Name: Aanal Patel

ID: A20528001

Network Intrusion Detection System

In [1]:
```
import pandas as pd
import numpy as np
```

In [122]:
```
data = pd.read_csv("C:/Users/91968/Desktop/IIT/2nd_sem/Machine_Learning/Project/IoT-Ne
```

In [3]:
```
data.head(5)
```

Out[3]:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	

5 rows × 45 columns

In [75]:
```
data.shape
```

Out[75]:
```
(175341, 44)
```

In [163]:
```
# Check for duplicate rows in the main dataset, excluding the 'id' column
duplicate_count_excluding_id = data.drop(columns=['id']).duplicated().sum()
duplicate_count_excluding_id
```

Out[163]:
```
67601
```

In [164]:
```
# Check the number of unique 'id' values in the main dataset
unique_id_count = data['id'].nunique()
unique_id_count
```

Out[164]:
```
175341
```

In [5]:
```
data_features = pd.read_csv("C:/Users/91968/Desktop/IIT/2nd_sem/Machine_Learning/Proje
```

In [6]:
```
data_features
```

Out[6]:

	No.	Name	Type	Description
0	1	srcip	nominal	Source IP address
1	2	sport	integer	Source port number
2	3	dstip	nominal	Destination IP address
3	4	dsport	integer	Destination port number
4	5	proto	nominal	Transaction protocol
5	6	state	nominal	Indicates to the state and its dependent proto...
6	7	dur	Float	Record total duration
7	8	sbytes	Integer	Source to destination transaction bytes
8	9	dbytes	Integer	Destination to source transaction bytes
9	10	sttl	Integer	Source to destination time to live value
10	11	dttl	Integer	Destination to source time to live value
11	12	sloss	Integer	Source packets retransmitted or dropped
12	13	dloss	Integer	Destination packets retransmitted or dropped
13	14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and ...
14	15	Sload	Float	Source bits per second
15	16	Dload	Float	Destination bits per second
16	17	Spkts	integer	Source to destination packet count
17	18	Dpkts	integer	Destination to source packet count
18	19	swin	integer	Source TCP window advertisement value
19	20	dwin	integer	Destination TCP window advertisement value
20	21	stcpb	integer	Source TCP base sequence number
21	22	dtcpb	integer	Destination TCP base sequence number
22	23	smeansz	integer	Mean of the ?ow packet size transmitted by the...
23	24	dmeansz	integer	Mean of the ?ow packet size transmitted by the...
24	25	trans_depth	integer	Represents the pipelined depth into the connec...
25	26	res_bdy_len	integer	Actual uncompressed content size of the data t...
26	27	Sjit	Float	Source jitter (mSec)
27	28	Djit	Float	Destination jitter (mSec)
28	29	Stime	Timestamp	record start time
29	30	Ltime	Timestamp	record last time
30	31	Sintpkt	Float	Source interpacket arrival time (mSec)
31	32	Dintpkt	Float	Destination interpacket arrival time (mSec)
32	33	tcprtt	Float	TCP connection setup round-trip time, the sum ...
33	34	synack	Float	TCP connection setup time, the time between th...
34	35	ackdat	Float	TCP connection setup time, the time between th...
35	36	is_sm_ips_ports	Binary	If source (1) and destination (3)IP addresses ...
36	37	ct_state_ttl	Integer	No. for each state (6) according to specific r...

	No.	Name	Type	Description
37	38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and ...
38	39	is_ftp_login	Binary	If the ftp session is accessed by user and pas...
39	40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
40	41	ct_srv_src	integer	No. of connections that contain the same servi...
41	42	ct_srv_dst	integer	No. of connections that contain the same servi...
42	43	ct_dst_ltm	integer	No. of connections of the same destination add...
43	44	ct_src_ ltm	integer	No. of connections of the same source address ...
44	45	ct_src_dport_ltm	integer	No of connections of the same source address (...
45	46	ct_dst_sport_ltm	integer	No of connections of the same destination addr...
46	47	ct_dst_src_ltm	integer	No of connections of the same source (1) and t...
47	48	attack_cat	nominal	The name of each attack category. In this data...
48	49	Label	binary	0 for normal and 1 for attack records

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
id                 0
dur                0
proto              0
service            0
state              0
spkts              0
dpkts              0
sbytes             0
dbytes             0
rate               0
sttl               0
dttl               0
sload              0
dload              0
sloss              0
dloss              0
sinpkt             0
dinpkt             0
sjit               0
djit               0
swin               0
stcpb              0
dtcpb              0
dwin               0
tcprtt             0
synack             0
ackdat             0
smean              0
dmean              0
trans_depth        0
response_body_len  0
ct_srv_src         0
ct_state_ttl       0
ct_dst_ltm         0
ct_src_dport_ltm   0
ct_dst_sport_ltm   0
ct_dst_src_ltm     0
is_ftp_login       0
```

```
ct_ftp_cmd           0
ct_flw_http_mthd     0
ct_src_ltm           0
ct_srv_dst           0
is_sm_ips_ports      0
attack_cat           0
label                0
dtype: int64
```

In [8]:
```
data['service'].unique()
```

Out[8]:
```
array(['-', 'ftp', 'smtp', 'snmp', 'http', 'ftp-data', 'dns', 'ssh',
       'radius', 'pop3', 'dhcp', 'ssl', 'irc'], dtype=object)
```

In [9]:
```
data.columns
```

Out[9]:
```
Index(['id', 'dur', 'proto', 'service', 'state', 'spkts', 'dpkts', 'sbytes',
       'dbytes', 'rate', 'sttl', 'dttl', 'sload', 'dload', 'sloss', 'dloss',
       'sinpkt', 'dinpkt', 'sjit', 'djit', 'swin', 'stcpb', 'dtcpb', 'dwin',
       'tcprtt', 'synack', 'ackdat', 'smean', 'dmean', 'trans_depth',
       'response_body_len', 'ct_srv_src', 'ct_state_ttl', 'ct_dst_ltm',
       'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm',
       'is_ftp_login', 'ct_ftp_cmd', 'ct_flw_http_mthd', 'ct_src_ltm',
       'ct_srv_dst', 'is_sm_ips_ports', 'attack_cat', 'label'],
      dtype='object')
```

In [10]:
```
for i in data.columns:
    print(data[i].unique())
```

```
[     1      2      3 ... 175339 175340 175341]
[0.121478 0.649902 1.623129 ... 3.71911  0.996503 1.557125]
['tcp' 'udp' 'arp' 'ospf' 'icmp' 'igmp' 'rtp' 'ddp' 'ipv6-frag' 'cftp'
 'wsn' 'pvp' 'wb-expak' 'mtp' 'pri-enc' 'sat-mon' 'cphb' 'sun-nd' 'iso-ip'
 'xtp' 'il' 'unas' 'mfe-nsp' '3pc' 'ipv6-route' 'idrp' 'bna' 'swipe'
 'kryptolan' 'cpnx' 'rsvp' 'wb-mon' 'vmtp' 'ib' 'dgp' 'eigrp' 'ax.25'
 'gmtp' 'pnni' 'sep' 'pgm' 'idpr-cmtp' 'zero' 'rvd' 'mobile' 'narp' 'fc'
 'pipe' 'ipcomp' 'ipv6-no' 'sat-expak' 'ipv6-opts' 'snp' 'ipcv'
 'br-sat-mon' 'ttp' 'tcf' 'nsfnet-igp' 'sprite-rpc' 'aes-sp3-d' 'sccopmce'
 'sctp' 'qnx' 'scps' 'etherip' 'aris' 'pim' 'compaq-peer' 'vrrp' 'iatp'
 'stp' 'l2tp' 'srp' 'sm' 'isis' 'smp' 'fire' 'ptp' 'crtp' 'sps'
 'merit-inp' 'idpr' 'skip' 'any' 'larp' 'ipip' 'micp' 'encap' 'ifmp'
 'tp++' 'a/n' 'ipv6' 'i-nlsp' 'ipx-n-ip' 'sdrp' 'tlsp' 'gre' 'mhrp' 'ddx'
 'ippc' 'visa' 'secure-vmtp' 'uti' 'vines' 'crudp' 'iplt' 'ggp' 'ip'
 'ipnip' 'st2' 'argus' 'bbn-rcc' 'egp' 'emcon' 'igp' 'nvp' 'pup' 'xnet'
 'chaos' 'mux' 'dcn' 'hmp' 'prm' 'trunk-1' 'xns-idp' 'leaf-1' 'leaf-2'
 'rdp' 'irtp' 'iso-tp4' 'netblt' 'trunk-2' 'cbt']
['-' 'ftp' 'smtp' 'snmp' 'http' 'ftp-data' 'dns' 'ssh' 'radius' 'pop3'
 'dhcp' 'ssl' 'irc']
['FIN' 'INT' 'CON' 'ECO' 'REQ' 'RST' 'PAR' 'URN' 'no']
[   6   14    8   12   10   62    2   24   28   30    1   16    4   36
   22  122   38   20   66   78   42   52   64   40   80    3   26  110
  106  360  354  366  352  108  362   18  342  332  336  308  322  328
  358  224  234  222  208  200   68   48   90  356   70  348  236   92
  240  452  450  368  434  420  238  432  288  112  650  656  364  548
   50  230  562  372  216  204   60  228  206  350  294  346  232   46
  226  218  296  190  286   88  338   32  344  326  340  318  212  242
  202  214  324  314  292  244  284   58  334  220  306  290  280   11
   54   34   84  128   86  436  422  438  414  440  126  320  198  302
  370  130  398  210  460  448  396  454  400  442  462  464   82  412
  468  456  330  406  446  424  466  402  132  444  428  458  310  268
  276   72  376  426   74  430  282  312  278  266  390  124  270  264
  618  622  298  304  632  274  262  104  254  620  256  628  614  616
```

```
  260  624  272   56  174  176  184  172  178  182  470  180  316   44
  374  170  116  162  158  152  146  142  136  120 1624  250  118 4316
  490  682  150   96  164  154  168  574 5418  160  102 4808   94  378
  144 5494  668  816 4018  166   98  188  140  194 6974   76  246  114
  148 5386  594  156  134  138  512  826  532 1268 4094 7040  644  196
  300  100  954  542 5966 1876  890 2578 1388 2478 1604  584 5404 7292
 9094  186  666 9492  486 4232  598 1062 3086  392  248  702 5708  818
 1634  680  710 1630  706  672  804 2698 1522  252  538  258  630 1664
  488  558  576  832 5370 4026 4694 3758 1254 1832 3974  556 1654  708
 1026 4040  754 1816 4410 1096  192 6680  722  494 1866  566  384  564
 1638 4416 5062 5424  586 8070  864 1138  714 1488 7990  998 5702  740
 2740 2462  792  964 2014 4624 6430  472  776  752  634 3818  686 5666
 4608  704  660  930  742  790  516  410  912 4370 4360 4174 1600 5392
 1248  810 2320  582 5402  552  662 1874 7684  612 9616  514 1072  636
 1142 6836 8882  610 5652 1422  484  608  590 1006 2446 2690  670 4806
 2436  382 4404  716 2922 5382 6262 6616  568 6776  578 1246  592 3074
 1870  530 1704 3844 2228 1110 2004  788  822   21  600 1622 5254 1500
 5442 7226  482 4938 7252  698 1770 7908  694  836 4212 1458 8324  854
 1834 2700 1892  978  696 2734  720 5412 4920  732 4450 1708 1012  524
 2722  798  784 1596  782  678  952 1228 6118 2066 1686 6584 6332 5444
 2750 4458 7194 5376  734 7660 3882  554 5686 3822  658  760  758  800
 5184 1456 1544 1044]
[    4    38    16    12     6     8    28     0    10    58    20     2
    24    18    22   126    30    40    68    66    80    42    54    26
    44    60     3   114    82   746    36    14   116   438   388    70
    50    92   858    64    90   560   112  1278  1078    48    72    94
    78    52     1   400    62    86   124    34   806   122    84   754
    88   128   512    74   594   130   682    76   684   510   192   236
   238   234   232    56   196   240   194   190   106   606    46  1716
    32  1452   252  2940   150   140   402   368    96   704   164   100
   274   242   810   288   324   148   350   160   570  1084   166   638
   158   156   748   230   108   318   412   138   268   110  2150   228
   558   828   132   626   340   286   394   182  2588   162  1036   826
   210   332   118   300   366  1530   170   244   208  3402   674   200
   602  1046   448   226  1234   146    98   346   316  1382   296   254
   304   102   562   168   370   152   362   918   616   310   144   364
   276   224   174   802   308   588   392   154   554   272   250   814
   142   104  1106   690   184   342  1368   656   204   120  1414   328
   702   612   636   222   650   280   198   564  2748   178   486   472
   816  1978  4560   808   860   134   136   306  5254  2636   352   260
   334  4526   800   444  1250   416  2272   248   338   426   220   336
  1594   258   348   270  1226  5164   314   694   528   188   548  1542
   446  1434   822   732   592  1418   282   398   652   216 10974   372
   422   292   668   408   374   202   172   556  2786   660   572  1154
  2144   996   356   320   176   246   516   670   818   780   618   278
  1216  1104   386  6676   878   186   214   566   624   180   492   354
   474   382 10850   390  4396  1394   490  1998   976   376   646   862
  2630  2612   312   738  4550   716   428  1560   812  2646  6088   536
  1052   700   696   360   662  1244   798   482   264   586   796   698
   610   824   326  1206   832   298  1458   544  1174  1272   266   550
   206  3350   600   664   418   420  1842   358  1030  4548  1350   468
   856  9346  3698  3664   344  3696   396   450  1632   432   724   552
   380   322   710  1256   290  2620  3036  1796   946   986  1032  4476
   302  1538   256  1048   330   804   576  3206   648   686  1222  1262
   642   568   436  2076  1088   520   762   378   424  1622  1110  1200
  1502   462  9660  1260  1364   514   434   608   864  1500  1310  2014
   466  6494  1644  3604   714  2642   430   692   476  2306  1388  1264
   928  1018   644   688   964   834  1114   596   680  3314   212  1160
   294  4910  5276   792  2004   384  1742  1656   404  2002  6994]
[   258    734    364 ... 272070  69997  12601]
[   172  42014  13186 ...  68382   3084 426483]
[74.08749  78.473372 14.170161 ... 49.171955 31.468251 33.612649]
[252  62 254   0   1  31  63  64  60  29 255]
[254 252   0  29  60  31]
```

```
[14158.94238   8395.112305  1572.271851 ...  7185.126465 62427.87109
  8826.286133]
[  8495.365234 503571.3125    60929.23047  ... 129476.7813     9586.899414
   4903.492188]
[   0    2    1    3   28    5    4    6   11   12   15   30    7   14
   18   26   21   19   38   55   47   27   35   36   34   24  144   13
   33   37   39   17   54    8   29   53  130  132   10   58   32    9
   88   84   85   83   78   76   73   70   68   65   63   57   51   52
   79   20  133   16  809   56 2157  337   69   45  118   22  140   74
   81   43  171   42  131  284   31 2706  113   97   92   49   77   71
  183  167   48 2403   41   62   44   72  157  186  111 2746  331  406
 2003   80   46   91   59   67   60   94  110   23 3484  135   95   50
  120   86 2690  295  151   75   64   40  105   66  129  410  631 2045
 3518  152  102  221  319  168   25  147   87   96  125  108  473  268
  170 2982  934  438  142 1288  691 1236  182  149  799  289 2699  278
 3643 4546   90  177  329 4745  240 2110  210  296  122   61  528  127
  165  124  313  164 1540  143  192  121 2851  154  340  156   89  349
  332 1346  758  123  266  126  107  137  119  315  830  109  216  114
  244  276  103  106  338  285  413 2682 2011 2343   82 1877  624  913
 1985  207  162  275   99  353  512  230 2019  376  100  138  184  905
 2203  548   93  145 3338  358  930  280  189  279  816 2205  181 2530
 2709  288 4033  451  566  354  741 3993  239  211 2848  196  307  366
 1367  348 1225  395  112  476 1004 2309 3213  233  385  116  375  314
 1906 2832  153 2303  327  464  370  394  257  159 2184 2179 2085  217
  797  146 2693  304  357  622  402 1157 2698  273  325  328  204 3838
  303 4803 2014  532  209  568 3417 4439  302 2822  150  232  241  495
 1220 1341  155 2401 1214  270  101  322  188 2201  355 1458 2688 3129
 3307  175  218  281 3386  294  286  619  339  166  352 1536  932  850
 1922 1111  552  999  391  408  297  808 2623  271  747  334 2718 3607
  238 2468  161 3624  346  881 3951  344  414 2101 4158  115  426  911
 1347  185  305  178  943 1364  356 2703 2457 2225  847  502  261 1358
  213  398  392  795  390  336  287  611 3057 1030  840 3290 3164  219
 2719 1372  399 2226 3592 2687  364 3830 1940 2840 1910  377 2591  726
  770  393  309]
[   0   17    6    3    1    8    2    5    4   27   11   10    9   32
    7   13   26   33   15   14   31  370  197  174   28   18   39  390
  253  583  491  180   24   36  366  342  231   30  269  199   20  369
  230   37   59   67   44  274   16   22  856   25  724   21   12 1467
   73  194  181   46  349  182   35   80  134  119  141   23  160  135
   72   45  172   75  283  540   77  317   74  113   52  157  204   34
  115  132   54   29   53 1072  111  276   95  124  168   56  195 1291
   79   78   19  103  164   57   51  148   64  763   83  120  102 1699
  335  221  615   42   71   68   41   47   43  171  155   40  689  146
   55  125  149   49  279  150  131   76   82  183  178  114   63  306
  136  110   84  398  152  292  275   38   69  192   50  343   86  326
  100   58  277  162  304  109  323  138   97   48  153   89 1368   96
  240  406  123  987 2277  402   70   66  151 2627 1315  173  165 2262
  220  623  116  122  206 1133  167  211  108  166  797  126  611 2576
  118  345   61  262   91  272  769  715   81  707  196  133  106 5484
  144   88  201  332   99  344 1390  284  572 1070   62   93  176  158
  179   85  256  407  307  310  137  606  550  191 3338  295   90  324
  105  234  280  121  244  175  203 5425 2198  694  243  997  117 1312
 1303  154 2273  356  212  142  185  778  403 1320 3039  266  163  524
  703  329  620  397  238  130  288  396  347  303  161  601  414  727
  270  139  294  634  273   92  101 1674  330  184  207  921  177 2272
  521 4672 1846 1831 1848  412  814  265  214  188 1316  410  625 1307
 1517  895 2234  298  767  522  399 1601  341  609  302  629  235  319
  282  147  112  216 1035  257  186  210  805  749 4829  680  254  215
  748  653 1005  301 3246  820 1799  143 1318  255  236 1151  692  233
  555  296   98 1654  104  321   87 2454 2633   65 1000  127  190  869
  825  200  281  140  999 3497]
[ 24.2956    49.915    231.875571 ...  52.447526  57.671293  54.400111]
[  8.375     15.432865 102.737203 ...  33.93738   73.69143   66.98057 ]
[   30.177547    61.426934 17179.58686  ...  3005.256004  3661.213103
```

```
  3721.068786]
[   11.830604  1387.77833  11420.92623  ...  2479.497222   112.41807
   120.177727]
[255   0  31 232  14 192 103  45  87 172 168 167  42]
[ 621772692 1417884146 2116150707 ...    5604755 1932059121 3518776216]
[2202533631 3077387971 2963114973 ...  575257391 2472223109 3453092386]
[255   0 244  70  48  37  40]
[0.       0.111897 0.128381 ... 0.044995 0.045137 0.09944 ]
[0.       0.061458 0.071147 ... 0.025596 0.008571 0.036895]
[0.       0.050439 0.057234 ... 0.037629 0.036566 0.062545]
[  43   52   46 ... 1440 1344  688]
[  43 1106  824 ... 1137  603  747]
[  0   1   2  80 155   3   5   4 163 172  39]
[    0   103   109 ... 12836 18663  3021]
[ 1 43  7 11  2  3  6  5 12 63  4  9  8 14 13 16 10 17 15 19 21 20 22 18
 27 23 37 46 39 28 36 34 29 40 25 31 38 35 52 33 24 26 44 45 30 32 47 41
 42 49 51 50]
[0 1 2 3 6]
[ 1  2  3  6  7 10  5  8  9 12  4 11 13 14 24 15 17 18 16 21 23 37 46 20
 40 25 22 41 35 45 27 19 26 44 31 32 38 33 39 36 43 48 29 34 47 30 28 42
 50 51]
[ 1  2  3  6 10  5  8  9  7  4 11 14 13 15 12 21 17 18 23 37 46 16 20 40
 25 19 33 22 31 27 34 28 26 32 29 30 24 36 45 35 39 42 41 38 50 51 43]
[ 1  2  3 23 37 46 14 20 40  4  5  6  9  8 10  7 18 17 15 16 13 12 11 24
 25 28 22 19 26 21 27 31]
[ 1  2  3 40  4  7  6 63  5  8  9 10 13 11 12 14 19 26 15 23 37 46 39 20
 28 36 34 22 29 16 25 32 38 35 51 33 18 31 41 21 30 17 27 44 43 45 24 54
 42 65 47 49 50 52]
[0 1 2 4]
[0 1 2 4]
[ 0  1  4  2  9 12  6 25 16 30  3]
[ 1  2  3  6 12 10  5  8  4  9 13  7 17 19 11 14 15 16 21 18 24 20 23 37
 46 40 25 22 45 27 26 39 41 44 34 30 28 35 29 42 47 43 31 60 33 36 38 32
 50 51]
[ 1  6 39  3  2  8  5 11  9  7 62 10  4 12 21 17 16 13 18 14 19 15 24 20
 26 23 37 46 28 36 34 22 29 30 40 25 31 38 35 51 33 27 44 42 45 32 41 43
 47 49 50 52]
[0 1]
['Normal' 'Backdoor' 'Analysis' 'Fuzzers' 'Shellcode' 'Reconnaissance'
 'Exploits' 'DoS' 'Worms' 'Generic']
[0 1]
```

In [11]:
```
# WE NEED TO FIX THE SERVICES COLUMN:
```

In [12]:
```
data[data['service'] == '-'].count()
```

Out[12]:
```
id                 94168
dur                94168
proto              94168
service            94168
state              94168
spkts              94168
dpkts              94168
sbytes             94168
dbytes             94168
rate               94168
sttl               94168
dttl               94168
sload              94168
dload              94168
sloss              94168
dloss              94168
```

```
sinpkt              94168
dinpkt              94168
sjit                94168
djit                94168
swin                94168
stcpb               94168
dtcpb               94168
dwin                94168
tcprtt              94168
synack              94168
ackdat              94168
smean               94168
dmean               94168
trans_depth         94168
response_body_len   94168
ct_srv_src          94168
ct_state_ttl        94168
ct_dst_ltm          94168
ct_src_dport_ltm    94168
ct_dst_sport_ltm    94168
ct_dst_src_ltm      94168
is_ftp_login        94168
ct_ftp_cmd          94168
ct_flw_http_mthd    94168
ct_src_ltm          94168
ct_srv_dst          94168
is_sm_ips_ports     94168
attack_cat          94168
label               94168
dtype: int64
```

In [13]:
```
# there are a lot of rows with - as the service, but we cannot just delete it. hence r
```

In [14]:
```
data['service'].replace('-',np.nan,inplace=True)
```

```
C:\Users\anjal\AppData\Local\Temp\ipykernel_24008\1851727398.py:1: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because th
e intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c
ol: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the
operation inplace on the original object.


  data['service'].replace('-',np.nan,inplace=True)
```

In [15]:
```
data['service'].unique()
```

Out[15]:
```
array([nan, 'ftp', 'smtp', 'snmp', 'http', 'ftp-data', 'dns', 'ssh',
       'radius', 'pop3', 'dhcp', 'ssl', 'irc'], dtype=object)
```

In [16]:
```
data.shape
```

Out[16]:
```
(175341, 45)
```

In [ ]:
```

```

In [17]:
```
data.dtypes
```

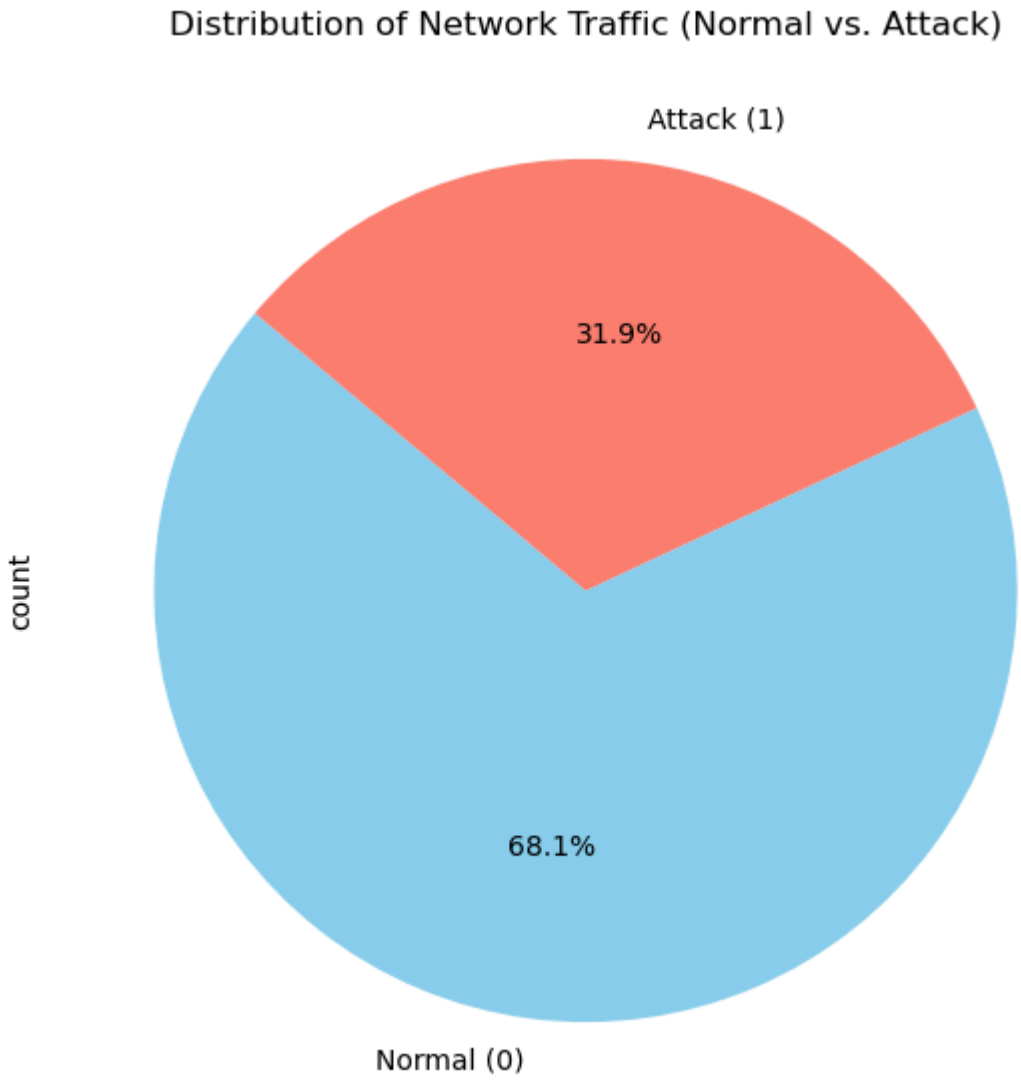
Out[17]:
```
id                   int64
dur                float64
proto               object
service             object
state               object
spkts                int64
dpkts                int64
sbytes               int64
dbytes               int64
rate               float64
sttl                 int64
dttl                 int64
sload              float64
dload              float64
sloss                int64
dloss                int64
sinpkt             float64
dinpkt             float64
sjit               float64
djit               float64
swin                 int64
stcpb                int64
dtcpb                int64
dwin                 int64
tcprtt             float64
synack             float64
ackdat             float64
smean                int64
dmean                int64
trans_depth          int64
response_body_len    int64
ct_srv_src           int64
ct_state_ttl         int64
ct_dst_ltm           int64
ct_src_dport_ltm     int64
ct_dst_sport_ltm     int64
ct_dst_src_ltm       int64
is_ftp_login         int64
ct_ftp_cmd           int64
ct_flw_http_mthd     int64
ct_src_ltm           int64
ct_srv_dst           int64
is_sm_ips_ports      int64
attack_cat          object
label                int64
dtype: object
```

Visualization and EDA

In [18]:
```
# Distribution of the 'label' variable (0 for normal, 1 for attack) as a pie chart
label_distribution = data['label'].value_counts()
```

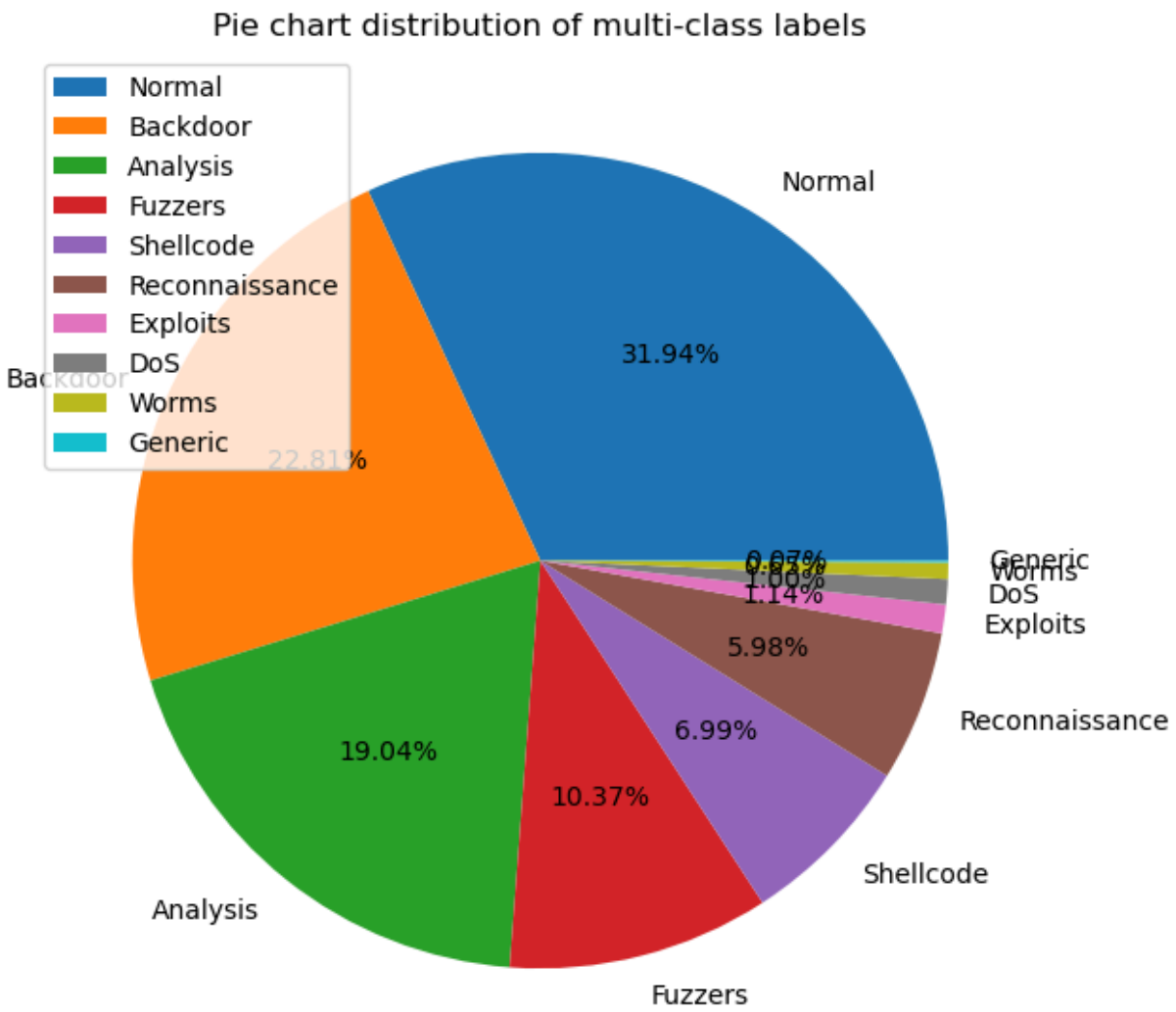
In [19]:
```
# Plotting the pie chart
pie_chart = label_distribution.plot(kind='pie', labels=['Normal (0)', 'Attack (1)'], a
                                    startangle=140, colors=['skyblue', 'salmon'], figs
                                    title='Distribution of Network Traffic (Normal vs.

# Save the plot
```

```
#pie_chart_path = '/mnt/data/label_distribution_pie_chart.png'
#pie_chart.savefig(pie_chart_path)
```



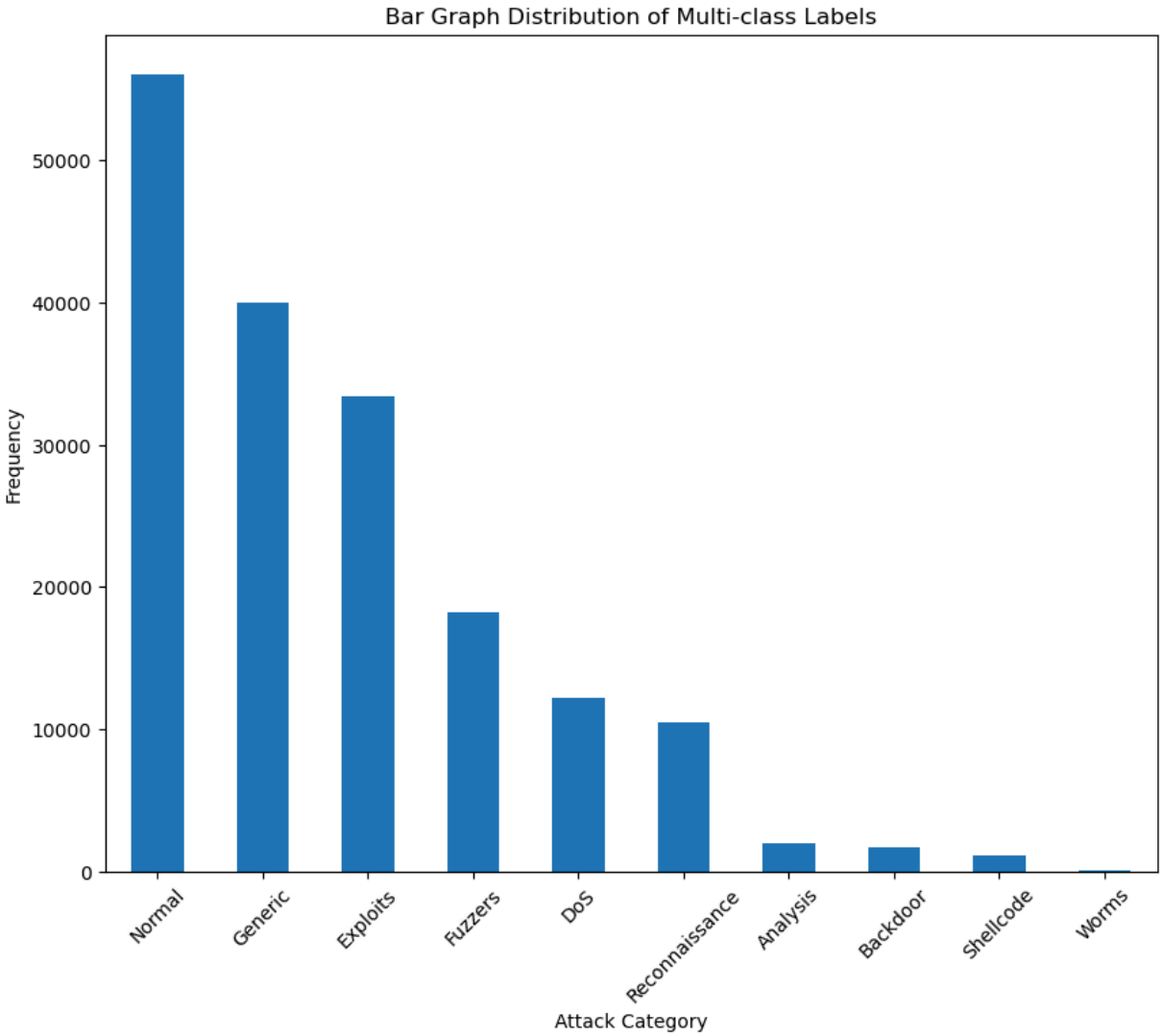
In [20]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7,7))
plt.pie(data.attack_cat.value_counts(),labels=data.attack_cat.unique(),autopct='%0.2f%
plt.title('Pie chart distribution of multi-class labels')
plt.legend(loc='best')
#plt.savefig('plots/Pie_chart_multi.png')
plt.show()
```



In [21]:

```
attack_cat_counts = data['attack_cat'].value_counts()

plt.figure(figsize=(10, 8))
attack_cat_counts.plot(kind='bar')
plt.title('Bar Graph Distribution of Multi-class Labels')
plt.xlabel('Attack Category')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
#plt.savefig('Pie_chart_multi.png')  # Save the figure
plt.show()
```



In [22]:
```
# analyzing the attacked records:

attack_data = data[data['label'] == 1]
```

In [23]:
```
attack_summary_statistics = attack_data.describe()
```

In [24]:
```
# dropping the ID column:

data = data.drop(['id'], axis=1)
```

In [25]:
```
data
```

Out[25]:

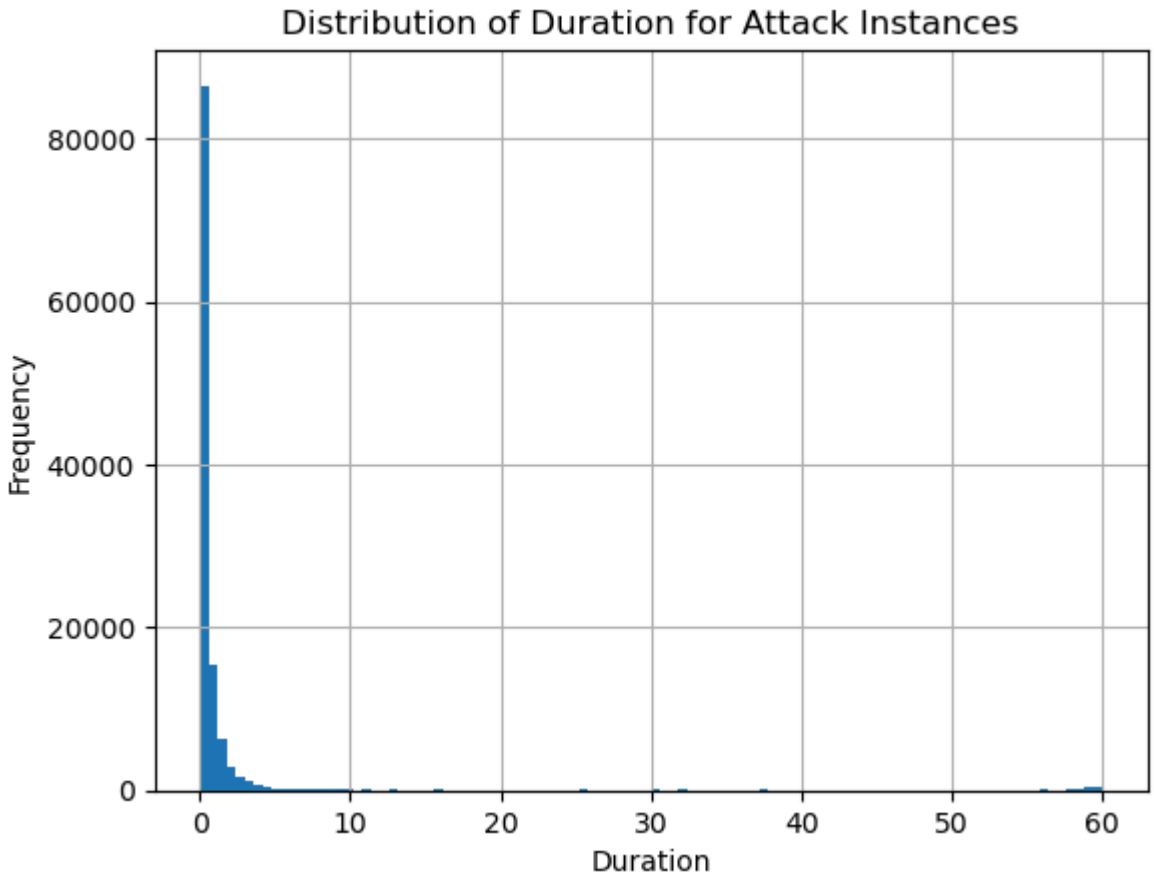
	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_s
0	0.121478	tcp	NaN	FIN	6	4	258	172	74.087490	252	...	
1	0.649902	tcp	NaN	FIN	14	38	734	42014	78.473372	62	...	
2	1.623129	tcp	NaN	FIN	8	16	364	13186	14.170161	62	...	
3	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	62	...	
4	0.449454	tcp	NaN	FIN	10	6	534	268	33.373826	254	...	
...	...	...	...	...	...	...	...	...	...	...	...	

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_s
175336	0.000009	udp	dns	INT	2	0	114	0	111111.107200	254	...	
175337	0.505762	tcp	NaN	FIN	10	8	620	354	33.612649	254	...	
175338	0.000009	udp	dns	INT	2	0	114	0	111111.107200	254	...	
175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	254	...	
175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	254	...	

175341 rows × 44 columns

In [26]:

```
import matplotlib.pyplot as plt

# Example for visualizing the distribution of 'dur' (duration)
attack_data['dur'].hist(bins=100)
plt.title('Distribution of Duration for Attack Instances')
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.show()
```



In [ ]:

In [27]:

```
attack_data
```

Out[27]:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_ds
47911	47912	0.000009	ddp	NaN	INT	2	0	200	0	111111.107200	...	

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_ds
47912	47913	0.000009	ipv6-frag	NaN	INT	2	0	200	0	111111.107200	...	
47913	47914	0.000009	cftp	NaN	INT	2	0	200	0	111111.107200	...	
47914	47915	0.000003	wsn	NaN	INT	2	0	200	0	333333.321500	...	
47915	47916	0.000008	pvp	NaN	INT	2	0	200	0	125000.000300	...	
...	...	...	...	...	...	...	...	...	...	...	...	
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	
175337	175338	0.505762	tcp	NaN	FIN	10	8	620	354	33.612649	...	
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	

119341 rows × 45 columns

In [ ]:

here, in the feature selection process, pre-selecting features manually can be done if we have good domain knowledge.

In [ ]:
```
# performing one hot encoding on the dataset:
```

In [ ]:
```
# first checking the datatypes of every column and converting it into numerical format
```

In [29]:
```
column_datatypes = data.dtypes
print(column_datatypes)
```

```
dur                 float64
proto                object
service              object
state                object
spkts                 int64
dpkts                 int64
sbytes                int64
dbytes                int64
rate                float64
sttl                  int64
dttl                  int64
sload               float64
dload               float64
sloss                 int64
dloss                 int64
sinpkt              float64
dinpkt              float64
sjit                float64
djit                float64
swin                  int64
stcpb                 int64
dtcpb                 int64
dwin                  int64
```

```
tcprtt               float64
synack               float64
ackdat               float64
smean                  int64
dmean                  int64
trans_depth            int64
response_body_len      int64
ct_srv_src             int64
ct_state_ttl           int64
ct_dst_ltm             int64
ct_src_dport_ltm       int64
ct_dst_sport_ltm       int64
ct_dst_src_ltm         int64
is_ftp_login           int64
ct_ftp_cmd             int64
ct_flw_http_mthd       int64
ct_src_ltm             int64
ct_srv_dst             int64
is_sm_ips_ports        int64
attack_cat            object
label                  int64
dtype: object
```

In [30]:
```
# filtering all the categorical columns from the 'data' and performing one hot encodin
```

In [31]:
```
categorical_cols = ['proto', 'service', 'state']

# one hot encoding:
data_encoded = pd.get_dummies(data, columns=categorical_cols)

data_encoded= data_encoded*1
```

In [32]:
```
data_encoded
```

Out[32]:

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload
0	0.121478	6	4	258	172	74.087490	252	254	1.415894e+04	8495.365234
1	0.649902	14	38	734	42014	78.473372	62	252	8.395112e+03	503571.312500
2	1.623129	8	16	364	13186	14.170161	62	252	1.572272e+03	60929.230470
3	1.681642	12	12	628	770	13.677108	62	252	2.740179e+03	3358.622070
4	0.449454	10	6	534	268	33.373826	254	252	8.561499e+03	3987.059814
...	...	...	...	...	...	...	...	...	...	...
175336	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000
175337	0.505762	10	8	620	354	33.612649	254	252	8.826286e+03	4903.492188
175338	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000
175339	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000
175340	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.000000

175341 rows × 195 columns

In [ ]:

In [33]:
```
data_encoded['label']
```

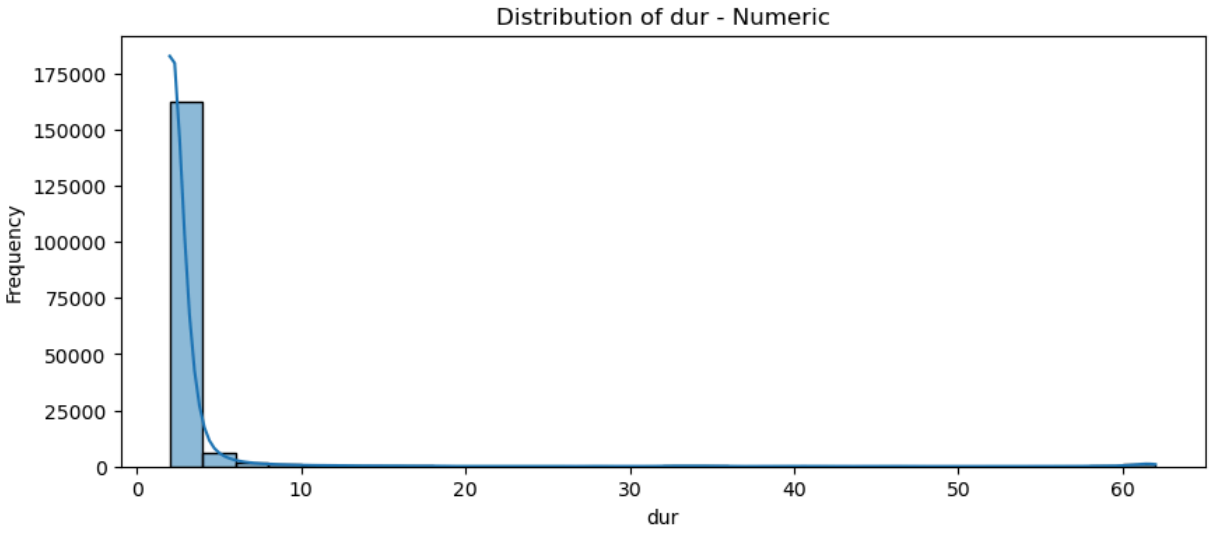
Out[33]:
```
0         0
1         0
2         0
3         0
4         0
         ..
175336    1
175337    1
175338    1
175339    1
175340    1
Name: label, Length: 175341, dtype: int64
```

In [34]:
```
data_encoded.shape
```

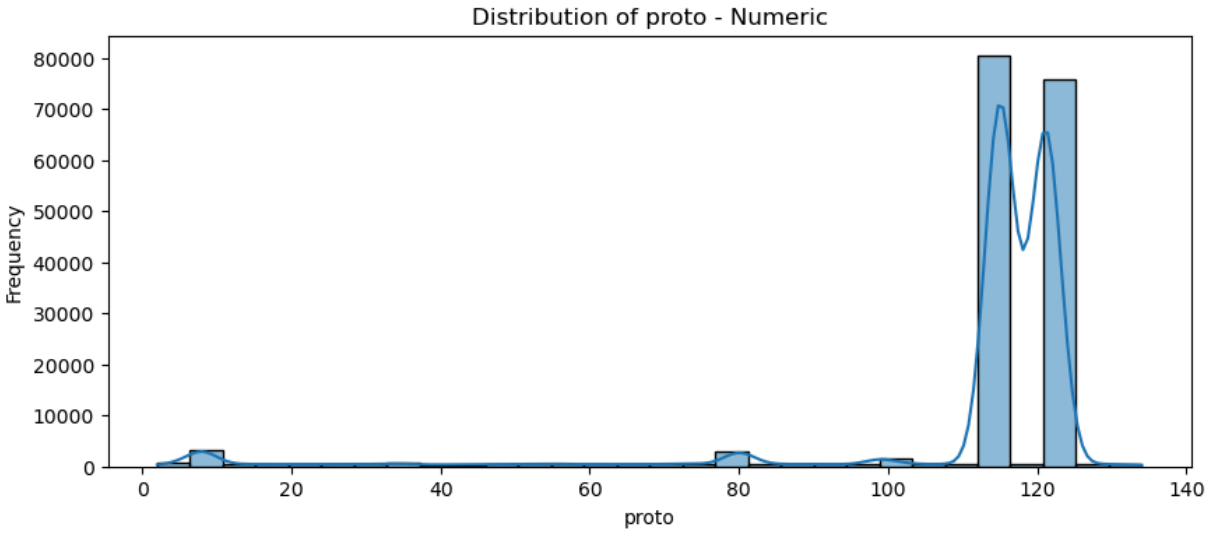
Out[34]:
```
(175341, 195)
```

In [161]:
```
# Before log transformation
# representation of skewed data:

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming 'data' is your DataFrame
# Iterate over each column in the DataFrame
for column in data_encoded.columns:
    plt.figure(figsize=(10, 4))
    # Check if the column is numeric or categorical
    if pd.api.types.is_numeric_dtype(data_encoded[column]):
        # Plot distribution of numeric columns
        sns.histplot(data_encoded[column], kde=True, bins=30)
        plt.title(f'Distribution of {column} - Numeric')
    else:
        # Plot count of categorical columns
        sns.countplot(y=column, data=data_encoded, order=data_encoded[column].value_co
        plt.title(f'Distribution of {column} - Categorical')

    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

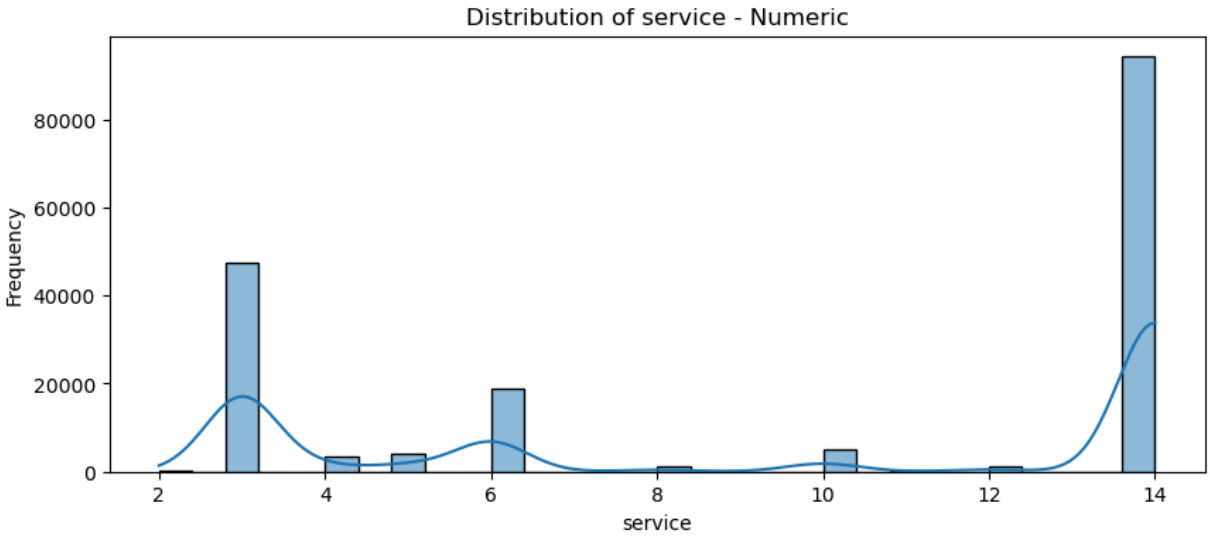
```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



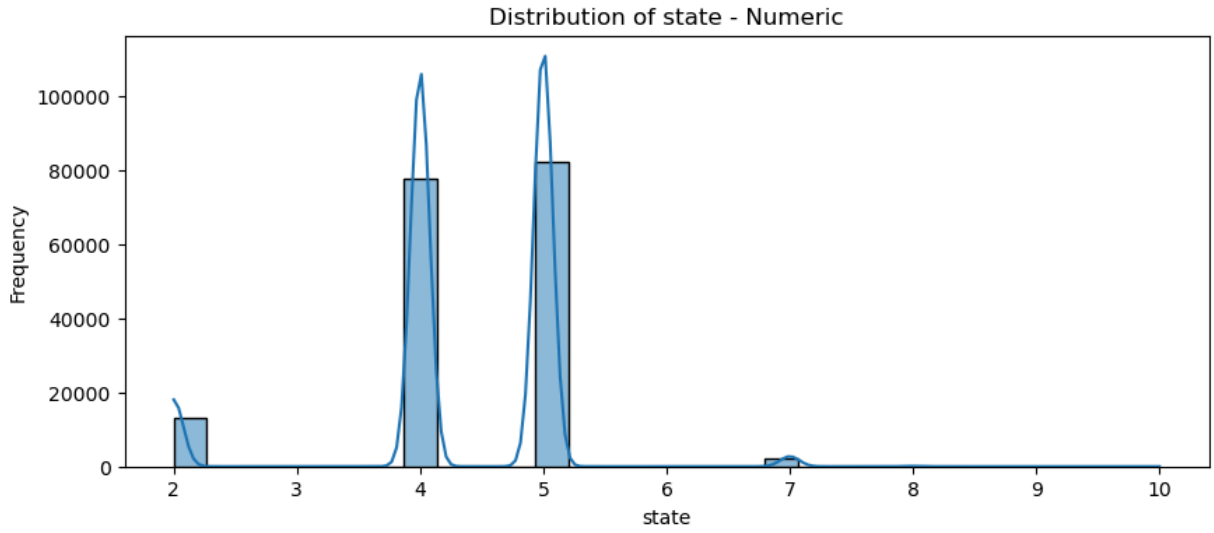

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



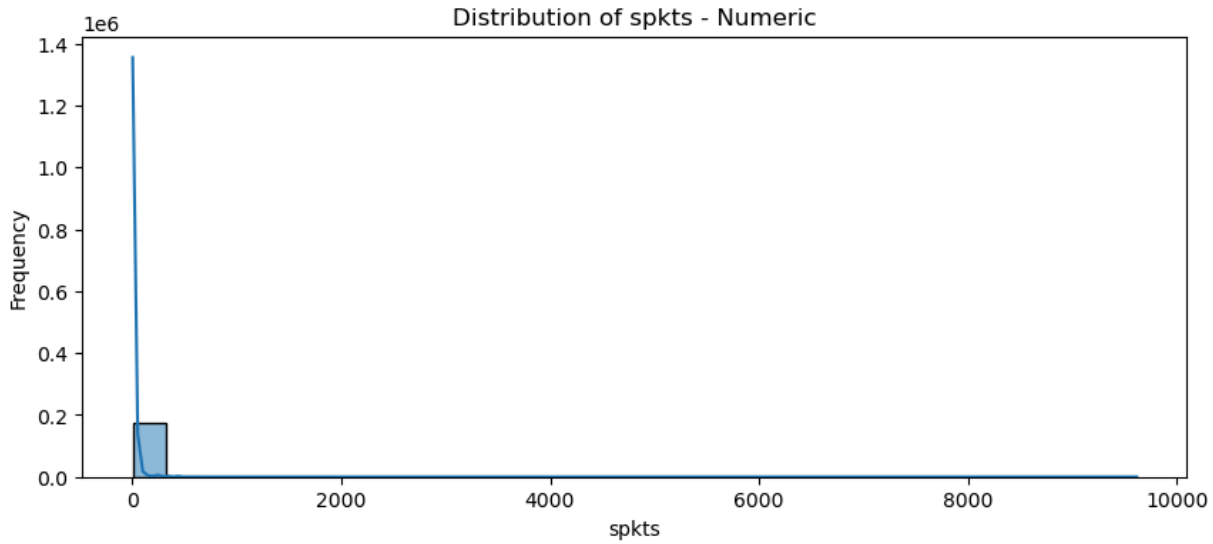

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



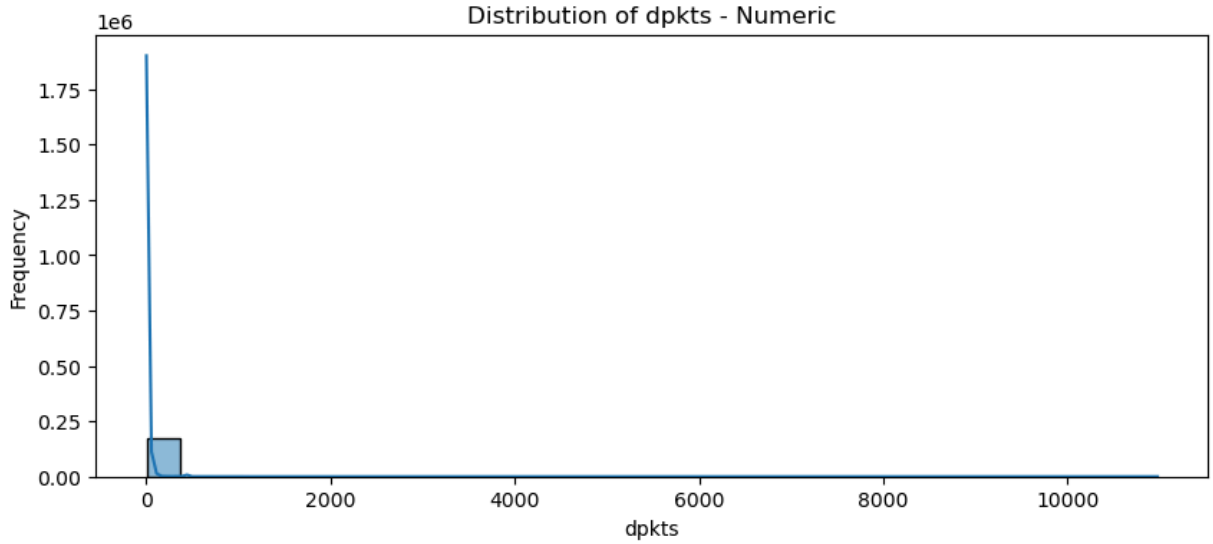

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

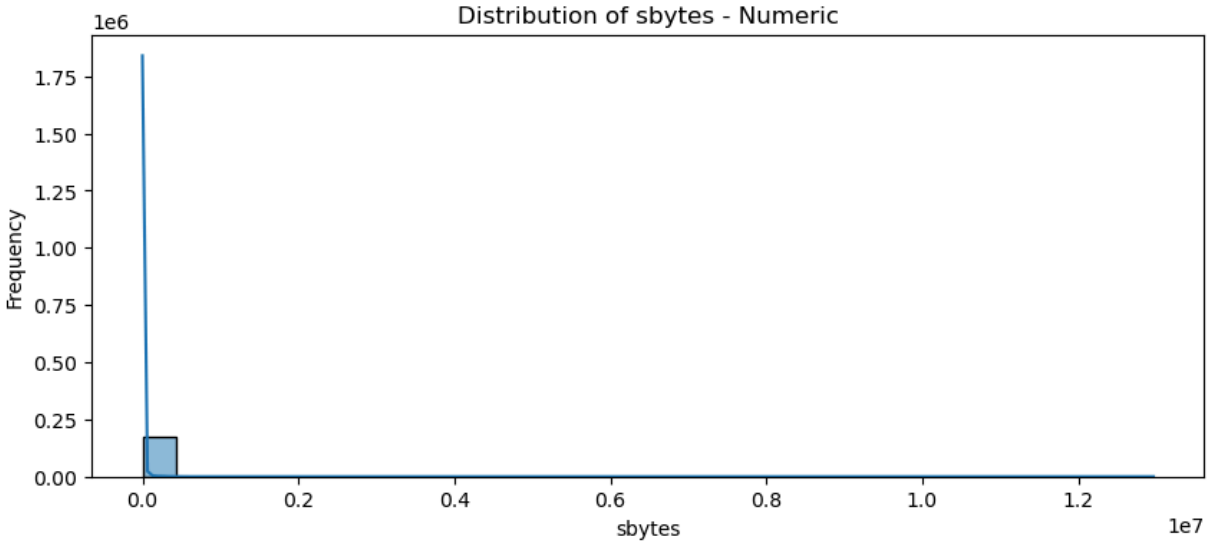



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

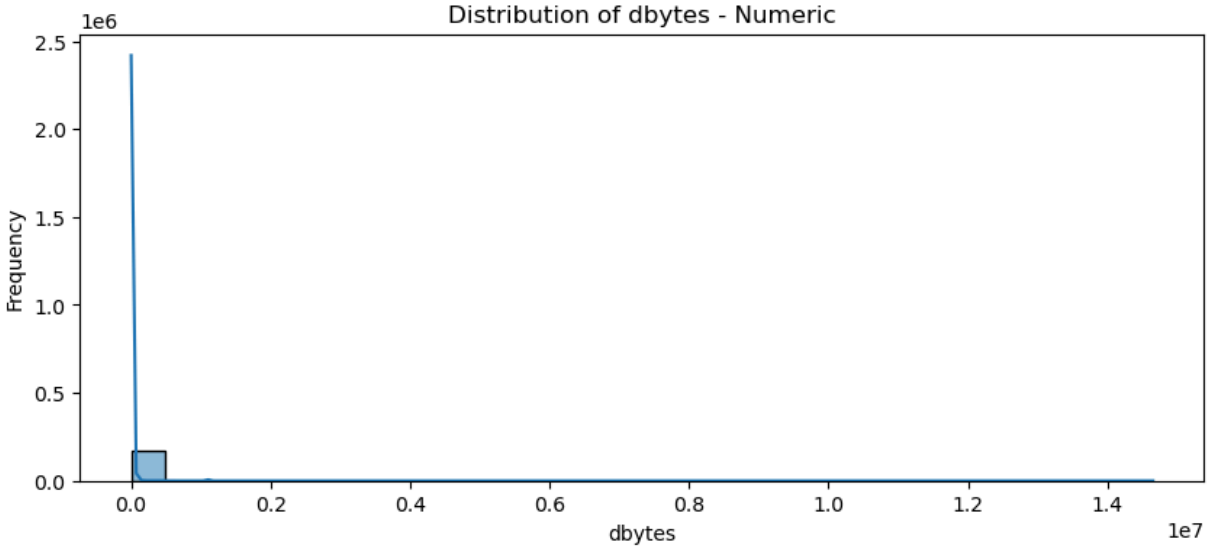



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

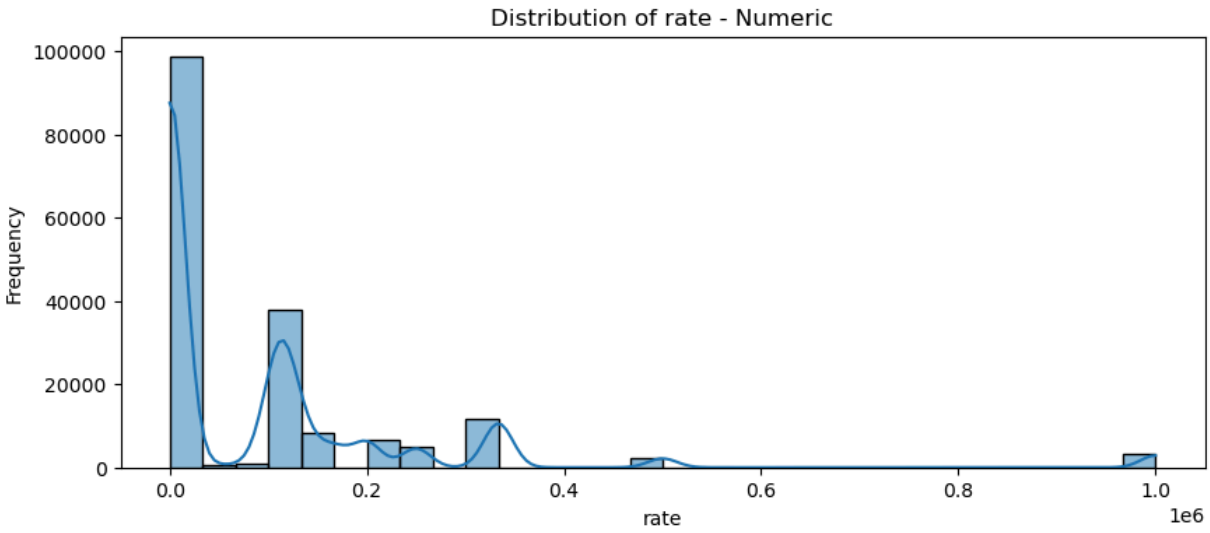
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



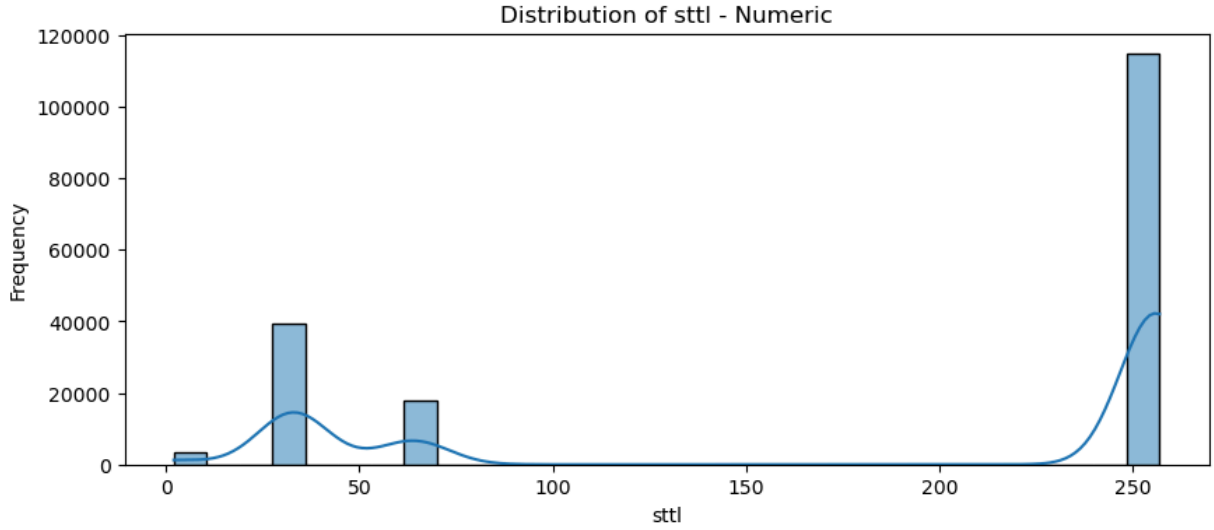

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



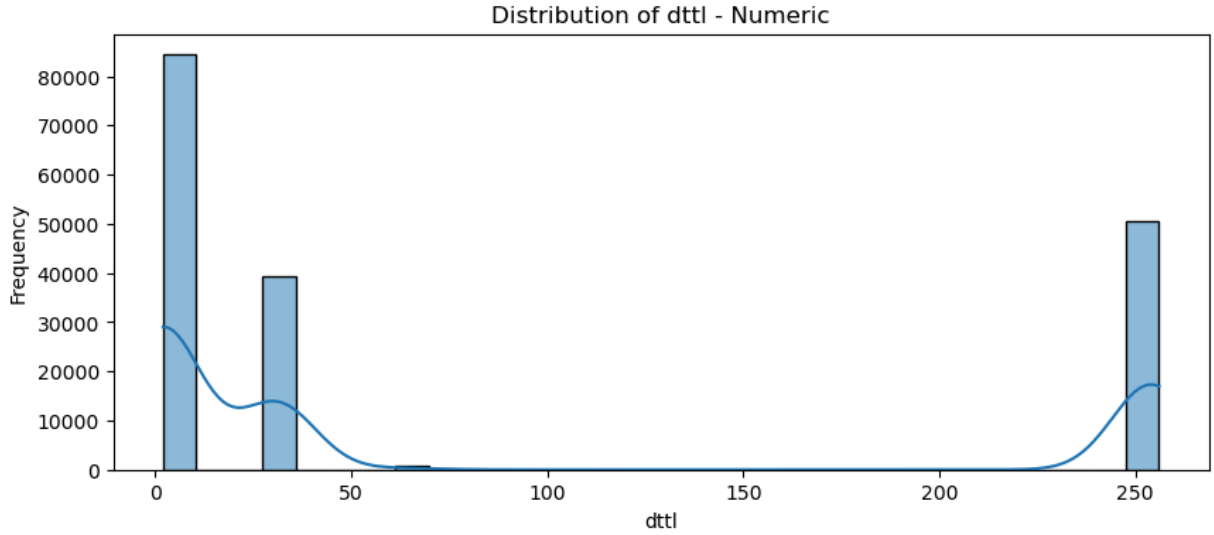

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



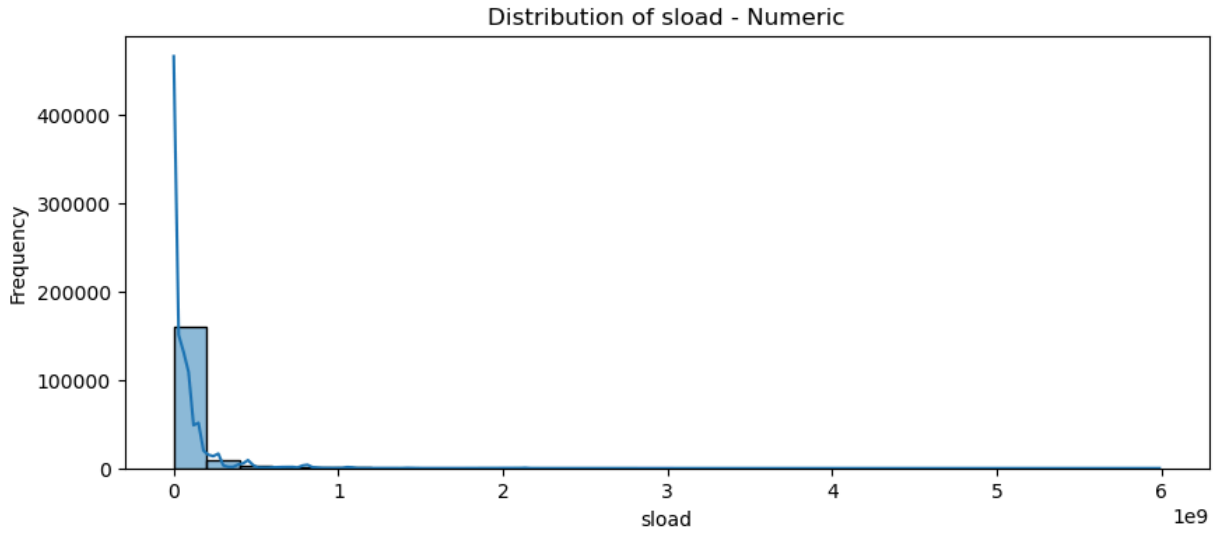
```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



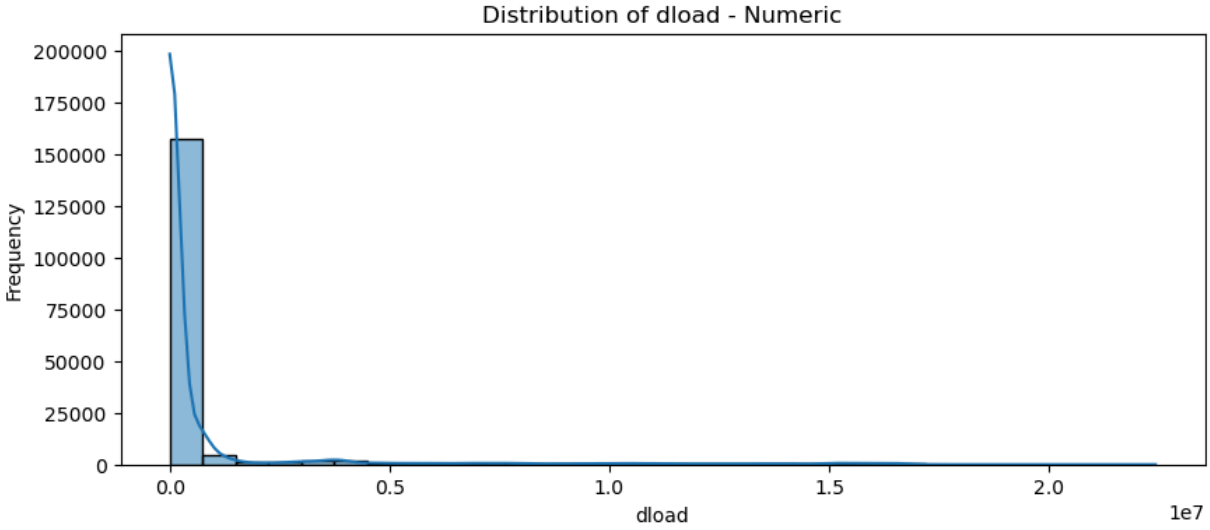

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



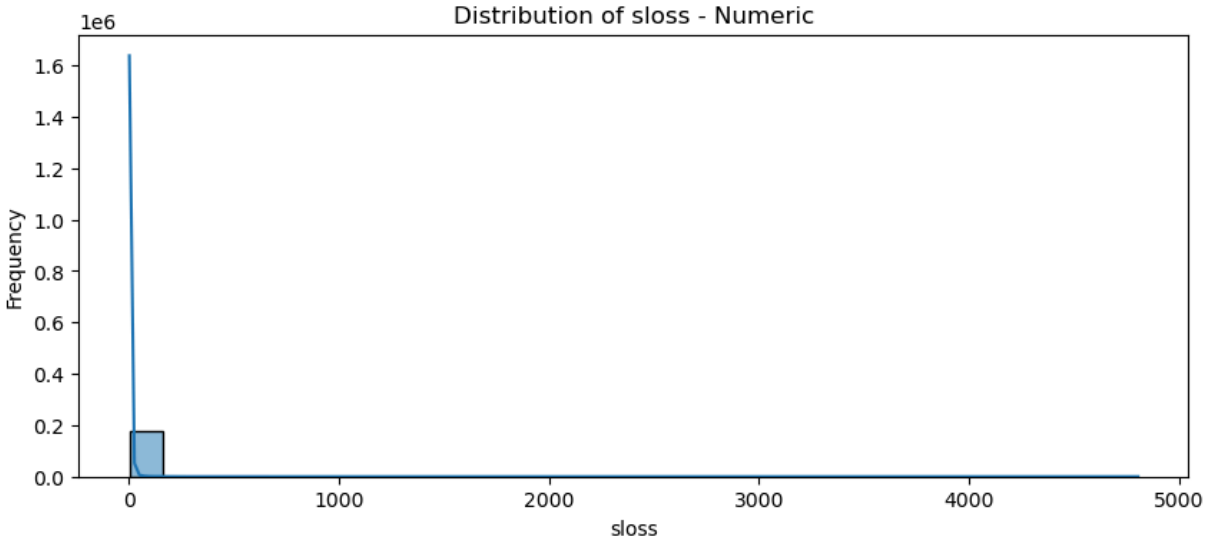

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



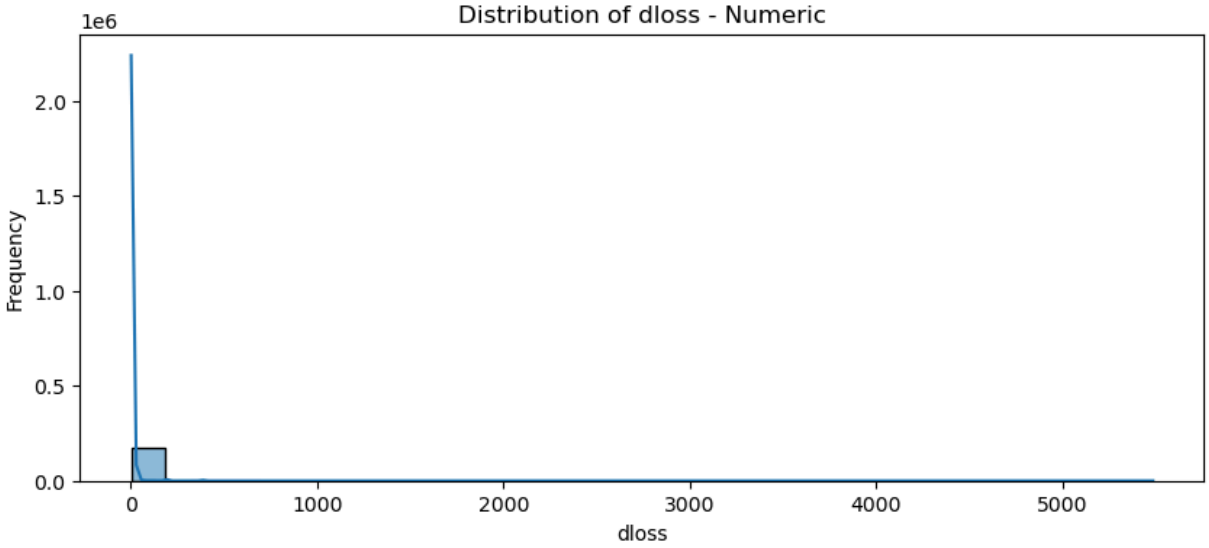
```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



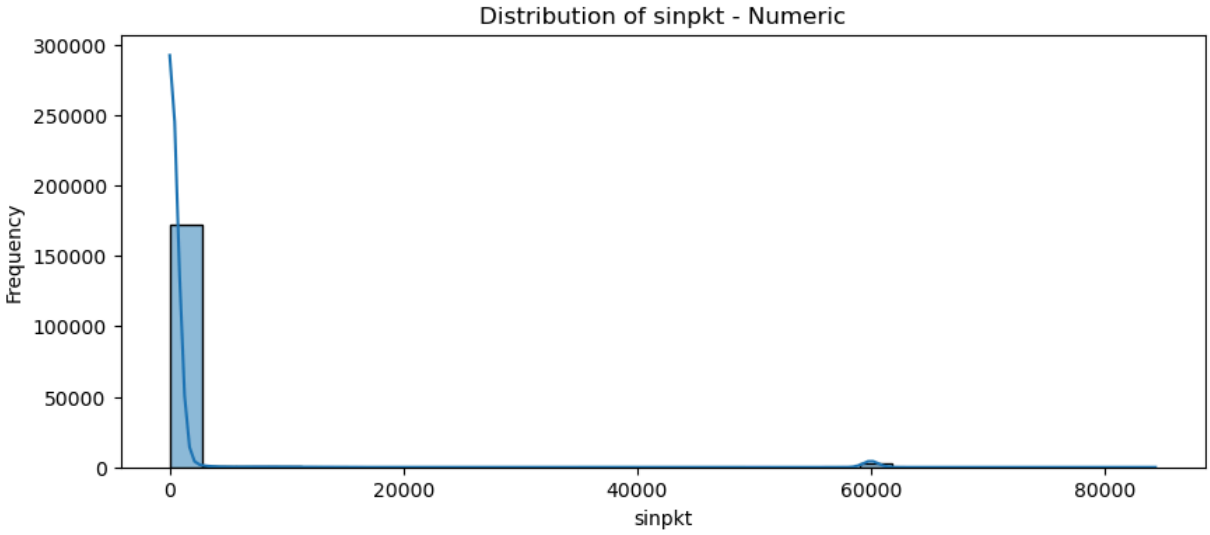

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



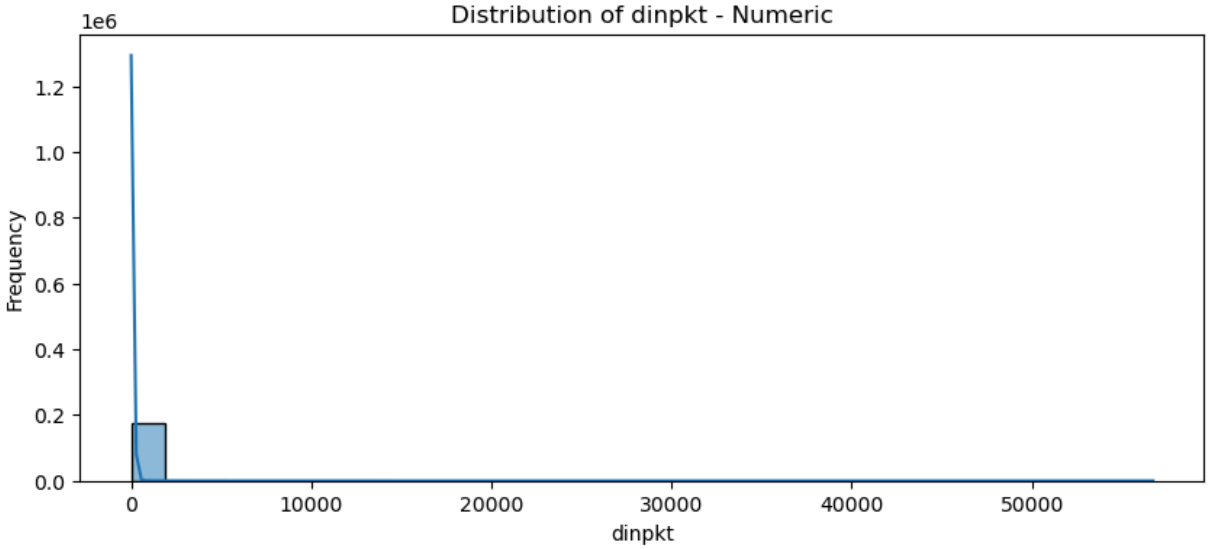

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

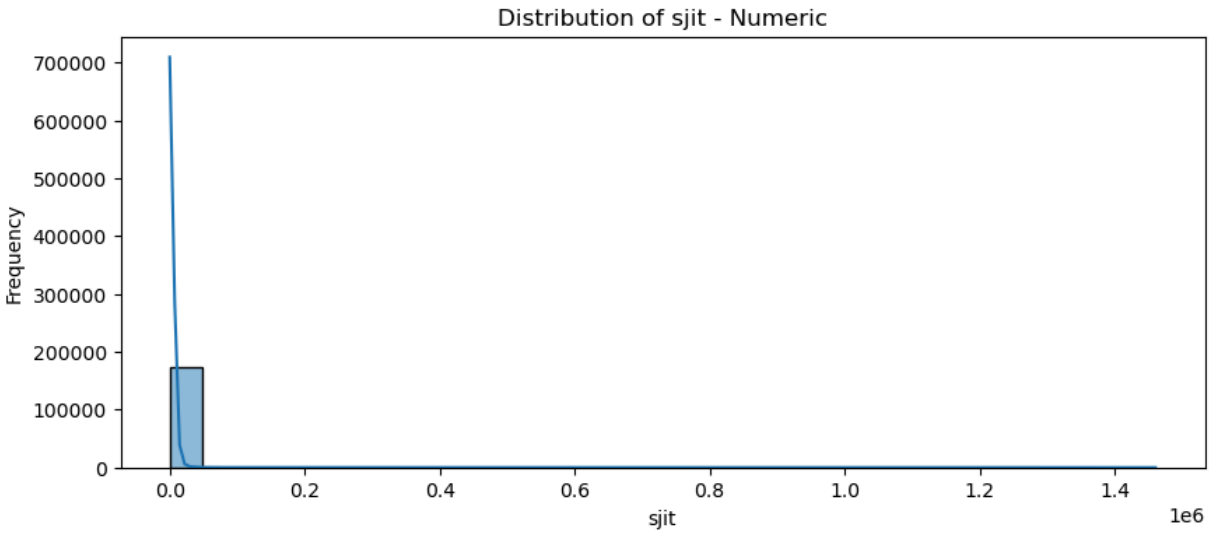



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

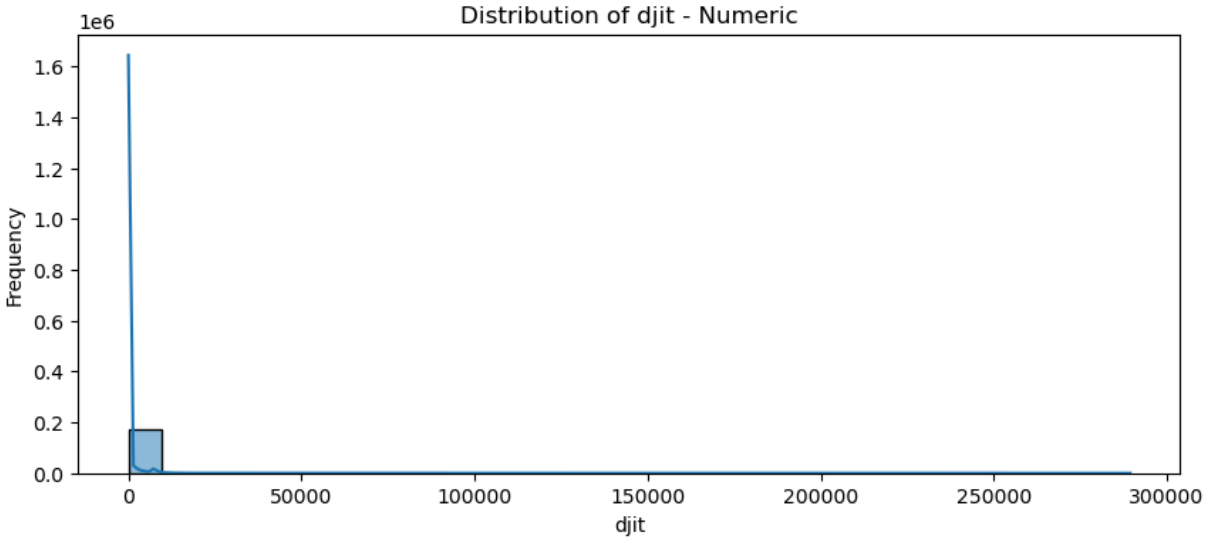



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

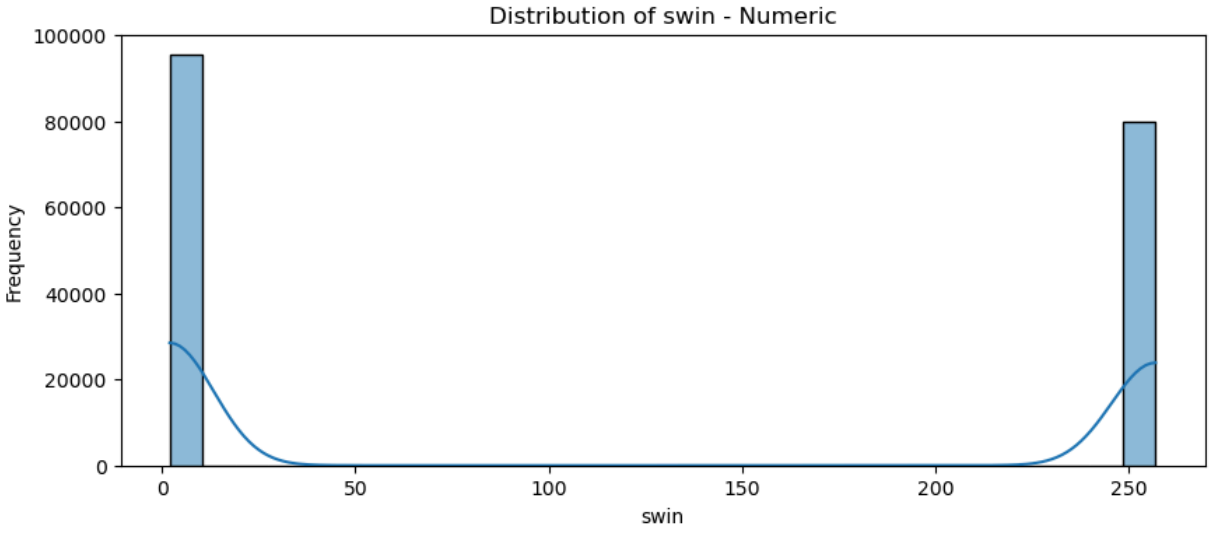
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

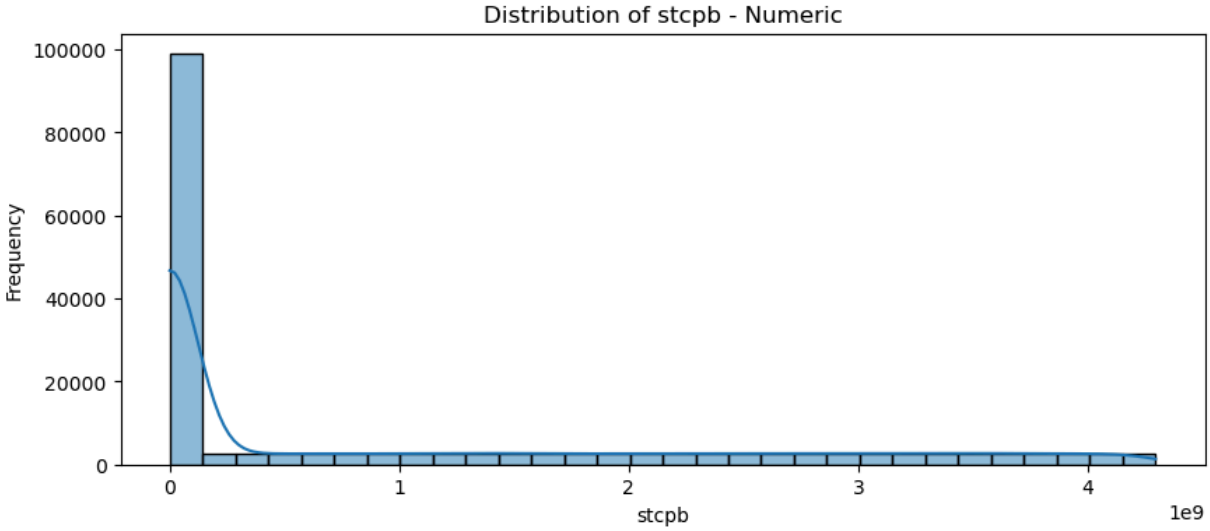



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

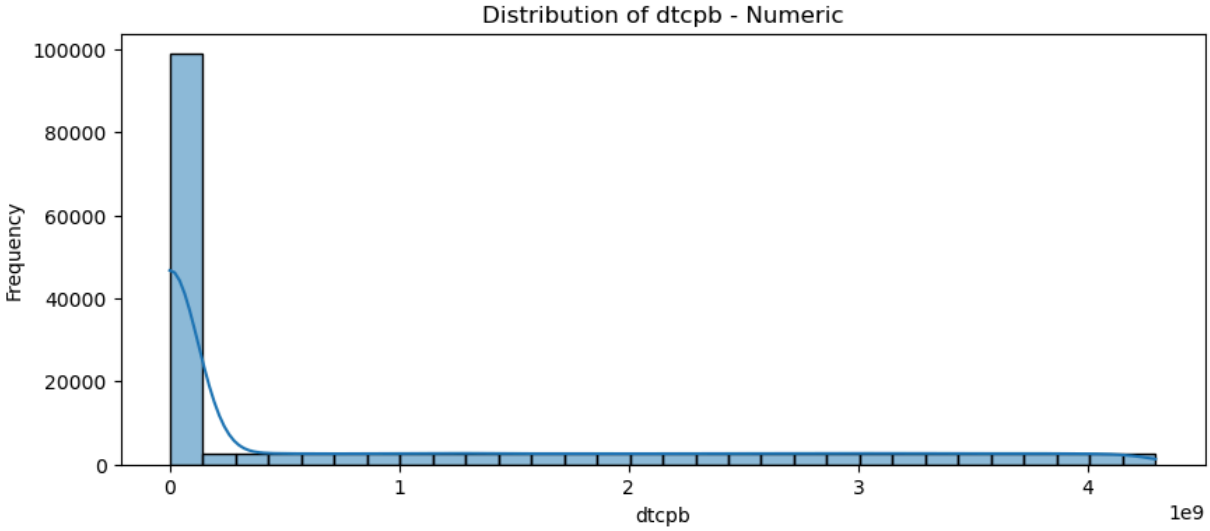



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

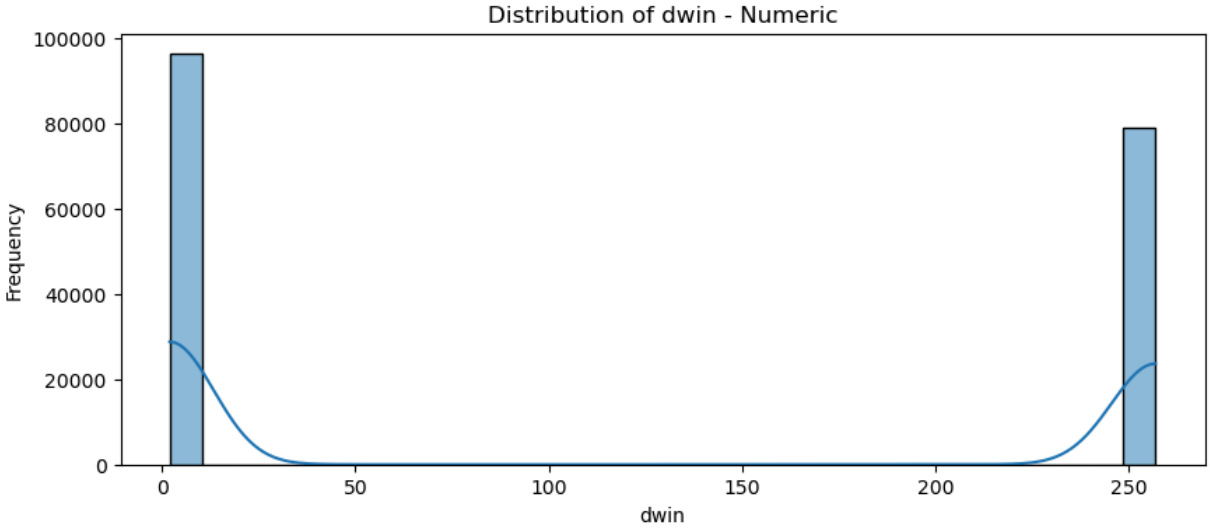
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

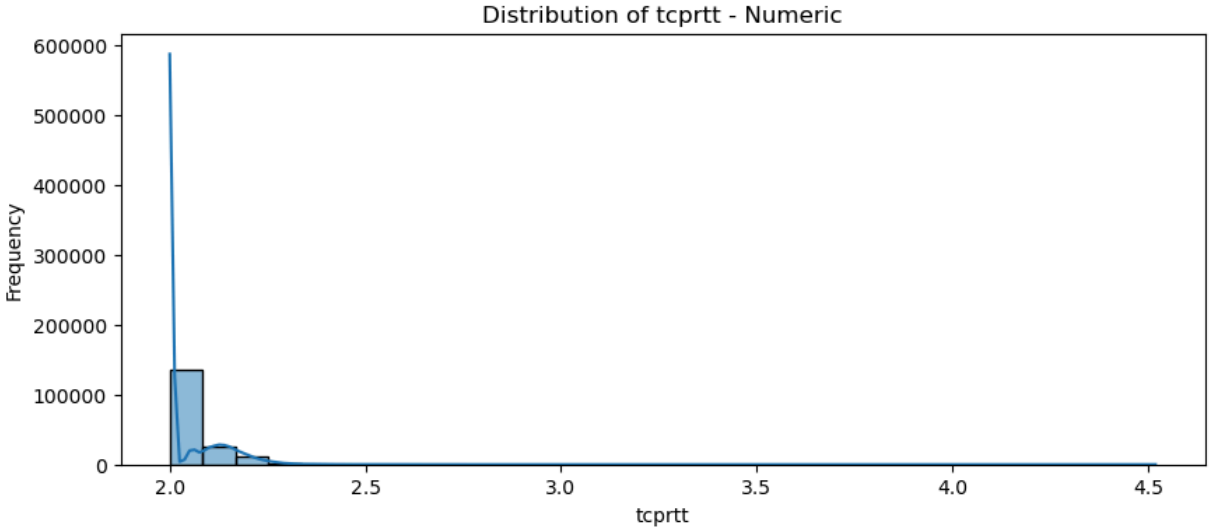



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

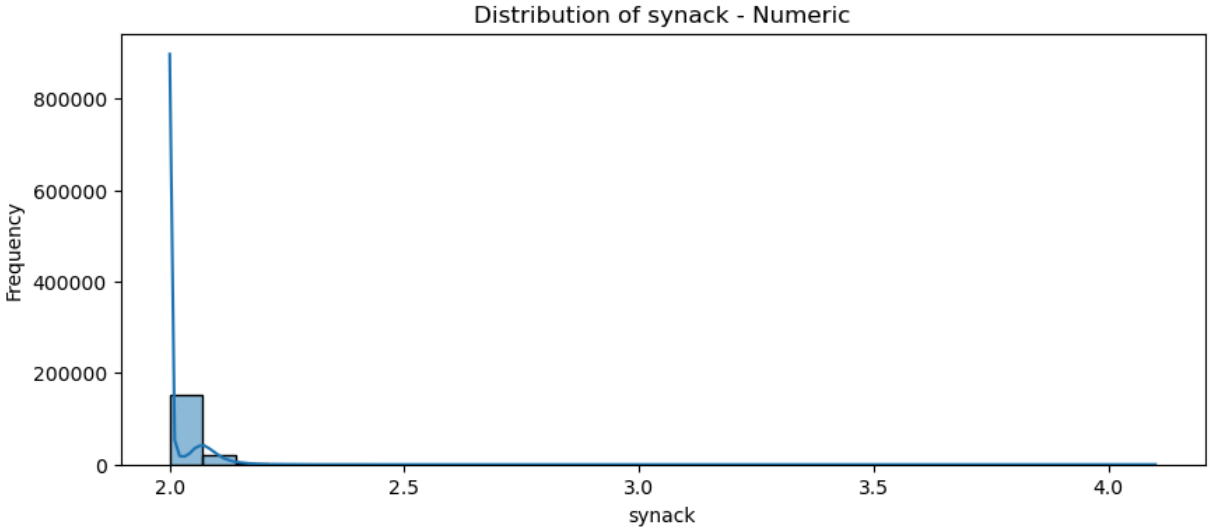



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

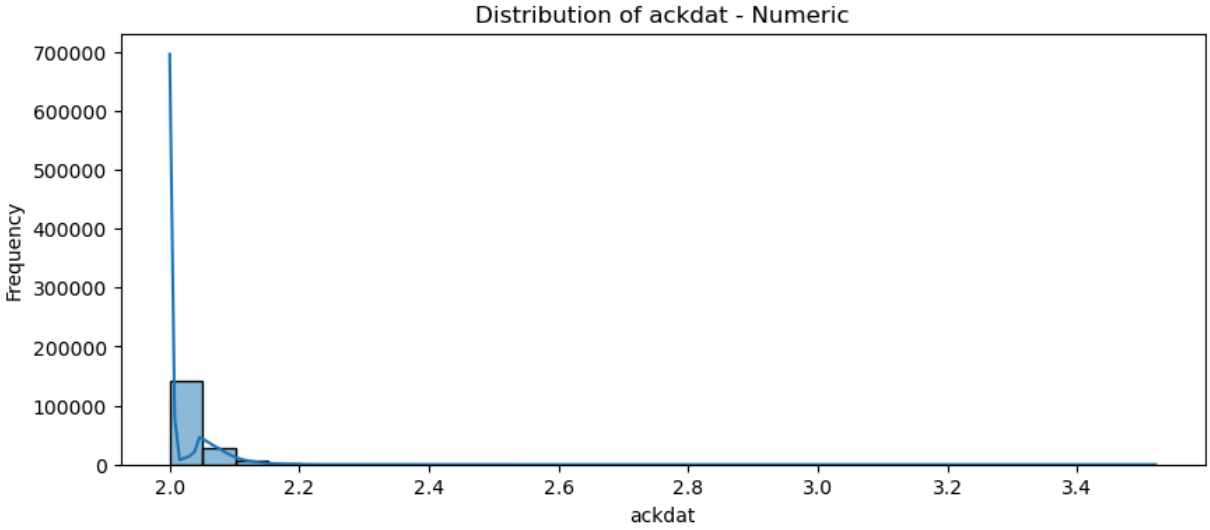
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

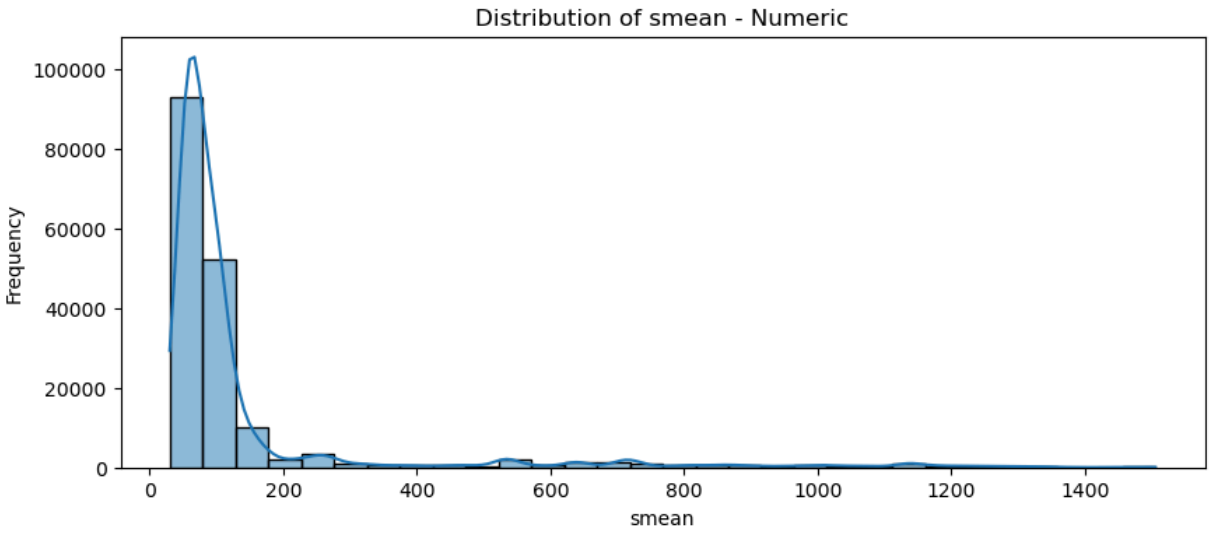



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

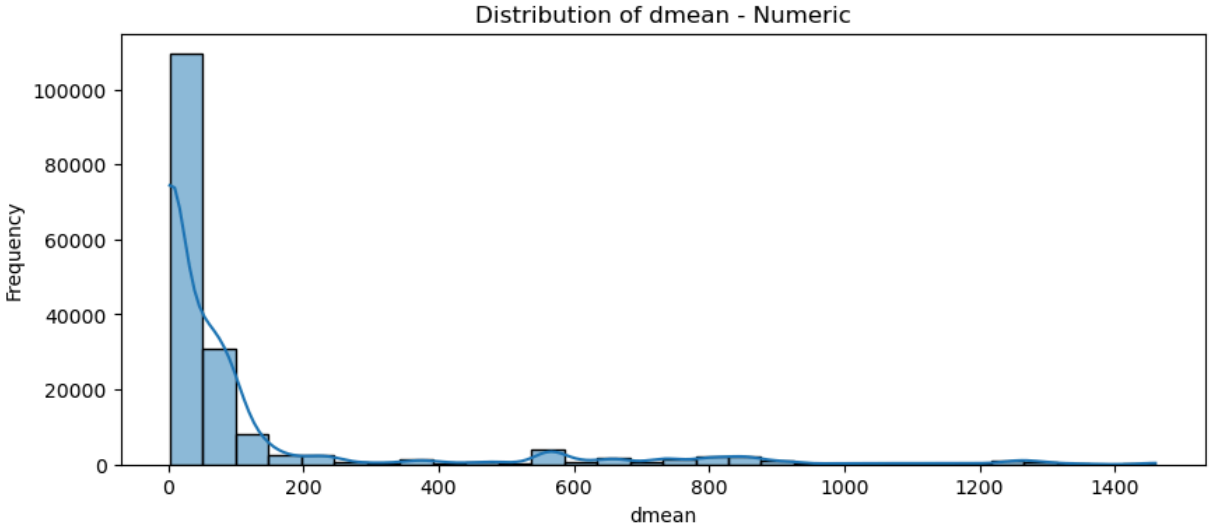



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

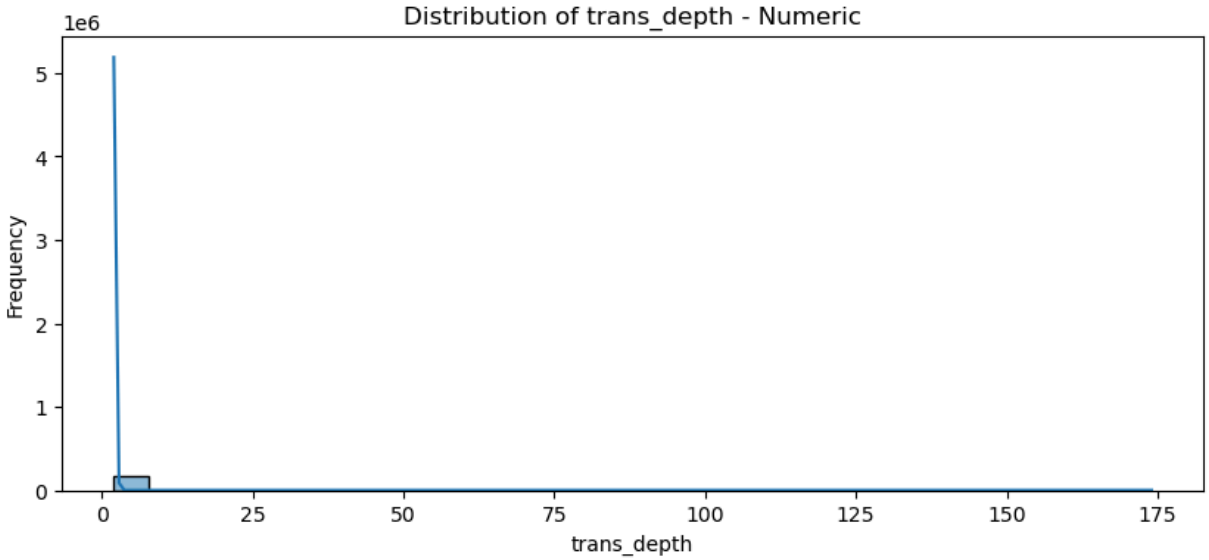
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



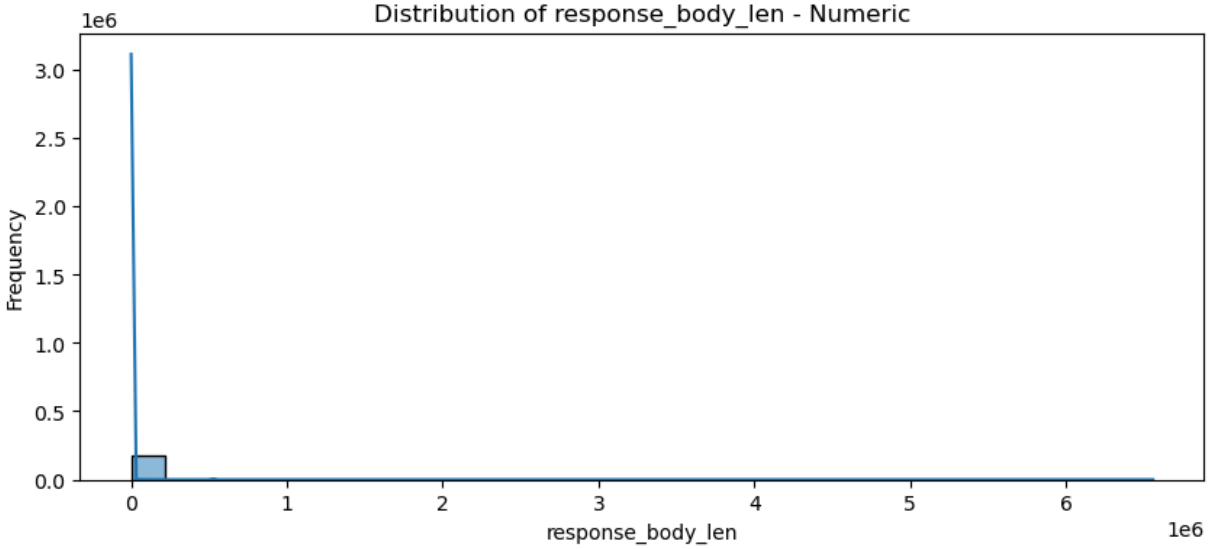

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



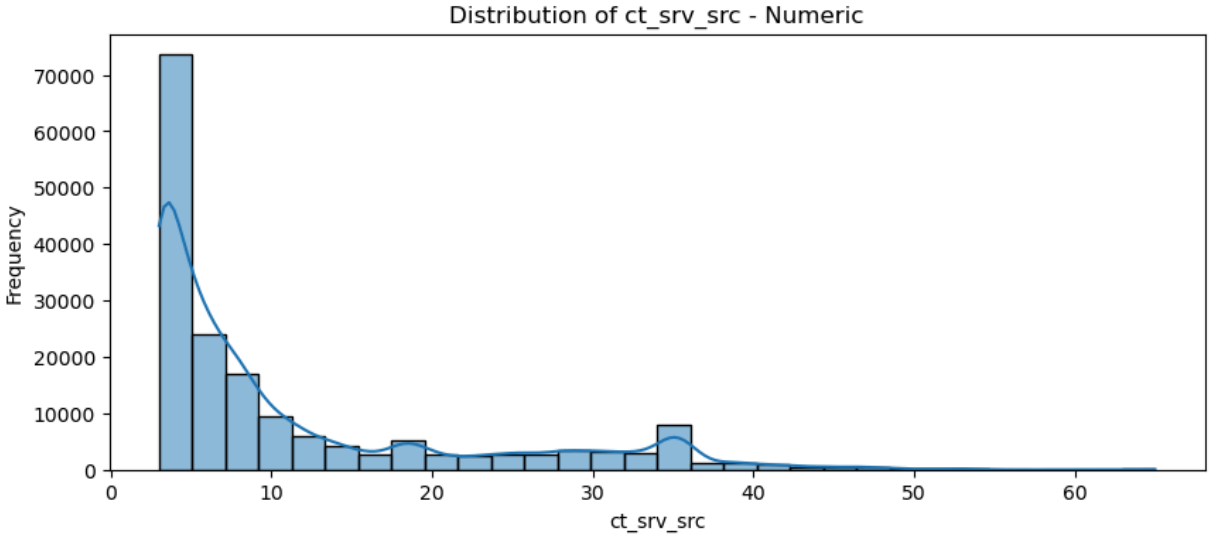

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



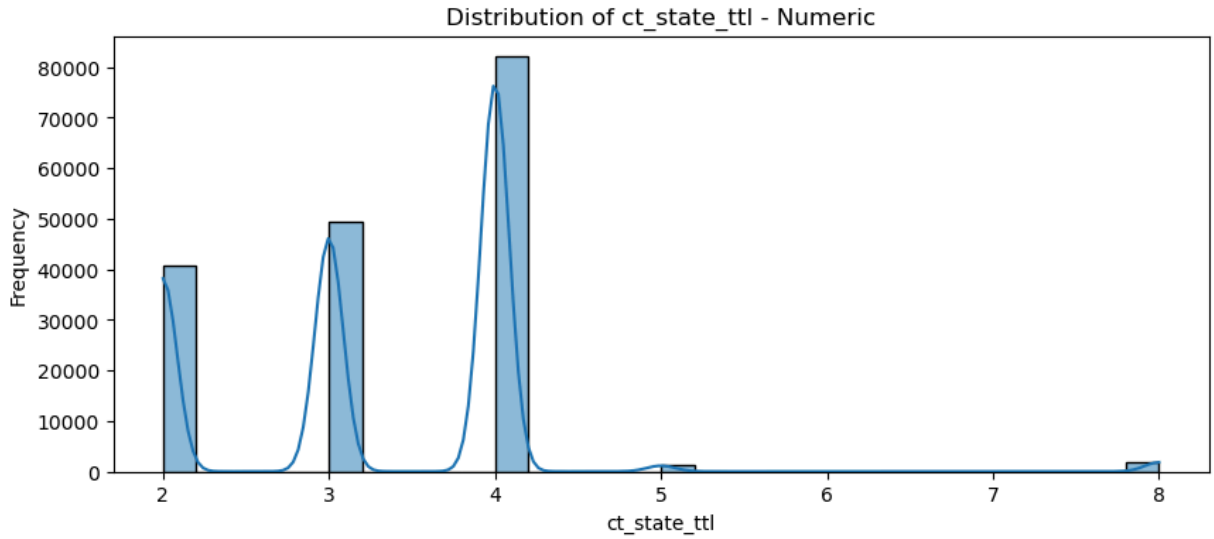
```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



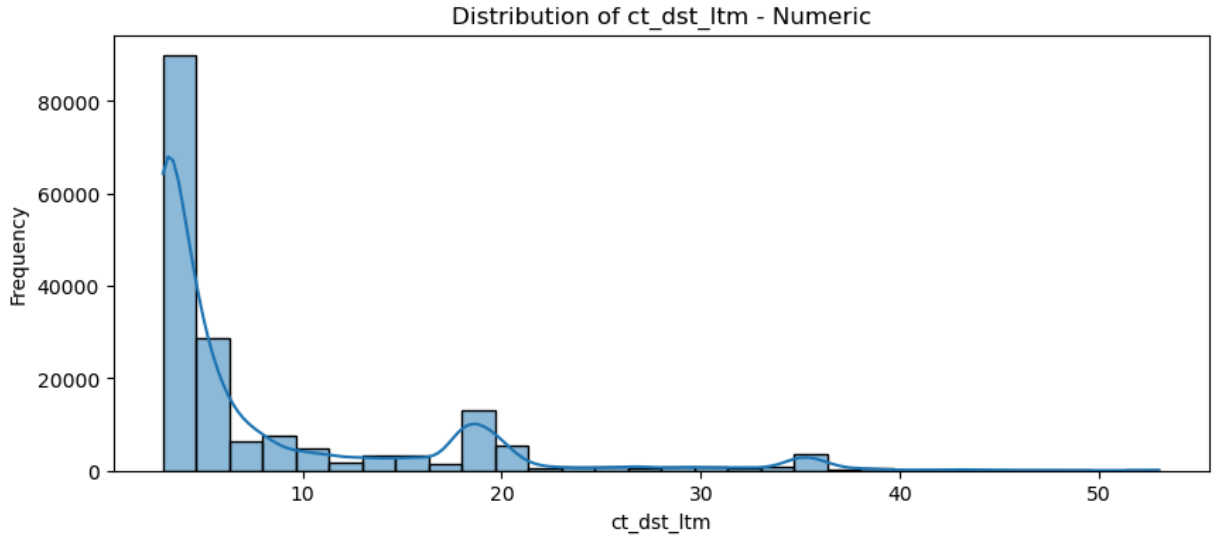

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



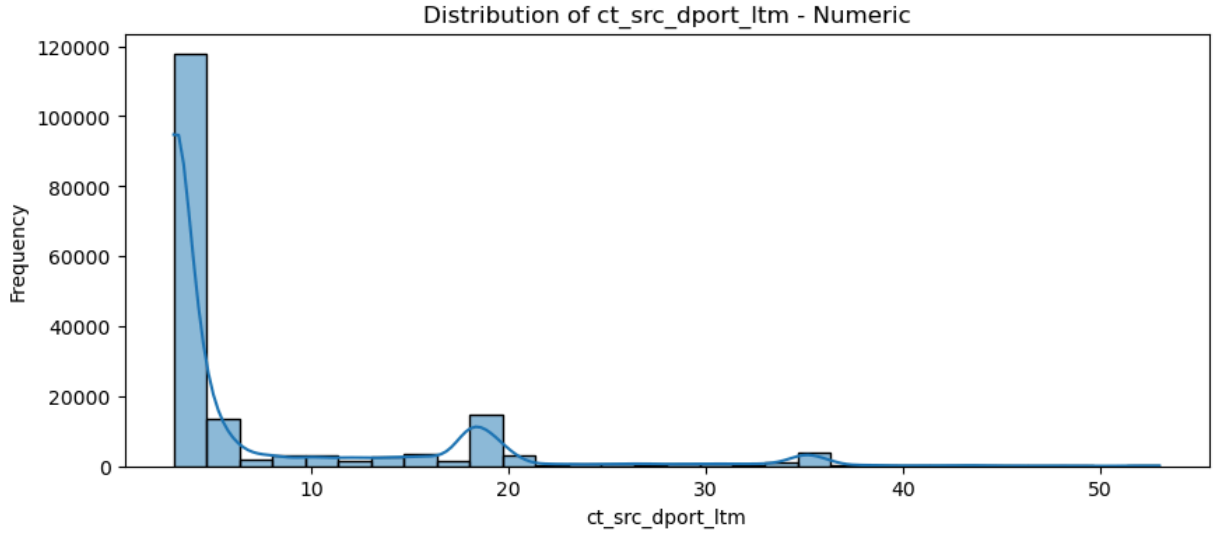

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



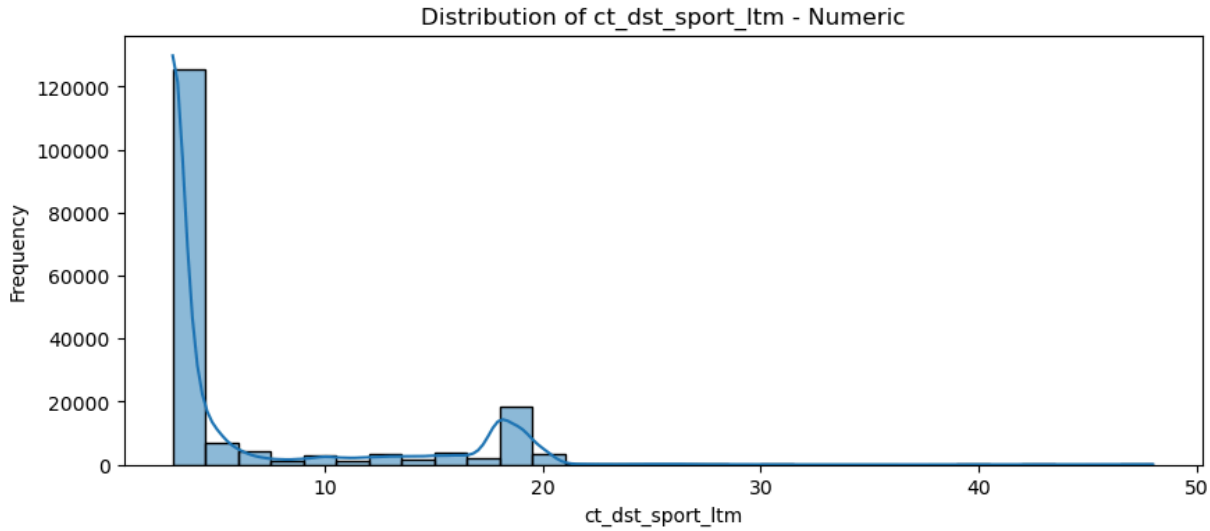

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



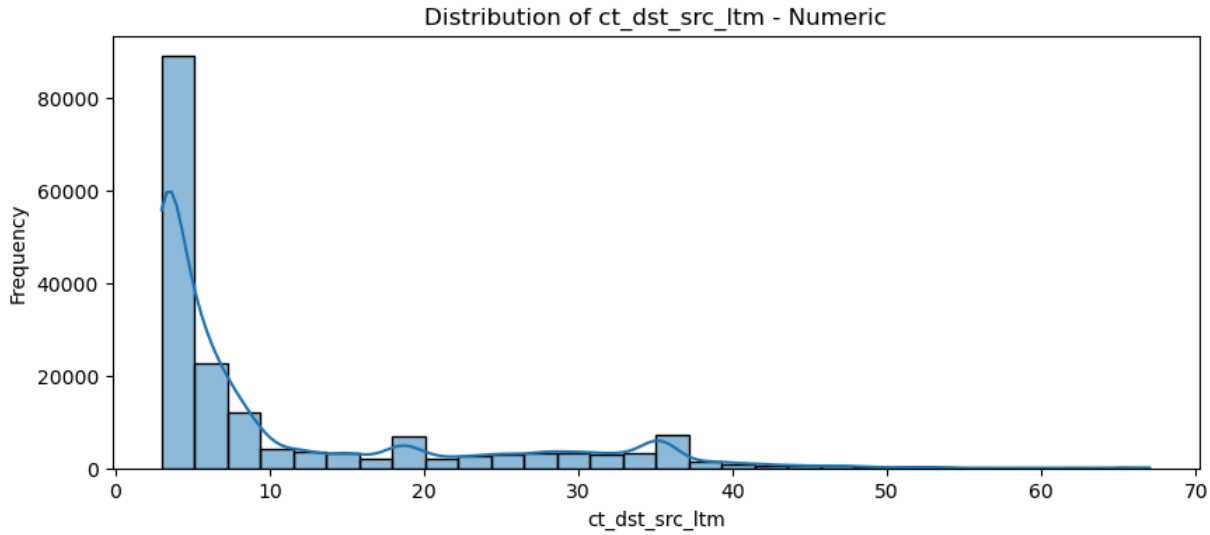

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



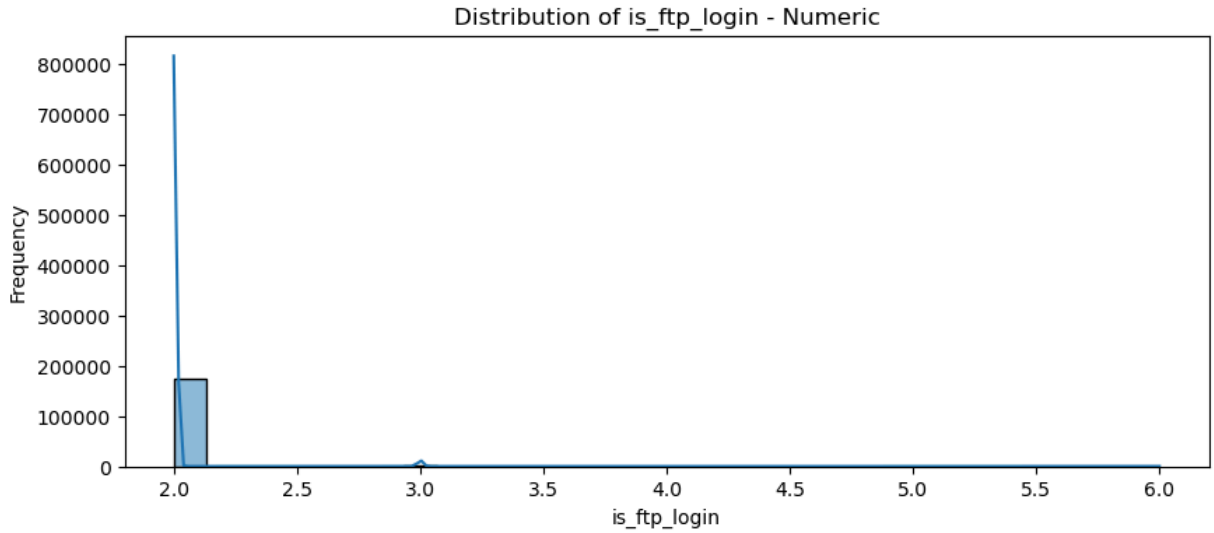

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



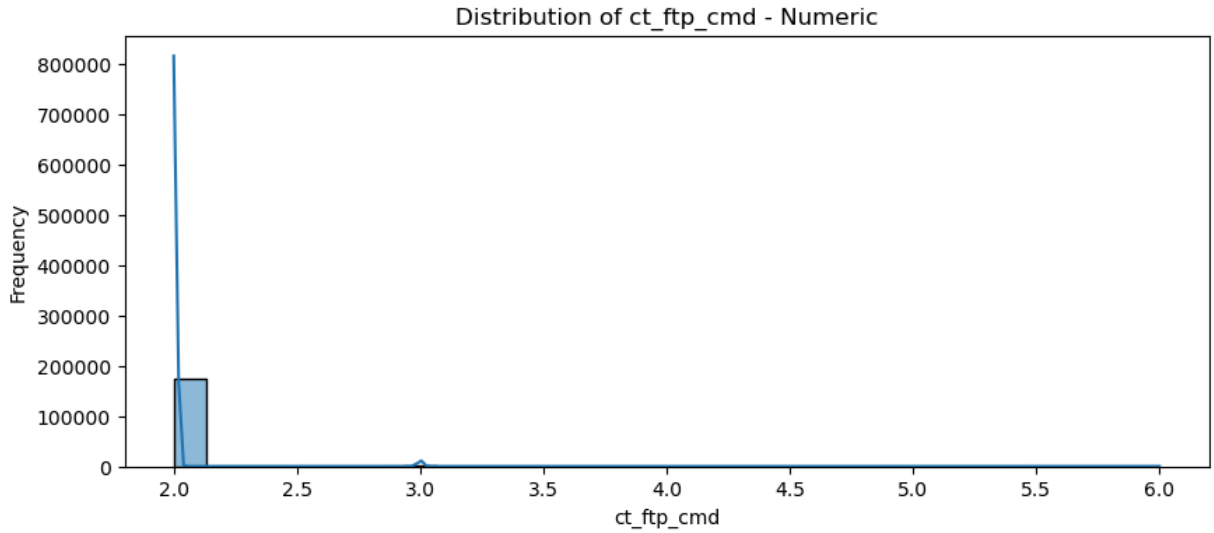

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



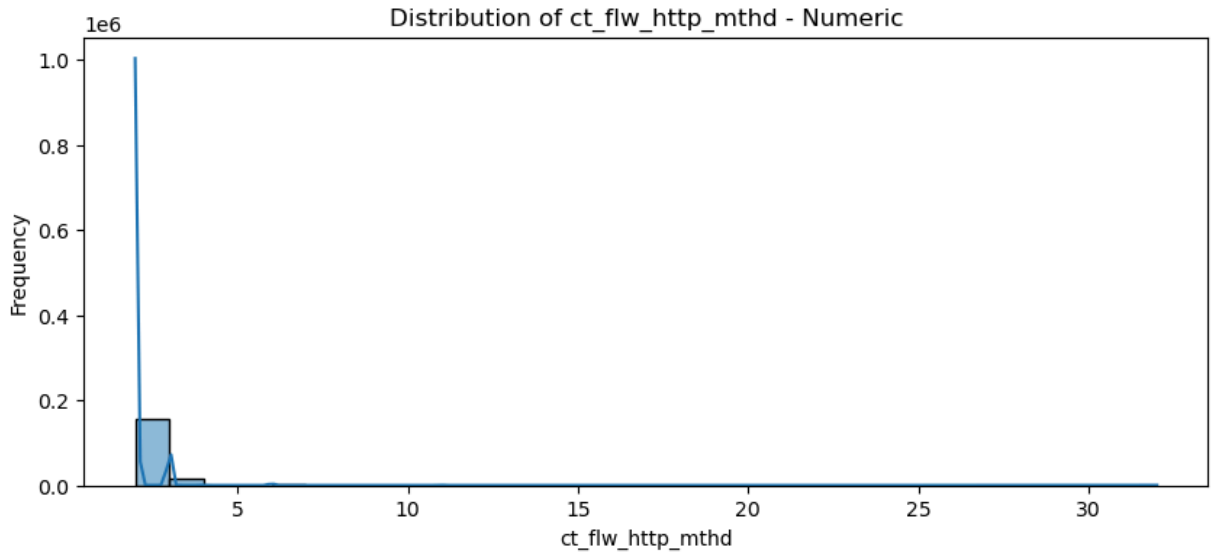

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



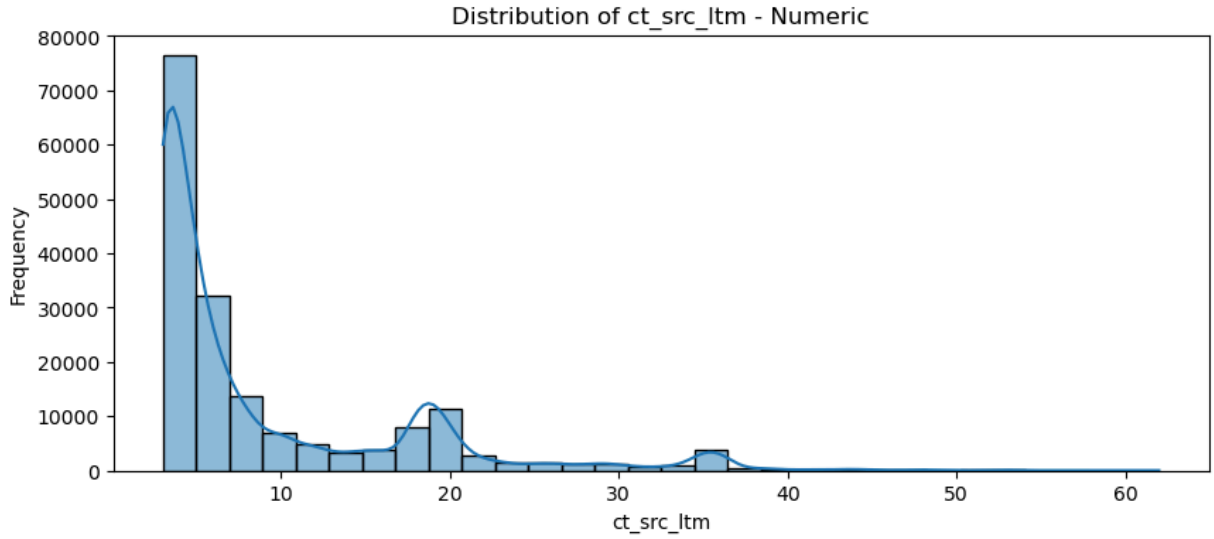

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

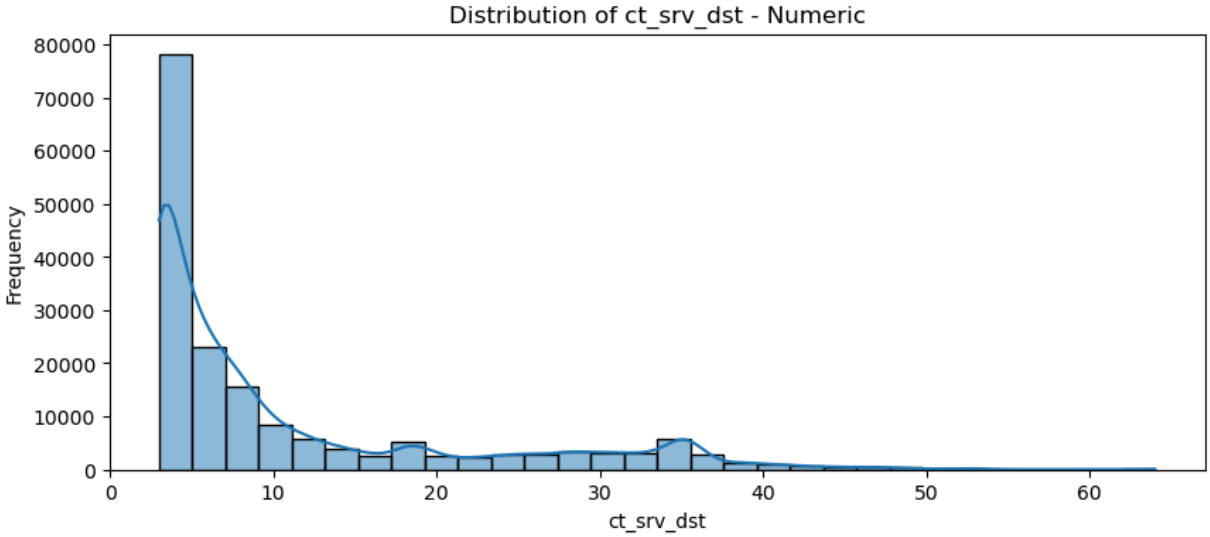



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

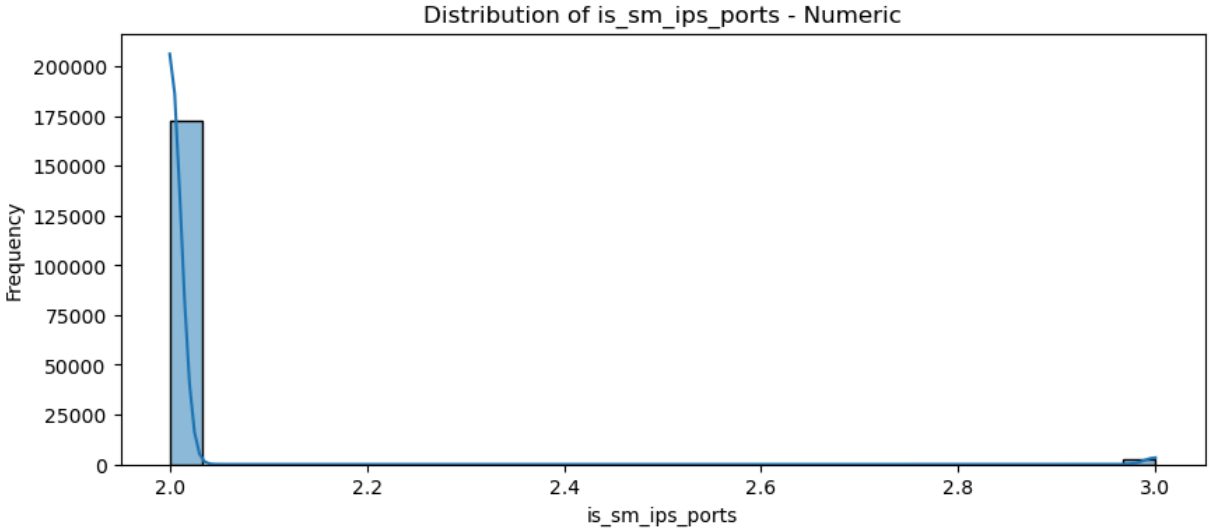



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

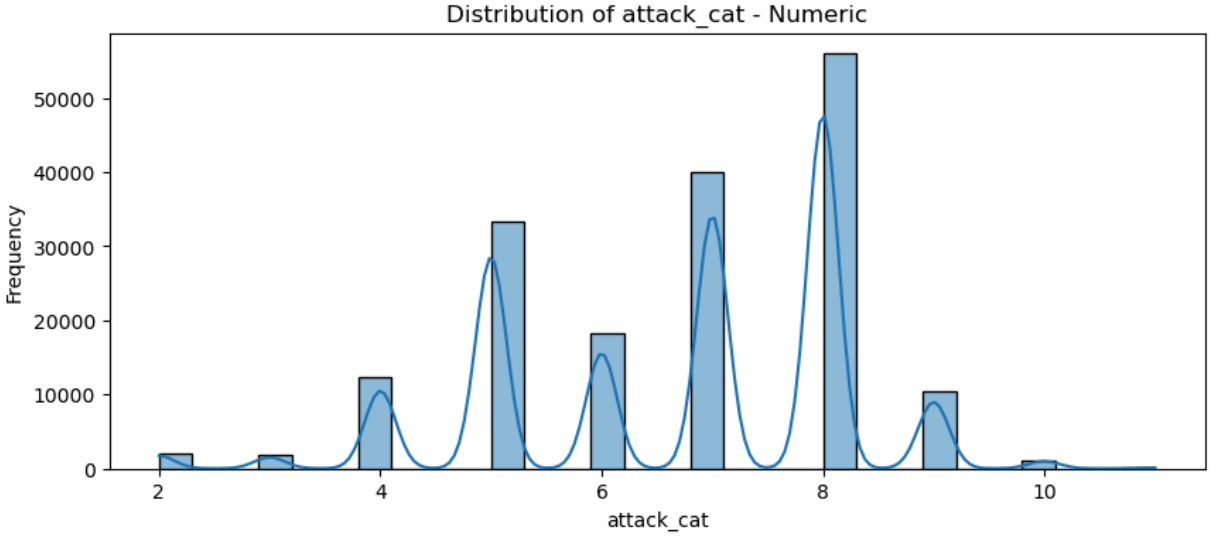
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

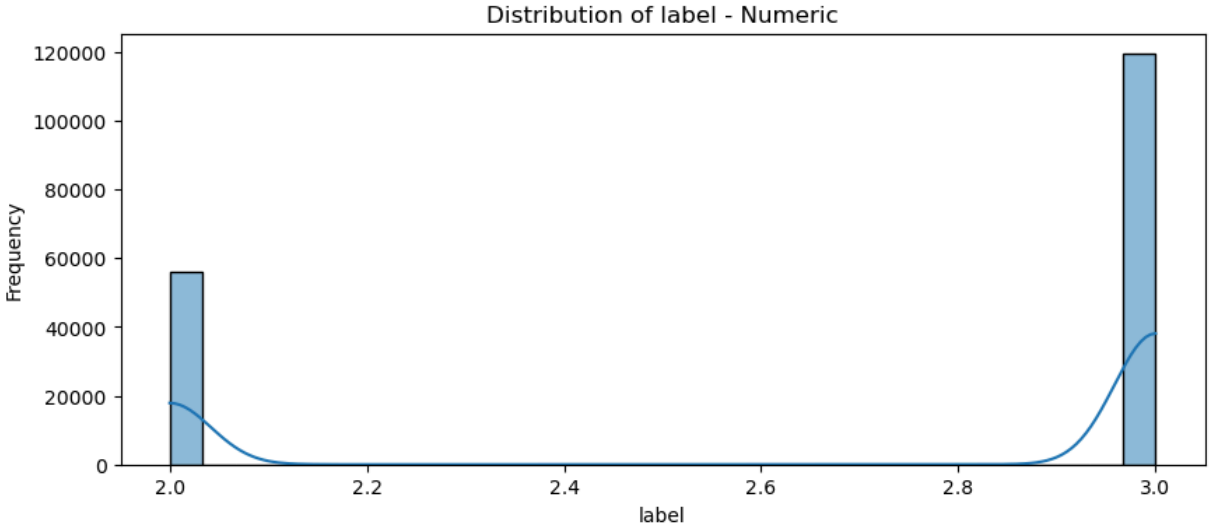



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [35]:
```
# normalization of the dataset into batches:
```

In [36]:
```
data1 = data.drop('attack_cat', axis = 1)
```

In [37]:
```
if 'attack_cat' in data_encoded.columns:
    y = data_encoded['attack_cat']  # or any other categorical columns
    data_encoded = data_encoded.drop(['attack_cat'], axis=1)  # Drop the categorical c
else:
    data_encoded = data_encoded
```

performing normalization

In [38]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode categorical columns
data_encoded = data.copy()  # Assuming data is your DataFrame
for col in data_encoded.columns:
    if data_encoded[col].dtype == 'object':  # Check if column contains categorical da
        data_encoded[col] = label_encoder.fit_transform(data_encoded[col])

# Now you can apply MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(data_encoded)
data_normalized = pd.DataFrame(data_normalized, columns=data_encoded.columns)
```

In [39]:
```
data_normalized
```

Out[39]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl
0	2.024634e-03	0.856061	1.000000	0.250	0.000520	0.000364	0.000018	0.000012	0.000074	0.988235

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl
1	1.083170e-02	0.856061	1.000000	0.250	0.001352	0.003463	0.000054	0.002867	0.000078	0.243137
2	2.705215e-02	0.856061	1.000000	0.250	0.000728	0.001458	0.000026	0.000900	0.000014	0.243137
3	2.802737e-02	0.856061	0.166667	0.250	0.001144	0.001093	0.000046	0.000053	0.000014	0.243137
4	7.490901e-03	0.856061	1.000000	0.250	0.000936	0.000547	0.000039	0.000018	0.000033	0.996078
...	...	...	...	...	...	...	...	...	...	...
175336	1.500000e-07	0.901515	0.083333	0.375	0.000104	0.000000	0.000007	0.000000	0.111111	0.996078
175337	8.429368e-03	0.856061	1.000000	0.250	0.000936	0.000729	0.000046	0.000024	0.000034	0.996078
175338	1.500000e-07	0.901515	0.083333	0.375	0.000104	0.000000	0.000007	0.000000	0.111111	0.996078
175339	1.500000e-07	0.901515	0.083333	0.375	0.000104	0.000000	0.000007	0.000000	0.111111	0.996078
175340	1.500000e-07	0.901515	0.083333	0.375	0.000104	0.000000	0.000007	0.000000	0.111111	0.996078

175341 rows × 44 columns

In [66]:
```
# importing the necessary libraries:
from sklearn.preprocessing import StandardScaler
```

In [67]:
```
data_encoded.dtypes.unique()
```

Out[67]:
```
array([dtype('float64'), dtype('int64'), dtype('bool')], dtype=object)
```

In [68]:
```
scaler = StandardScaler()

# Declaring the batch size
batch_size = 100

# Compute the number of batches
n_batches = int(np.ceil(data_encoded.shape[0] / batch_size))

# Initializing a scaled DataFrame
scaled_features = pd.DataFrame(index=data_encoded.index, columns=data_encoded.columns)

# Scaling in batches
for i in range(n_batches):
    start_i = i * batch_size
    end_i = min(start_i + batch_size, data_encoded.shape[0])  # Adjust to not exceed t

    # Selecting the batch
    data_encoded_batch = data_encoded.iloc[start_i:end_i]

    # Fitting the scaler on the first batch and transform, then only transform the res
    if i == 0:
        scaler.fit(data_encoded_batch)
```

```
    # Applying the transformation
    data_encoded_scaled_batch = scaler.transform(data_encoded_batch)

    # Replacing the original data with the scaled data
    scaled_features.iloc[start_i:end_i] = data_encoded_scaled_batch
```

In [69]:
```
scaled_features
```

Out[69]:

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload
0	-0.280141	-0.33471	-0.342067	-0.265402	-0.239901	-0.527975	0.642806	0.866227	-0.512984
1	-0.227075	0.608135	3.433996	-0.189186	4.64474	-0.527907	-1.3023	0.850028	-0.513015
2	-0.129341	-0.098999	0.990661	-0.248429	1.279355	-0.528903	-1.3023	0.850028	-0.513052
3	-0.123464	0.372424	0.546418	-0.206158	-0.17009	-0.52891	-1.3023	0.850028	-0.513045
4	-0.247205	0.136713	-0.119946	-0.221209	-0.228694	-0.528605	0.663281	0.850028	-0.513014
...	...	...	...	...	...	...	...	...	...
175336	-0.29234	-0.806133	-0.78631	-0.288459	-0.25998	1.191388	0.663281	-1.191012	-0.240204
175337	-0.24155	0.136713	0.102176	-0.207439	-0.218654	-0.528602	0.663281	0.850028	-0.513013
175338	-0.29234	-0.806133	-0.78631	-0.288459	-0.25998	1.191388	0.663281	-1.191012	-0.240204
175339	-0.29234	-0.806133	-0.78631	-0.288459	-0.25998	1.191388	0.663281	-1.191012	-0.240204
175340	-0.29234	-0.806133	-0.78631	-0.288459	-0.25998	1.191388	0.663281	-1.191012	-0.240204

175341 rows × 194 columns

In [70]:
```
y.unique()
```

Out[70]:
```
array(['Normal', 'Backdoor', 'Analysis', 'Fuzzers', 'Shellcode',
       'Reconnaissance', 'Exploits', 'DoS', 'Worms', 'Generic'],
      dtype=object)
```

In [70]:

In [70]:

In [70]:

In [70]:

performing outlier testing on a part of 'data':

In [70]:

In [71]:
```
# applying log transformation:
```

In [40]:
```
# Adding a small constant if there are zeros in the dataset
data_encoded += 1

# Apply log transformation
data_encoded_log = np.log1p(data_normalized)


# Checking for infinite values
if np.isinf(data_encoded_log).any().any():
    print("Infinite values detected")


data_encoded_log.replace(-np.inf, np.nan, inplace=True)
# You can then impute the NaN values or remove those rows.
```

In [41]:
```
data_encoded
```

Out[41]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_s
0	1.121478	114	13	3	7	5	259	173	75.087490	253	...	
1	1.649902	114	13	3	15	39	735	42015	79.473372	63	...	
2	2.623129	114	13	3	9	17	365	13187	15.170161	63	...	
3	2.681642	114	3	3	13	13	629	771	14.677108	63	...	
4	1.449454	114	13	3	11	7	535	269	34.373826	255	...	
...	...	...	...	...	...	...	...	...	...	...	...	
175336	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	
175337	1.505762	114	13	3	11	9	621	355	34.612649	255	...	
175338	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	
175339	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	
175340	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	

175341 rows × 44 columns

In [42]:
```
data_encoded_log
```

Out[42]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	s
0	2.022587e-03	0.618456	0.693147	0.223144	0.000520	0.000364	0.000018	0.000012	0.000074	0.687
1	1.077346e-02	0.618456	0.693147	0.223144	0.001351	0.003457	0.000054	0.002863	0.000078	0.217
2	2.669271e-02	0.618456	0.693147	0.223144	0.000728	0.001457	0.000026	0.000899	0.000014	0.217
3	2.764179e-02	0.618456	0.154151	0.223144	0.001143	0.001093	0.000046	0.000053	0.000014	0.217
4	7.462984e-03	0.618456	0.693147	0.223144	0.000936	0.000547	0.000039	0.000018	0.000033	0.691

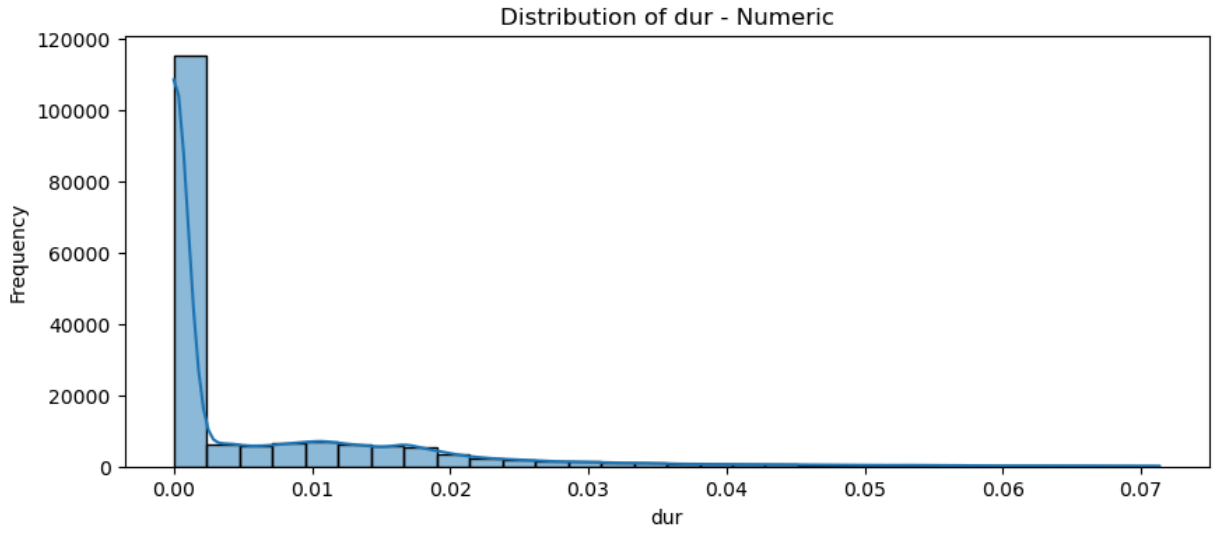
	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	s
...	...	...	...	...	...	...	...	...	...	
175336	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691
175337	8.394039e-03	0.618456	0.693147	0.223144	0.000936	0.000729	0.000046	0.000024	0.000034	0.691
175338	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691
175339	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691
175340	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691

175341 rows × 44 columns

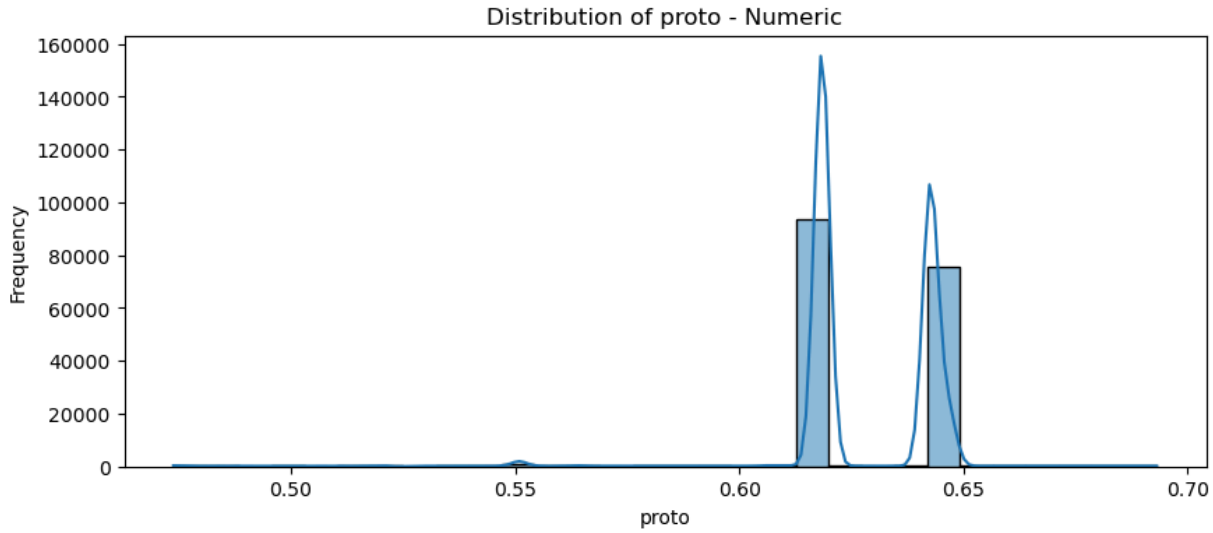
In [162]:

```
# Before log transformation
# representation of skewed data:

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming 'data' is your DataFrame
# Iterate over each column in the DataFrame
for column in data_encoded_log.columns:
    plt.figure(figsize=(10, 4))
    # Check if the column is numeric or categorical
    if pd.api.types.is_numeric_dtype(data_encoded_log[column]):
        # Plot distribution of numeric columns
        sns.histplot(data_encoded_log[column], kde=True, bins=30)
        plt.title(f'Distribution of {column} - Numeric')
    else:
        # Plot count of categorical columns
        sns.countplot(y=column, data=data_encoded_log, order=data_encoded_log[column].
        plt.title(f'Distribution of {column} - Categorical')

    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

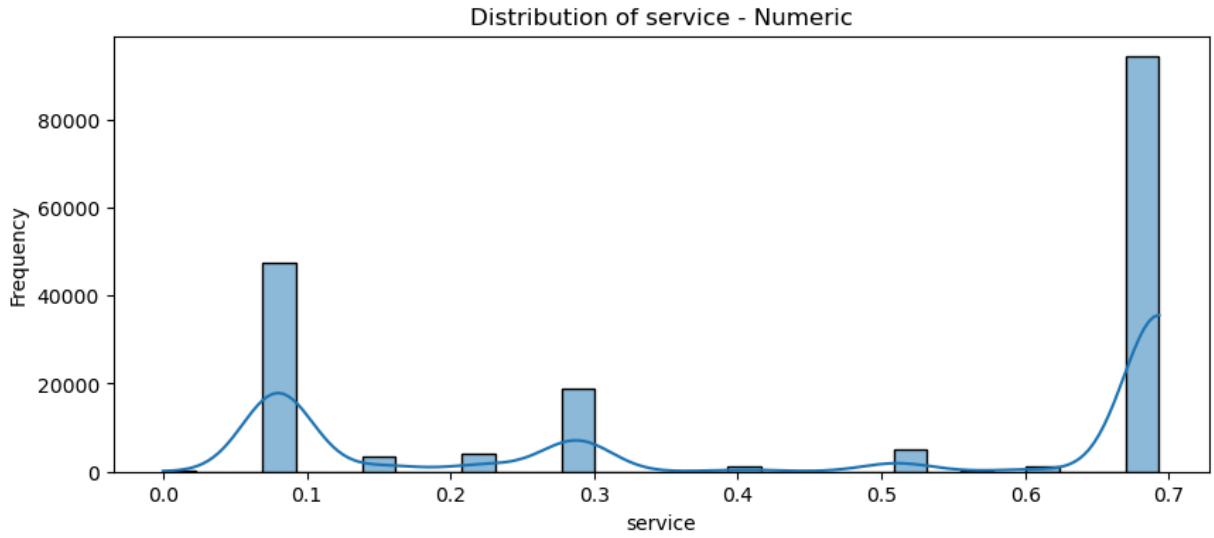
```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



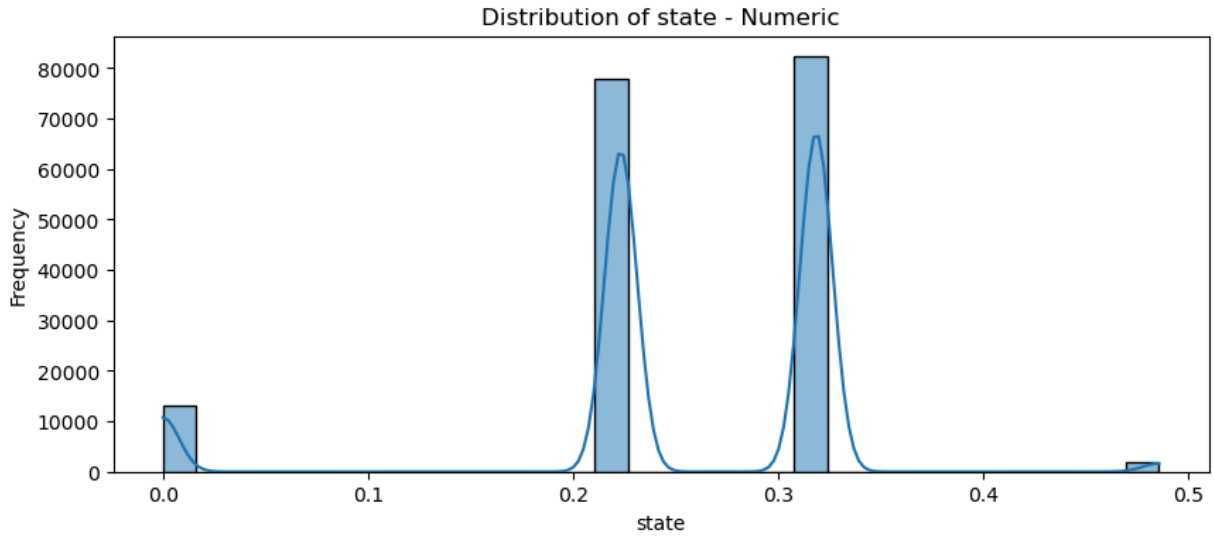

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



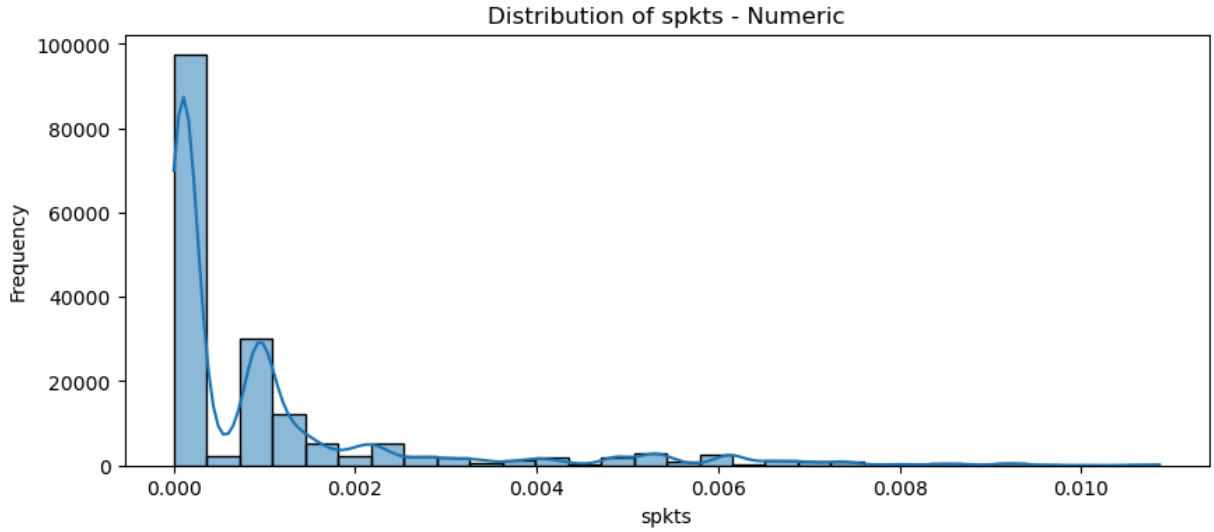

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



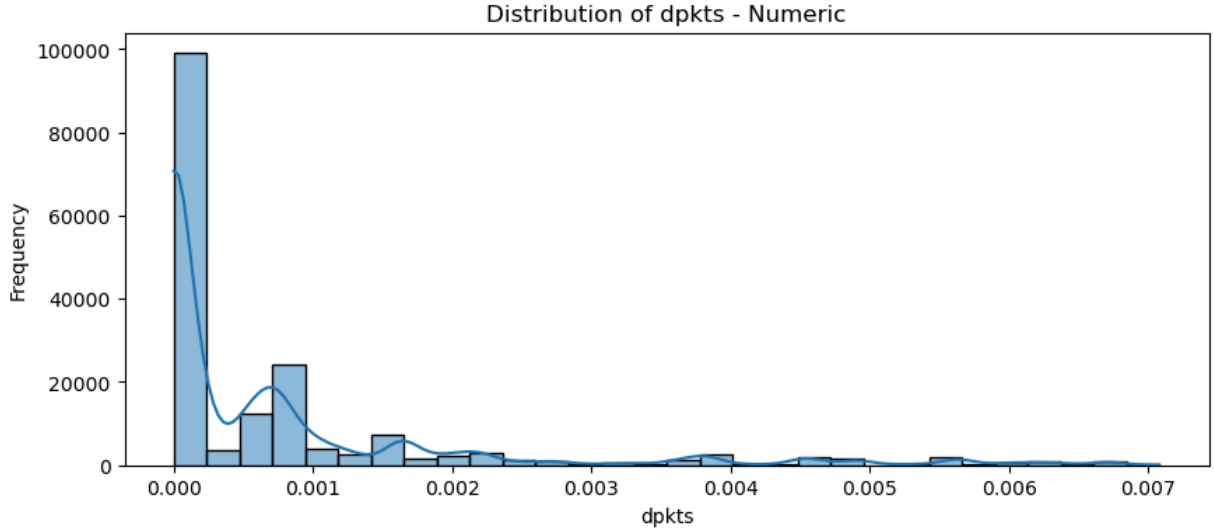

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



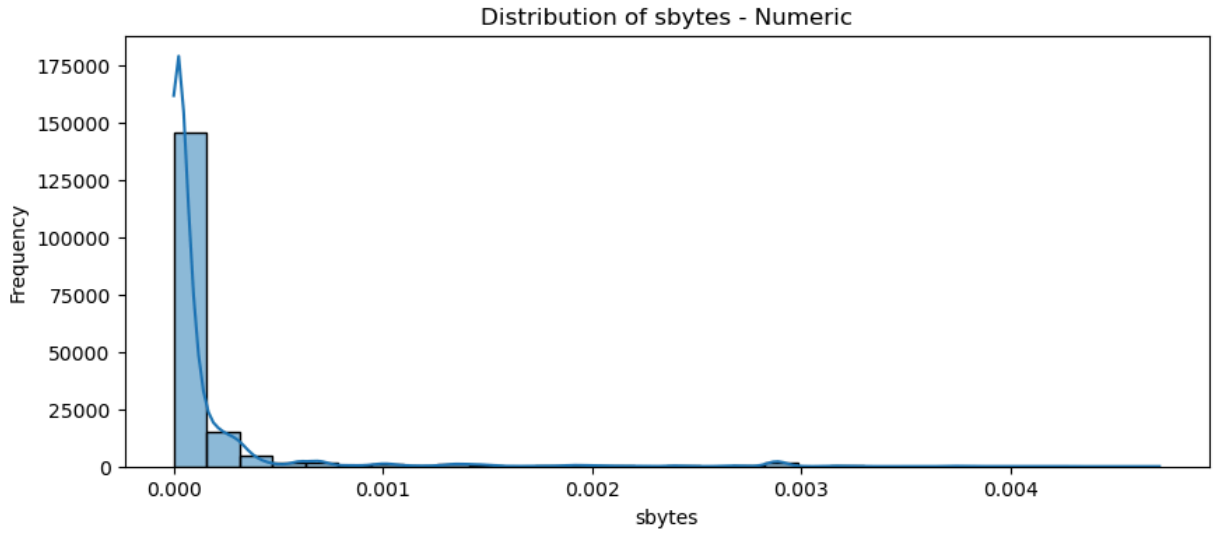

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



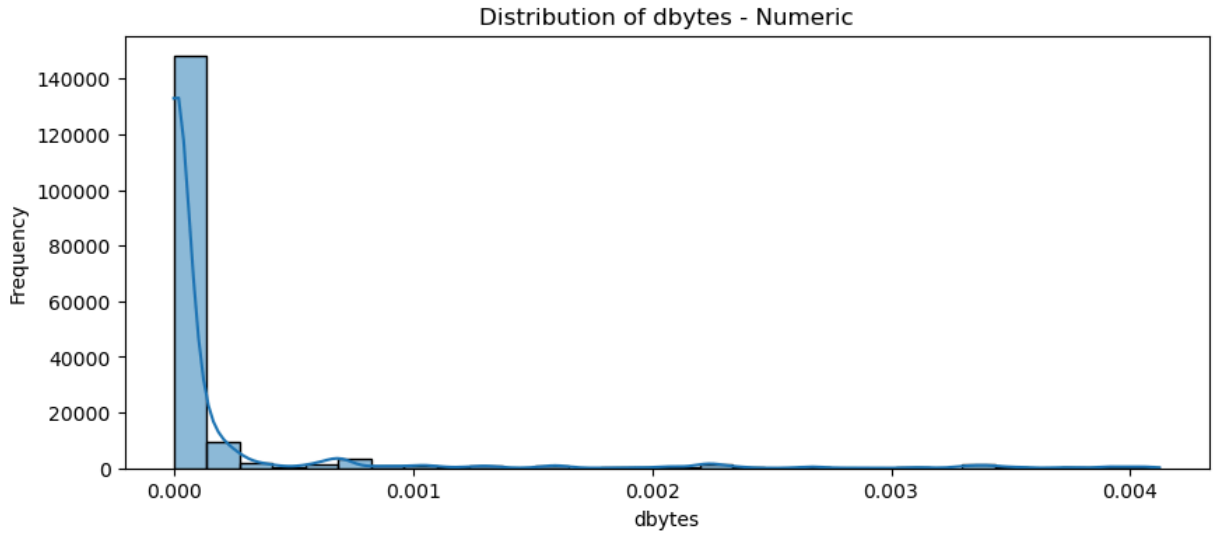

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



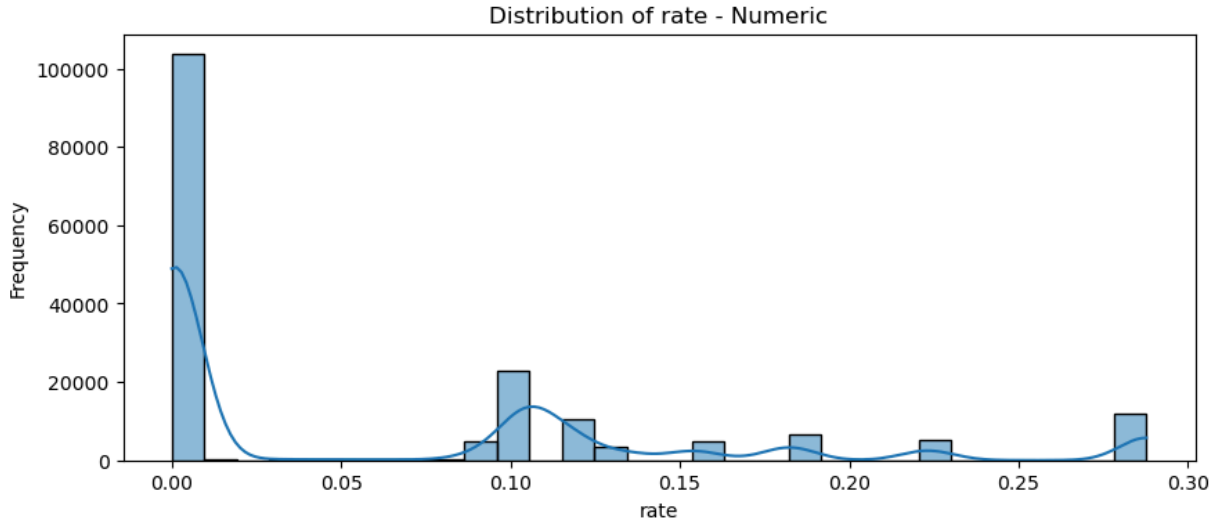

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



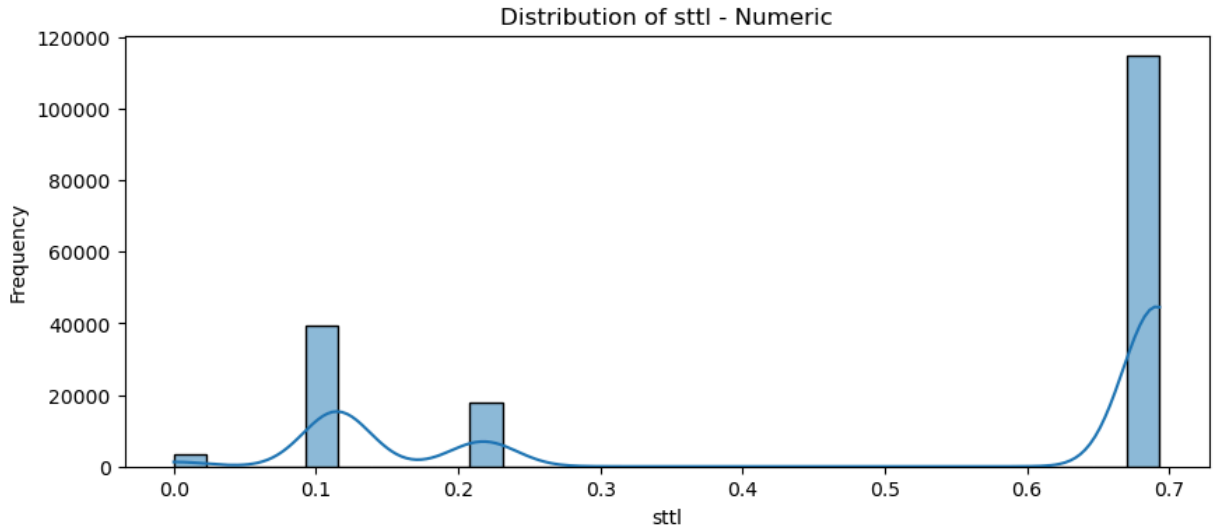

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



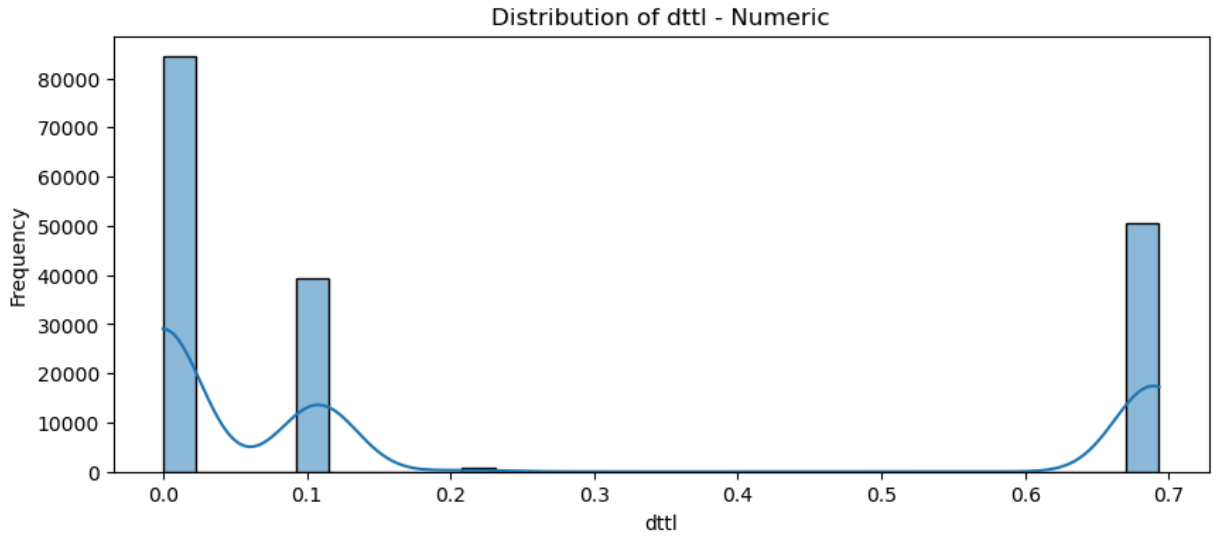

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



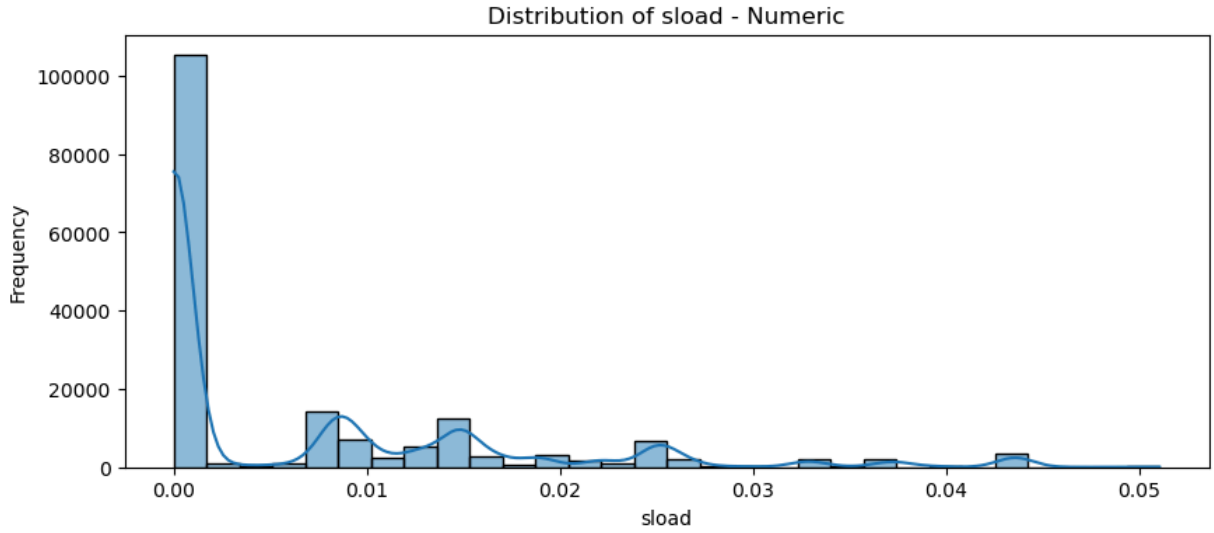

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



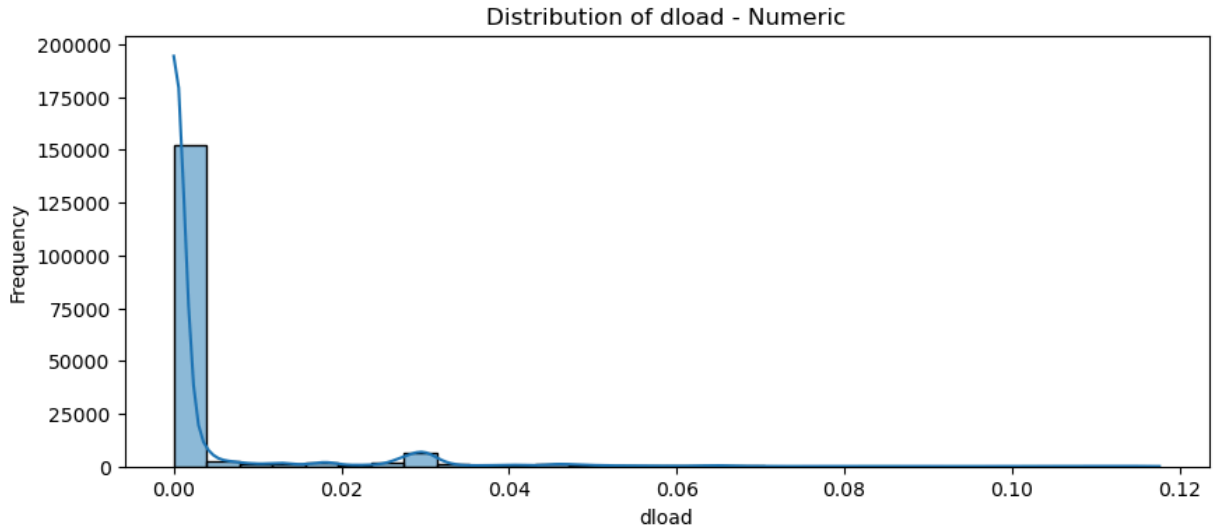

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



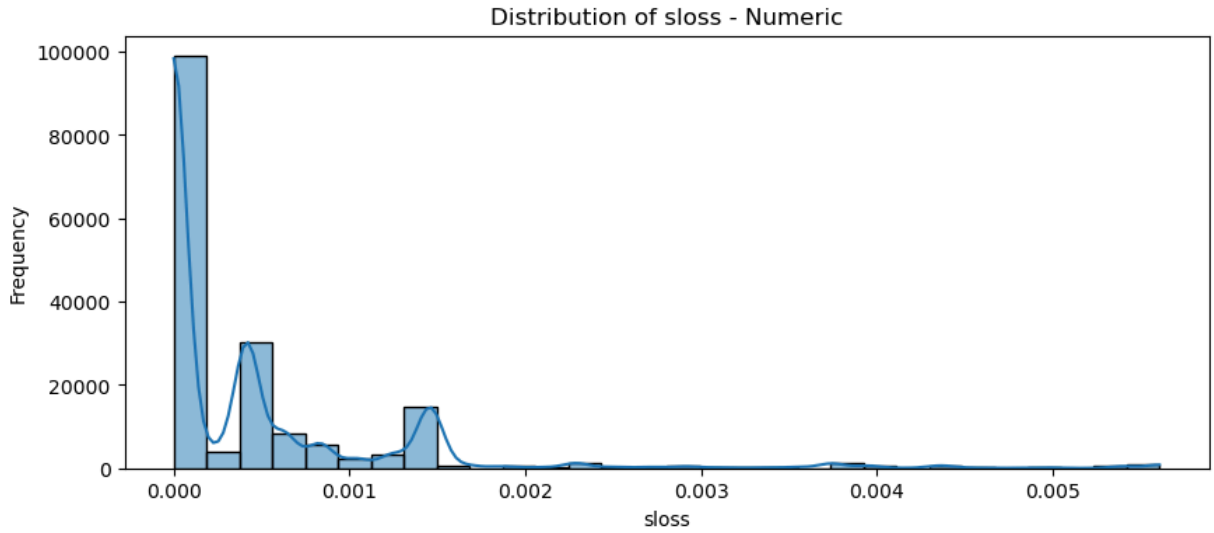

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



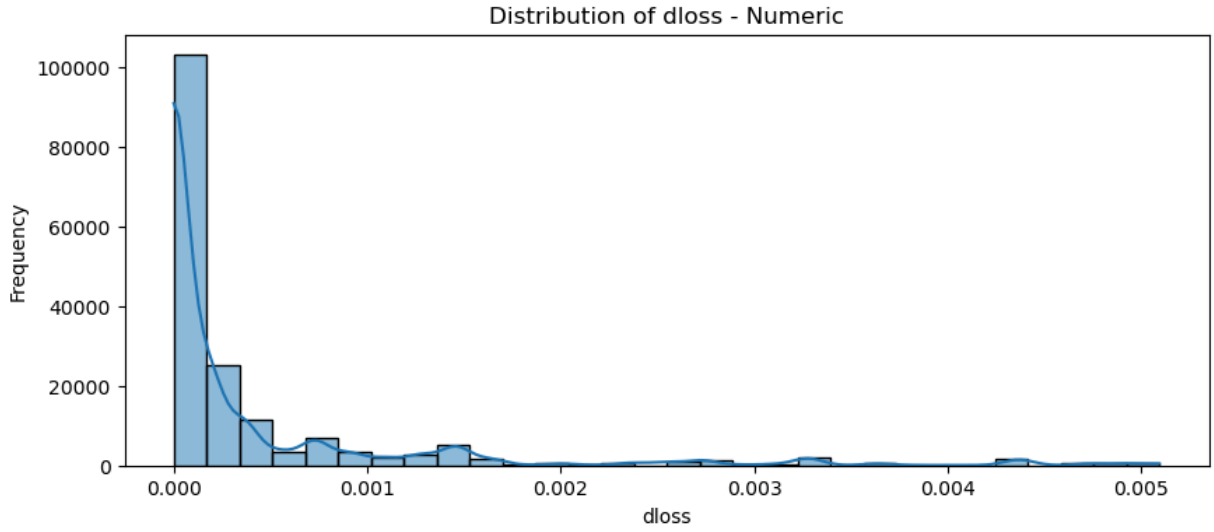

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



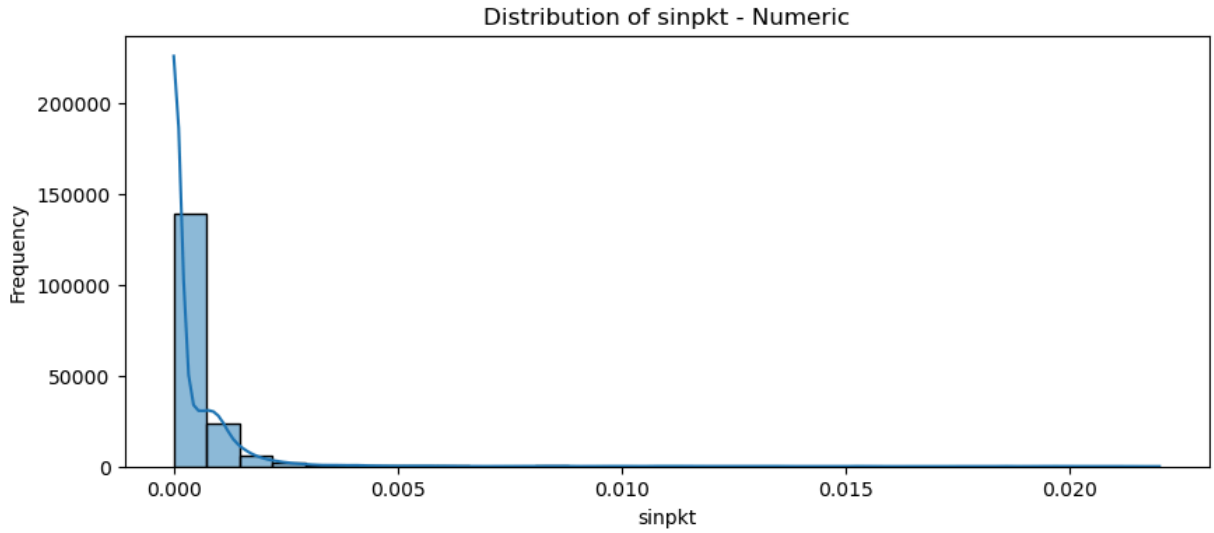

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



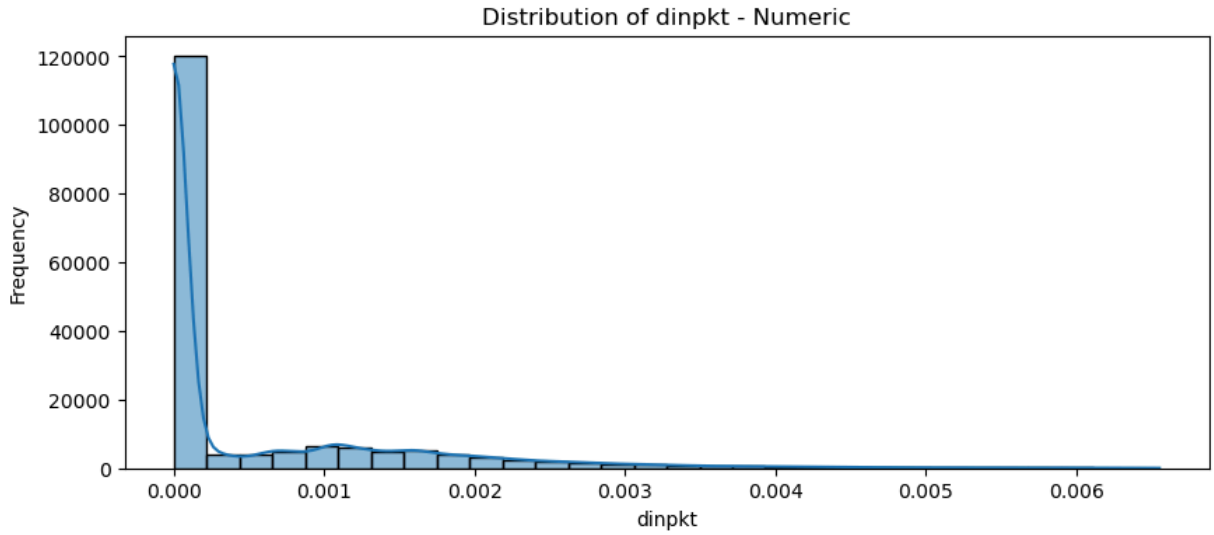

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



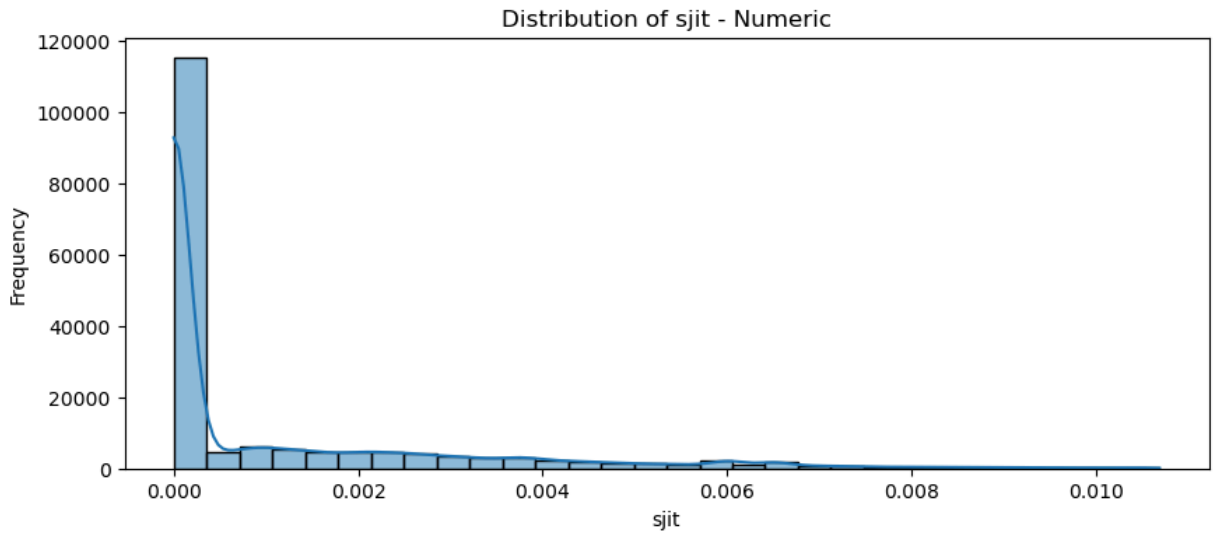

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



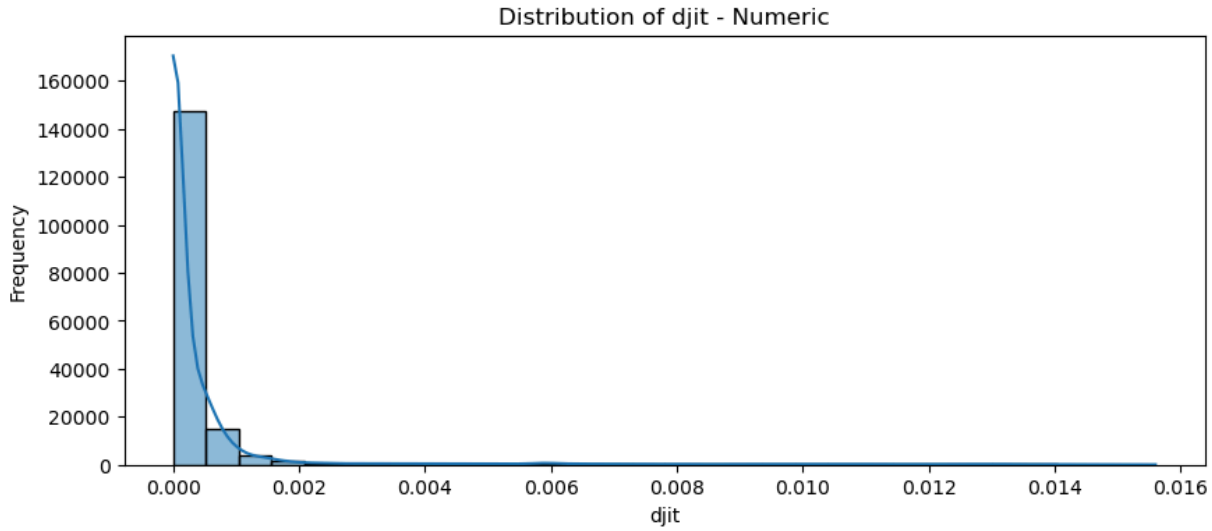

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



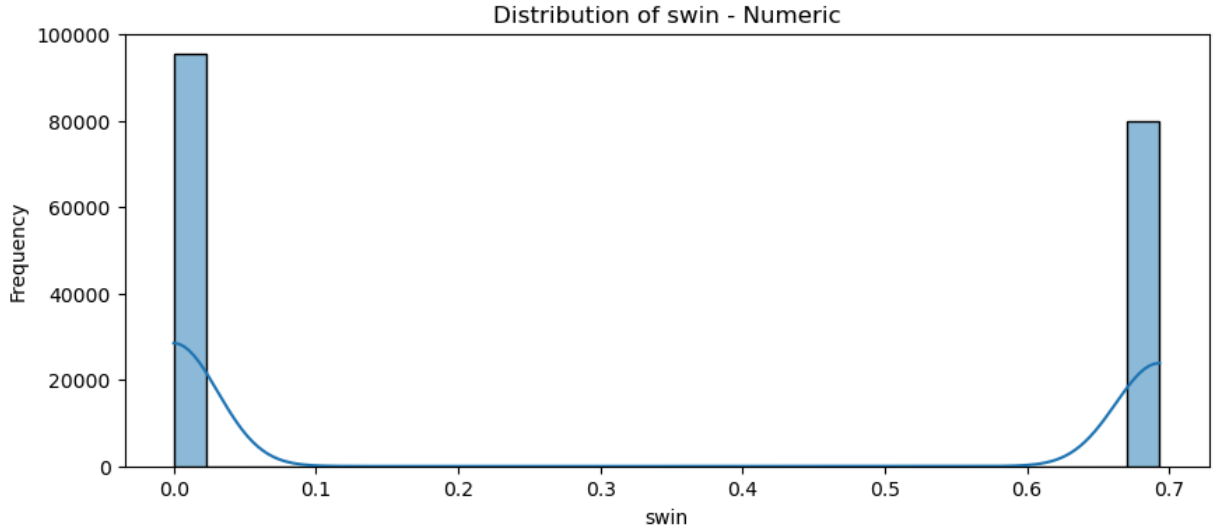

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



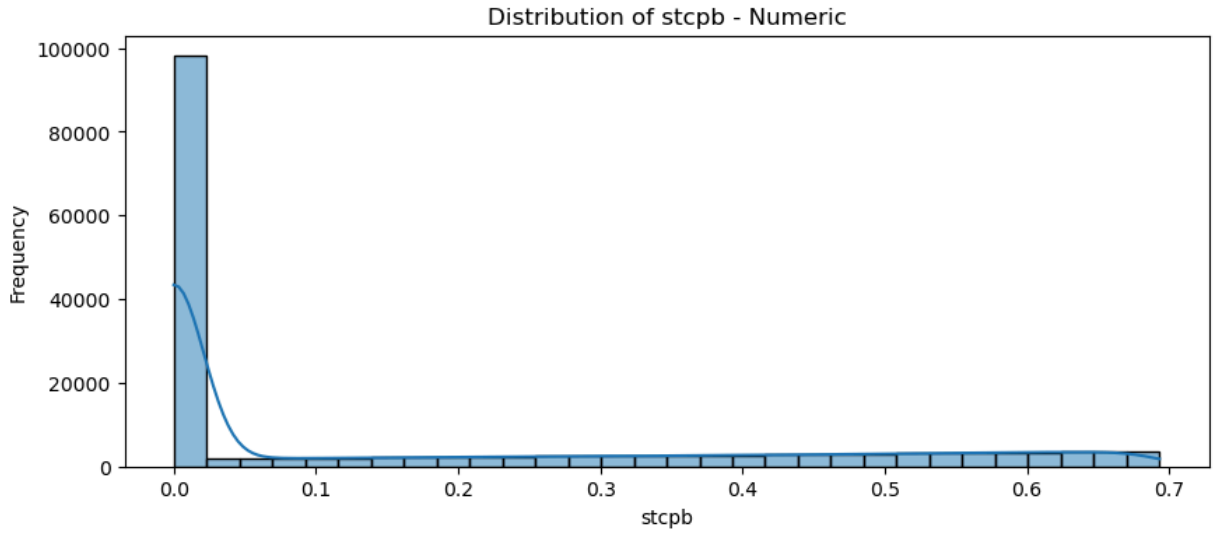

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



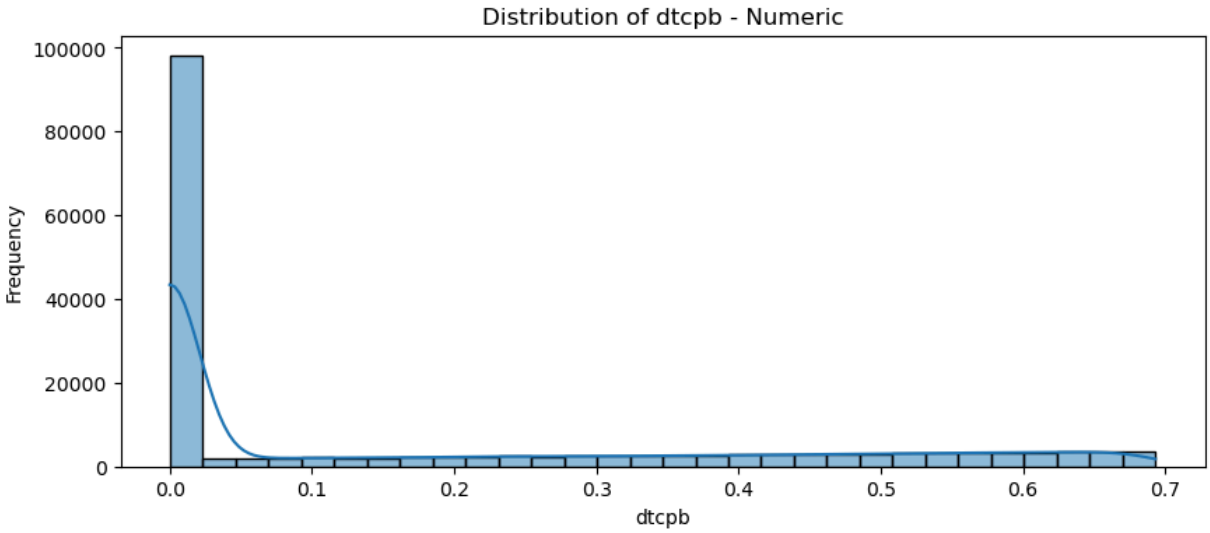

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



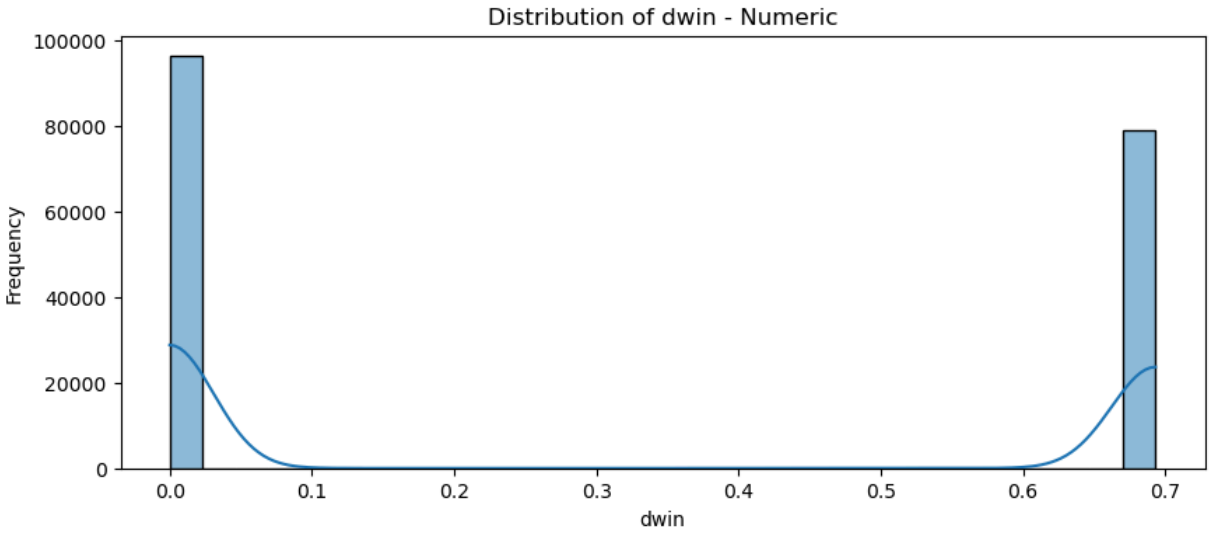

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



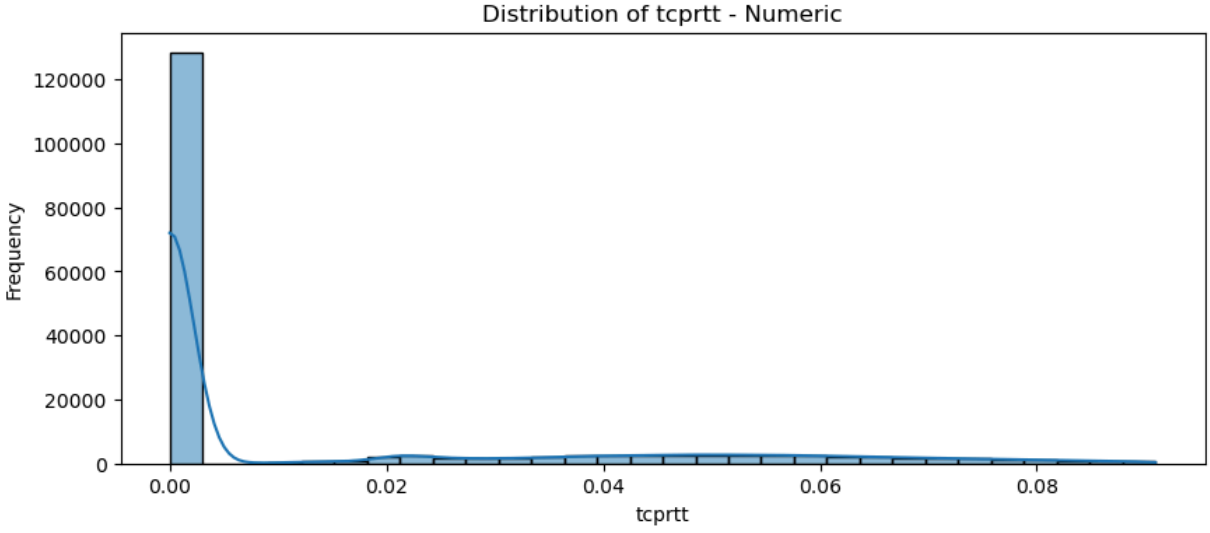

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



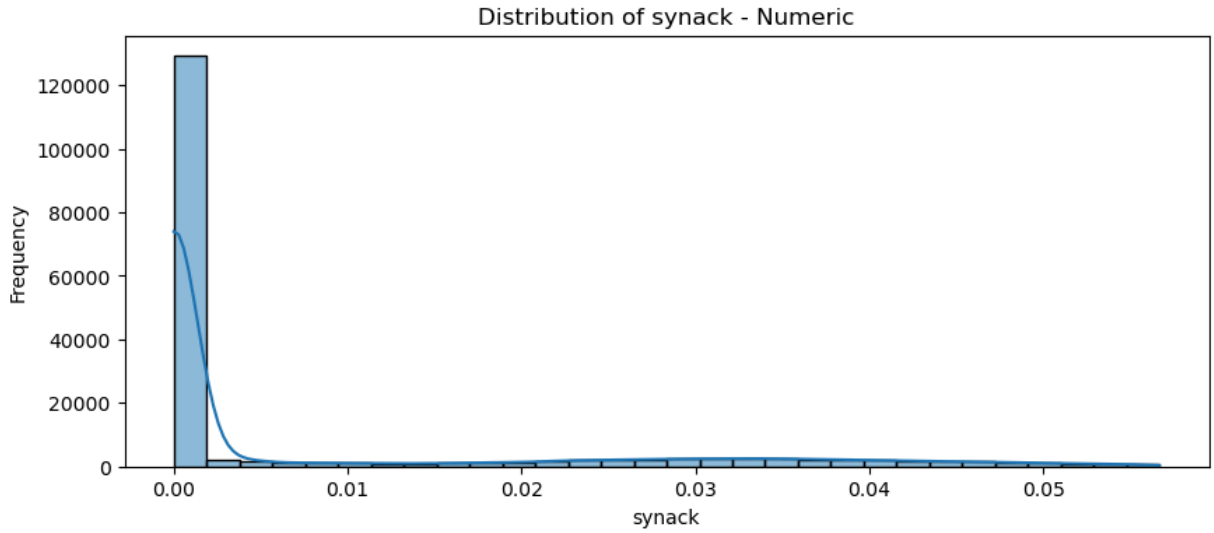

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



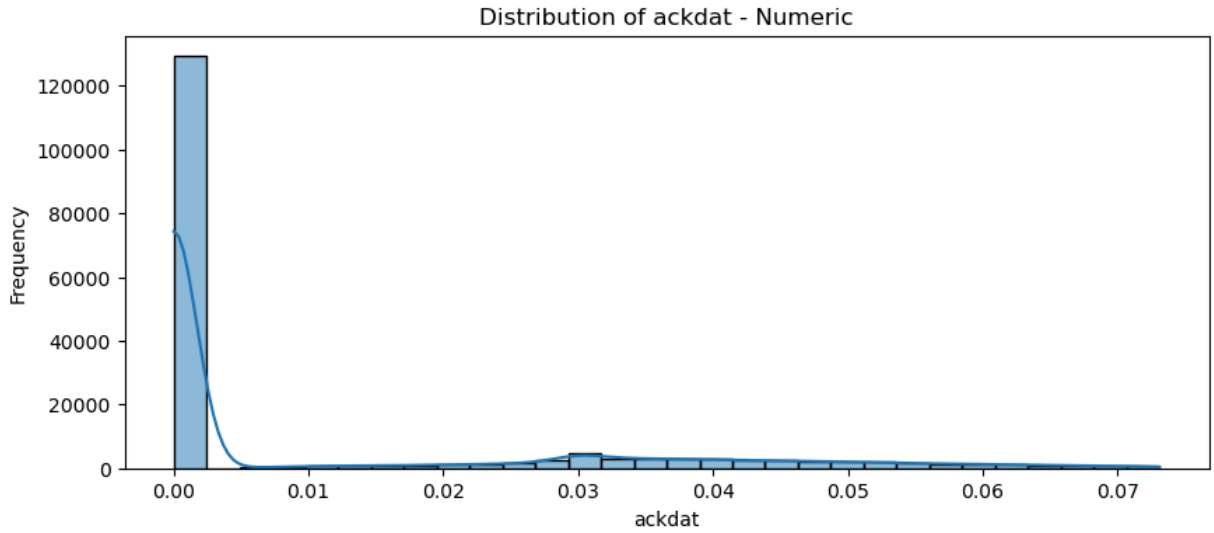

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



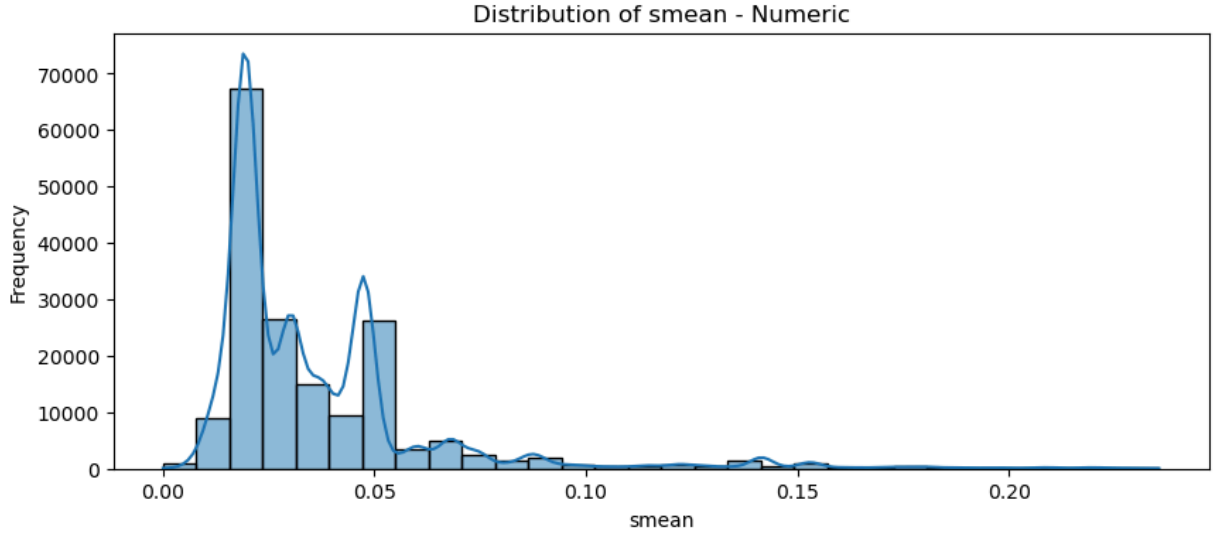

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



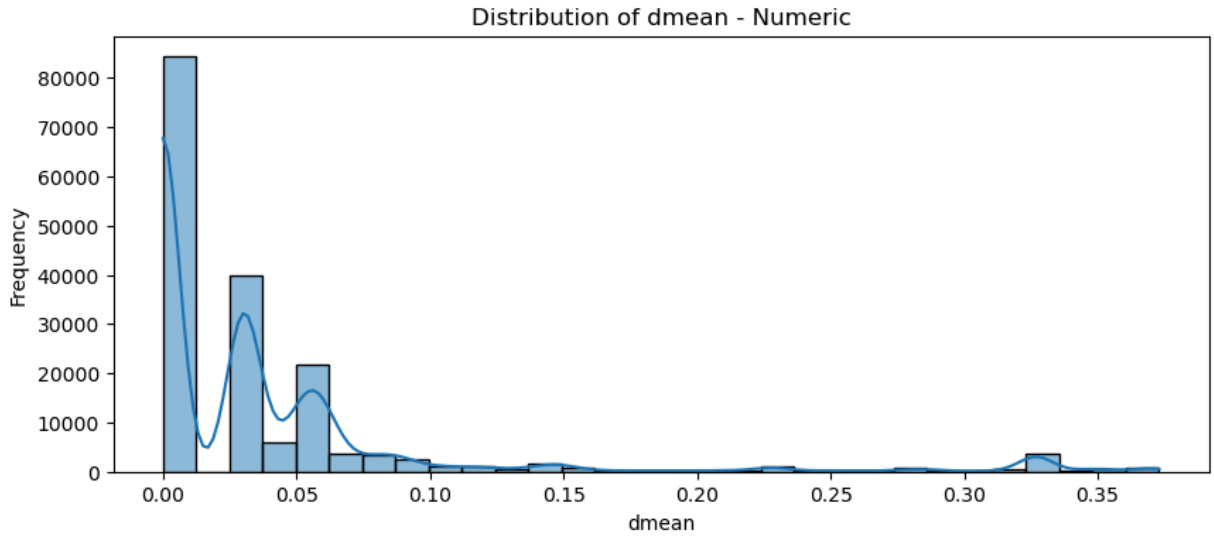

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



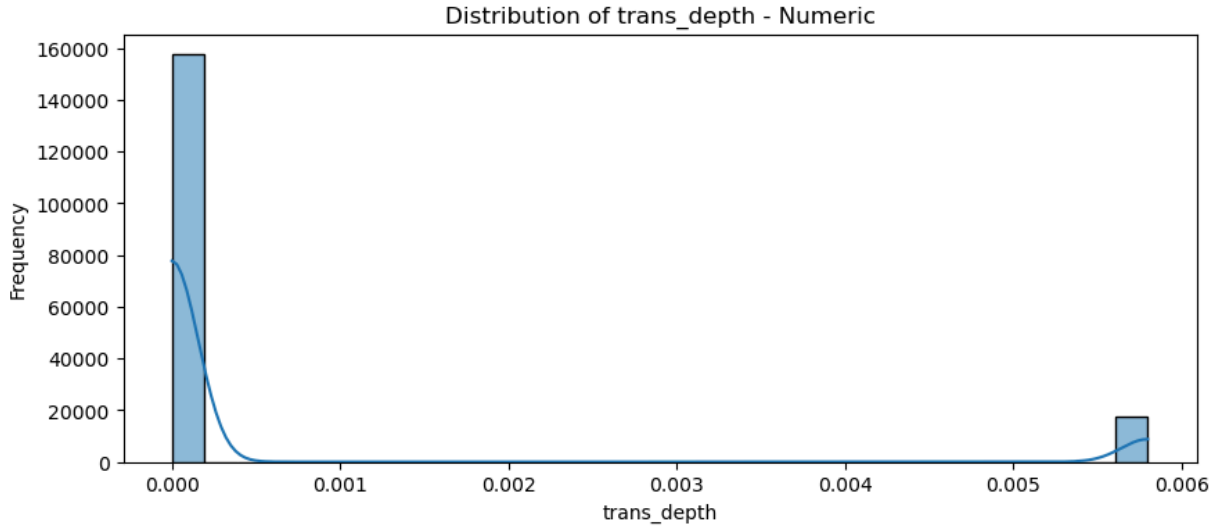

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



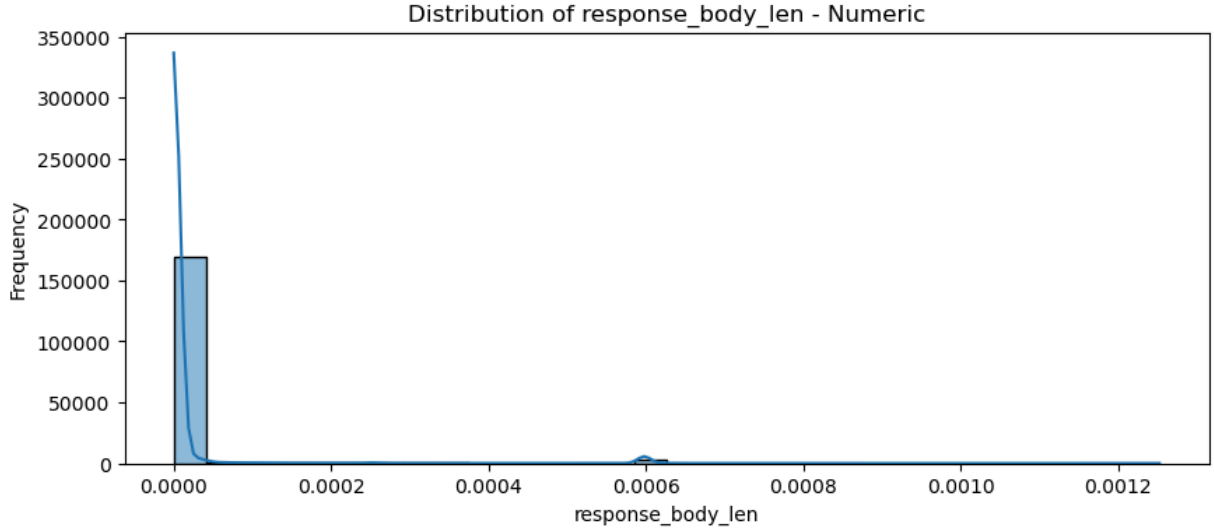

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



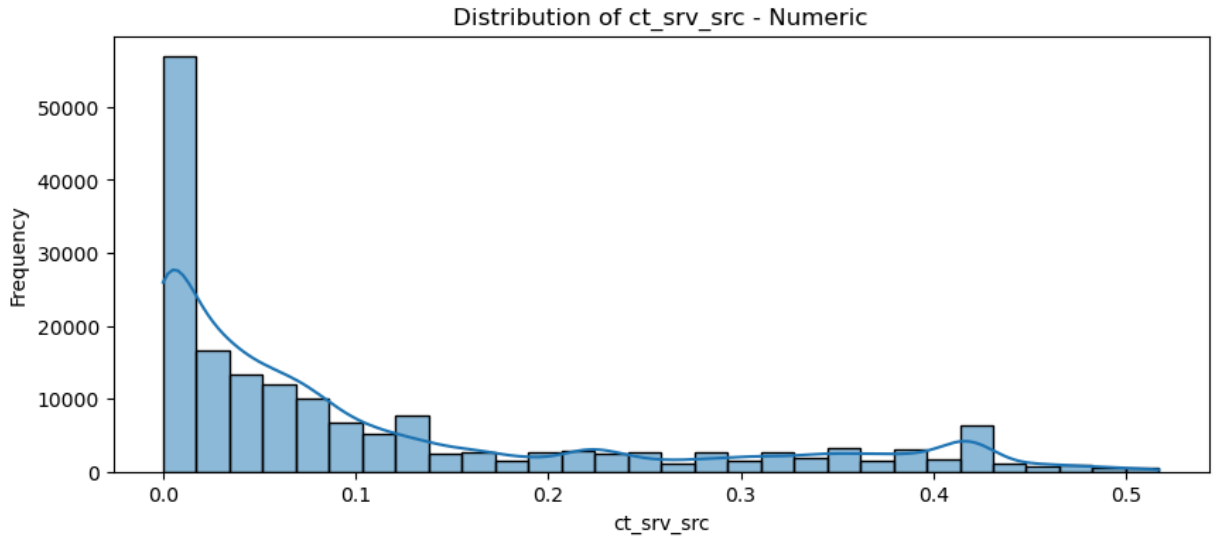

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



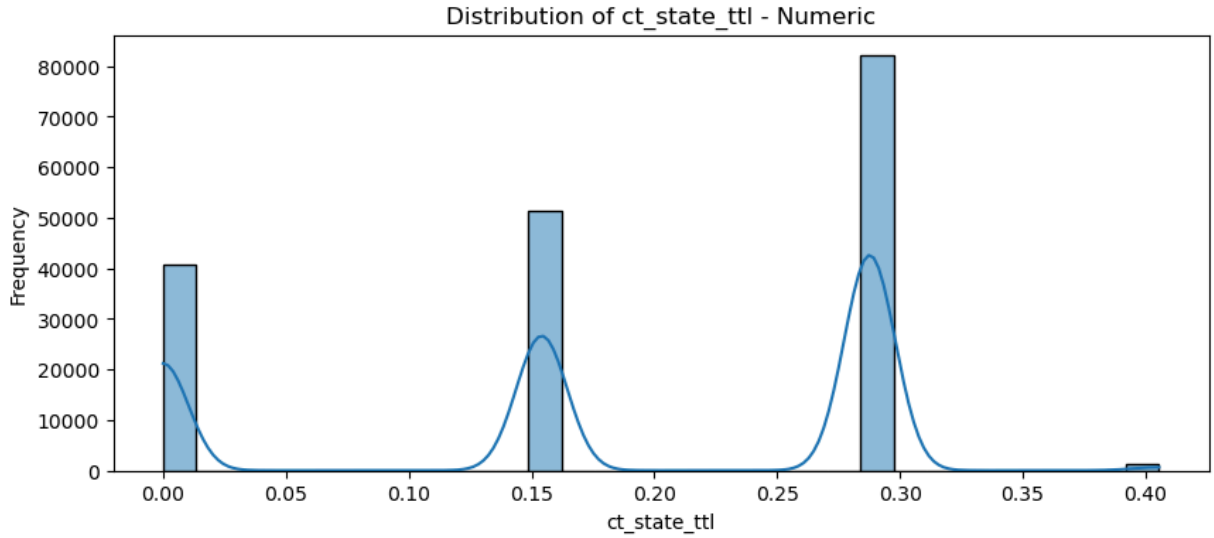

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



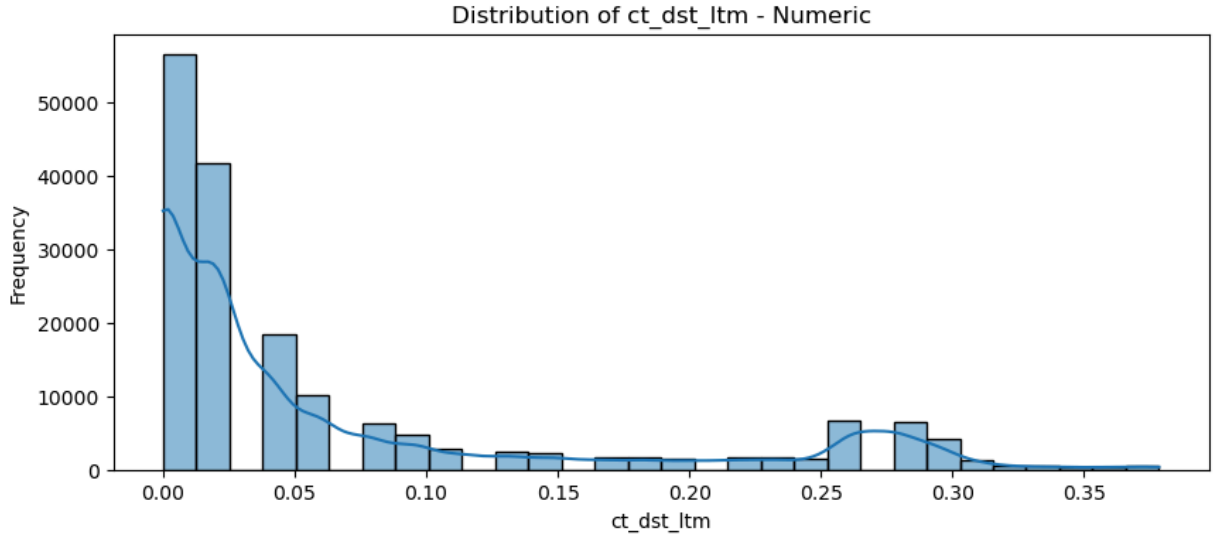

```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

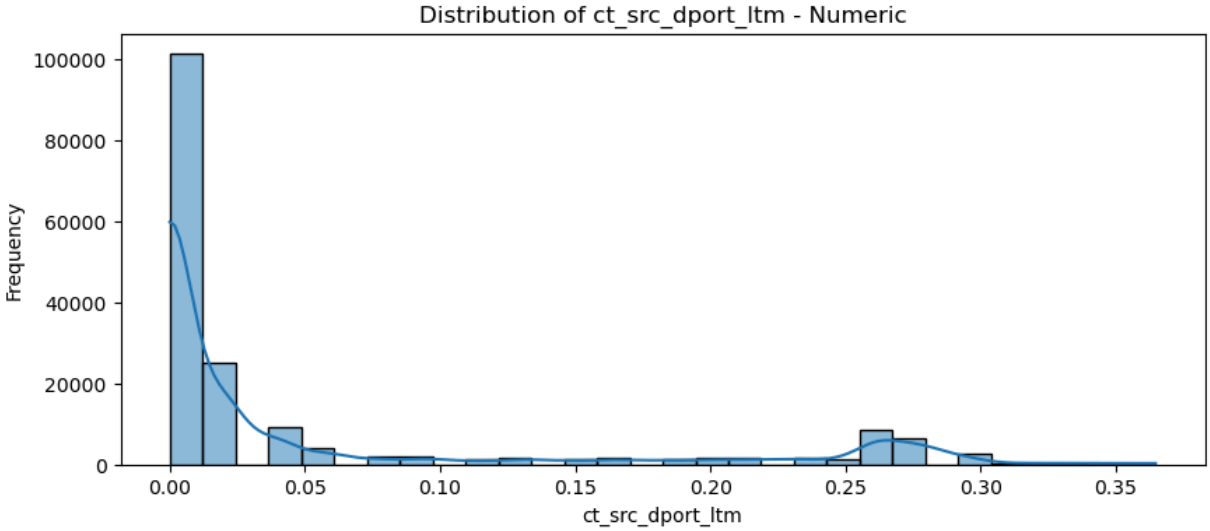



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

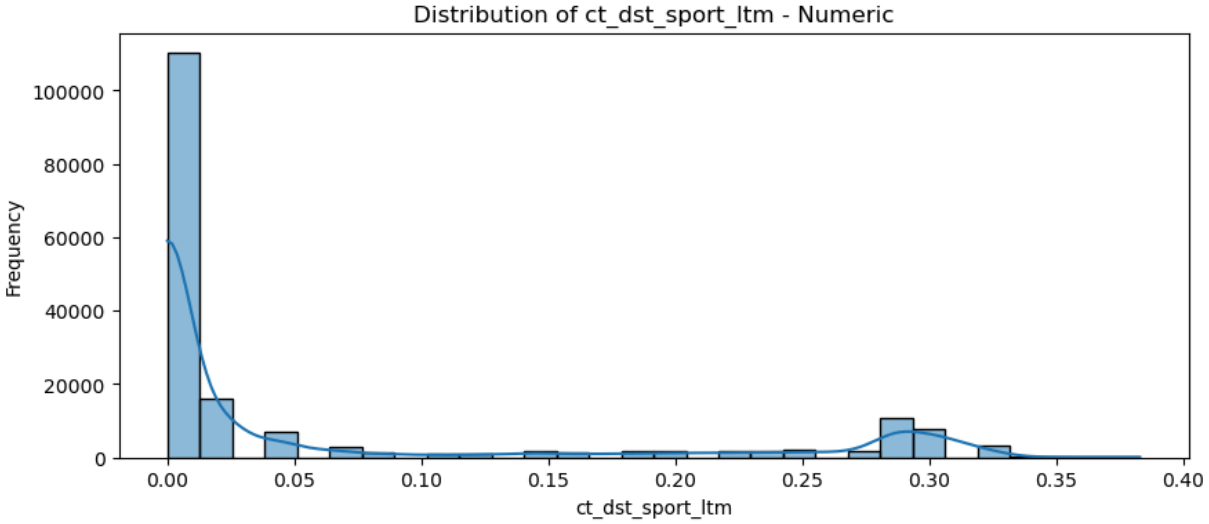



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

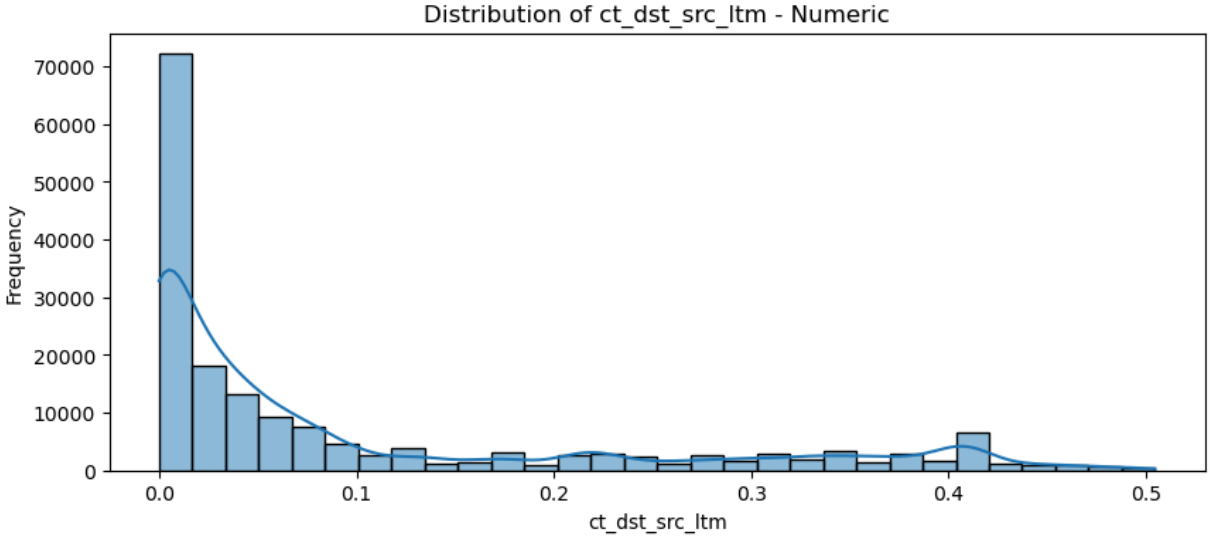
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

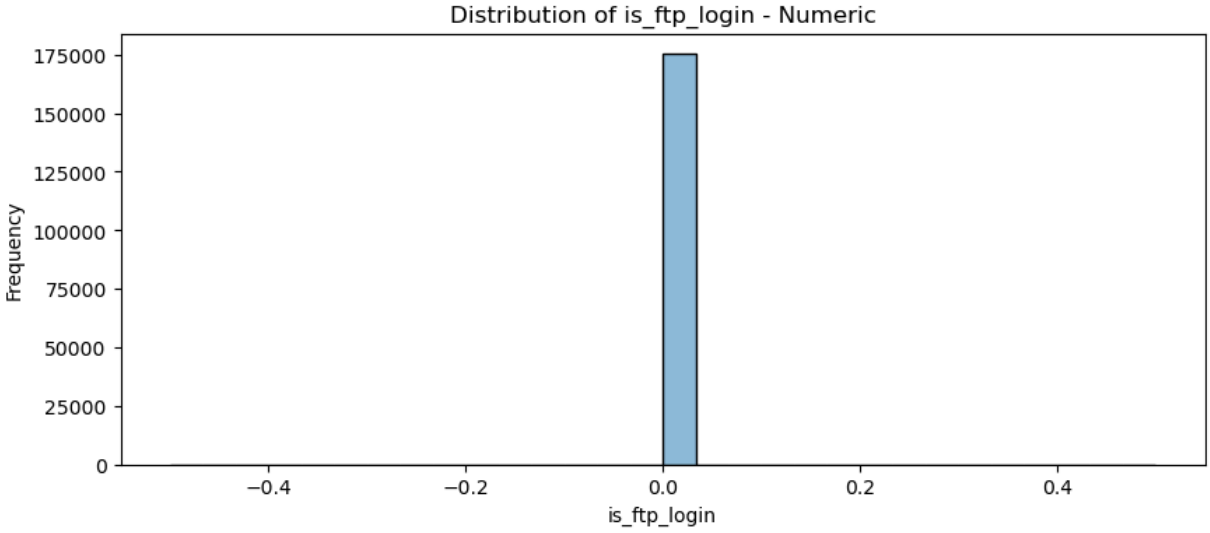



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

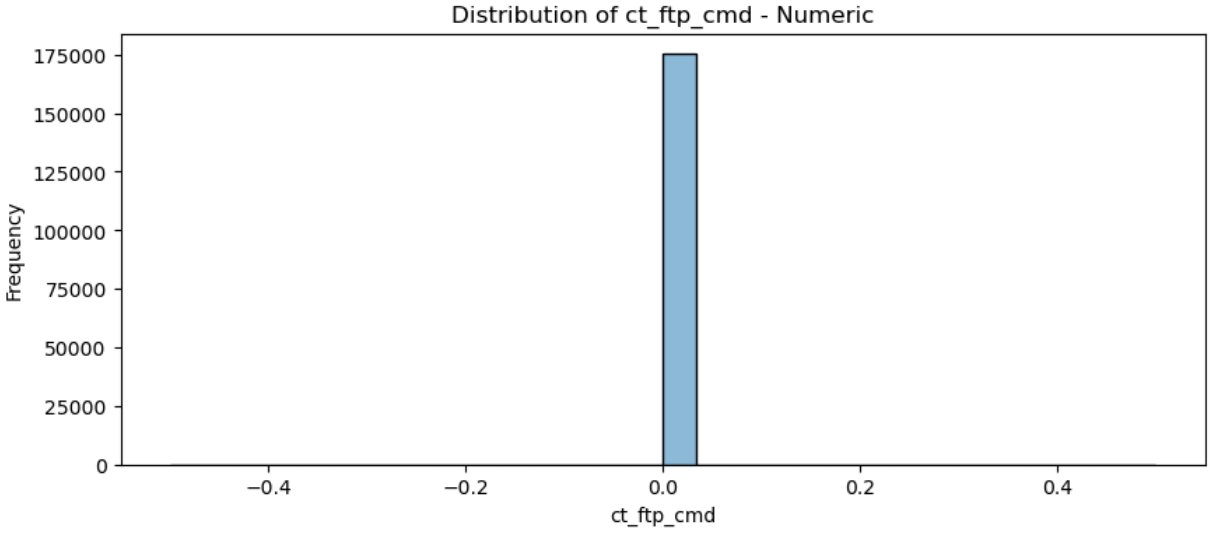



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

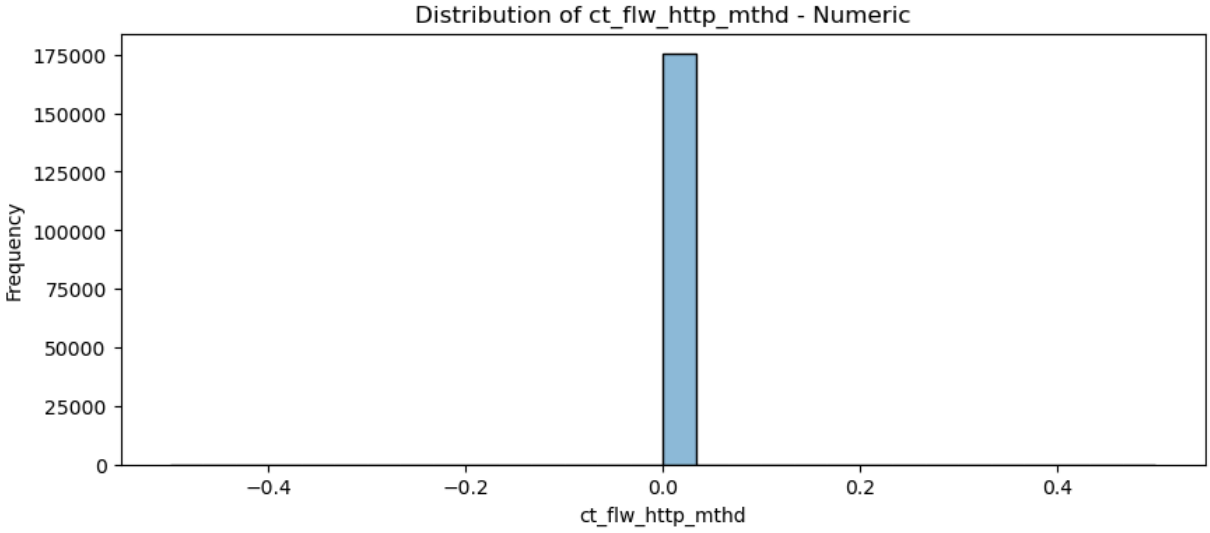
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

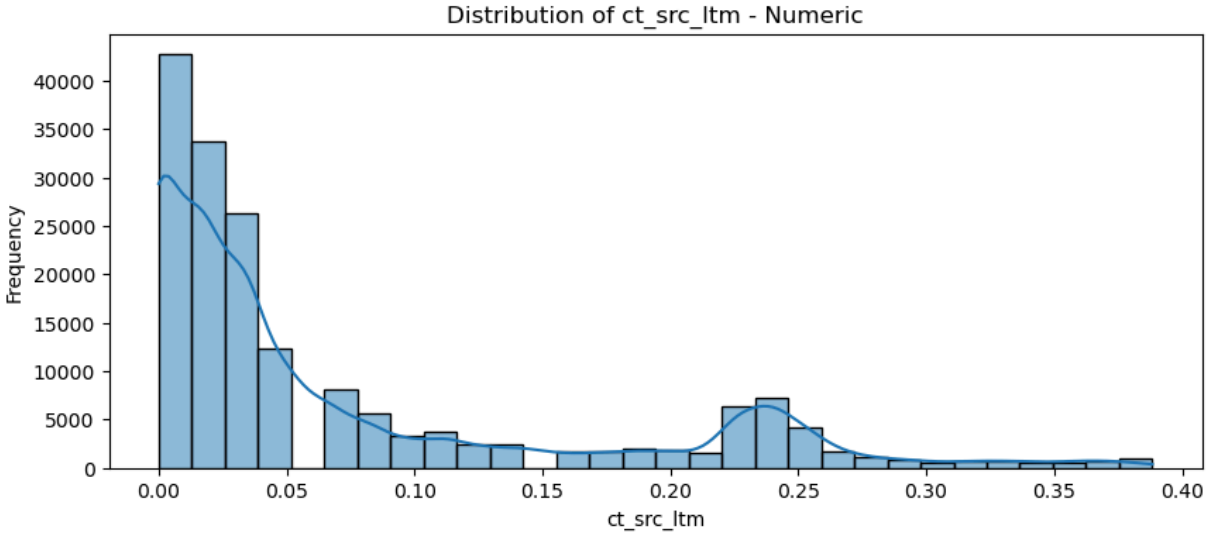



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

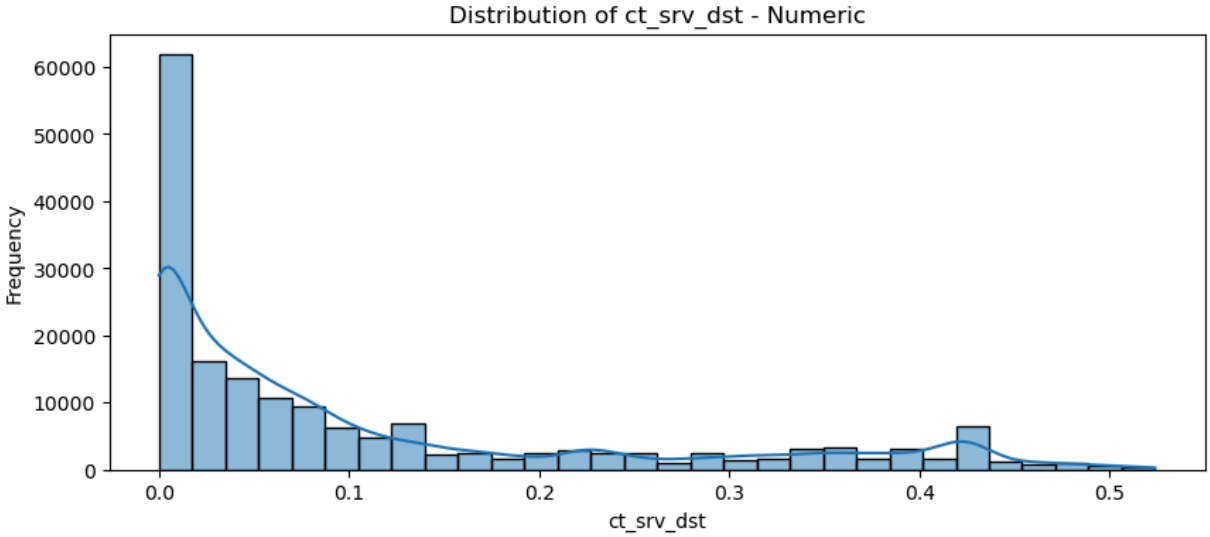



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

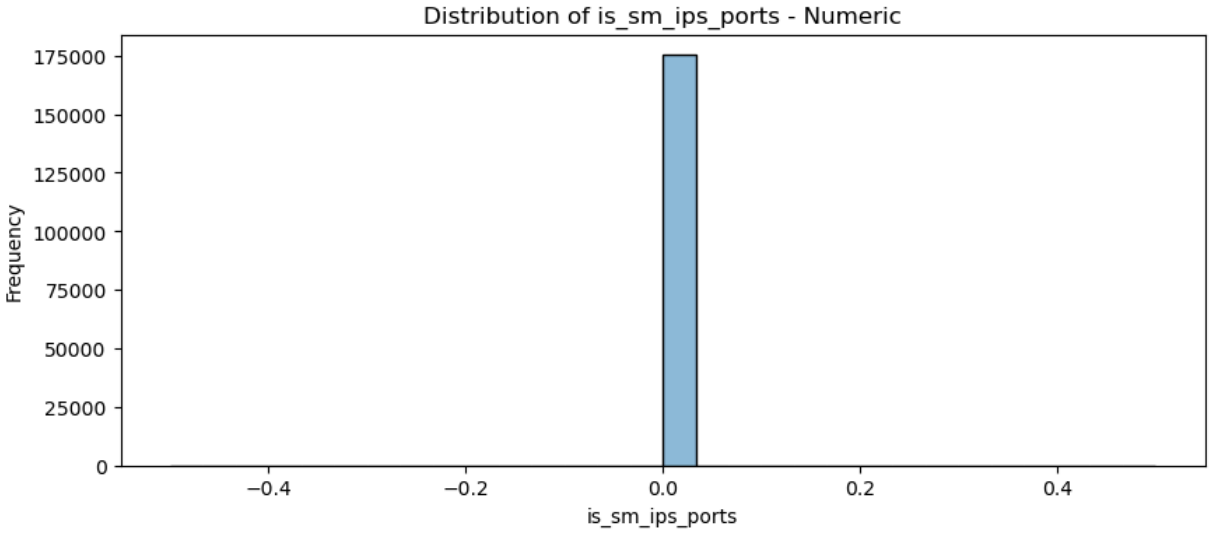
```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

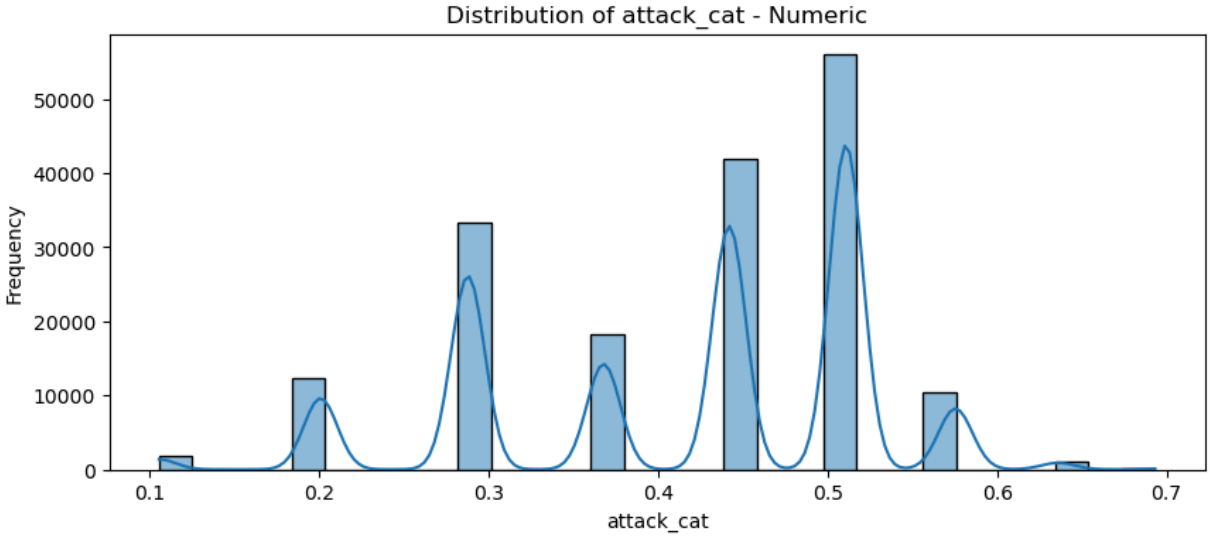



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

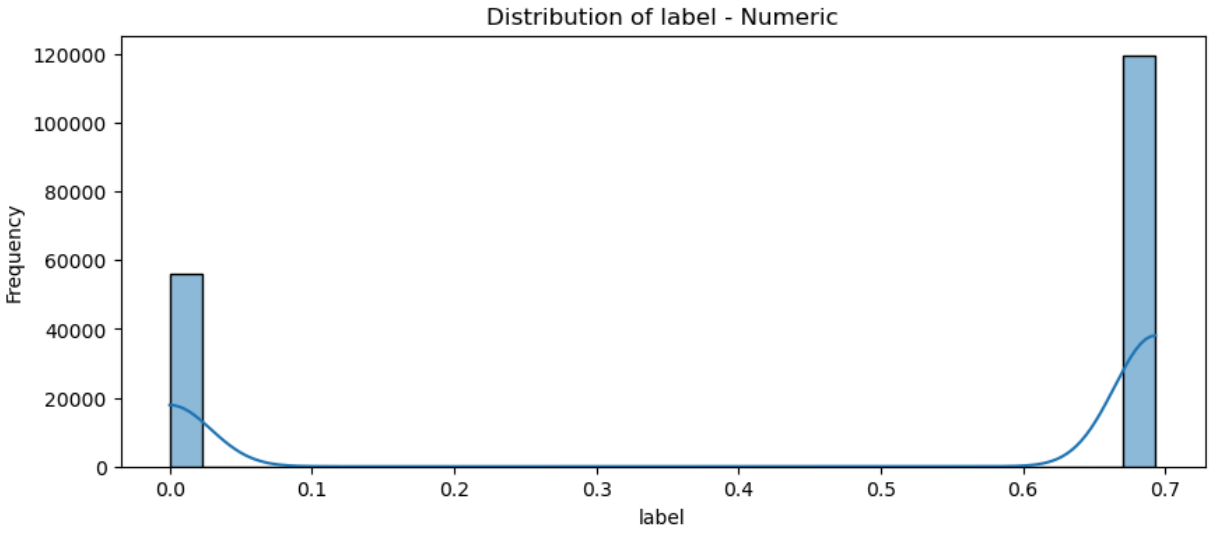



```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
```

```
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\anjal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:
```
# applied z-score for detecting outliers:
```

In [76]:
```
import numpy as np
from scipy import stats

# Assuming `data_encoded` is your DataFrame with all numeric data
# Calculate Z-scores
z_scores = np.abs(stats.zscore(data_encoded_log))

# Define a threshold
threshold = 3

# Get boolean array where true represents the presence of an outlier
outliers = (z_scores > threshold)

# Optional: Get the actual data points that are considered outliers
outlier_points = data_encoded[(z_scores > threshold).any(axis=1)]
```

In [77]:
```
outlier_points
```

Out[77]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_s
2	2.623129	114	13	3	9	17	365	13187	15.170161	63	...	
3	2.681642	114	3	3	13	13	629	771	14.677108	63	...	
11	3.093085	114	9	3	63	29	56330	2213	43.520967	63	...	
12	1.416952	114	13	3	11	7	535	269	36.975363	255	...	
15	1.000002	120	10	4	3	1	139	1	500001.001300	255	...	
...	...	...	...	...	...	...	...	...	...	...	...	
175334	1.000006	120	2	4	3	1	115	1	166667.660800	255	...	
175335	1.000006	120	2	4	3	1	115	1	166667.660800	255	...	
175336	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	
175339	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	
175340	1.000009	120	2	4	3	1	115	1	111112.107200	255	...	

64097 rows × 44 columns

In [ ]:

In [78]:
```
# Example: Replacing outliers with the median
median = data_encoded_log.median()
z_scores = np.abs(stats.zscore(data_encoded_log))
outlier_positions = z_scores > 3
data_encoded_log[outlier_positions] = np.nan
data_encoded_log.fillna(median, inplace=True)
```

In [125]:
```
data_encoded_log
```

Out[125]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	s
0	2.022587e-03	0.618456	0.693147	0.223144	0.000520	0.000364	0.000018	0.000012	0.000074	0.687
1	1.077346e-02	0.618456	0.693147	0.223144	0.001351	0.003457	0.000054	0.002863	0.000078	0.217
2	2.669271e-02	0.618456	0.693147	0.223144	0.000728	0.001457	0.000026	0.000899	0.000014	0.217
3	2.764179e-02	0.618456	0.154151	0.223144	0.001143	0.001093	0.000046	0.000053	0.000014	0.217
4	7.462984e-03	0.618456	0.693147	0.223144	0.000936	0.000547	0.000039	0.000018	0.000033	0.691
...	...	...	...	...	...	...	...	...	...	
175336	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691
175337	8.394039e-03	0.618456	0.693147	0.223144	0.000936	0.000729	0.000046	0.000024	0.000034	0.691
175338	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	s
175339	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691
175340	1.500000e-07	0.642651	0.080043	0.318454	0.000104	0.000000	0.000007	0.000000	0.105361	0.691

175341 rows × 44 columns

In [ ]:

applying embedded method:

In [ ]:
```
# for feature selection:
```

In [126]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel

# Assuming 'X' is your set of features and 'y' is the categorical target variable
rf_clf = RandomForestClassifier(n_estimators=100)
rf_clf.fit(data_encoded_log, y)

# Get feature importances
importances = rf_clf.feature_importances_

# Create a model selector object
model_selector = SelectFromModel(rf_clf, prefit=True, threshold='mean')  # or use a cu

# Select features based on importance
X_important = model_selector.transform(data_encoded_log)

# Now X_important contains the most important features as determined by the Random For
```

```
C:\Users\anjal\anaconda3\Lib\site-packages\sklearn\base.py:432: UserWarning: X has fea
ture names, but SelectFromModel was fitted without feature names
  warnings.warn(
```

In [127]:
```
X_important.shape
```

Out[127]:
```
(175341, 11)
```

In [128]:
```
X_important
```

Out[128]:
```
array([[6.93147181e-01, 1.77396316e-05, 6.87247458e-01, ...,
        0.00000000e+00, 5.10825624e-01, 0.00000000e+00],
       [6.93147181e-01, 5.44519565e-05, 2.17638229e-01, ...,
        7.87808779e-02, 5.10825624e-01, 0.00000000e+00],
       [6.93147181e-01, 2.59151819e-05, 2.17638229e-01, ...,
        7.87808779e-02, 5.10825624e-01, 0.00000000e+00],
       ...,
       [8.00427077e-02, 6.63311647e-06, 6.91184471e-01, ...,
        1.65792255e-01, 4.41832752e-01, 6.93147181e-01],
       [8.00427077e-02, 6.63311647e-06, 6.91184471e-01, ...,
        3.88935806e-01, 4.41832752e-01, 6.93147181e-01],
```

```
       [8.00427077e-02, 6.63311647e-06, 6.91184471e-01, ...,
        3.88935806e-01, 4.41832752e-01, 6.93147181e-01]])
```

In [129]:
```
# Assuming 'model_selector' is your SelectFromModel object and 'X' is the original Dat
selected_features = data_encoded_log.columns[model_selector.get_support()]

print(selected_features)
```

```
Index(['service', 'sbytes', 'sttl', 'smean', 'ct_srv_src', 'ct_state_ttl',
       'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'ct_srv_dst', 'attack_cat',
       'label'],
      dtype='object')
```

In [55]:
```
# difference between feature selection and feature extraction
```

In [130]:
```
final_df = data_encoded_log[selected_features]

# Show the first few rows of the new DataFrame to verify it looks correct
print(final_df)
```

```
         service    sbytes      sttl     smean  ct_srv_src  ct_state_ttl  \
0       0.693147  0.000018  0.687247  0.010111    0.000000      0.000000
1       0.693147  0.000054  0.217638  0.016129    0.517257      0.154151
2       0.693147  0.000026  0.217638  0.012121    0.092373      0.154151
3       0.154151  0.000046  0.217638  0.016129    0.000000      0.154151
4       0.693147  0.000039  0.691184  0.016796    0.517257      0.154151
...          ...       ...       ...       ...         ...           ...
175336  0.080043  0.000007  0.691184  0.019457    0.315517      0.287682
175337  0.693147  0.000046  0.691184  0.022774    0.000000      0.154151
175338  0.080043  0.000007  0.691184  0.019457    0.163325      0.287682
175339  0.080043  0.000007  0.691184  0.019457    0.383725      0.287682
175340  0.080043  0.000007  0.691184  0.019457    0.383725      0.287682

        ct_dst_sport_ltm  ct_dst_src_ltm  ct_srv_dst  attack_cat     label
0               0.000000        0.000000    0.000000    0.510826  0.000000
1               0.000000        0.015504    0.078781    0.510826  0.000000
2               0.000000        0.030772    0.078781    0.510826  0.000000
3               0.000000        0.030772    0.000000    0.510826  0.000000
4               0.000000        0.475846    0.484246    0.510826  0.000000
...                  ...             ...         ...         ...       ...
175336          0.236389        0.307025    0.319943    0.441833  0.693147
175337          0.000000        0.015504    0.000000    0.635989  0.693147
175338          0.043485        0.171850    0.165792    0.441833  0.693147
175339          0.253781        0.373716    0.388936    0.441833  0.693147
175340          0.287682        0.373716    0.388936    0.441833  0.693147

[175341 rows x 11 columns]
```

In [56]:
```
y.value_counts()
```

Out[56]:
```
attack_cat
Normal            56000
Generic           40000
Exploits          33393
Fuzzers           18184
DoS               12264
Reconnaissance    10491
Analysis           2000
Backdoor           1746
Shellcode          1133
```

```
Worms               130
Name: count, dtype: int64
```

In [53]:
```
y.shape
```

Out[53]:
```
(175341,)
```

In [ ]:
```

```

In [ ]:
```
# checking the distribution of the data, before balancing the data:
```

In [131]:
```
print("Original Distribution")
print(y.value_counts(normalize=True))  # Normalized to show proportions

#print("\nSampled Distribution")
#print(final_df['target'].val|ue_counts(normalize=True))  # Should be similar to the o
```

```
Original Distribution
attack_cat
Normal            0.319378
Generic           0.228127
Exploits          0.190446
Fuzzers           0.103706
DoS               0.069944
Reconnaissance    0.059832
Analysis          0.011406
Backdoor          0.009958
Shellcode         0.006462
Worms             0.000741
Name: proportion, dtype: float64
```

In [61]:
```
# the data is imbalanced.
```

In [ ]:
```

```

In [62]:
```
!pip install -U imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\anjal\anaconda3\lib\site-p
ackages (0.12.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\anjal\anaconda3\lib\site-pack
ages (from imbalanced-learn) (1.26.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\anjal\anaconda3\lib\site-packa
ges (from imbalanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\anjal\anaconda3\lib\sit
e-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\anjal\anaconda3\lib\site-pack
ages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\anjal\anaconda3\lib\si
te-packages (from imbalanced-learn) (2.2.0)
```

In [ ]:
```
# balancing the dataset after feature selection. Applying SMOTE on the training datase
```

In [ ]:
```
# dividing the 'final_df' dataset into training and testing.
```

In [ ]:
```
# appling: Stratified sampling ----> dividing the data into training and testing---->
```

In [102]:
```
# applying modelling:
```

In [133]:
```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Assuming 'final_df' is your features and 'y' is your target.

# Step 1: Stratified sampling to keep 20% of the data.
X_sample__, X_discard__, y_sample__, y_discard__ = train_test_split(
    final_df, y,
    test_size=0.5,   # Discarding 50% of the data
    stratify=y,
    random_state=42
)

# Step 2: Dividing sampled data into training and test sets.
X_train1, X_test1, y_train1, y_test1 = train_test_split(
    X_sample__, y_sample__,
    test_size=0.3,
    random_state = 42
)

# Step 3: Applying SMOTE to the training dataset to balance it.
smote = SMOTE(random_state=100)
X_balanced1, y_balanced1 = smote.fit_resample(X_train1, y_train1)

# Step 4: Encoding the balanced target variable if necessary.
label_encoder = LabelEncoder()
y_balanced_encoded = label_encoder.fit_transform(y_balanced1)
```

In [135]:
```
y_balanced_encoded
```

Out[135]:
```
array([5, 7, 4, ..., 9, 9, 9])
```

In [136]:
```
results_df = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': y_test_pred})
results_df
```

Out[136]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6

	Actual	Predicted
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [81]:
```
pip install xgboost
```

```
Collecting xgboost
  Obtaining dependency information for xgboost from https://files.pythonhosted.org/pac
kages/24/ec/ad387100fa3cc2b9b81af0829b5ecfe75ec5bb19dd7c19d4fea06fb81802/xgboost-2.0.3
-py3-none-win_amd64.whl.metadata
  Downloading xgboost-2.0.3-py3-none-win_amd64.whl.metadata (2.0 kB)
Requirement already satisfied: numpy in c:\users\anjal\anaconda3\lib\site-packages (fr
om xgboost) (1.26.3)
Requirement already satisfied: scipy in c:\users\anjal\anaconda3\lib\site-packages (fr
om xgboost) (1.11.4)
Downloading xgboost-2.0.3-py3-none-win_amd64.whl (99.8 MB)
   ---------------------------------------- 0.0/99.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/99.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/99.8 MB 487.6 kB/s eta 0:03:25
   ---------------------------------------- 0.3/99.8 MB 3.1 MB/s eta 0:00:33
   ---------------------------------------- 0.8/99.8 MB 5.1 MB/s eta 0:00:20
   ---------------------------------------- 0.8/99.8 MB 5.1 MB/s eta 0:00:20
   ---------------------------------------- 1.2/99.8 MB 4.6 MB/s eta 0:00:22
    --------------------------------------- 1.9/99.8 MB 6.2 MB/s eta 0:00:16
    --------------------------------------- 2.4/99.8 MB 7.1 MB/s eta 0:00:14
   - -------------------------------------- 3.3/99.8 MB 8.1 MB/s eta 0:00:12
   - -------------------------------------- 3.8/99.8 MB 8.5 MB/s eta 0:00:12
   - -------------------------------------- 4.6/99.8 MB 9.4 MB/s eta 0:00:11
   -- ------------------------------------- 5.1/99.8 MB 9.7 MB/s eta 0:00:10
   -- ------------------------------------- 5.9/99.8 MB 10.2 MB/s eta 0:00:10
   -- ------------------------------------- 6.3/99.8 MB 9.8 MB/s eta 0:00:10
   -- ------------------------------------- 7.0/99.8 MB 10.1 MB/s eta 0:00:10
   --- ------------------------------------ 7.5/99.8 MB 10.2 MB/s eta 0:00:10
   --- ------------------------------------ 8.3/99.8 MB 10.8 MB/s eta 0:00:09
   --- ------------------------------------ 9.5/99.8 MB 11.4 MB/s eta 0:00:08
   ---- ----------------------------------- 10.6/99.8 MB 13.6 MB/s eta 0:00:07
   ---- ----------------------------------- 11.9/99.8 MB 16.4 MB/s eta 0:00:06
   ----- ---------------------------------- 13.3/99.8 MB 17.7 MB/s eta 0:00:05
   ----- ---------------------------------- 14.2/99.8 MB 18.2 MB/s eta 0:00:05
   ------ --------------------------------- 15.6/99.8 MB 19.8 MB/s eta 0:00:05
   ------ --------------------------------- 16.3/99.8 MB 21.1 MB/s eta 0:00:04
   ------ --------------------------------- 17.3/99.8 MB 23.4 MB/s eta 0:00:04
   ------- -------------------------------- 18.7/99.8 MB 25.1 MB/s eta 0:00:04
   ------- -------------------------------- 19.9/99.8 MB 25.2 MB/s eta 0:00:04
   -------- ------------------------------- 21.3/99.8 MB 26.2 MB/s eta 0:00:03
   --------- ------------------------------ 22.6/99.8 MB 25.2 MB/s eta 0:00:04
   --------- ------------------------------ 24.1/99.8 MB 26.2 MB/s eta 0:00:03
   ---------- ----------------------------- 25.4/99.8 MB 28.5 MB/s eta 0:00:03
   ---------- ----------------------------- 26.7/99.8 MB 28.4 MB/s eta 0:00:03
   ----------- ---------------------------- 28.2/99.8 MB 28.5 MB/s eta 0:00:03
   ----------- ---------------------------- 29.6/99.8 MB 29.7 MB/s eta 0:00:03
   ------------ --------------------------- 31.1/99.8 MB 29.8 MB/s eta 0:00:03
   ------------- -------------------------- 32.5/99.8 MB 29.7 MB/s eta 0:00:03
   ------------- -------------------------- 33.7/99.8 MB 28.4 MB/s eta 0:00:03
   -------------- ------------------------- 35.2/99.8 MB 31.2 MB/s eta 0:00:03
   -------------- ------------------------- 36.7/99.8 MB 29.7 MB/s eta 0:00:03
   --------------- ------------------------ 38.2/99.8 MB 29.8 MB/s eta 0:00:03
```

```
   -------------- ------------------------ 39.6/99.8 MB 29.8 MB/s eta 0:00:03
   --------------- ----------------------- 40.7/99.8 MB 28.5 MB/s eta 0:00:03
   --------------- ----------------------- 42.3/99.8 MB 29.7 MB/s eta 0:00:02
   ---------------- ---------------------- 43.5/99.8 MB 29.7 MB/s eta 0:00:02
   ---------------- ---------------------- 44.5/99.8 MB 28.4 MB/s eta 0:00:02
   ----------------- --------------------- 45.8/99.8 MB 27.3 MB/s eta 0:00:02
   ----------------- --------------------- 47.3/99.8 MB 27.3 MB/s eta 0:00:02
   ------------------ -------------------- 48.7/99.8 MB 28.5 MB/s eta 0:00:02
   ------------------- ------------------- 50.0/99.8 MB 27.3 MB/s eta 0:00:02
   ------------------- ------------------- 51.3/99.8 MB 28.5 MB/s eta 0:00:02
   -------------------- ------------------ 52.1/99.8 MB 27.3 MB/s eta 0:00:02
   -------------------- ------------------ 52.9/99.8 MB 26.2 MB/s eta 0:00:02
   -------------------- ------------------ 54.3/99.8 MB 28.5 MB/s eta 0:00:02
   --------------------- ----------------- 55.6/99.8 MB 26.2 MB/s eta 0:00:02
   --------------------- ----------------- 57.0/99.8 MB 26.2 MB/s eta 0:00:02
   ---------------------- ---------------- 58.1/99.8 MB 25.2 MB/s eta 0:00:02
   ---------------------- ---------------- 59.5/99.8 MB 25.2 MB/s eta 0:00:02
   ----------------------- --------------- 60.7/99.8 MB 26.2 MB/s eta 0:00:02
   ----------------------- --------------- 62.0/99.8 MB 25.2 MB/s eta 0:00:02
   ------------------------ -------------- 63.4/99.8 MB 28.5 MB/s eta 0:00:02
   ------------------------ -------------- 64.6/99.8 MB 27.3 MB/s eta 0:00:02
   ------------------------- ------------- 65.9/99.8 MB 27.3 MB/s eta 0:00:02
   ------------------------- ------------- 67.2/99.8 MB 28.5 MB/s eta 0:00:02
   -------------------------- ------------ 68.4/99.8 MB 27.3 MB/s eta 0:00:02
   -------------------------- ------------ 69.7/99.8 MB 27.3 MB/s eta 0:00:02
   --------------------------- ----------- 71.1/99.8 MB 27.3 MB/s eta 0:00:02
   --------------------------- ----------- 72.4/99.8 MB 27.3 MB/s eta 0:00:02
   ---------------------------- ---------- 73.8/99.8 MB 28.5 MB/s eta 0:00:01
   ---------------------------- ---------- 75.0/99.8 MB 27.3 MB/s eta 0:00:01
   ----------------------------- --------- 76.3/99.8 MB 27.3 MB/s eta 0:00:01
   ----------------------------- --------- 77.7/99.8 MB 28.4 MB/s eta 0:00:01
   ----------------------------- --------- 79.0/99.8 MB 28.5 MB/s eta 0:00:01
   ------------------------------ -------- 80.3/99.8 MB 27.3 MB/s eta 0:00:01
   ------------------------------ -------- 81.6/99.8 MB 27.3 MB/s eta 0:00:01
   ------------------------------- ------- 82.9/99.8 MB 28.5 MB/s eta 0:00:01
   ------------------------------- ------- 84.2/99.8 MB 28.5 MB/s eta 0:00:01
   -------------------------------- ------ 85.5/99.8 MB 26.2 MB/s eta 0:00:01
   -------------------------------- ------ 87.0/99.8 MB 27.3 MB/s eta 0:00:01
   --------------------------------- ----- 88.3/99.8 MB 28.4 MB/s eta 0:00:01
   --------------------------------- ----- 89.1/99.8 MB 27.3 MB/s eta 0:00:01
   ---------------------------------- ---- 90.4/99.8 MB 26.2 MB/s eta 0:00:01
   ---------------------------------- ---- 92.0/99.8 MB 28.5 MB/s eta 0:00:01
   ----------------------------------- --- 93.5/99.8 MB 28.4 MB/s eta 0:00:01
   ----------------------------------- --- 94.8/99.8 MB 27.3 MB/s eta 0:00:01
   ------------------------------------ -- 96.3/99.8 MB 28.5 MB/s eta 0:00:01
   ------------------------------------ -- 97.6/99.8 MB 28.5 MB/s eta 0:00:01
   ------------------------------------- - 99.1/99.8 MB 29.7 MB/s eta 0:00:01
   ------------------------------------- - 99.7/99.8 MB 29.7 MB/s eta 0:00:01
   ------------------------------------- - 99.7/99.8 MB 29.7 MB/s eta 0:00:01
   ------------------------------------- - 99.7/99.8 MB 29.7 MB/s eta 0:00:01
   ------------------------------------- - 99.7/99.8 MB 29.7 MB/s eta 0:00:01
   ---------------------------------------- 99.8/99.8 MB 17.7 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-2.0.3
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

In [ ]:
```
# applying the models inidividually:
```

In [137]:
```
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
```

In [138]:
```
from sklearn.preprocessing import LabelEncoder

# Create the LabelEncoder instance
label_encoder = LabelEncoder()
label_encoder.fit(y_test1)
```

Out[138]:
```
▾ LabelEncoder

LabelEncoder()
```

In [139]:
```
# KNN:
knn= KNeighborsClassifier(n_neighbors=5)

knn.fit(X_balanced1, y_balanced_encoded)

y_train_pred3 = knn.predict(X_balanced1)
train_accuracy3 = accuracy_score(y_balanced_encoded, y_train_pred3)
print(f"Training Accuracy: {train_accuracy3}")

y_test_pred3 = knn.predict(X_test1)
test_accuracy3 = accuracy_score(y_test_encoded, y_test_pred3)
print(f"Testing Accuracy: {test_accuracy3}")

# Generate the classification report
report = classification_report(y_test_encoded, y_test_pred3, target_names=class_names)

print(report)

result3 = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': y_test_pred3})
result3
```

```
Training Accuracy: 0.9955895912304067
Testing Accuracy: 0.9965020341431885
                precision    recall  f1-score   support

      Analysis       0.84      0.94      0.89       300
      Backdoor       1.00      1.00      1.00       244
           DoS       1.00      1.00      1.00      1846
      Exploits       1.00      1.00      1.00      4988
       Fuzzers       1.00      1.00      1.00      2710
       Generic       1.00      0.99      0.99      5948
        Normal       1.00      1.00      1.00      8478
Reconnaissance       1.00      1.00      1.00      1583
     Shellcode       1.00      0.99      1.00       187
         Worms       1.00      1.00      1.00        17

      accuracy                           1.00     26301
     macro avg       0.98      0.99      0.99     26301
  weighted avg       1.00      1.00      1.00     26301
```

Out[139]:

	Actual	Predicted
0	6	6

	Actual	Predicted
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [155]:
```
# knn:
from sklearn.metrics import confusion_matrix
conf_matrix3 = confusion_matrix(y_test_encoded, y_test_pred3)
conf_matrix3
```

Out[155]:
```
array([[ 282,    0,    0,    0,    0,   18,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1841,    4,    1,    0,    0,    0,    0,    0],
       [   0,    0,    5, 4977,    6,    0,    0,    0,    0,    0],
       [   0,    0,    0,    1, 2708,    1,    0,    0,    0,    0],
       [  53,    0,    0,    1,    1, 5893,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1583,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    1,  186,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [94]:
```
logr = LogisticRegression()

logr.fit(X_balanced1, y_balanced_encoded)

y_train_pred2 = logr.predict(X_balanced1)
train_accuracy2 = accuracy_score(y_balanced_encoded, y_train_pred2)
print(f"Training Accuracy: {train_accuracy2}")

# y_pred2 = logr.predict(X_test1)

y_test_pred2 = logr.predict(X_test1)
test_accuracy2 = accuracy_score(y_test_encoded, y_test_pred2)
print(f"Testing Accuracy: {test_accuracy2}")

# Generate the classification report
report = classification_report(y_test_encoded, y_test_pred2, target_names=class_names)

print(report)

result2 = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': y_test_pred2})
result2
```

```
Training Accuracy: 0.9660331933203565
Testing Accuracy: 0.971750123569446
```

```
                precision    recall  f1-score   support

      Analysis       0.41      0.98      0.58       300
      Backdoor       1.00      1.00      1.00       244
           DoS       1.00      1.00      1.00      1846
      Exploits       1.00      0.95      0.97      4988
       Fuzzers       0.91      0.88      0.89      2710
       Generic       1.00      0.98      0.99      5948
        Normal       1.00      1.00      1.00      8478
Reconnaissance       1.00      0.96      0.98      1583
     Shellcode       0.89      0.98      0.94       187
         Worms       0.67      0.94      0.78        17

      accuracy                           0.97     26301
     macro avg       0.89      0.97      0.91     26301
  weighted avg       0.98      0.97      0.97     26301
```

```
C:\Users\anjal\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[94]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [144]:

```
# Logistic regression:
from sklearn.metrics import confusion_matrix
conf_matrix2 = confusion_matrix(y_test_encoded, y_test_pred2)
conf_matrix2
```

Out[144]:

```
array([[ 294,    0,    0,    0,    6,    0,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1843,    3,    0,    0,    0,    0,    0,    0],
       [  34,    0,    0, 4760,  191,    3,    0,    0,    0,    0],
       [ 279,    0,    0,   17, 2390,   24,    0,    0,    0,    0],
       [  70,    0,    0,    0,   48, 5830,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [  35,    0,    0,    0,    0,    0,    0, 1519,   21,    8],
```

```
       [   0,    0,    0,    0,    0,    0,    0,    3,  184,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    1,   16]],
      dtype=int64)
```

In [ ]:
```
# applying all the models individually:
from sklearn import metrics

rf_bin = RandomForestClassifier(random_state=123)
rf_bin.fit(X_balanced1, y_balanced_encoded)

y_test_pred1 = rf_bin.predict(X_test1)

print("Accuracy - ",accuracy_score(y_test_encoded,y_test_pred1)*100)

cls_report = classification_report(y_true=y_test_encoded, y_pred=y_test_pred, target_n
print(cls_report)

result1 = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': y_test_pred1})
result1
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [64]:
```
import pandas as pd

# Assuming y_train_pred and y_test_pred are the predictions for training and testing s
# Ensure both arrays have the same length
min_length = min(len(y_train_pred), len(y_test_pred))
y_train_pred = y_train_pred[:min_length]
y_test_pred = y_test_pred[:min_length]

# Create a DataFrame to compare predictions
comparison_df = pd.DataFrame({
    'y_train_pred': y_train_pred,
    'y_test_pred': y_test_pred
})

# Display the DataFrame
print(comparison_df)
```

```
       y_train_pred  y_test_pred
0                 5            6
```

```
1                7           3
2                4           3
3                6           6
4                6           6
...            ...         ...
26296            6           7
26297            3           6
26298            6           3
26299            6           6
26300            6           3

[26301 rows x 2 columns]
```

In [ ]:

In [ ]:

Comparing model performances

In [ ]:

In [ ]:

performing without scaling, on the data: data_encoded

In [95]:

```
data_encoded += 1

# Apply log transformation
data_encoded_log_ = np.log1p(data_encoded)


# Checking for infinite values
if np.isinf(data_encoded_log_).any().any():
    print("Infinite values detected")


data_encoded_log_.replace(-np.inf, np.nan, inplace=True)
```

In [96]:

```
data_encoded_log_
```

Out[96]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	
0	1.138307	4.753590	2.708050	1.609438	2.197225	1.945910	5.564520	5.164786	4.344941	5.541
1	1.294700	4.753590	2.708050	1.609438	2.833213	3.713572	6.602588	10.645830	4.400276	4.174
2	1.531072	4.753590	2.708050	1.609438	2.397895	2.944439	5.905362	9.487138	2.843173	4.174
3	1.543649	4.753590	1.609438	1.609438	2.708050	2.708050	6.447306	6.650279	2.814037	4.174
4	1.238216	4.753590	2.708050	1.609438	2.564949	2.197225	6.285998	5.602119	3.593849	5.549
...	...	...	...	...	...	...	...	...	...	
175336	1.098615	4.804021	1.386294	1.791759	1.609438	1.098612	4.762174	1.098612	11.618313	5.549

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	
175337	1.254408	4.753590	2.708050	1.609438	2.564949	2.397895	6.434547	5.877736	3.600394	5.549
175338	1.098615	4.804021	1.386294	1.791759	1.609438	1.098612	4.762174	1.098612	11.618313	5.549
175339	1.098615	4.804021	1.386294	1.791759	1.609438	1.098612	4.762174	1.098612	11.618313	5.549
175340	1.098615	4.804021	1.386294	1.791759	1.609438	1.098612	4.762174	1.098612	11.618313	5.549

175341 rows × 44 columns

In [98]:

```
import numpy as np
from scipy import stats

# Assuming `data_encoded` is your DataFrame with all numeric data
# Calculate Z-scores
z_scores = np.abs(stats.zscore(data_encoded_log_))

# Define a threshold
threshold = 3

# Get boolean array where true represents the presence of an outlier
outliers_ = (z_scores > threshold)

# Optional: Get the actual data points that are considered outliers
outlier_points_ = data_encoded[(z_scores > threshold).any(axis=1)]
```

In [99]:

```
outlier_points_
```

Out[99]:

	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	...	ct_dst_spor
3	3.681642	115	4	4	14	14	630	772	15.677108	64	...	
11	4.093085	115	10	4	64	30	56331	2214	44.520967	64	...	
22	2.964656	115	4	2	16	14	692	952	27.915974	64	...	
30	2.725996	115	6	4	12	18	824	12222	36.435451	64	...	
32	2.337456	115	6	4	12	10	802	2262	52.376940	64	...	
...	...	...	...	...	...	...	...	...	...	...	...	
175112	6.079689	115	10	4	160	44	182787	2626	50.778228	64	...	
175182	4.724847	115	10	4	62	32	51669	2064	34.662383	64	...	
175248	4.013791	115	10	4	228	52	272072	3086	138.558356	64	...	
175267	3.914309	115	10	4	76	32	69999	2134	55.805315	64	...	
175277	5.719110	115	14	4	68	342	3088	426485	110.897021	64	...	

45801 rows × 44 columns

In [114]:

```
# Example: Replacing outliers with the median

median_ = data_encoded_log_.median()
z_scores_ = np.abs(stats.zscore(data_encoded_log_))
outlier_positions_ = z_scores_ > 3
```

```
data_encoded_log_[outlier_positions_] = np.nan
data_encoded_log_.fillna(median_, inplace=True)
```

In [ ]:
```
# performing embedded method:
```

In [116]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel

# Assuming 'X' is your set of features and 'y' is the categorical target variable
rf_clf_ = RandomForestClassifier(n_estimators=100)
rf_clf_.fit(data_encoded_log_, y)

# Get feature importances
importances_ = rf_clf_.feature_importances_

# Create a model selector object
model_selector = SelectFromModel(rf_clf_, prefit=True, threshold='mean')

# Select features based on importance
X_important_ = model_selector.transform(data_encoded_log_)
```

```
C:\Users\anjal\anaconda3\Lib\site-packages\sklearn\base.py:432: UserWarning: X has fea
ture names, but SelectFromModel was fitted without feature names
  warnings.warn(
```

In [117]:
```
X_important_.shape
```

Out[117]:
```
(175341, 11)
```

In [118]:
```
y
```

Out[118]:
```
0            Normal
1            Normal
2            Normal
3            Normal
4            Normal
            ...
175336      Generic
175337    Shellcode
175338      Generic
175339      Generic
175340      Generic
Name: attack_cat, Length: 175341, dtype: object
```

In [119]:
```
final_df_ = data_encoded_log_[selected_features]

# Show the first few rows of the new DataFrame to verify it looks correct
final_df_
```

Out[119]:

	service	sbytes	sttl	smean	ct_srv_src	ct_state_ttl	ct_dst_sport_ltm	ct_dst_src_ltm
0	2.708050	5.564520	5.541264	3.828641	1.386294	1.098612	1.386294	1.386294
1	2.708050	6.602588	4.174387	4.007333	3.828641	1.386294	1.386294	1.609438
2	2.708050	5.905362	4.174387	3.891820	2.302585	1.386294	1.386294	1.791759
3	1.609438	6.447306	4.174387	4.007333	1.386294	1.386294	1.386294	1.791759
4	2.708050	6.285998	5.549076	4.025352	3.828641	1.386294	1.386294	3.761200

	service	sbytes	sttl	smean	ct_srv_src	ct_state_ttl	ct_dst_sport_ltm	ct_dst_src_ltm
...	...	...	...	...	...	...	...	...
175336	1.386294	4.762174	5.549076	4.094345	3.295837	1.609438	2.772589	3.295837
175337	2.708050	6.434547	5.549076	4.174387	1.386294	1.386294	1.386294	1.609438
175338	1.386294	4.762174	5.549076	4.094345	2.708050	1.609438	1.791759	2.772589
175339	1.386294	4.762174	5.549076	4.094345	3.496508	1.609438	2.833213	3.496508
175340	1.386294	4.762174	5.549076	4.094345	3.496508	1.609438	2.944439	3.496508

175341 rows × 11 columns

In [109]:

```
# Step 1: Stratified sampling to keep 20% of the data.
X_sample1, X_discard1, y_sample1, y_discard1 = train_test_split(
    final_df_, y,
    test_size = 0.5,   # Discarding 50% of the data
    stratify = y,
    random_state = 42
)

# Step 2: Dividing sampled data into training and test sets.
X_train11, X_test11, y_train11, y_test11 = train_test_split(
    X_sample1, y_sample1,
    test_size = 0.3,
    random_state = 42
)

# Step 3: Applying SMOTE to the training dataset to balance it.
smote = SMOTE(random_state = 100)
X_balanced11, y_balanced11 = smote.fit_resample(X_train11, y_train11)

# Step 4: Encoding the balanced target variable if necessary.
label_encoder_ = LabelEncoder()
y_balanced_encoded_ = label_encoder_.fit_transform(y_balanced1)
```

In [156]:

```
X_sample1
```

Out[156]:

	service	sbytes	sttl	smean	ct_srv_src	ct_state_ttl	ct_dst_sport_ltm	ct_dst_src_ltm
15018	2.708050	7.711101	3.526361	4.234107	2.484907	1.098612	1.386294	1.386294
118015	2.708050	6.415097	5.549076	4.158883	2.079442	1.386294	1.386294	1.386294
34682	1.945910	7.430707	3.526361	4.812184	1.386294	1.098612	1.386294	1.386294
62000	2.708050	5.141664	5.549076	4.465908	1.609438	1.609438	1.609438	1.609438
48397	2.708050	6.285998	5.549076	4.025352	1.945910	1.386294	1.386294	1.945910
...	...	...	...	...	...	...	...	...
146539	1.386294	4.762174	5.549076	4.094345	3.218876	1.609438	2.995732	3.218876
71376	2.708050	5.313206	5.549076	4.634729	2.302585	1.609438	1.791759	2.302585
145740	1.386294	4.762174	5.549076	4.094345	3.850148	1.609438	2.833213	3.850148
100153	2.708050	7.090910	4.174387	4.043051	1.386294	1.386294	1.386294	1.386294
39221	1.386294	4.890349	3.526361	4.219508	1.609438	1.098612	1.386294	1.386294

87670 rows × 11 columns

In [157]:
```
X_test11
```

Out[157]:

	service	sbytes	sttl	smean	ct_srv_src	ct_state_ttl	ct_dst_sport_ltm	ct_dst_src_ltm
29106	2.708050	7.629004	3.526361	4.564348	2.079442	1.098612	1.386294	1.945910
63790	2.397895	10.455705	4.174387	6.630683	1.386294	1.386294	1.386294	1.386294
92451	2.708050	6.415097	5.549076	4.158883	1.386294	1.386294	1.386294	1.386294
86006	2.708050	7.472501	5.549076	4.330733	2.639057	1.609438	1.386294	2.564949
36391	1.945910	7.430707	3.526361	4.812184	1.386294	1.098612	1.386294	1.386294
...	...	...	...	...	...	...	...	...
77952	2.708050	5.141664	5.549076	4.465908	1.791759	1.609438	1.386294	1.386294
33698	1.386294	4.890349	3.526361	4.219508	1.791759	1.098612	1.386294	1.386294
64589	2.708050	8.015327	5.549076	4.709530	1.386294	1.609438	1.386294	1.386294
37076	1.791759	6.056784	3.526361	4.025352	1.609438	1.098612	1.386294	1.609438
115083	2.708050	5.313206	5.549076	4.634729	1.609438	1.609438	1.609438	1.609438

26301 rows × 11 columns

performing Modelling:

In [120]:
```
dt = DecisionTreeClassifier(random_state=123)

dt.fit(X_balanced11, y_balanced_encoded_)

y_train_pred_4 = dt.predict(X_balanced11)
train_accuracy_4 = accuracy_score(y_balanced_encoded_, y_train_pred_4)
print(f"Training Accuracy: {train_accuracy_4}")

y_test_pred_4 = dt.predict(X_test11)
y_test_encoded_ = label_encoder.transform(y_test11)
test_accuracy_4 = accuracy_score(y_test_encoded_, y_test_pred_4)
print(f"Testing Accuracy: {test_accuracy_4}")


from sklearn.metrics import classification_report
class_names = label_encoder.classes_

# Generate the classification report
report_4 = classification_report(y_test_encoded_, y_test_pred_4, target_names=class_na
print(report)


result_4 = pd.DataFrame({'Actual': y_test_encoded_, 'Predicted': y_test_pred_4})
result_4
```

```
Training Accuracy: 0.9993699416043438
Testing Accuracy: 0.9980228888635413
              precision    recall  f1-score   support
```

```
                precision    recall  f1-score   support

      Analysis       0.41      0.98      0.58       300
      Backdoor       1.00      1.00      1.00       244
           DoS       1.00      1.00      1.00      1846
      Exploits       1.00      0.95      0.97      4988
       Fuzzers       0.91      0.88      0.89      2710
       Generic       1.00      0.98      0.99      5948
        Normal       1.00      1.00      1.00      8478
Reconnaissance       1.00      0.96      0.98      1583
     Shellcode       0.89      0.98      0.94       187
         Worms       0.67      0.94      0.78        17

      accuracy                           0.97     26301
     macro avg       0.89      0.97      0.91     26301
  weighted avg       0.98      0.97      0.97     26301
```

Out[120]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [153]:

```
# decision tree:
conf_matrix4 = confusion_matrix(y_test_encoded, y_test_pred_4)
conf_matrix4
```

Out[153]:

```
array([[ 298,    0,    0,    0,    0,    2,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1846,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 4988,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 2710,    0,    0,    0,    0,    0],
       [  50,    0,    0,    0,    0, 5898,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1583,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  187,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [ ]:

In [158]:

```
gb_ = GradientBoostingClassifier(random_state=42)

gb_.fit(X_balanced11, y_balanced_encoded_)
```

```
y_train_pred_5 = gb_.predict(X_balanced11)
train_accuracy_5 = accuracy_score(y_balanced_encoded_, y_train_pred_5)
print(f"Training Accuracy: {train_accuracy_5}")

y_test_pred_5 = gb_.predict(X_test11)
y_test_encoded_ = label_encoder.transform(y_test11)
test_accuracy_5 = accuracy_score(y_test_encoded_, y_test_pred_5)
print(f"Testing Accuracy: {test_accuracy_5}")


from sklearn.metrics import classification_report
class_names = label_encoder.classes_

# Generate the classification report
report_5 = classification_report(y_test_encoded_, y_test_pred_5, target_names=class_na
print(report)


result_5 = pd.DataFrame({'Actual': y_test_encoded_, 'Predicted': y_test_pred_5})
result_5
```

```
Training Accuracy: 0.9991804118430488
Testing Accuracy: 0.9978327820234972
                precision    recall  f1-score   support

      Analysis       0.84      0.94      0.89       300
      Backdoor       1.00      1.00      1.00       244
           DoS       1.00      1.00      1.00      1846
      Exploits       1.00      1.00      1.00      4988
       Fuzzers       1.00      1.00      1.00      2710
       Generic       1.00      0.99      0.99      5948
        Normal       1.00      1.00      1.00      8478
Reconnaissance       1.00      1.00      1.00      1583
     Shellcode       1.00      0.99      1.00       187
         Worms       1.00      1.00      1.00        17

      accuracy                           1.00     26301
     macro avg       0.98      0.99      0.99     26301
  weighted avg       1.00      1.00      1.00     26301
```

Out[158]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [159]:
```
# gradient boosting classifier:

from sklearn.metrics import confusion_matrix

conf_matrix5 = confusion_matrix(y_test_encoded, y_test_pred_5)

conf_matrix5
```

Out[159]:
```
array([[ 297,    0,    0,    0,    0,    3,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1846,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 4988,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 2710,    0,    0,    0,    0,    0],
       [  54,    0,    0,    0,    0, 5894,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1583,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  187,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [160]:
```
conf_matrix5
```

Out[160]:
```
array([[ 297,    0,    0,    0,    0,    3,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1846,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 4988,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 2710,    0,    0,    0,    0,    0],
       [  54,    0,    0,    0,    0, 5894,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1583,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  187,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [151]:
```
xgb_ = xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='mloglo

xgb_.fit(X_balanced11, y_balanced_encoded_)

y_train_pred_6 = xgb_.predict(X_balanced11)
train_accuracy_6 = accuracy_score(y_balanced_encoded_, y_train_pred_6)
print(f"Training Accuracy: {train_accuracy_6}")

y_test_pred_6 = gb_.predict(X_test11)
y_test_encoded_ = label_encoder.transform(y_test11)
test_accuracy_6 = accuracy_score(y_test_encoded_, y_test_pred_6)
print(f"Testing Accuracy: {test_accuracy_6}")

cls_report = classification_report(y_true=y_test_encoded, y_pred=y_test_pred_6, target
print(cls_report)

result6 = pd.DataFrame({'Actual': y_test_encoded, 'Predicted': y_test_pred_6})
result6
```

```
Training Accuracy: 0.9993340846224772
Testing Accuracy: 0.9978327820234972
              precision    recall  f1-score   support

    Analysis       0.85      0.99      0.91       300
    Backdoor       1.00      1.00      1.00       244
         DoS       1.00      1.00      1.00      1846
    Exploits       1.00      1.00      1.00      4988
```

```
              Fuzzers       1.00      1.00      1.00      2710
              Generic       1.00      0.99      1.00      5948
               Normal       1.00      1.00      1.00      8478
       Reconnaissance       1.00      1.00      1.00      1583
            Shellcode       1.00      1.00      1.00       187
                Worms       1.00      1.00      1.00        17

             accuracy                           1.00     26301
            macro avg       0.98      1.00      0.99     26301
         weighted avg       1.00      1.00      1.00     26301
```

Out[151]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [152]:

```
# xgboost:

conf_matrix6 = confusion_matrix(y_test_encoded, y_test_pred_6)
conf_matrix6
```

Out[152]:

```
array([[ 297,    0,    0,    0,    0,    3,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1846,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 4988,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 2710,    0,    0,    0,    0,    0],
       [  54,    0,    0,    0,    0, 5894,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1583,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  187,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [ ]:

In [ ]:

In [147]:

```
clf_ = RandomForestClassifier(random_state=100)

clf_.fit(X_balanced11, y_balanced_encoded_)
```

```
y_train_pred_1 = clf_.predict(X_balanced11)
train_accuracy_1 = accuracy_score(y_balanced_encoded_, y_train_pred_1)
print(f"Training Accuracy: {train_accuracy_1}")

y_test_pred_1 = clf_.predict(X_test11)
y_test_encoded_ = label_encoder.transform(y_test11)
test_accuracy_1 = accuracy_score(y_test_encoded_, y_test_pred_1)
print(f"Testing Accuracy: {test_accuracy_1}")


from sklearn.metrics import classification_report
class_names = label_encoder.classes_

# Generate the classification report
report_1 = classification_report(y_test_encoded_, y_test_pred_1, target_names=class_na
print(report)


result_1 = pd.DataFrame({'Actual': y_test_encoded_, 'Predicted': y_test_pred_1})
result_1
```

```
Training Accuracy: 0.9993699416043438
Testing Accuracy: 0.998098931599559
                precision    recall  f1-score   support

      Analysis       0.84      0.94      0.89       300
      Backdoor       1.00      1.00      1.00       244
           DoS       1.00      1.00      1.00      1846
      Exploits       1.00      1.00      1.00      4988
       Fuzzers       1.00      1.00      1.00      2710
       Generic       1.00      0.99      0.99      5948
        Normal       1.00      1.00      1.00      8478
Reconnaissance       1.00      1.00      1.00      1583
     Shellcode       1.00      0.99      1.00       187
         Worms       1.00      1.00      1.00        17

      accuracy                           1.00     26301
     macro avg       0.98      0.99      0.99     26301
  weighted avg       1.00      1.00      1.00     26301
```

Out[147]:

	Actual	Predicted
0	6	6
1	3	3
2	3	3
3	6	6
4	6	6
...	...	...
26296	7	7
26297	6	6
26298	3	3
26299	6	6
26300	3	3

26301 rows × 2 columns

In [149]:
```
# Random forest classifier

from sklearn.metrics import confusion_matrix

conf_matrix3 = confusion_matrix(y_test_encoded, y_test_pred_1)

conf_matrix_values
conf_matrix3
```

Out[149]:
```
array([[ 295,    0,    0,    0,    1,    4,    0,    0,    0,    0],
       [   0,  244,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1846,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 4988,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 2710,    0,    0,    0,    0,    0],
       [  41,    0,    0,    0,    3, 5904,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 8478,    0,    0,    0],
       [   1,    0,    0,    0,    0,    0,    0, 1582,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  187,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,   17]],
      dtype=int64)
```

In [ ]:

In [ ]: