1','f2']],final_df['target'])
```

In [200…
```
x.shape
```

Out[200…
```
(1788, 2)
```

In [201…
```
y.shape
```

Out[201…
```
(1788,)
```

In [202…
```
len(y[y==0])
```

Out[202…
```
894
```

In [203…
```
len(x[x==0])
```

Out[203…
```
1788
```

In [204…
```
df1 = pd.DataFrame(x,columns = ['f1','f2'])
df2 = pd.DataFrame(y,columns = ['target'])
oversample_df = pd.concat([df1,df2],axis = 1)
oversample_df
```

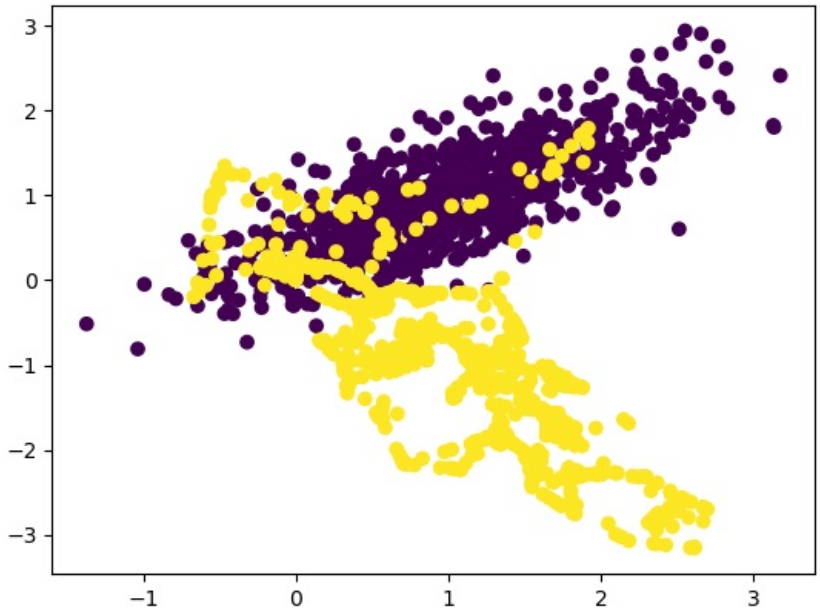
Out[204…

	f1	f2	target
0	1.536830	-1.398694	1
1	1.551108	1.810329	0
2	1.293619	1.010946	0
3	1.119889	1.632518	0
4	1.042356	1.121529	0
...	...	...	...
1783	0.379066	0.905933	1
1784	1.756310	-2.345462	1
1785	2.334889	-2.820627	1
1786	1.104103	-0.147551	1
1787	1.028514	-1.372499	1

1788 rows × 3 columns

In [205…
```
plt.scatter(oversample_df['f1'],oversample_df['f2'],c = oversample_df['target'])
```

Out[205… <matplotlib.collections.PathCollection at 0x1ce468b6e40>

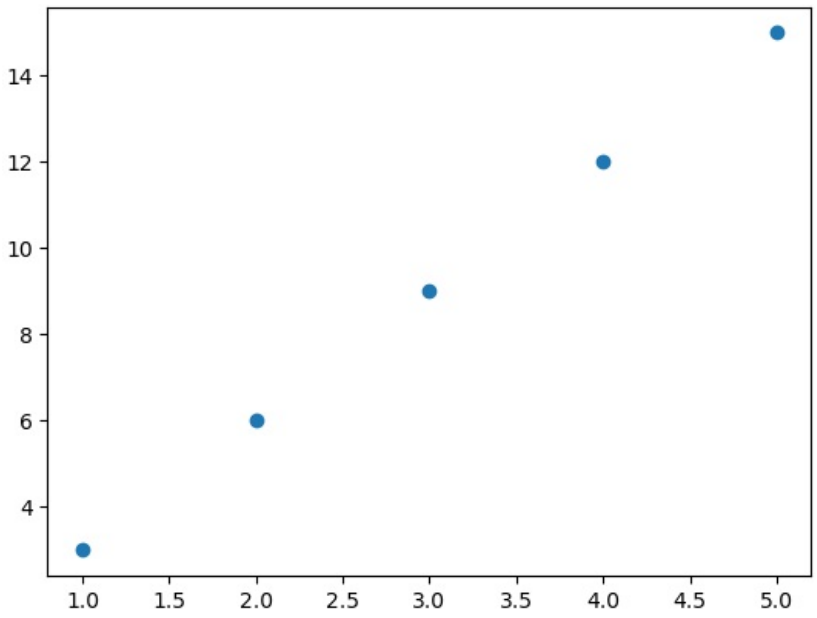


In [206…
```
#data interpolation>> is of estimating data between given range
```

In [207…
```
#1. linear interpolation
#2. cubic interpolation
#3. polynomial interpolation
```

In [208…
```
#linear interpolation
x = np.array([1,2,3,4,5])
y = np.array([3,6,9,12,15])
plt.scatter(x,y)
```

Out[208… <matplotlib.collections.PathCollection at 0x1ce4688d5b0>



In [209…
```
x_new = np.linspace(2,6,10)
```

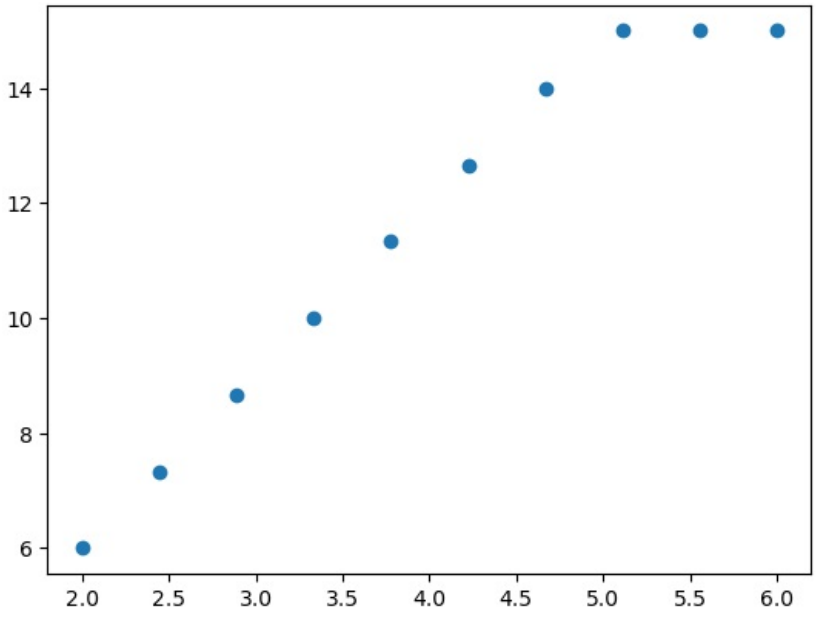
In [210…
```
y_interp = np.interp(x_new,x,y)
```

In [211…
```
y_interp
```

Out[211…
```
array([ 6.        ,  7.33333333,  8.66666667, 10.        , 11.33333333,
       12.66666667, 14.        , 15.        , 15.        , 15.        ])
```

In [212…
```
plt.scatter(x_new,y_interp)
```

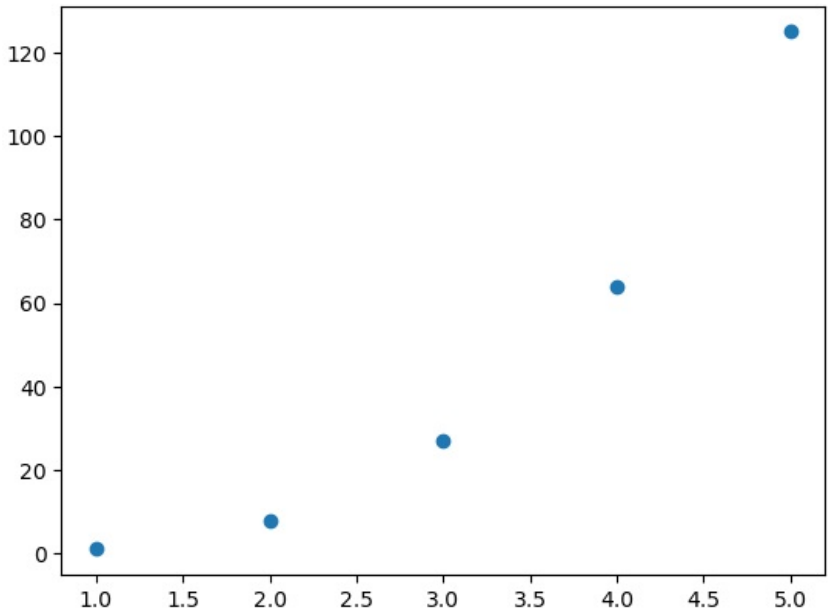
Out[212…
```
<matplotlib.collections.PathCollection at 0x1ce46ae0740>
```



In [213…
```
x = np.array([1,2,3,4,5])
```

```
y = np.array([1,8,27,64,125])
plt.scatter(x,y)
```

Out[213… <matplotlib.collections.PathCollection at 0x1ce4688ef60>



In [255…
```
from scipy.interpolate import interp1d
```

In [256…
```
f = interp1d(x,y, kind = 'cubic')
```

In [257…
```
x_new = np.linspace(1,5,10)
y_interp = f(x_new)
```

In [258…
```
y_interp
```

Out[258… array([1.   , 0.1659808 , 3.01920439, 6.92592593, 9.25240055,
7.52949246, 3.07407407, 0.98902606, 6.54183813, 25.   ])

plt.scatter(x_new,y_interp)

In [ ]:
```
```

In [259…
```
#polynomial interpolation
```

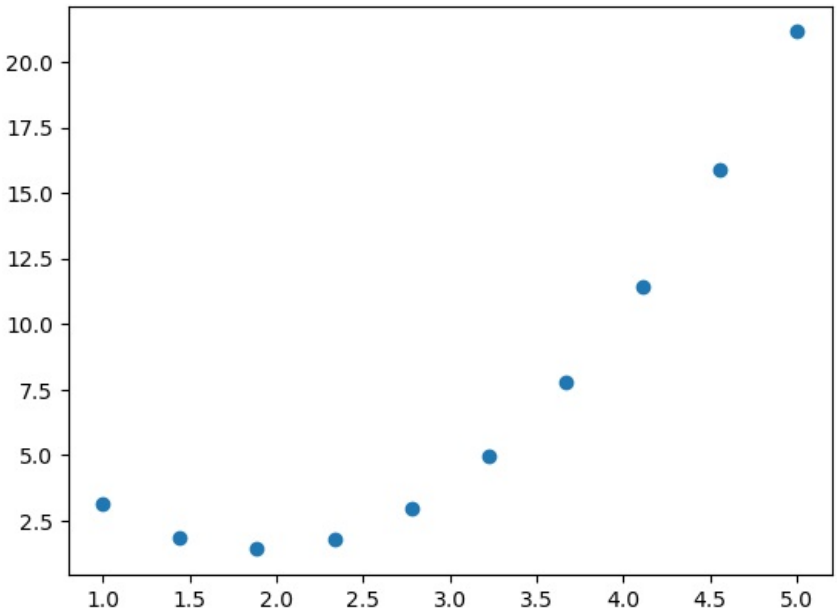
In [260…
```
x =np.array([1,2,3,4,5])
y = np.array([1,4,9,1,25])
```

In [261…
```
p = np.polyfit(x,y,2)
```

In [262…
```
x_new = np.linspace(1,5,10)
y_interp = np.polyval(p,x_new)
```

In [263…
```
plt.scatter(x_new,y_interp)
```

Out[263… <matplotlib.collections.PathCollection at 0x1ce47d881a0>



In [264…
```
import warnings
warnings.filterwarnings('ignore')
salary = [11,40,50,40,68,78,90,74,91,92,88,57,89,99,39,49]
```

In [265…
```
np.quantile(salary,[0,0.25,.50,.75,.1])
```

Out[265… array([11. , 46.75, 71. , 89.25, 39.5])

In [266…
```
df = pd.DataFrame(salary,columns = ['salary'])
```

In [267…
```
df.describe()  # to see 5 point summary
```

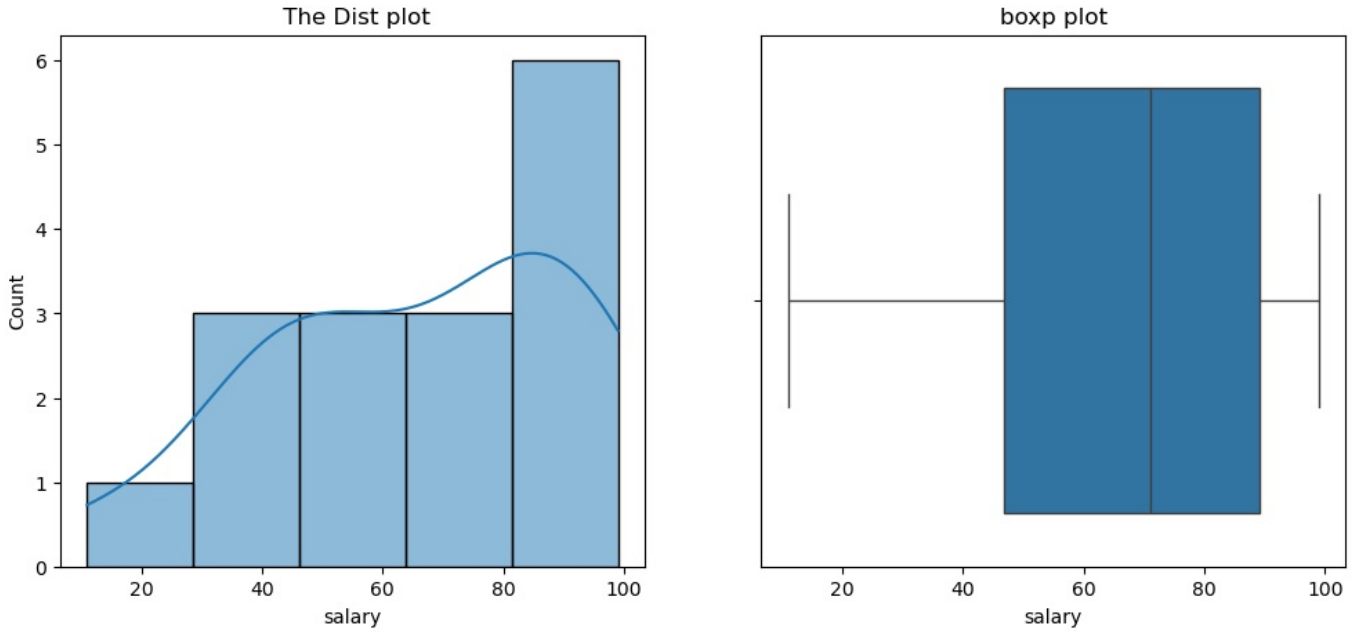
Out[267…

	salary
count	16.000000
mean	65.937500
std	25.720209
min	11.000000
25%	46.750000
50%	71.000000
75%	89.250000
max	99.000000

In [268…
```
#to check outlier
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
sns.histplot(df['salary'],kde = True)
plt.title(" The Dist plot")
```

```
plt.subplot(1,2,2)
sns.boxplot(data = df,x = 'salary')
plt.title("boxp plot")
```

Out[268… Text(0.5, 1.0, 'boxp plot')



In [269…
```
#dropping the outlier
Q1 =df['salary'].quantile(0.25)
Q3 = df['salary'].quantile(0.75)
IQR = Q3-Q1
lower_fence = Q1 - 1.5 *IQR
upper_fence = Q3 + 1.5 *IQR
```

In [270…
```
lower_fence
```

Out[270… -17.0

In [271…
```
upper_fence
```

Out[271… 153.0

In [272…
```
df_filtered = df[(df.salary>= lower_fence) & (df.salary<=upper_fence)]
```

In [273…
```
df_filtered
```

Out[273…

	salary
0	11
1	40
2	50
3	40
4	68
5	78
6	90
7	74
8	91
9	92
10	88
11	57
12	89
13	99
14	39
15	49

In [274…
```
df.shape
```

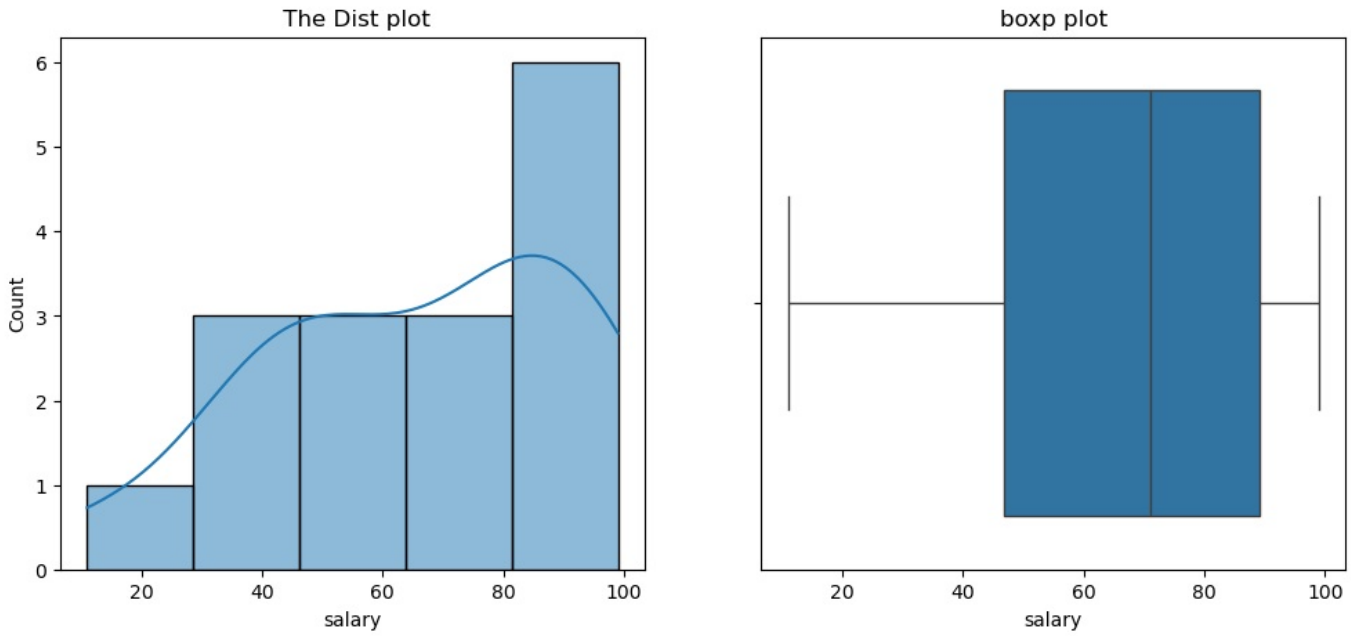
Out[274… (16, 1)

In [275…
```
df_filtered.shape
```

Out[275… (16, 1)

In [276…
```
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
sns.histplot(df_filtered['salary'],kde = True)
plt.title(" The Dist plot")

plt.subplot(1,2,2)
sns.boxplot(data = df_filtered,x = 'salary')
plt.title("boxp plot")
```

Out[276… Text(0.5, 1.0, 'boxp plot')



In [277…
```
#imputtion with mean & media   #to replace outlier
df['salary_imputed_mean'] = np.where((df.salary >= upper_fence)| (df.salary <= lower_fence), df['salary'].mean(
```

In [278…
```
df
```

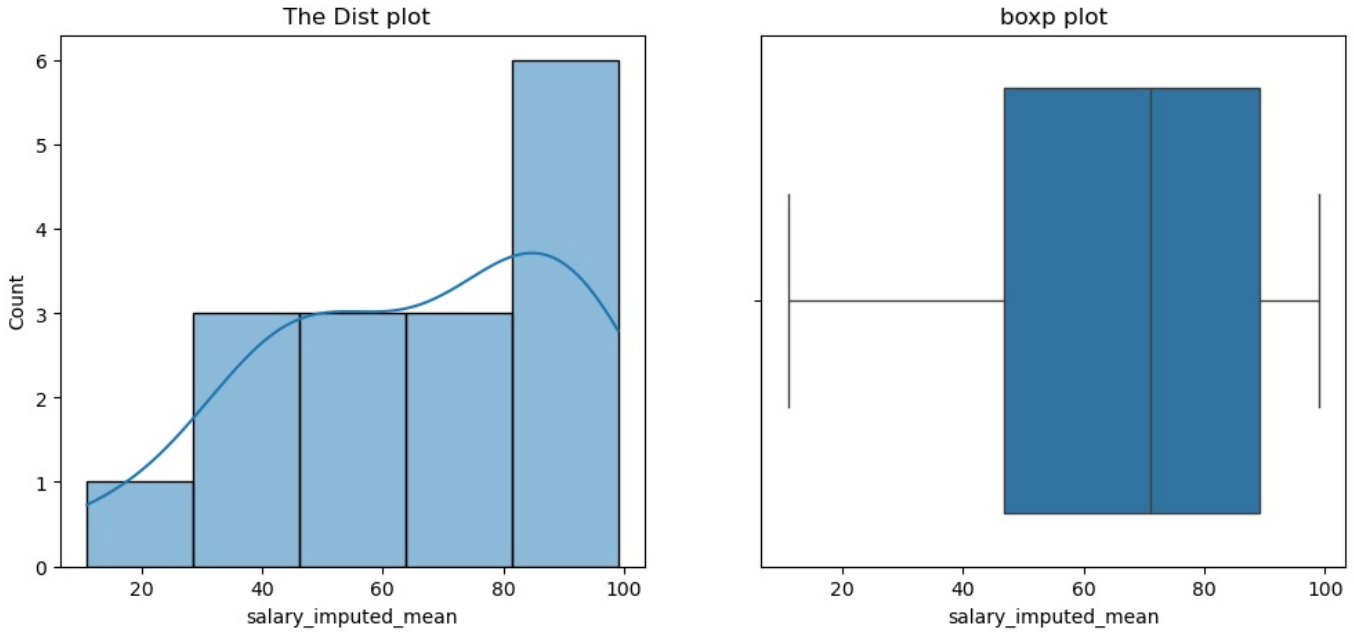
Out[278…

	salary	salary_imputed_mean
0	11	11.0
1	40	40.0
2	50	50.0
3	40	40.0
4	68	68.0
5	78	78.0
6	90	90.0
7	74	74.0
8	91	91.0
9	92	92.0
10	88	88.0
11	57	57.0
12	89	89.0
13	99	99.0
14	39	39.0
15	49	49.0

In [279…
```
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
sns.histplot(df['salary_imputed_mean'],kde = True)
plt.title(" The Dist plot")
```

```
plt.subplot(1,2,2)
sns.boxplot(data = df,x = 'salary_imputed_mean')
plt.title("boxp plot")
```

Out[279… Text(0.5, 1.0, 'boxp plot')



In [290…
```
# to replace with median
df['salary_imputed_median'] = np.where((df.salary >= upper_fence)| (df.salary <= lower_fence), df['salary'].med
```

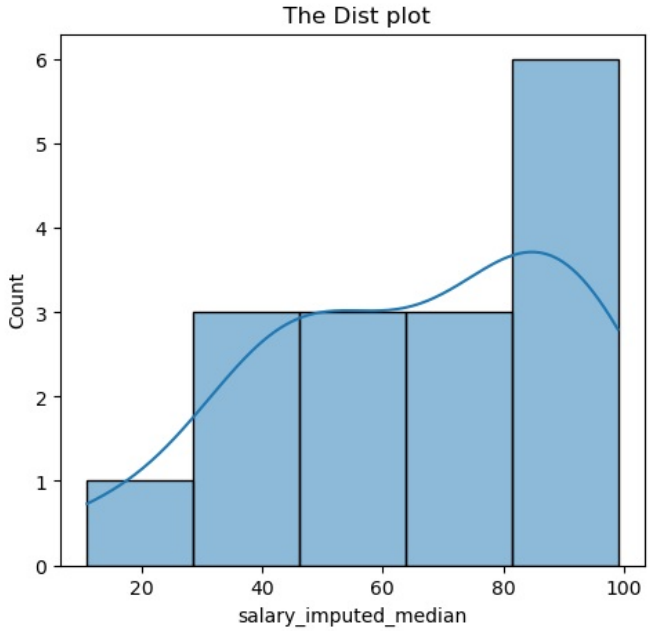
In [291…
```
df
```

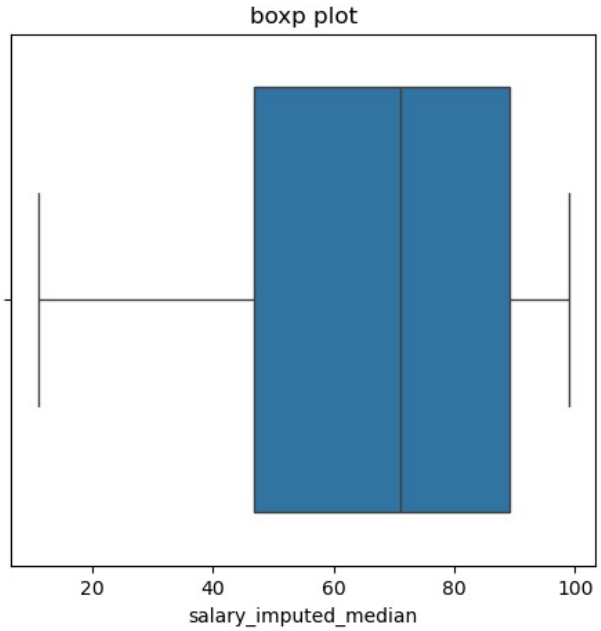
Out[291…

	salary	salary_imputed_mean	salary_imputed_median
0	11	11.0	11.0
1	40	40.0	40.0
2	50	50.0	50.0
3	40	40.0	40.0
4	68	68.0	68.0
5	78	78.0	78.0
6	90	90.0	90.0
7	74	74.0	74.0
8	91	91.0	91.0
9	92	92.0	92.0
10	88	88.0	88.0
11	57	57.0	57.0
12	89	89.0	89.0
13	99	99.0	99.0
14	39	39.0	39.0
15	49	49.0	49.0

In [292…
```
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
sns.histplot(df['salary_imputed_median'],kde = True)
plt.title(" The Dist plot")

plt.subplot(1,2,2)
sns.boxplot(data = df,x = 'salary_imputed_median')
plt.title("boxp plot")
```

Out[292… Text(0.5, 1.0, 'boxp plot')





In [293…
```
#capping
df
```

Out[293…

	salary	salary_imputed_mean	salary_imputed_median
0	11	11.0	11.0
1	40	40.0	40.0
2	50	50.0	50.0
3	40	40.0	40.0
4	68	68.0	68.0
5	78	78.0	78.0
6	90	90.0	90.0
7	74	74.0	74.0
8	91	91.0	91.0
9	92	92.0	92.0
10	88	88.0	88.0
11	57	57.0	57.0
12	89	89.0	89.0
13	99	99.0	99.0
14	39	39.0	39.0
15	49	49.0	49.0

In [294…
```
#capping>>replacing with the nearest value which is not outlier
```

In [295…
```
lower_cap = df['salary'].quantile(0.05)  # lower cap with 5th percentile
upper_cap = df['salary'].quantile(0.95 ) #upper cap with 95th percentile
```

In [296…
```
lower_cap
```

Out[296… 32.0

In [297…
```
upper_cap
```

Out[297… 93.75

In [304…
```
#replace the outlier with cap
df['salary_capped']= np.where(df['salary'] <lower_cap,lower_cap,
        np.where(df['salary']> upper_cap,upper_cap,
                 df['salary']))
```

In [305…
```
df
```

Out[305…

	salary	salary_imputed_mean	salary_imputed_median	salary_capped
0	11	11.0	11.0	32.00
1	40	40.0	40.0	40.00
2	50	50.0	50.0	50.00
3	40	40.0	40.0	40.00
4	68	68.0	68.0	68.00
5	78	78.0	78.0	78.00
6	90	90.0	90.0	90.00
7	74	74.0	74.0	74.00
8	91	91.0	91.0	91.00
9	92	92.0	92.0	92.00
10	88	88.0	88.0	88.00
11	57	57.0	57.0	57.00
12	89	89.0	89.0	89.00
13	99	99.0	99.0	93.75
14	39	39.0	39.0	39.00
15	49	49.0	49.0	49.00

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: