In [1]:
```
import numpy as np
```
In [2]:
```
import matplotlib.pyplot as plt
```
In [4]:
```
X = np.array([[1,2],[3,4]])
print(X)
```
```
[[1 2]
 [3 4]]
```
In [9]:
```
print (X.shape)
```
```
(2, 2)
```
In [10]:
```
print(X.size)
```
```
4
```
In [13]:
```
print(X.ndim)
```
```
2
```
In [14]:
```
print (X.dtype)
```
```
int32
```
In [15]:
```
print(type(X))
```
```
<class 'numpy.ndarray'>
```
In [16]:
```
X.size
```
Out[16]:
```
4
```
In [17]:
```
x = np.empty(shape=[2,2], dtype = int)
print(x)
```
```
[[-2145025664         513]
 [          0           0]]
```
In [18]:
```
x = np.zeros(shape=(2,3), dtype='float', order = 'C')
print(x)
```
```
[[0. 0. 0.]
 [0. 0. 0.]]
```
In [19]:
```
x = np.ones(shape=(2,3), dtype = float, order = 'C')
print(x)
```
```
[[1. 1. 1.]
 [1. 1. 1.]]
```
In [20]:
```
x = np.arange(start=100,stop=1000,step=100,dtype=int)
print(x)
```
```
[100 200 300 400 500 600 700 800 900]
```
In [21]:
```
print(np.linspace(1, 10))
```
```
[ 1.          1.18367347  1.36734694  1.55102041  1.73469388  1.91836735
  2.10204082  2.28571429  2.46938776  2.65306122  2.83673469  3.02040816
  3.20408163  3.3877551   3.57142857  3.75510204  3.93877551  4.12244898
  4.30612245  4.48979592  4.67346939  4.85714286  5.04081633  5.2244898
  5.40816327  5.59183673  5.7755102   5.95918367  6.14285714  6.32653061
  6.51020408  6.69387755  6.87755102  7.06122449  7.24489796  7.42857143
  7.6122449   7.79591837  7.97959184  8.16326531  8.34693878  8.53061224
  8.71428571  8.89795918  9.08163265  9.26530612  9.44897959  9.63265306
  9.81632653 10.        ]
```
In [22]:
```
x = np.arange(50,100)
print(x)
```
```
[50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99]
```
In [25]:
```
x[5]
```
Out[25]:
```
55
```
In [26]:
```
print(x[5:70:1])
```
```
[55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99]
```
In [27]:
```
print(x[40:])
```
```
[90 91 92 93 94 95 96 97 98 99]
```
In [32]:
```
x = np.arange(1,10)
```
In [33]:
```
x
```
Out[33]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```
In [34]:
```
x[7]
```
Out[34]:
```
8
```
In [35]:
```
x[0]
```
Out[35]:
```
1
```
In [36]:
```
x = np.array([0, -9, 5, 22, -10, 95])
```
In [38]:
```
arr1 = np.array([3,4,67,45,55])
ref_arr = arr1
ref_arr[3] = 33
```
In [39]:
```
print("Contents of the original array :",arr1)
print("Contents of the referenced array :",ref_arr)
```
```
Contents of the original array : [ 3  4 67 33 55]
Contents of the referenced array : [ 3  4 67 33 55]
```
In [40]:
```
n_arr = np.array([34,2,33,66,89])
```
In [41]:
```
n_arr1  = np.ndarray(5, dtype='object')
n_arr1[:] = n_arr
```
In [42]:
```
n_arr1[1] = 19
n_arr1[4] = 77
```
In [43]:
```
print(n_arr1)
print(n_arr)
```
```
[34 19 33 66 77]
[34  2 33 66 89]
```
In [45]:
```
a=np.array([[0,1],[2,3],[5,6]])
print(a.shape)
```
```
(3, 2)
```
In [46]:
```
a_ = a.reshape((2,3))
print(a_,a_.shape)
```
```
[[0 1 2]
 [3 5 6]] (2, 3)
```
In [47]:
```
x = np.ones((3,4))
```
In [48]:
```
print(x.shape)
```
```
(3, 4)
```
In [49]:
```
x
```
Out[49]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```
In [50]:
```
x_ = x.flatten()
```
In [52]:
```
print(x_)
```
```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```
In [53]:
```
print(x_.shape)
```
```
(12,)
```
In [54]:
```
x = np.ones((3,4,5))
```
In [55]:
```
x
```
Out[55]:
```
array([[[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]]])
```
In [56]:
```
x_=x.ravel()
```
In [57]:
```
x_
```
Out[57]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1.])
```
In [58]:
```
x = np.ones((3,4))
```
In [59]:
```
x
```
Out[59]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```
In [61]:
```
x.shape
```
Out[61]:
```
(3, 4)
```
In [60]:
```
x.T
```
Out[60]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```
In [62]:
```
x.T.shape
```
Out[62]:
```
(4, 3)
```
In [63]:
```
x = np.array([[1,2],[3,4]])
```
In [64]:
```
y = np.linalg.inv(x)
```
In [65]:
```
print('Original array:',x)
print('Inverse array:',y)
```
```
Original array: [[1 2]
 [3 4]]
Inverse array: [[-2.   1. ]
 [ 1.5 -0.5]]
```
In [66]:
```
x = np.ones((2,2))
print (x)
```
```
[[1. 1.]
 [1. 1.]]
```
In [67]:
```
y = np.zeros((2,3))
print(y)
```
```
[[0. 0. 0.]
 [0. 0. 0.]]
```
In [68]:
```
z = np.hstack((x,y))
print(z)
```
```
[[1. 1. 0. 0. 0.]
 [1. 1. 0. 0. 0.]]
```
In [69]:
```
x = np.arange(3)
print('x : \n',x)
y = np.array(['a','b','c'])
print('y : \n',y)

z = np.hstack((x,y))
print('\n\nhstack(x,y) :\n ',z)
print('New shape : ',z.shape)
```
```
x :
 [0 1 2]
y :
 ['a' 'b' 'c']


hstack(x,y) :
  ['0' '1' '2' 'a' 'b' 'c']
New shape :  (6,)
```
In [10]:
```
x1 = np.arange(5)
```
In [11]:
```
print(x1)
```
```
[0 1 2 3 4]
```
In [12]:
```
print(np.cumsum(x1))
```
```
[ 0  1  3  6 10]
```
In [13]:
```
a = np.array(['hyd','mum','hyd','kol','mum'])
print(a)
np.unique(a)
```
```
['hyd' 'mum' 'hyd' 'kol' 'mum']
```
Out[13]:
```
array(['hyd', 'kol', 'mum'], dtype='<U3')
```
In [14]:
```
print(np.unique(a))
```
```
['hyd' 'kol' 'mum']
```
In [15]:
```
len(np.unique(a))
```
Out[15]:
```
3
```
In [16]:
```
x = np.array([[4,5,6], [7,8,9], [1,2,3]])
x
```
Out[16]:
```
array([[4, 5, 6],
       [7, 8, 9],
       [1, 2, 3]])
```
In [17]:
```
print(np.sum(x))
```
```
45
```
In [18]:
```
print(np.min(x))
```
```
1
```
In [19]:
```
print(np.max(x))
```
```
9
```
In [20]:
```
print(np.argmin(x))
```
```
6
```
In [22]:
```
print(np.argmax(x))
```
```
5
```
In [24]:
```
print(np.sort(x,0))
```
```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```
In [26]:
```
a = np.array([[1,2,4,7], [9,88,6,45], [9,76,3,4]])
np.unique(a)
```
Out[26]:
```
array([ 1,  2,  3,  4,  6,  7,  9, 45, 76, 88])
```
In [27]:
```
a = np.array([1,2])
b = np.arange(8)
print(np.in1d(a,b))
```
```
[ True  True]
```
In [28]:
```
a = np.array([1,2])
b = np.arange(8)
print(np.intersect1d(a,b))
```
```
[1 2]
```
In [29]:
```
a = np.array(['a','b'])
b = np.arange(8)
print(np.union1d(a,b))
```
```
['0' '1' '2' '3' '4' '5' '6' '7' 'a' 'b']
```
In [30]:
```
a = np.array([1,2])
b = np.arange(8)
print(np.setdiff1d(a,b))
print(np.setdiff1d(b,a))
```
```
[]
[0 3 4 5 6 7]
```
In [32]:
```
f = 2
type(f)
```
Out[32]:
```
int
```
In [38]:
```
cities = ['Mumbai', 'Bangalore', 'Hyderabad', 'Chennai', 'Delhi']
print(cities)
```
```
['Mumbai', 'Bangalore', 'Hyderabad', 'Chennai', 'Delhi']
```
In [39]:
```
len(cities)
```
Out[39]:
```
5
```
In [40]:
```
cities[1]
```
Out[40]:
```
'Bangalore'
```
In [41]:
```
cities[-1]
```
Out[41]:
```
'Delhi'
```
In [42]:
```
cities[0:3]
```
Out[42]:
```
['Mumbai', 'Bangalore', 'Hyderabad']
```
In [43]:
```
cities[:2]
```
Out[43]:
```
['Mumbai', 'Bangalore']
```
In [44]:
```
ities = ['Mumbai', 'Bangalore', 'Hyderabad', 'Chennai', 'Delhi']
print (cities[0:4:1])
```
```
['Mumbai', 'Bangalore', 'Hyderabad', 'Chennai']
```
In [45]:
```
cities[0:4:2]
```
Out[45]:
```
['Mumbai', 'Hyderabad']
```
In [46]:
```
cities.extend(['Pune','Noida','Gurgaon'])
cities
```
Out[46]:
```
['Mumbai',
 'Bangalore',
 'Hyderabad',
 'Chennai',
 'Delhi',
 'Pune',
 'Noida',
 'Gurgaon']
```
In [52]:
```
cities.sort()
cities
```
Out[52]:
```
['Bangalore',
 'Chennai',
 'Delhi',
 'Gurgaon',
 'Hyderabad',
 'Mumbai',
 'Noida',
 'Pune']
```
In [53]:
```
cities.pop()
cities
```
Out[53]:
```
['Bangalore', 'Chennai', 'Delhi', 'Gurgaon', 'Hyderabad', 'Mumbai', 'Noida']
```
In [54]:
```
cities.remove('Delhi')
cities
```
Out[54]:
```
['Bangalore', 'Chennai', 'Gurgaon', 'Hyderabad', 'Mumbai', 'Noida']
```
In [55]:
```
sample_dict = {'a':'AI','b':'Business','c':'cost','d':'Data'}
print(sample_dict)
```
```
{'a': 'AI', 'b': 'Business', 'c': 'cost', 'd': 'Data'}
```
In [56]:
```
print(sample_dict.get('a'))
sample_dict['a']
```
```
AI
```
Out[56]:
```
'AI'
```
In [57]:
```
sample_dict.keys()
```
Out[57]:
```
dict_keys(['a', 'b', 'c', 'd'])
```
In [58]:
```
sample_dict.values()
```
Out[58]:
```
dict_values(['AI', 'Business', 'cost', 'Data'])
```
In [59]:
```
sample_dict.items()
```
Out[59]:
```
dict_items([('a', 'AI'), ('b', 'Business'), ('c', 'cost'), ('d', 'Data')])
```
In [60]:
```
for item in sample_dict.items():
    print(item)
```
```
('a', 'AI')
('b', 'Business')
('c', 'cost')
('d', 'Data')
```
In [61]:
```
for item in sample_dict.keys():
    print(sample_dict[item])
```
```
AI
Business
cost
Data
```
In [62]:
```
for key in sample_dict.keys():
    print(key)
```
```
a
b
c
d
```
In [63]:
```
sample_dict.pop('a')
```
Out[63]:
```
'AI'
```
In [64]:
```
s = "Hey there! what should this string be?"
print(len(s))
```
```
38
```
In [65]:
```
combine = 'Supervised' + 'Learning'
combine
```
Out[65]:
```
'SupervisedLearning'
```
In [66]:
```
repetition = 'AI!!' * 5
repetition
```
Out[66]:
```
'AI!!AI!!AI!!AI!!AI!!'
```
In [3]:
```
s = "Hey there! what should this string be?"
print(s[:5])
```
```
Hey t
```
In [4]:
```
print(s[12])
```
```
h
```
In [5]:
```
print(s[4:10])
```
```
there!
```
In [11]:
```
print(s[-5:])
```
```
bing?
```
In [6]:
```
s = 'How are you doing?'
s = s[:4]+'do'+s[7:]
s
```
Out[6]:
```
'How do you doing?'
```
In [8]:
```
s = s.replace('do','ab')
print(s)
```
```
How ab you abing?
```
In [10]:
```
split_string = 'Welcome to Supervised Learning'
split_string.upper()
```
Out[10]:
```
'WELCOME TO SUPERVISED LEARNING'
```
In [13]:
```
a = -5
if a < 0:
    print('Negative Number')
elif a > 0:
    print('Positive Number')
else:
    print('Zero')
```
```
Negative Number
```
In [16]:
```
temperature = 42
if temperature > 42:
   print('It is very sunny today')
else:
    print('The weather is ok')
```
```
The weather is ok
```
In [17]:
```
import pandas as pd
```
In [18]:
```
df = pd.read_csv('creditcard.csv')
```
In [19]:
```
df.head()
```
Out[19]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

In [20]:
```
df.describe()
```
Out[20]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927028e-16	-3.137024e-15	...	1.537294e-16	7.959909e-16	5.367590e-16	4.458112e-15	1.4
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.2
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.0
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.1
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.6
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.5
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.5

8 rows × 31 columns

In [21]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```
In [22]:
```
print('Dimensions :',df.shape)
```
```
Dimensions : (284807, 31)
```
In [23]:
```
print('Columns :', df.columns)
```
```
Columns : Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```
In [24]:
```
df.set_index("V1").head()
```
Out[24]:

V1	Time	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
-1.359807	0.0	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1.191857	0.0	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
-1.358354	1.0	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
-0.966272	1.0	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
-1.158233	2.0	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 30 columns

In [25]:
```
df = df.drop(columns='V2')
df.head()
```
Out[25]:

	Time	V1	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 30 columns

In [26]:
```
df_slice = df.iloc[0]
df_slice
```
Out[26]:
```
Time        0.000000
V1         -1.359807
V3          2.536347
V4          1.378155
V5         -0.338321
V6          0.462388
V7          0.239599
V8          0.098698
V9          0.363787
V10         0.090794
V11        -0.551600
V12        -0.617801
V13        -0.991390
V14        -0.311169
V15         1.468177
V16        -0.470401
V17         0.207971
V18         0.025791
V19         0.403993
V20         0.251412
V21        -0.018307
V22         0.277838
V23        -0.110474
V24         0.066928
V25         0.128539
V26        -0.189115
V27         0.133558
V28        -0.021053
Amount    149.620000
Class       0.000000
Name: 0, dtype: float64
```
In [28]:
```
df_slice1 = df.iloc[2,2]
df_slice1
```
Out[28]:
```
1.77320934263119
```
In [29]:
```
df_slice4 = df.iloc[:, 5:10]
df_slice4
```
Out[29]:

	V6	V7	V8	V9	V10
0	0.462388	0.239599	0.098698	0.363787	0.090794
1	-0.082361	-0.078803	0.085102	-0.255425	-0.166974
2	1.800499	0.791461	0.247676	-1.514654	0.207643
3	1.247203	0.237609	0.377436	-1.387024	-0.054952
4	0.095921	0.592941	-0.270533	0.817739	0.753074
...	...	...	...	...	...
284802	-2.606837	-4.918215	7.305334	1.914428	4.356170
284803	1.058415	0.024330	0.294869	0.584800	-0.975926
284804	3.031260	-0.296827	0.708417	0.432454	-0.484782
284805	0.623708	-0.686180	0.679145	0.392087	-0.399126
284806	-0.649617	1.577006	-0.414650	0.486180	-0.915427

284807 rows × 5 columns

In [30]:
```
df_slice5 = df.iloc[:, :6]
df_slice5
```
Out[30]:

	Time	V1	V3	V4	V5	V6
0	0.0	-1.359807	2.536347	1.378155	-0.338321	0.462388
1	0.0	1.191857	0.166480	0.448154	0.060018	-0.082361
2	1.0	-1.358354	1.773209	0.379780	-0.503198	1.800499
3	1.0	-0.966272	1.792993	-0.863291	-0.010309	1.247203
4	2.0	-1.158233	1.548718	0.403034	-0.407193	0.095921
...	...	...	...	...	...	...
284802	172786.0	-11.881118	-9.834783	-2.066656	-5.364473	-2.606837
284803	172787.0	-0.732789	2.035030	-0.738589	0.868229	1.058415
284804	172788.0	1.919565	-3.249640	-0.557828	2.630515	3.031260
284805	172788.0	-0.240440	0.702510	0.689799	-0.377961	0.623708
284806	172792.0	-0.533413	0.703337	-0.506271	-0.012546	-0.649617

284807 rows × 6 columns

In [31]:
```
df_slice_func = df.iloc[lambda x: x.index % 2 == 0]
df_slice_func.head()
```
Out[31]:

	Time	V1	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
2	1.0	-1.358354	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
4	2.0	-1.158233	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
6	4.0	1.229658	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	-0.099254	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	-0.257237	0.034507	0.005168	4.99	0
8	7.0	-0.894286	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	-0.410430	...	-0.073425	-0.268092	-0.204233	1.011592	0.373205	-0.384157	0.011747	0.142404	93.20	0

5 rows × 30 columns

In [32]:
```
df_sl_loc = df.loc[3]
df_sl_loc.head()
```
Out[32]:
```
Time    1.000000
V1     -0.966272
V3      1.792993
V4     -0.863291
V5     -0.010309
Name: 3, dtype: float64
```
In [33]:
```
df_sl_loc1 = df.loc[[2,3,4]]
df_sl_loc1
```
Out[33]:

	Time	V1	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
2	1.0	-1.358354	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

3 rows × 30 columns

In [34]:
```
df['V4'].value_counts()
```
Out[34]:
```
-0.842316    77
 0.635322    77
 0.454212    62
-1.904603    60
-0.578356    53
             ..
 2.834682     1
 0.564523     1
-1.193373     1
-0.711142     1
 1.337592     1
Name: V4, Length: 275663, dtype: int64
```
In [35]:
```
df = df.sort_values(by=['V4'],ascending=[True])
df.head()
```
Out[35]:

	Time	V1	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
180814	124712.0	-0.965429	-2.449502	-5.683171	2.140891	2.758972	0.597103	-0.222219	0.562963	-0.568171	...	0.485689	-0.123492	-0.056423	0.674867	0.030486	-0.971031	0.819602	0.387628	112.65	0
182620	125479.0	-1.787397	-2.962769	-5.600607	1.987672	2.531584	0.052127	0.207669	0.625643	-0.447870	...	0.464088	-0.195545	-0.229420	0.693335	0.054848	-0.966504	0.885878	0.280081	96.82	0
224907	144017.0	-0.459458	-2.326057	-5.560118	2.371447	2.334309	0.112838	1.061090	-0.046589	-1.422468	...	-0.119929	0.167820	-0.115230	0.682162	-0.073734	-0.989914	0.364937	0.195323	4.08	0
132217	79904.0	-0.210836	-0.988081	-5.519697	2.282025	2.371594	0.373510	0.460417	0.602629	-1.131138	...	-0.377830	-0.276392	-0.233912	0.984375	0.138031	-1.095317	0.285157	0.005901	7.80	0
172737	121221.0	-0.607549	-2.095909	-5.416315	2.803473	2.271774	0.529851	0.637980	0.115044	-1.253305	...	-0.062860	0.145264	0.168607	0.668462	-0.033718	-0.995497	0.436403	0.277180	109.99	0

5 rows × 30 columns

In [ ]: