ML Project 2

April 20, 2020

It is important we first import important libraries to us with our analysis. I have imported numpy, pandas etc

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas import Series, DataFrame
```

The NSL-KDD dataset version 1 and version 2 were both imported and explored to gain more insight about the data

[2]:
```
# Version-1 importation
version1 = pd.read_csv("nslkdd-version1.csv")
```

[3]:
```
version1.columns
```

[3]:
```
Index(['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'a11',
       'a12', 'a13', 'a14', 'a15', 'a16', 'a17', 'a18', 'a19', 'a20', 'a21',
       'a22', 'a23', 'a24', 'a25', 'a26', 'a27', 'a28', 'a29', 'a30', 'a31',
       'a32', 'a33', 'a34', 'a35', 'a36', 'a37', 'a38', 'a39', 'a40', 'a41',
       'a42'],
      dtype='object')
```

[4]:
```
#first five rows in Version-1
version1.head()
```

[4]:
```
   a1   a2        a3  a4   a5    a6  a7  a8  a9  a10  ...  a33   a34   a35  \
0   0  tcp  ftp_data  SF  491     0   0   0   0    0  ...   25  0.17  0.03
1   0  udp     other  SF  146     0   0   0   0    0  ...    1  0.00  0.60
2   0  tcp   private  S0    0     0   0   0   0    0  ...   26  0.10  0.05
3   0  tcp      http  SF  232  8153   0   0   0    0  ...  255  1.00  0.00
4   0  tcp      http  SF  199   420   0   0   0    0  ...  255  1.00  0.00

    a36   a37   a38   a39   a40   a41      a42
0  0.17  0.00  0.00  0.00  0.05  0.00   normal
1  0.88  0.00  0.00  0.00  0.00  0.00   normal
2  0.00  0.00  1.00  1.00  0.00  0.00  neptune
3  0.03  0.04  0.03  0.01  0.00  0.01   normal
4  0.00  0.00  0.00  0.00  0.00  0.00   normal
```

```
[5 rows x 42 columns]
```

[5]:
```
#Last five rows in Varesion-1
version1.tail()
```

[5]:
```
       a1   a2        a3    a4   a5  a6  a7  a8  a9  a10  ...  a33   a34  \
25187   0  tcp      exec  RSTO    0   0   0   0   0    0  ...    7  0.03
25188   0  tcp  ftp_data    SF  334   0   0   0   0    0  ...   39  1.00
25189   0  tcp   private   REJ    0   0   0   0   0    0  ...   13  0.05
25190   0  tcp      nnsp    S0    0   0   0   0   0    0  ...   20  0.08
25191   0  tcp    finger    S0    0   0   0   0   0    0  ...   49  0.19

        a35   a36   a37  a38  a39  a40  a41          a42
25187  0.06  0.00  0.00  0.0  0.0  1.0  1.0      neptune
25188  0.00  1.00  0.18  0.0  0.0  0.0  0.0  warezclient
25189  0.07  0.00  0.00  0.0  0.0  1.0  1.0      neptune
25190  0.06  0.00  0.00  1.0  1.0  0.0  0.0      neptune
25191  0.03  0.01  0.00  1.0  1.0  0.0  0.0      neptune

[5 rows x 42 columns]
```

[6]:
```
#Version-2 importation
version2 = pd.read_csv("nslkdd-version2.csv")
```

[7]:
```
version2.columns
```

[7]:
```
Index(['a7', 'a8', 'a9', 'a10', 'a11', 'a12', 'a13', 'a14', 'a15', 'a16',
       'a17', 'a18', 'a19', 'a20', 'a21', 'a22', 'a23', 'a24', 'a25', 'a26',
       'a27', 'a28', 'a29', 'a30', 'a31', 'a32', 'a33', 'a34', 'a35', 'a36',
       'a37', 'a38', 'a39', 'a40', 'a41', 'a42'],
      dtype='object')
```

[8]:
```
# first five rows in Version 1
version1.head()
```

[8]:
```
   a1   a2        a3  a4   a5    a6  a7  a8  a9  a10  ...  a33   a34   a35  \
0   0  tcp  ftp_data  SF  491     0   0   0   0    0  ...   25  0.17  0.03
1   0  udp     other  SF  146     0   0   0   0    0  ...    1  0.00  0.60
2   0  tcp   private  S0    0     0   0   0   0    0  ...   26  0.10  0.05
3   0  tcp      http  SF  232  8153   0   0   0    0  ...  255  1.00  0.00
4   0  tcp      http  SF  199   420   0   0   0    0  ...  255  1.00  0.00

    a36   a37   a38   a39   a40   a41      a42
0  0.17  0.00  0.00  0.00  0.05  0.00   normal
1  0.88  0.00  0.00  0.00  0.00  0.00   normal
2  0.00  0.00  1.00  1.00  0.00  0.00  neptune
3  0.03  0.04  0.03  0.01  0.00  0.01   normal
4  0.00  0.00  0.00  0.00  0.00  0.00   normal

[5 rows x 42 columns]
```

[9]:
```python
# Expression to show that all columns can be viewed

pd.set_option('display.max_columns', None)
version1.head()
```

[9]:
```
   a1   a2        a3  a4   a5    a6  a7  a8  a9  a10  a11  a12  a13  a14  a15  \
0   0  tcp  ftp_data  SF  491     0   0   0   0    0    0    0    0    0    0
1   0  udp     other  SF  146     0   0   0   0    0    0    0    0    0    0
2   0  tcp   private  S0    0     0   0   0   0    0    0    0    0    0    0
3   0  tcp      http  SF  232  8153   0   0   0    0    0    1    0    0    0
4   0  tcp      http  SF  199   420   0   0   0    0    0    1    0    0    0

   a16  a17  a18  a19  a20  a21  a22  a23  a24  a25  a26  a27  a28   a29  \
0    0    0    0    0    0    0    0    2    2  0.0  0.0  0.0  0.0  1.00
1    0    0    0    0    0    0    0   13    1  0.0  0.0  0.0  0.0  0.08
2    0    0    0    0    0    0    0  123    6  1.0  1.0  0.0  0.0  0.05
3    0    0    0    0    0    0    0    5    5  0.2  0.2  0.0  0.0  1.00
4    0    0    0    0    0    0    0   30   32  0.0  0.0  0.0  0.0  1.00

    a30   a31  a32  a33   a34   a35   a36   a37   a38   a39   a40   a41  \
0  0.00  0.00  150   25  0.17  0.03  0.17  0.00  0.00  0.00  0.05  0.00
1  0.15  0.00  255    1  0.00  0.60  0.88  0.00  0.00  0.00  0.00  0.00
2  0.07  0.00  255   26  0.10  0.05  0.00  0.00  1.00  1.00  0.00  0.00
3  0.00  0.00   30  255  1.00  0.00  0.03  0.04  0.03  0.01  0.00  0.01
4  0.00  0.09  255  255  1.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00

       a42
0   normal
1   normal
2  neptune
3   normal
4   normal
```

[10]:
```python
# first five rows of Version-2
version2.head()
```

[10]:
```
   a7  a8  a9  a10  a11  a12  a13  a14  a15  a16  a17  a18  a19  a20  a21  \
0   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0
1   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0
2   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0
3   0   0   0    0    0    0    0    1    0    0    0    0    0    0    0
4   0   0   0    0    0    0    0    1    0    0    0    0    0    0    0

   a22  a23  a24  a25  a26  a27  a28   a29   a30   a31  a32  a33   a34   a35  \
0    0    2    2  0.0  0.0  0.0  0.0  1.00  0.00  0.00  150   25  0.17  0.03
1    0   13    1  0.0  0.0  0.0  0.0  0.08  0.15  0.00  255    1  0.00  0.60
2    0  123    6  1.0  1.0  0.0  0.0  0.05  0.07  0.00  255   26  0.10  0.05
3    0    5    5  0.2  0.2  0.0  0.0  1.00  0.00  0.00   30  255  1.00  0.00
```

```
4    0    30    32   0.0   0.0   0.0   0.0   1.00   0.00   0.09   255   255   1.00   0.00

    a36   a37   a38   a39   a40   a41  a42
0  0.17  0.00  0.00  0.00  0.05  0.00    0
1  0.88  0.00  0.00  0.00  0.00  0.00    0
2  0.00  0.00  1.00  1.00  0.00  0.00    1
3  0.03  0.04  0.03  0.01  0.00  0.01    0
4  0.00  0.00  0.00  0.00  0.00  0.00    0
```

[11]:
```python
# Version-1 dimension/shape
version1.shape
```

[11]: (25192, 42)

There are 25192 observations in NSL-KDD version 1 and 42 variables

[12]:
```python
# Version-2 dimension/shape
version2.shape
```

[12]: (25192, 36)

There are 25192 observations in NSL-KDD version 2 and 36 variables

[13]:
```python
# To check if both data sets are equal
version1.equals(version2)
```

[13]: False

[14]:
```python
#Futher information about the data

version1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25192 entries, 0 to 25191
Data columns (total 42 columns):
a1      25192 non-null int64
a2      25192 non-null object
a3      25192 non-null object
a4      25192 non-null object
a5      25192 non-null int64
a6      25192 non-null int64
a7      25192 non-null int64
a8      25192 non-null int64
a9      25192 non-null int64
a10     25192 non-null int64
a11     25192 non-null int64
a12     25192 non-null int64
a13     25192 non-null int64
a14     25192 non-null int64
a15     25192 non-null int64
a16     25192 non-null int64
a17     25192 non-null int64
a18     25192 non-null int64
```

```
a19    25192 non-null int64
a20    25192 non-null int64
a21    25192 non-null int64
a22    25192 non-null int64
a23    25192 non-null int64
a24    25192 non-null int64
a25    25192 non-null float64
a26    25192 non-null float64
a27    25192 non-null float64
a28    25192 non-null float64
a29    25192 non-null float64
a30    25192 non-null float64
a31    25192 non-null float64
a32    25192 non-null int64
a33    25192 non-null int64
a34    25192 non-null float64
a35    25192 non-null float64
a36    25192 non-null float64
a37    25192 non-null float64
a38    25192 non-null float64
a39    25192 non-null float64
a40    25192 non-null float64
a41    25192 non-null float64
a42    25192 non-null object
dtypes: float64(15), int64(23), object(4)
memory usage: 8.1+ MB
```

[15]:
```
#Futher information about the data

version2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25192 entries, 0 to 25191
Data columns (total 36 columns):
a7     25192 non-null int64
a8     25192 non-null int64
a9     25192 non-null int64
a10    25192 non-null int64
a11    25192 non-null int64
a12    25192 non-null int64
a13    25192 non-null int64
a14    25192 non-null int64
a15    25192 non-null int64
a16    25192 non-null int64
a17    25192 non-null int64
a18    25192 non-null int64
a19    25192 non-null int64
```

```
a20    25192 non-null int64
a21    25192 non-null int64
a22    25192 non-null int64
a23    25192 non-null int64
a24    25192 non-null int64
a25    25192 non-null float64
a26    25192 non-null float64
a27    25192 non-null float64
a28    25192 non-null float64
a29    25192 non-null float64
a30    25192 non-null float64
a31    25192 non-null float64
a32    25192 non-null int64
a33    25192 non-null int64
a34    25192 non-null float64
a35    25192 non-null float64
a36    25192 non-null float64
a37    25192 non-null float64
a38    25192 non-null float64
a39    25192 non-null float64
a40    25192 non-null float64
a41    25192 non-null float64
a42    25192 non-null int64
dtypes: float64(15), int64(21)
memory usage: 6.9 MB
```

[16]:
```
# Version-1 Statistical Summary
version1.describe()
```

[16]:
```
                 a1            a5            a6            a7            a8  \
count  25192.000000  2.519200e+04  2.519200e+04  25192.000000  25192.000000
mean     305.054104  2.433063e+04  3.491847e+03      0.000079      0.023738
std     2686.555640  2.410805e+06  8.883072e+04      0.008910      0.260221
min        0.000000  0.000000e+00  0.000000e+00      0.000000      0.000000
25%        0.000000  0.000000e+00  0.000000e+00      0.000000      0.000000
50%        0.000000  4.400000e+01  0.000000e+00      0.000000      0.000000
75%        0.000000  2.790000e+02  5.302500e+02      0.000000      0.000000
max    42862.000000  3.817091e+08  5.151385e+06      1.000000      3.000000

                a9           a10           a11           a12           a13  \
count  25192.00000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.00004      0.198039      0.001191      0.394768      0.227850
std        0.00630      2.154202      0.045418      0.488811     10.417352
min        0.00000      0.000000      0.000000      0.000000      0.000000
25%        0.00000      0.000000      0.000000      0.000000      0.000000
50%        0.00000      0.000000      0.000000      0.000000      0.000000
75%        0.00000      0.000000      0.000000      1.000000      0.000000
max        1.00000     77.000000      4.000000      1.000000    884.000000
```

```
                a14          a15          a16          a17          a18  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.001548      0.001350      0.249841      0.014727      0.000357
std        0.039316      0.048785     11.500842      0.529602      0.018898
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000      0.000000
max        1.000000      2.000000    975.000000     40.000000      1.000000

                a19      a20      a21          a22          a23  \
count  25192.000000  25192.0  25192.0  25192.000000  25192.000000
mean       0.004327      0.0      0.0      0.009130     84.591180
std        0.098524      0.0      0.0      0.095115    114.673451
min        0.000000      0.0      0.0      0.000000      1.000000
25%        0.000000      0.0      0.0      0.000000      2.000000
50%        0.000000      0.0      0.0      0.000000     14.000000
75%        0.000000      0.0      0.0      0.000000    144.000000
max        8.000000      0.0      0.0      1.000000    511.000000

                a24          a25          a26          a27          a28  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean      27.698754      0.286338      0.283762      0.118630      0.120260
std       72.468242      0.447312      0.447599      0.318745      0.322335
min        1.000000      0.000000      0.000000      0.000000      0.000000
25%        2.000000      0.000000      0.000000      0.000000      0.000000
50%        8.000000      0.000000      0.000000      0.000000      0.000000
75%       18.000000      1.000000      1.000000      0.000000      0.000000
max      511.000000      1.000000      1.000000      1.000000      1.000000

                a29          a30          a31          a32          a33  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.660559      0.062363      0.095931    182.532074    115.063036
std        0.439637      0.178550      0.256583     98.993895    110.646850
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.090000      0.000000      0.000000     84.000000     10.000000
50%        1.000000      0.000000      0.000000    255.000000     61.000000
75%        1.000000      0.060000      0.000000    255.000000    255.000000
max        1.000000      1.000000      1.000000    255.000000    255.000000

                a34          a35          a36          a37          a38  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.519791      0.082539      0.147453      0.031844      0.285800
std        0.448944      0.187191      0.308367      0.110575      0.445316
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.050000      0.000000      0.000000      0.000000      0.000000
```

```
50%        0.510000      0.030000      0.000000      0.000000      0.000000
75%        1.000000      0.070000      0.060000      0.020000      1.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

                a39           a40           a41
count  25192.000000  25192.000000  25192.000000
mean       0.279846      0.117800      0.118769
std        0.446075      0.305869      0.317333
min        0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000
75%        1.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000
```

[17]:
```
# a2 value count
version1['a2'].value_counts()
```

[17]:
```
tcp     20526
udp      3011
icmp     1655
Name: a2, dtype: int64
```

[18]:
```
# a4 value count
version1['a4'].value_counts()
```

[18]:
```
SF        14973
S0         7009
REJ        2216
RSTR        497
RSTO        304
S1           88
SH           43
S2           21
RSTOS0       21
S3           15
OTH           5
Name: a4, dtype: int64
```

[19]:
```
# a3 value count
version1['a3'].value_counts()
```

[19]:
```
http         8003
private      4351
domain_u     1820
smtp         1449
ftp_data     1396
             ...
urh_i           4
pm_dump         3
red_i           3
```

```
tim_i           2
http_8001       1
Name: a3, Length: 66, dtype: int64
```

[20]:
```
# a42 value count

version1['a42'].value_counts()
```

[20]:
```
normal             13449
neptune             8282
ipsweep              710
satan                691
portsweep            587
smurf                529
nmap                 301
back                 196
teardrop             188
warezclient          181
pod                   38
guess_passwd          10
warezmaster            7
buffer_overflow        6
imap                   5
rootkit                4
phf                    2
multihop               2
ftp_write              1
land                   1
spy                    1
loadmodule             1
Name: a42, dtype: int64
```

[21]:
```
#Version-2 statistical summary

version2.describe()
```

[21]:
```
                 a7            a8            a9           a10          a11  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.00000
mean       0.000079      0.023738      0.000079      0.023738      0.00004
std        0.008910      0.260221      0.008910      0.260221      0.00630
min        0.000000      0.000000      0.000000      0.000000      0.00000
25%        0.000000      0.000000      0.000000      0.000000      0.00000
50%        0.000000      0.000000      0.000000      0.000000      0.00000
75%        0.000000      0.000000      0.000000      0.000000      0.00000
max        1.000000      3.000000      1.000000      3.000000      1.00000

                a12           a13           a14           a15           a16  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.198039      0.001191      0.394768      0.227850      0.001548
```

```
std         2.154202      0.045418      0.488811     10.417352      0.039316
min         0.000000      0.000000      0.000000      0.000000      0.000000
25%         0.000000      0.000000      0.000000      0.000000      0.000000
50%         0.000000      0.000000      0.000000      0.000000      0.000000
75%         0.000000      0.000000      1.000000      0.000000      0.000000
max        77.000000      4.000000      1.000000    884.000000      1.000000

                 a17           a18           a19           a20           a21  \
count   25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean        0.001350      0.249841      0.014727      0.000357      0.004327
std         0.048785     11.500842      0.529602      0.018898      0.098524
min         0.000000      0.000000      0.000000      0.000000      0.000000
25%         0.000000      0.000000      0.000000      0.000000      0.000000
50%         0.000000      0.000000      0.000000      0.000000      0.000000
75%         0.000000      0.000000      0.000000      0.000000      0.000000
max         2.000000    975.000000     40.000000      1.000000      8.000000

                 a22           a23           a24           a25           a26  \
count   25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean        0.009130     84.591180     27.698754      0.286338      0.283762
std         0.095115    114.673451     72.468242      0.447312      0.447599
min         0.000000      1.000000      1.000000      0.000000      0.000000
25%         0.000000      2.000000      2.000000      0.000000      0.000000
50%         0.000000     14.000000      8.000000      0.000000      0.000000
75%         0.000000    144.000000     18.000000      1.000000      1.000000
max         1.000000    511.000000    511.000000      1.000000      1.000000

                 a27           a28           a29           a30           a31  \
count   25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean        0.118630      0.120260      0.660559      0.062363      0.095931
std         0.318745      0.322335      0.439637      0.178550      0.256583
min         0.000000      0.000000      0.000000      0.000000      0.000000
25%         0.000000      0.000000      0.090000      0.000000      0.000000
50%         0.000000      0.000000      1.000000      0.000000      0.000000
75%         0.000000      0.000000      1.000000      0.060000      0.000000
max         1.000000      1.000000      1.000000      1.000000      1.000000

                 a32           a33           a34           a35           a36  \
count   25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean      182.532074    115.063036      0.519791      0.082539      0.147453
std        98.993895    110.646850      0.448944      0.187191      0.308367
min         0.000000      0.000000      0.000000      0.000000      0.000000
25%        84.000000     10.000000      0.050000      0.000000      0.000000
50%       255.000000     61.000000      0.510000      0.030000      0.000000
75%       255.000000    255.000000      1.000000      0.070000      0.060000
max       255.000000    255.000000      1.000000      1.000000      1.000000
```

```
                a37           a38           a39           a40           a41  \
count  25192.000000  25192.000000  25192.000000  25192.000000  25192.000000
mean       0.031844      0.285800      0.279846      0.117800      0.118769
std        0.110575      0.445316      0.446075      0.305869      0.317333
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.020000      1.000000      1.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

                a42
count  25192.000000
mean       1.171364
std        2.222340
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max       21.000000
```

[22]:

```
# Version-2 Info
version2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25192 entries, 0 to 25191
Data columns (total 36 columns):
a7      25192 non-null int64
a8      25192 non-null int64
a9      25192 non-null int64
a10     25192 non-null int64
a11     25192 non-null int64
a12     25192 non-null int64
a13     25192 non-null int64
a14     25192 non-null int64
a15     25192 non-null int64
a16     25192 non-null int64
a17     25192 non-null int64
a18     25192 non-null int64
a19     25192 non-null int64
a20     25192 non-null int64
a21     25192 non-null int64
a22     25192 non-null int64
a23     25192 non-null int64
a24     25192 non-null int64
a25     25192 non-null float64
a26     25192 non-null float64
a27     25192 non-null float64
a28     25192 non-null float64
```

```
a29     25192 non-null float64
a30     25192 non-null float64
a31     25192 non-null float64
a32     25192 non-null int64
a33     25192 non-null int64
a34     25192 non-null float64
a35     25192 non-null float64
a36     25192 non-null float64
a37     25192 non-null float64
a38     25192 non-null float64
a39     25192 non-null float64
a40     25192 non-null float64
a41     25192 non-null float64
a42     25192 non-null int64
dtypes: float64(15), int64(21)
memory usage: 6.9 MB
```

[23]:

```
# Checking for missing values in version-1

version1.isnull().sum()
```

[23]:

```
a1      0
a2      0
a3      0
a4      0
a5      0
a6      0
a7      0
a8      0
a9      0
a10     0
a11     0
a12     0
a13     0
a14     0
a15     0
a16     0
a17     0
a18     0
a19     0
a20     0
a21     0
a22     0
a23     0
a24     0
a25     0
a26     0
a27     0
```

```
a28      0
a29      0
a30      0
a31      0
a32      0
a33      0
a34      0
a35      0
a36      0
a37      0
a38      0
a39      0
a40      0
a41      0
a42      0
dtype: int64
```

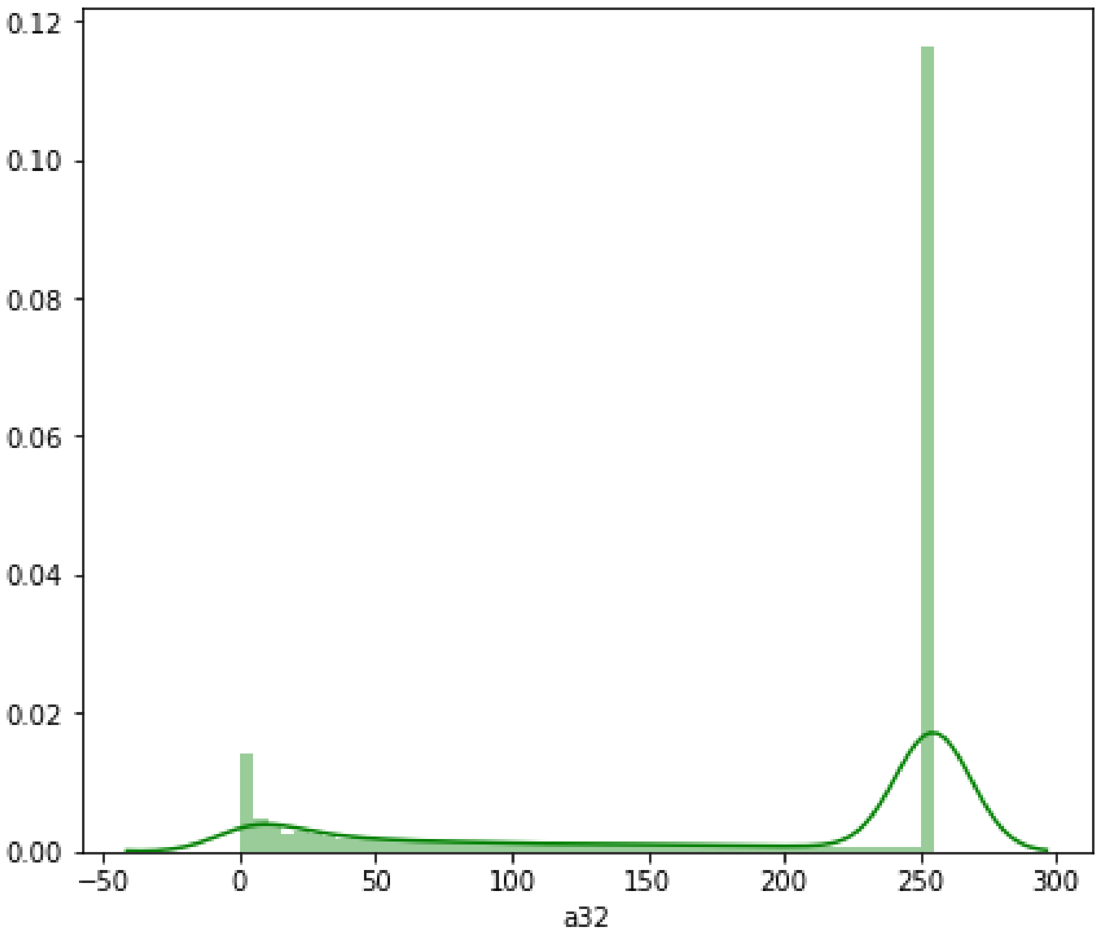
[24]:
```python
# Checking for missing values in version-2
version2.isnull().sum()
```

[24]:
```
a7       0
a8       0
a9       0
a10      0
a11      0
a12      0
a13      0
a14      0
a15      0
a16      0
a17      0
a18      0
a19      0
a20      0
a21      0
a22      0
a23      0
a24      0
a25      0
a26      0
a27      0
a28      0
a29      0
a30      0
a31      0
a32      0
a33      0
a34      0
```

```
a35    0
a36    0
a37    0
a38    0
a39    0
a40    0
a41    0
a42    0
dtype: int64
```

[25]:
```
#A32 distribution plot

plt.figure(figsize = (7,6))
sns.distplot(version1['a32'], color = 'g', bins = 50)
```

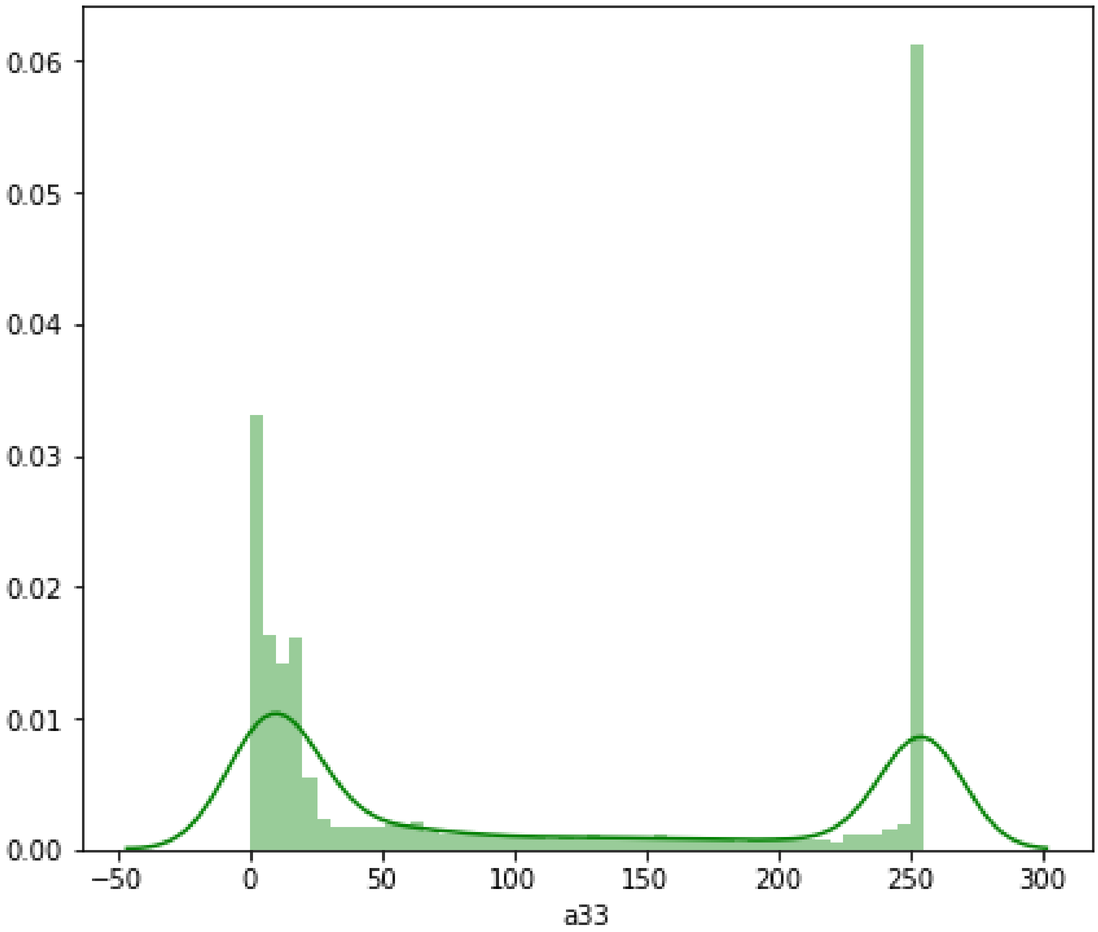
[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46310290f0>
```



[26]:
```
#A33 distribution plot
```

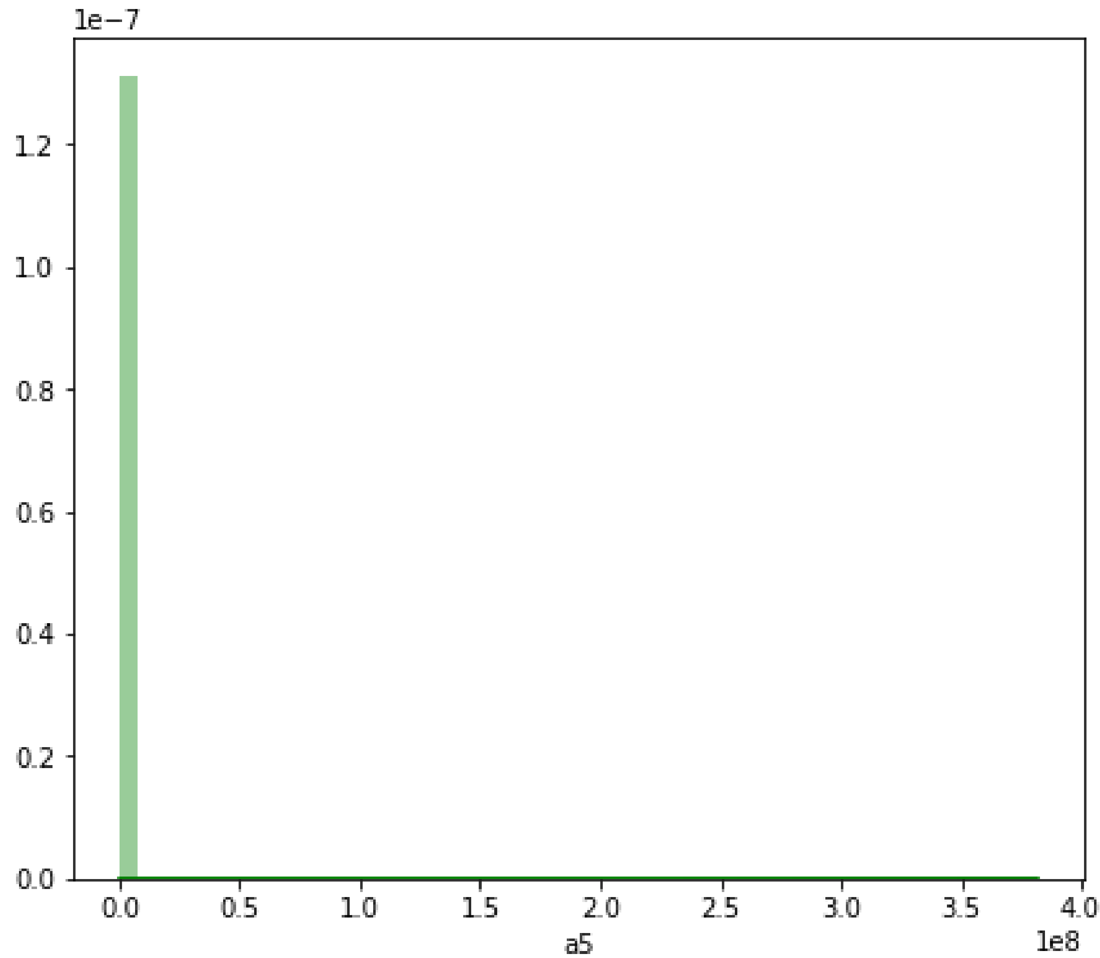
```python
plt.figure(figsize = (7,6))
sns.distplot(version1['a33'], color = 'g', bins = 50)
```

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4631037940>

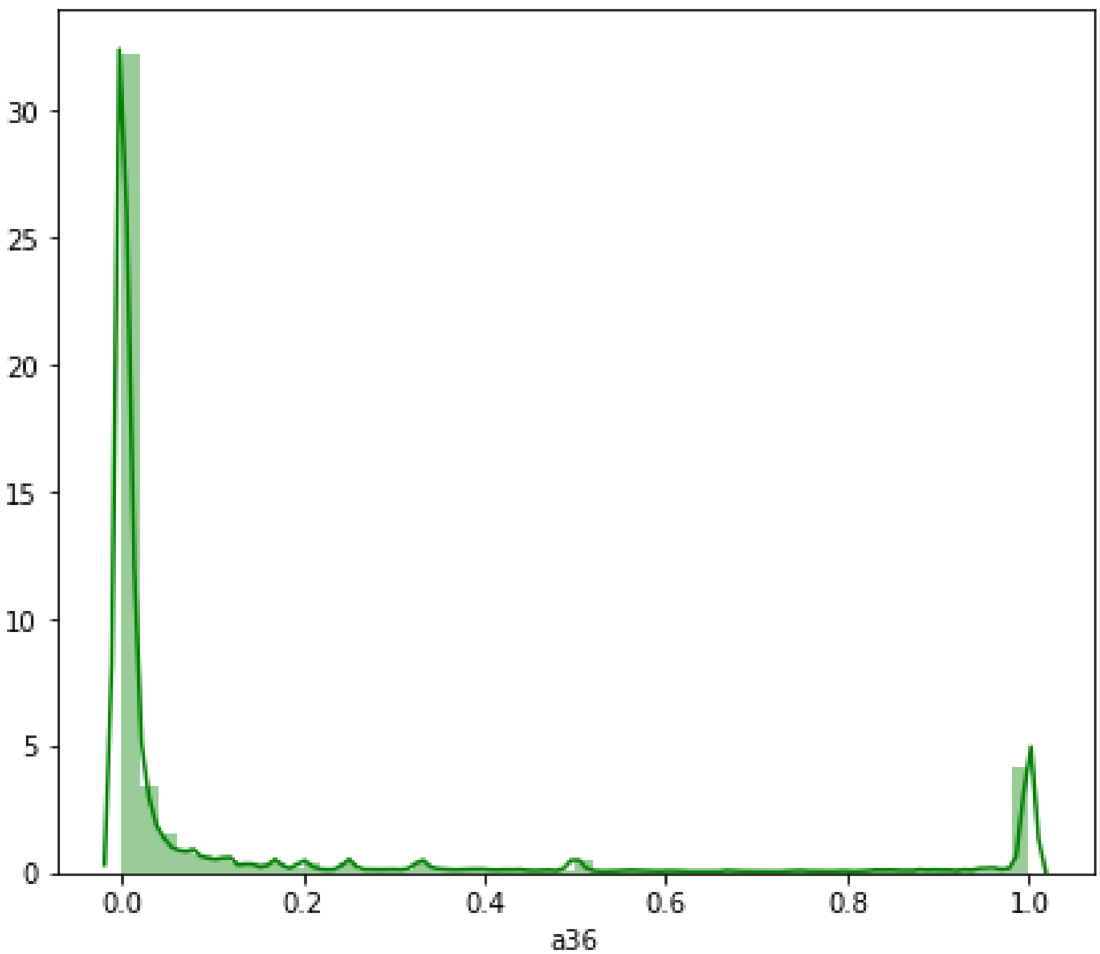


[27]:
```python
#a36 and a5 distribution plot

plt.figure(figsize = (7,6))
sns.distplot(version1['a5'], color = 'g', bins = 50)

plt.figure(figsize = (7,6))
sns.distplot(version1['a36'], color = 'g', bins = 50)
```

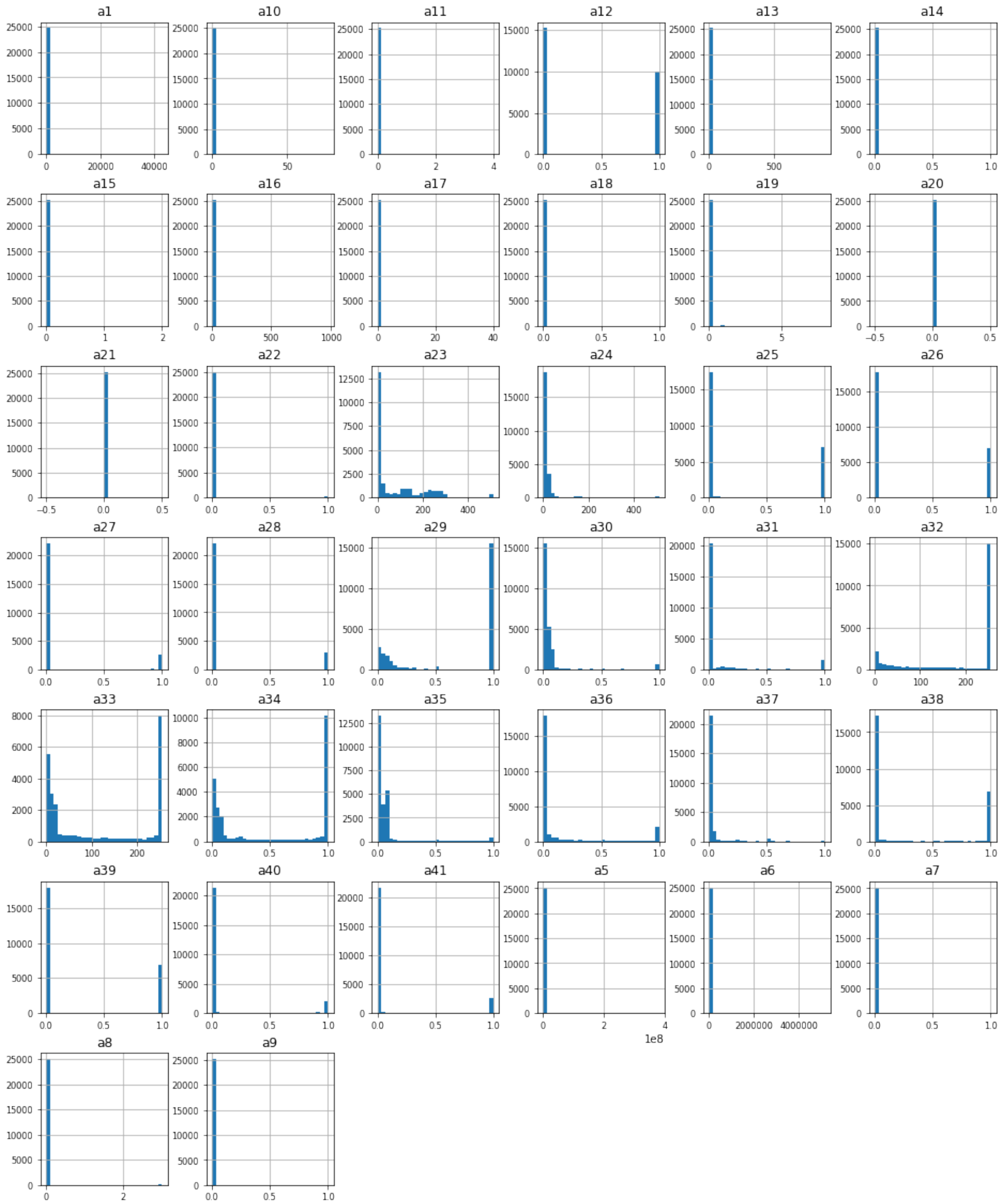
[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4630742390>



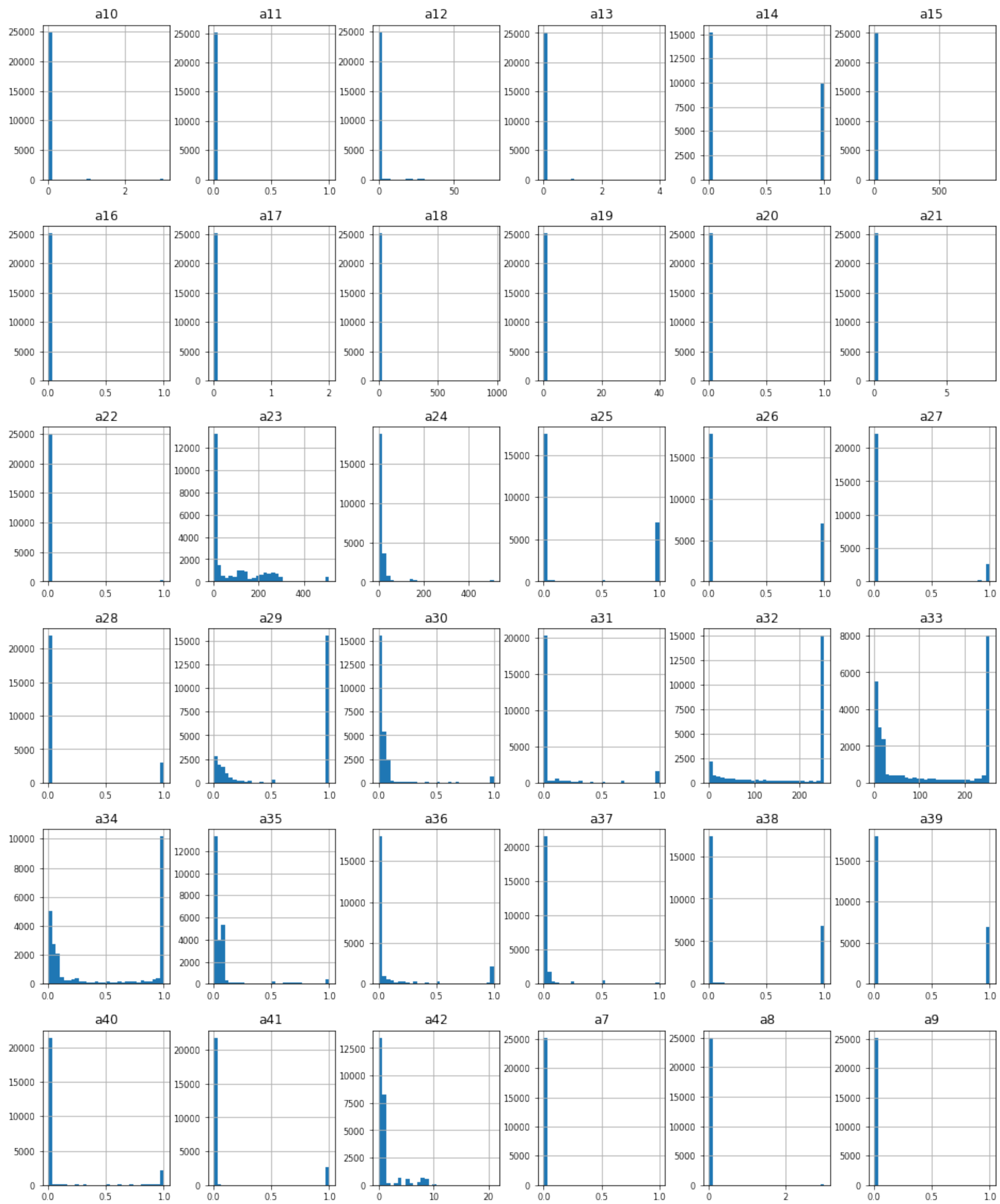
1e−7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
a5
1e8



[28]:
```
#Each variable histogramfor Version-1

version1.hist(figsize = (16,20), bins = 30, xlabelsize =8, ylabelsize = 8)
plt.show()
```

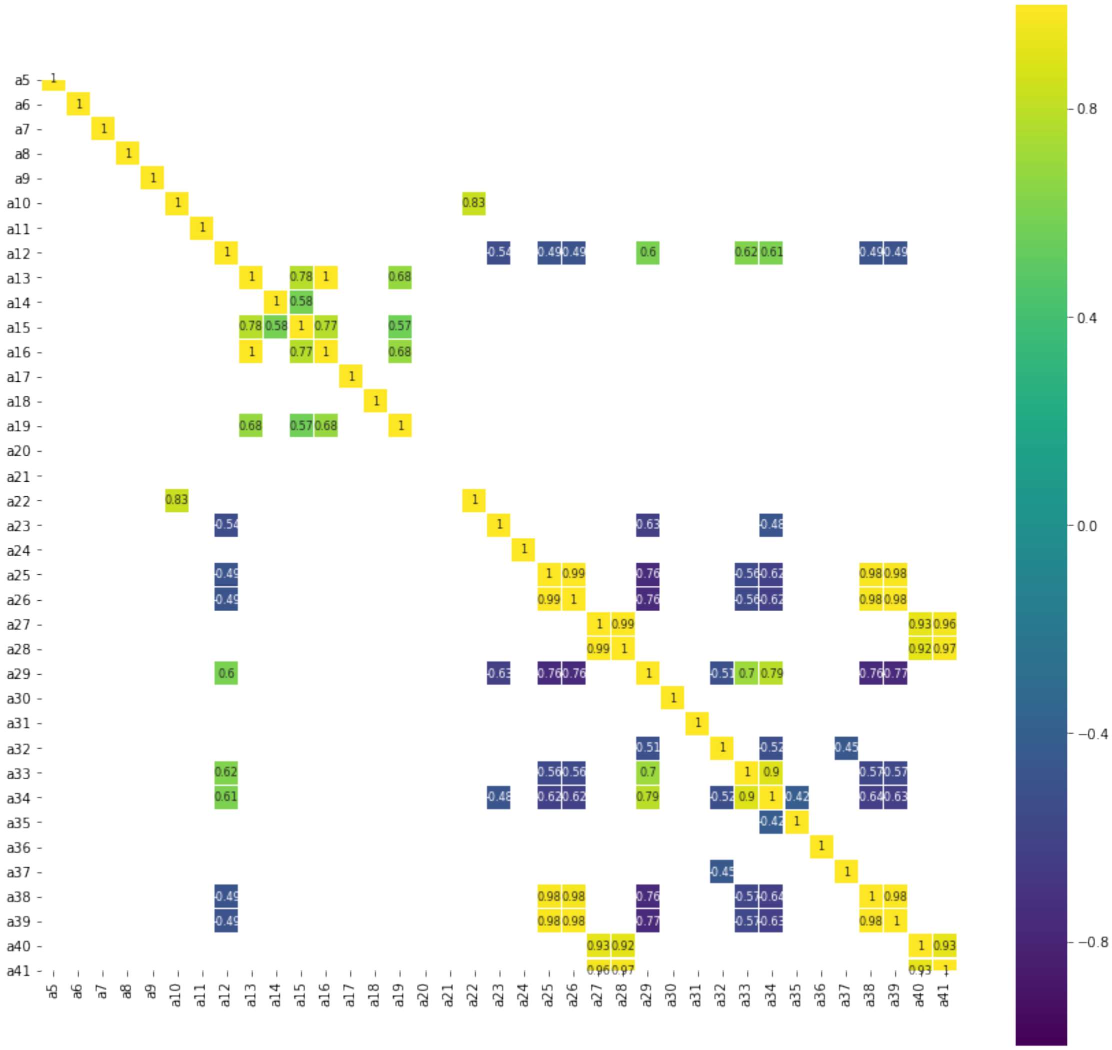


[29]:
```
#Each variable histogramfor Version-2

version2.hist(figsize = (16,20), bins = 30, xlabelsize =8, ylabelsize = 8)
plt.show()
```

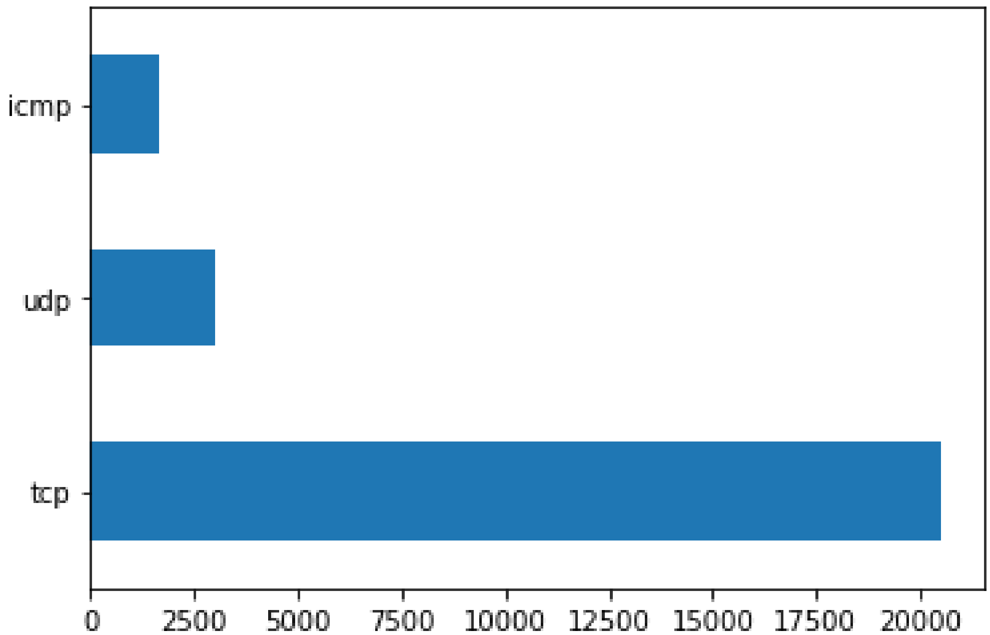


[30]:
```
#Correlation diagram/heatmap for version-1

plt.figure(figsize = (15,14))
corr = version1.drop('a1', axis=1).corr()

sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.4)],
cmap = 'viridis', vmax = 1.0, vmin = -1.0, linewidths = 0.1,
```

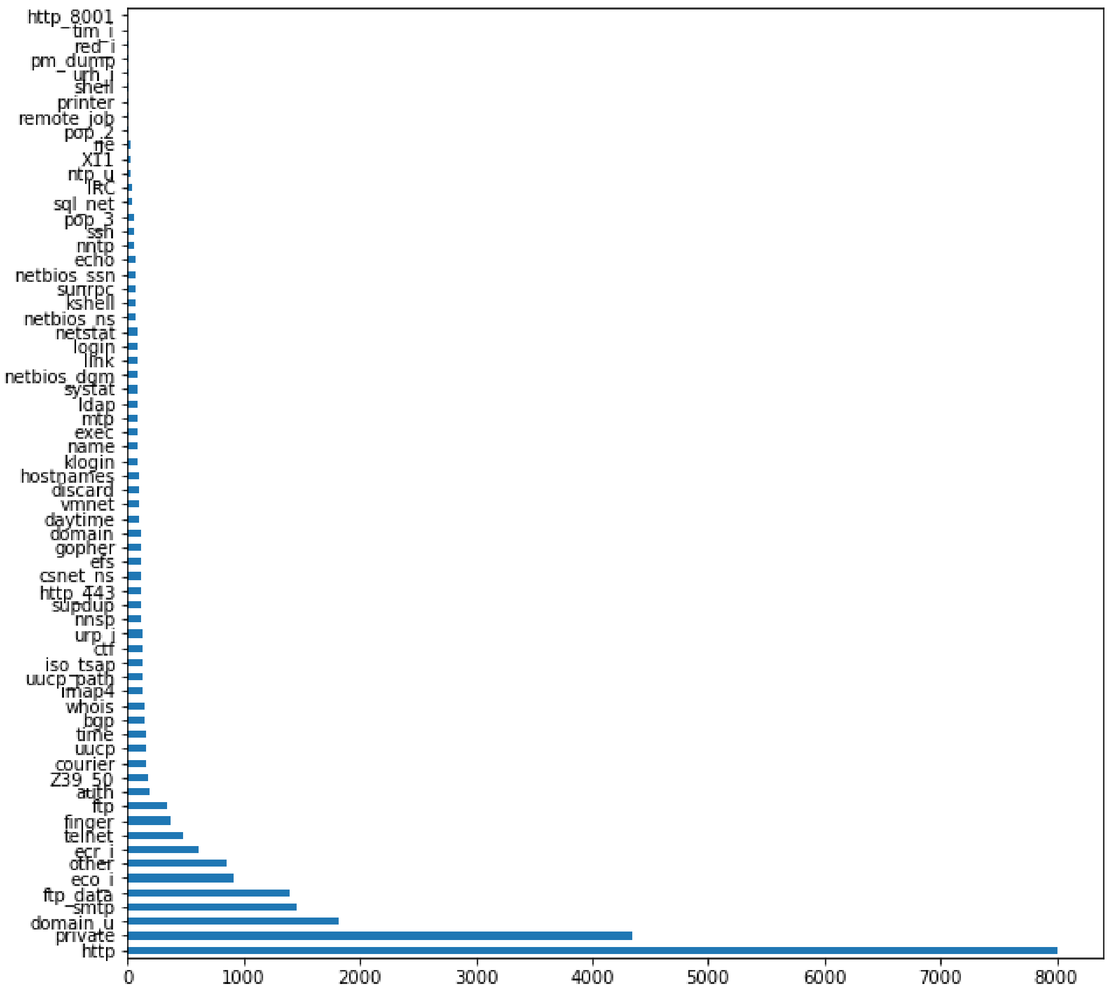
```python
annot = True, annot_kws = {"size":8}, square = True);
```



[31]:
```python
# Representation of a2 value count on bar plot

version1['a2'].value_counts().plot.barh()
plt.show()
```

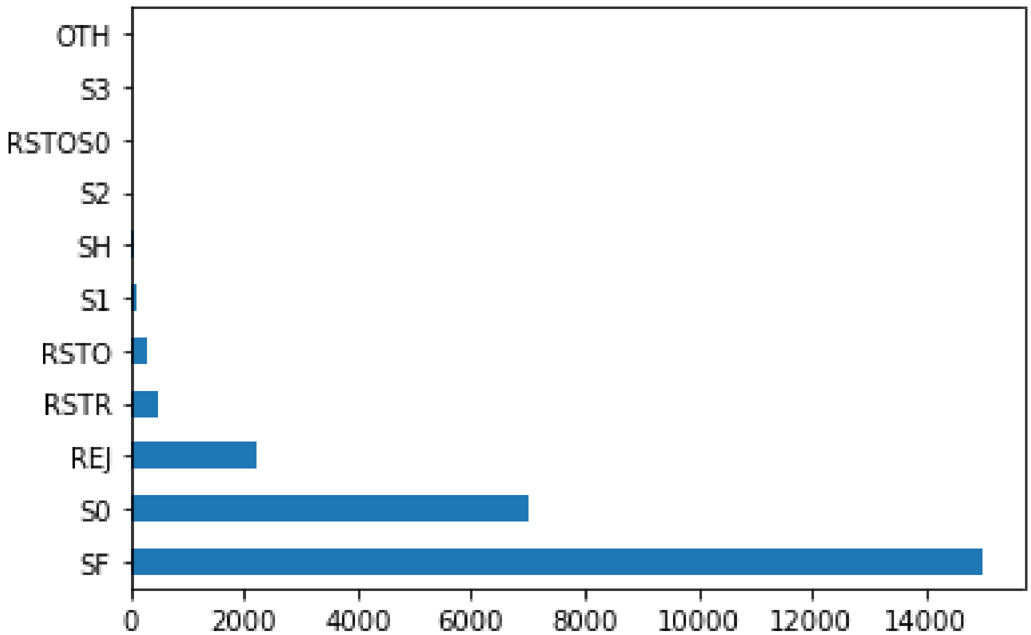


[32]:
```
# Representation of a3 value count on bar plot

plt.figure(figsize=(10,10))
version1['a3'].value_counts().plot.barh()
plt.show()
```

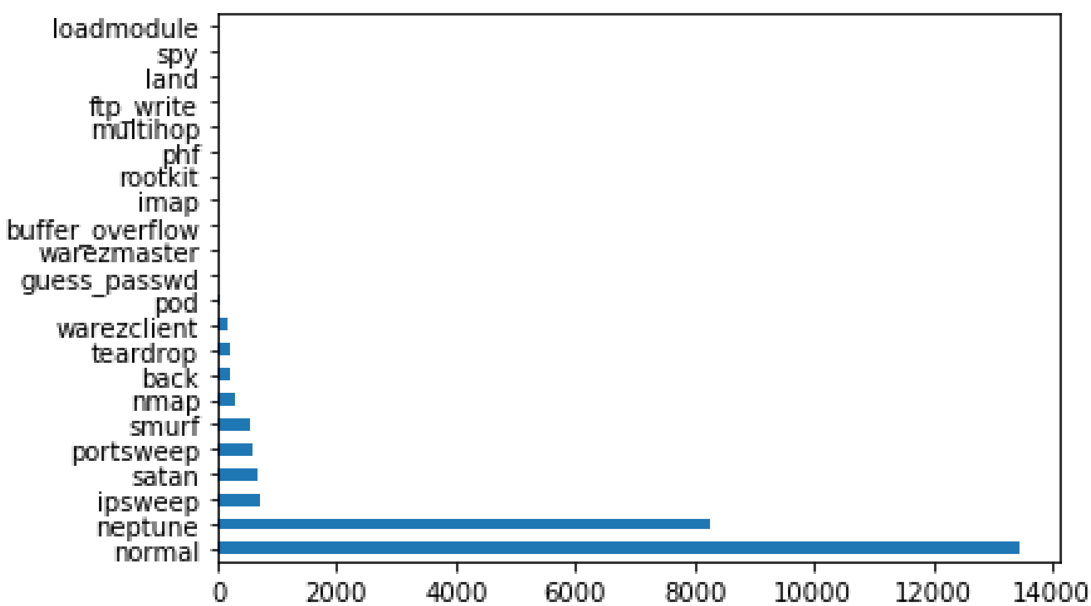


[33]:

```
# Representation of a4 value count on bar plot

version1['a4'].value_counts().plot.barh()
plt.show()
```



[34]:
```
# Representation of a42 value count on bar plot

version1['a42'].value_counts().plot.barh()
plt.show()
```



[35]:
```
# Correlation representation in values

corr_matrix = version1.corr().abs()
corr_matrix.head()
```

[35]:

	a1	a5	a6	a7	a8	a9	a10
a1	1.000000	0.084864	0.013258	0.001012	0.010358	0.000486	0.004202
a5	0.084864	1.000000	0.003611	0.000090	0.000916	0.000062	0.000995
a6	0.013258	0.003611	1.000000	0.000350	0.003586	0.000345	0.002539
a7	0.001012	0.000090	0.000350	1.000000	0.000813	0.000056	0.000819
a8	0.010358	0.000916	0.003586	0.000813	1.000000	0.000575	0.008386

	a11	a12	a13	a14	a15	a16	a17
a1	0.011108	0.063703	0.095215	0.050547	0.094243	0.094066	0.088272
a5	0.000260	0.002040	0.000196	0.000383	0.000267	0.000209	0.000218
a6	0.005197	0.012704	0.035852	0.020214	0.035041	0.035171	0.008456
a7	0.000234	0.007196	0.000195	0.000351	0.000247	0.000194	0.000248
a8	0.002392	0.073674	0.001995	0.003592	0.002524	0.001982	0.002537

	a18	a19	a20	a21	a22	a23	a24	a25
a1	0.001585	0.070206	NaN	NaN	0.002050	0.081787	0.040642	0.072458
a5	0.000158	0.000422	NaN	NaN	0.000932	0.007302	0.003623	0.006312
a6	0.000146	0.024142	NaN	NaN	0.001161	0.027824	0.012524	0.022390
a7	0.000168	0.000391	NaN	NaN	0.000855	0.006495	0.003221	0.014216
a8	0.001725	0.004006	NaN	NaN	0.008756	0.023241	0.023377	0.045228

	a26	a27	a28	a29	a30	a31	a32
a1	0.071832	0.209441	0.208354	0.075723	0.012009	0.041115	0.055174
a5	0.006225	0.016015	0.015816	0.007673	0.003098	0.003077	0.009764
a6	0.022443	0.013843	0.013664	0.030018	0.012300	0.007560	0.030930
a7	0.014259	0.003316	0.003324	0.006880	0.003112	0.014033	0.016340
a8	0.057834	0.033464	0.034035	0.056683	0.027428	0.028744	0.040020

	a33	a34	a35	a36	a37	a38	a39
a1	0.112530	0.119321	0.263489	0.240970	0.025485	0.066513	0.066240
a5	0.008520	0.006776	0.001026	0.002316	0.001238	0.006346	0.006227
a6	0.000980	0.022392	0.012971	0.024078	0.006006	0.015584	0.014543
a7	0.008743	0.009531	0.003929	0.024635	0.053037	0.014291	0.005596
a8	0.047256	0.051845	0.053177	0.034670	0.020174	0.053786	0.057230

	a40	a41
a1	0.187070	0.208435
a5	0.002130	0.006190
a6	0.014094	0.012803
a7	0.003432	0.003335
a8	0.027718	0.034143

[36]:
```
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
  ↪bool))
upper.head()
```

[36]:
```
     a1        a5        a6        a7        a8        a9       a10       a11  \
a1  NaN  0.084864  0.013258  0.001012  0.010358  0.000486  0.004202  0.011108
a5  NaN       NaN  0.003611  0.000090  0.000916  0.000062  0.000995  0.000260
a6  NaN       NaN       NaN  0.000350  0.003586  0.000345  0.002539  0.005197
a7  NaN       NaN       NaN       NaN  0.000813  0.000056  0.000819  0.000234
a8  NaN       NaN       NaN       NaN       NaN  0.000575  0.008386  0.002392

         a12       a13       a14       a15       a16       a17       a18  \
a1  0.063703  0.095215  0.050547  0.094243  0.094066  0.088272  0.001585
a5  0.002040  0.000196  0.000383  0.000267  0.000209  0.000218  0.000158
a6  0.012704  0.035852  0.020214  0.035041  0.035171  0.008456  0.000146
a7  0.007196  0.000195  0.000351  0.000247  0.000194  0.000248  0.000168
a8  0.073674  0.001995  0.003592  0.002524  0.001982  0.002537  0.001725

         a19  a20  a21       a22       a23       a24       a25       a26  \
a1  0.070206  NaN  NaN  0.002050  0.081787  0.040642  0.072458  0.071832
a5  0.000422  NaN  NaN  0.000932  0.007302  0.003623  0.006312  0.006225
a6  0.024142  NaN  NaN  0.001161  0.027824  0.012524  0.022390  0.022443
a7  0.000391  NaN  NaN  0.000855  0.006495  0.003221  0.014216  0.014259
a8  0.004006  NaN  NaN  0.008756  0.023241  0.023377  0.045228  0.057834

         a27       a28       a29       a30       a31       a32       a33  \
a1  0.209441  0.208354  0.075723  0.012009  0.041115  0.055174  0.112530
a5  0.016015  0.015816  0.007673  0.003098  0.003077  0.009764  0.008520
a6  0.013843  0.013664  0.030018  0.012300  0.007560  0.030930  0.000980
a7  0.003316  0.003324  0.006880  0.003112  0.014033  0.016340  0.008743
a8  0.033464  0.034035  0.056683  0.027428  0.028744  0.040020  0.047256

         a34       a35       a36       a37       a38       a39       a40  \
a1  0.119321  0.263489  0.240970  0.025485  0.066513  0.066240  0.187070
a5  0.006776  0.001026  0.002316  0.001238  0.006346  0.006227  0.002130
a6  0.022392  0.012971  0.024078  0.006006  0.015584  0.014543  0.014094
a7  0.009531  0.003929  0.024635  0.053037  0.014291  0.005596  0.003432
a8  0.051845  0.053177  0.034670  0.020174  0.053786  0.057230  0.027718

         a41
a1  0.208435
a5  0.006190
a6  0.012803
a7  0.003335
a8  0.034143
```

[37]:
```
#Values that correlates
```

```python
def find_correlation(version1, threshold=0.50, remove_negative=False):
    corr_mat = version1.corr()
    if remove_negative:
        corr_mat = np.abs(corr_mat)
    corr_mat.loc[:, :] = np.tril(corr_mat, k=-1)
    already_in = set()
    result = []
    for col in corr_mat:
        perfect_corr = corr_mat[col][corr_mat[col] > threshold].index.tolist()
        if perfect_corr and col not in already_in:
            already_in.update(set(perfect_corr))
            perfect_corr.append(col)
            result.append(perfect_corr)
    select_nested = [f[1:] for f in result]
    select_flat = [i for j in select_nested for i in j]
    return select_flat
```

[38]:
```python
find_correlation(version1)
```

[38]:
```
['a10',
 'a33',
 'a34',
 'a12',
 'a16',
 'a19',
 'a13',
 'a14',
 'a38',
 'a39',
 'a25',
 'a40',
 'a41',
 'a27']
```

[39]:
```python
# Values that correlates

corr_matrix = version1.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
 ↪bool))

# Find index of feature columns with correlation greater than 0.95
upper

to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]
# Drop features
to_drop
```

[39]: ['a16', 'a26', 'a28', 'a38', 'a39', 'a41']

[40]:
```
# Values with high correlation

def corr_df(version1, corr_val):
    '''
    Obj: Drops features that are strongly correlated to other features.
          This lowers model complexity, and aids in generalizing the model.
    Inputs:
          df: features df (x)
          corr_val: Columns are dropped relative to the corr_val input (e.g. 0.
 ↪8)
    Output: df that only includes uncorrelated features
    '''

    # Creates Correlation Matrix and Instantiates
    corr_matrix = version1.corr()
    iters = range(len(corr_matrix.columns) - 1)
    drop_cols = []

    # Iterates through Correlation Matrix Table to find correlated columns
    for i in iters:
        for j in range(i):
            item = corr_matrix.iloc[j:(j+1), (i+1):(i+2)]
            col = item.columns
            row = item.index
            val = item.values
            if abs(val) >= corr_val:
                # Prints the correlated feature set and the corr val
                print(col.values[0], "|", row.values[0], "|", round(val[0][0],␣
 ↪2))
                drop_cols.append(i)

    drops = sorted(set(drop_cols))[::-1]

    # Drops the correlated columns
    for i in drops:
        col = x.iloc[:, (i+1):(i+2)].columns.values
        x = x.drop(col, axis=1)
    return x
```

[41]:
```
corr_df(version1, 0.90)
```

```
a16 | a13 | 1.0
a38 | a25 | 0.98
a38 | a26 | 0.98
a39 | a25 | 0.98
a39 | a26 | 0.98
```

```
a40 | a27 | 0.93
a40 | a28 | 0.92
a41 | a27 | 0.96
a41 | a28 | 0.97
```

```
 ␣
↪---------------------------------------------------------------------------

        UnboundLocalError                         Traceback (most recent call␣
↪last)

        <ipython-input-41-4368195cf4a6> in <module>
    ----> 1 corr_df(version1, 0.90)


        <ipython-input-40-c2d506e1794f> in corr_df(version1, corr_val)
         30     # Drops the correlated columns
         31     for i in drops:
    ---> 32         col = x.iloc[:, (i+1):(i+2)].columns.values
         33         x = x.drop(col, axis=1)
         34     return x


        UnboundLocalError: local variable 'x' referenced before assignment
```

[43]:
```
# a2 unique values
version1['a2'].unique()
```

[43]:
```
array(['tcp', 'udp', 'icmp'], dtype=object)
```

[44]:
```
# a3 unique values

version1['a3'].unique()
```

[44]:
```
array(['ftp_data', 'other', 'private', 'http', 'remote_job', 'name',
       'netbios_ns', 'eco_i', 'mtp', 'telnet', 'finger', 'domain_u',
       'supdup', 'uucp_path', 'Z39_50', 'smtp', 'csnet_ns', 'uucp',
       'netbios_dgm', 'urp_i', 'auth', 'domain', 'ftp', 'bgp', 'ldap',
       'ecr_i', 'gopher', 'vmnet', 'systat', 'http_443', 'efs', 'whois',
       'imap4', 'iso_tsap', 'echo', 'klogin', 'link', 'sunrpc', 'login',
       'kshell', 'sql_net', 'time', 'hostnames', 'exec', 'ntp_u',
       'discard', 'nntp', 'courier', 'ctf', 'ssh', 'daytime', 'shell',
       'netstat', 'pop_3', 'nnsp', 'IRC', 'pop_2', 'printer', 'tim_i',
       'pm_dump', 'red_i', 'netbios_ssn', 'rje', 'X11', 'urh_i',
       'http_8001'], dtype=object)
```

[45]:
```
# a4 unique values
```

```
version1['a4'].unique()
```

[45]:
```
array(['SF', 'S0', 'REJ', 'RSTR', 'SH', 'RSTO', 'S1', 'RSTOS0', 'S3',
       'S2', 'OTH'], dtype=object)
```

[46]:
```
# a42 unique values

version1['a42'].unique()
```

[46]:
```
array(['normal', 'neptune', 'warezclient', 'ipsweep', 'portsweep',
       'teardrop', 'nmap', 'satan', 'smurf', 'pod', 'back',
       'guess_passwd', 'ftp_write', 'multihop', 'rootkit',
       'buffer_overflow', 'imap', 'warezmaster', 'phf', 'land',
       'loadmodule', 'spy'], dtype=object)
```

[47]:
```
# Applying label encoding to label the target variable

# import label encoder

from sklearn import preprocessing

le = preprocessing.LabelEncoder()
```

[48]:
```
version1['a2'] = le.fit_transform(version1['a2'])
version1['a2'].unique()
```

[48]:
```
array([1, 2, 0])
```

[49]:
```
version1['a4'] = le.fit_transform(version1['a4'])
version1['a4'].unique()
```

[49]:
```
array([ 9,  5,  1,  4, 10,  2,  6,  3,  8,  7,  0])
```

[50]:
```
version1['a42'] = le.fit_transform(version1['a42'])
version1['a42'].unique()
```

[50]:
```
array([11,  9, 20,  5, 14, 19, 10, 16, 17, 13,  0,  3,  2,  8, 15,  1,  4,
       21, 12,  6,  7, 18])
```

[51]:
```
version1['a3'] = le.fit_transform(version1['a3'])
version1['a3'].unique()
```

[51]:
```
array([19, 41, 46, 22, 48, 33, 35, 13, 32, 57, 17, 11, 55, 63,  2, 51,  6,
       62, 34, 61,  3, 10, 18,  4, 29, 14, 20, 64, 56, 23, 15, 65, 25, 26,
       12, 27, 30, 54, 31, 28, 52, 59, 21, 16, 40,  9, 39,  5,  7, 53,  8,
       50, 37, 44, 38,  0, 43, 45, 58, 42, 47, 36, 49,  1, 60, 24])
```

[52]:
```
# Statistical Information of the version1 dataset

version1_drop = version1.drop(columns = ['a2', 'a3', 'a4', 'a42'])
statistical_summary_version1 = pd.DataFrame(version1_drop.describe()).T
statistical_summary_version1.head
```

[52]:

```
<bound method NDFrame.head of          count          mean           std  min
25%      50%      75%  \
a1   25192.0    305.054104  2.686556e+03  0.0    0.00    0.00    0.00
a5   25192.0  24330.628215  2.410805e+06  0.0    0.00   44.00  279.00
a6   25192.0   3491.847174  8.883072e+04  0.0    0.00    0.00  530.25
a7   25192.0      0.000079  8.909946e-03  0.0    0.00    0.00    0.00
a8   25192.0      0.023738  2.602208e-01  0.0    0.00    0.00    0.00
a9   25192.0      0.000040  6.300408e-03  0.0    0.00    0.00    0.00
a10  25192.0      0.198039  2.154202e+00  0.0    0.00    0.00    0.00
a11  25192.0      0.001191  4.541818e-02  0.0    0.00    0.00    0.00
a12  25192.0      0.394768  4.888105e-01  0.0    0.00    0.00    1.00
a13  25192.0      0.227850  1.041735e+01  0.0    0.00    0.00    0.00
a14  25192.0      0.001548  3.931635e-02  0.0    0.00    0.00    0.00
a15  25192.0      0.001350  4.878505e-02  0.0    0.00    0.00    0.00
a16  25192.0      0.249841  1.150084e+01  0.0    0.00    0.00    0.00
a17  25192.0      0.014727  5.296023e-01  0.0    0.00    0.00    0.00
a18  25192.0      0.000357  1.889822e-02  0.0    0.00    0.00    0.00
a19  25192.0      0.004327  9.852398e-02  0.0    0.00    0.00    0.00
a20  25192.0      0.000000  0.000000e+00  0.0    0.00    0.00    0.00
a21  25192.0      0.000000  0.000000e+00  0.0    0.00    0.00    0.00
a22  25192.0      0.009130  9.511512e-02  0.0    0.00    0.00    0.00
a23  25192.0     84.591180  1.146735e+02  1.0    2.00   14.00  144.00
a24  25192.0     27.698754  7.246824e+01  1.0    2.00    8.00   18.00
a25  25192.0      0.286338  4.473123e-01  0.0    0.00    0.00    1.00
a26  25192.0      0.283762  4.475989e-01  0.0    0.00    0.00    1.00
a27  25192.0      0.118630  3.187455e-01  0.0    0.00    0.00    0.00
a28  25192.0      0.120260  3.223354e-01  0.0    0.00    0.00    0.00
a29  25192.0      0.660559  4.396374e-01  0.0    0.09    1.00    1.00
a30  25192.0      0.062363  1.785500e-01  0.0    0.00    0.00    0.06
a31  25192.0      0.095931  2.565828e-01  0.0    0.00    0.00    0.00
a32  25192.0    182.532074  9.899390e+01  0.0   84.00  255.00  255.00
a33  25192.0    115.063036  1.106469e+02  0.0   10.00   61.00  255.00
a34  25192.0      0.519791  4.489439e-01  0.0    0.05    0.51    1.00
a35  25192.0      0.082539  1.871911e-01  0.0    0.00    0.03    0.07
a36  25192.0      0.147453  3.083666e-01  0.0    0.00    0.00    0.06
a37  25192.0      0.031844  1.105750e-01  0.0    0.00    0.00    0.02
a38  25192.0      0.285800  4.453165e-01  0.0    0.00    0.00    1.00
a39  25192.0      0.279846  4.460753e-01  0.0    0.00    0.00    1.00
a40  25192.0      0.117800  3.058692e-01  0.0    0.00    0.00    0.00
a41  25192.0      0.118769  3.173335e-01  0.0    0.00    0.00    0.00

             max
a1       42862.0
a5   381709090.0
a6     5151385.0
a7           1.0
a8           3.0
```

```
a9           1.0
a10         77.0
a11          4.0
a12          1.0
a13        884.0
a14          1.0
a15          2.0
a16        975.0
a17         40.0
a18          1.0
a19          8.0
a20          0.0
a21          0.0
a22          1.0
a23        511.0
a24        511.0
a25          1.0
a26          1.0
a27          1.0
a28          1.0
a29          1.0
a30          1.0
a31          1.0
a32        255.0
a33        255.0
a34          1.0
a35          1.0
a36          1.0
a37          1.0
a38          1.0
a39          1.0
a40          1.0
a41          1.0  >
```

[53]:
```
# Statistical Information of the version2 dataset

statistical_summary_version2 = pd.DataFrame(version2.describe()).T
statistical_summary_version2
```

[53]:

	count	mean	std	min	25%	50%	75%	max
a7	25192.0	0.000079	0.008910	0.0	0.00	0.00	0.00	1.0
a8	25192.0	0.023738	0.260221	0.0	0.00	0.00	0.00	3.0
a9	25192.0	0.000079	0.008910	0.0	0.00	0.00	0.00	1.0
a10	25192.0	0.023738	0.260221	0.0	0.00	0.00	0.00	3.0
a11	25192.0	0.000040	0.006300	0.0	0.00	0.00	0.00	1.0
a12	25192.0	0.198039	2.154202	0.0	0.00	0.00	0.00	77.0
a13	25192.0	0.001191	0.045418	0.0	0.00	0.00	0.00	4.0
a14	25192.0	0.394768	0.488811	0.0	0.00	0.00	1.00	1.0

a15	25192.0	0.227850	10.417352	0.0	0.00	0.00	0.00	884.0
a16	25192.0	0.001548	0.039316	0.0	0.00	0.00	0.00	1.0
a17	25192.0	0.001350	0.048785	0.0	0.00	0.00	0.00	2.0
a18	25192.0	0.249841	11.500842	0.0	0.00	0.00	0.00	975.0
a19	25192.0	0.014727	0.529602	0.0	0.00	0.00	0.00	40.0
a20	25192.0	0.000357	0.018898	0.0	0.00	0.00	0.00	1.0
a21	25192.0	0.004327	0.098524	0.0	0.00	0.00	0.00	8.0
a22	25192.0	0.009130	0.095115	0.0	0.00	0.00	0.00	1.0
a23	25192.0	84.591180	114.673451	1.0	2.00	14.00	144.00	511.0
a24	25192.0	27.698754	72.468242	1.0	2.00	8.00	18.00	511.0
a25	25192.0	0.286338	0.447312	0.0	0.00	0.00	1.00	1.0
a26	25192.0	0.283762	0.447599	0.0	0.00	0.00	1.00	1.0
a27	25192.0	0.118630	0.318745	0.0	0.00	0.00	0.00	1.0
a28	25192.0	0.120260	0.322335	0.0	0.00	0.00	0.00	1.0
a29	25192.0	0.660559	0.439637	0.0	0.09	1.00	1.00	1.0
a30	25192.0	0.062363	0.178550	0.0	0.00	0.00	0.06	1.0
a31	25192.0	0.095931	0.256583	0.0	0.00	0.00	0.00	1.0
a32	25192.0	182.532074	98.993895	0.0	84.00	255.00	255.00	255.0
a33	25192.0	115.063036	110.646850	0.0	10.00	61.00	255.00	255.0
a34	25192.0	0.519791	0.448944	0.0	0.05	0.51	1.00	1.0
a35	25192.0	0.082539	0.187191	0.0	0.00	0.03	0.07	1.0
a36	25192.0	0.147453	0.308367	0.0	0.00	0.00	0.06	1.0
a37	25192.0	0.031844	0.110575	0.0	0.00	0.00	0.02	1.0
a38	25192.0	0.285800	0.445316	0.0	0.00	0.00	1.00	1.0
a39	25192.0	0.279846	0.446075	0.0	0.00	0.00	1.00	1.0
a40	25192.0	0.117800	0.305869	0.0	0.00	0.00	0.00	1.0
a41	25192.0	0.118769	0.317333	0.0	0.00	0.00	0.00	1.0
a42	25192.0	1.171364	2.222340	0.0	0.00	0.00	1.00	21.0

[54]:
```
version1.head()
```

[54]:

	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	\
0	0	1	19	9	491	0	0	0	0	0	0	0	0	0	0	0	
1	0	2	41	9	146	0	0	0	0	0	0	0	0	0	0	0	
2	0	1	46	5	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	1	22	9	232	8153	0	0	0	0	0	1	0	0	0	0	
4	0	1	22	9	199	420	0	0	0	0	0	1	0	0	0	0	

	a17	a18	a19	a20	a21	a22	a23	a24	a25	a26	a27	a28	a29	a30	\
0	0	0	0	0	0	0	2	2	0.0	0.0	0.0	0.0	1.00	0.00	
1	0	0	0	0	0	0	13	1	0.0	0.0	0.0	0.0	0.08	0.15	
2	0	0	0	0	0	0	123	6	1.0	1.0	0.0	0.0	0.05	0.07	
3	0	0	0	0	0	0	5	5	0.2	0.2	0.0	0.0	1.00	0.00	
4	0	0	0	0	0	0	30	32	0.0	0.0	0.0	0.0	1.00	0.00	

	a31	a32	a33	a34	a35	a36	a37	a38	a39	a40	a41	a42
0	0.00	150	25	0.17	0.03	0.17	0.00	0.00	0.00	0.05	0.00	11
1	0.00	255	1	0.00	0.60	0.88	0.00	0.00	0.00	0.00	0.00	11

```
2  0.00   255    26  0.10  0.05  0.00  0.00  1.00  1.00  0.00  0.00     9
3  0.00    30   255  1.00  0.00  0.03  0.04  0.03  0.01  0.00  0.01    11
4  0.09   255   255  1.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00    11
```

[55]:
```
y = version1['a42']
```

[56]:
```
y.head()
```

[56]:
```
0    11
1    11
2     9
3    11
4    11
Name: a42, dtype: int64
```

[57]:
```
y.value_counts()
```

[57]:
```
11    13449
9      8282
5       710
16      691
14      587
17      529
10      301
0       196
19      188
20      181
13       38
3        10
21        7
1         6
4         5
15        4
8         2
12        2
7         1
6         1
18        1
2         1
Name: a42, dtype: int64
```

[58]:
```
X = version1.drop(['a42'],axis=1)
```

[59]:
```
X.head()
```

[59]:
```
   a1  a2  a3  a4   a5    a6  a7  a8  a9  a10  a11  a12  a13  a14  a15  a16  \
0   0   1  19   9  491     0   0   0   0    0    0    0    0    0    0    0
1   0   2  41   9  146     0   0   0   0    0    0    0    0    0    0    0
2   0   1  46   5    0     0   0   0   0    0    0    0    0    0    0    0
3   0   1  22   9  232  8153   0   0   0    0    0    1    0    0    0    0
4   0   1  22   9  199   420   0   0   0    0    0    1    0    0    0    0
```

```
   a17  a18  a19  a20  a21  a22  a23  a24  a25  a26  a27  a28   a29   a30  \
0    0    0    0    0    0    0    2    2  0.0  0.0  0.0  0.0  1.00  0.00
1    0    0    0    0    0    0   13    1  0.0  0.0  0.0  0.0  0.08  0.15
2    0    0    0    0    0    0  123    6  1.0  1.0  0.0  0.0  0.05  0.07
3    0    0    0    0    0    0    5    5  0.2  0.2  0.0  0.0  1.00  0.00
4    0    0    0    0    0    0   30   32  0.0  0.0  0.0  0.0  1.00  0.00

    a31  a32  a33   a34   a35   a36   a37   a38   a39   a40   a41
0  0.00  150   25  0.17  0.03  0.17  0.00  0.00  0.00  0.05  0.00
1  0.00  255    1  0.00  0.60  0.88  0.00  0.00  0.00  0.00  0.00
2  0.00  255   26  0.10  0.05  0.00  0.00  1.00  1.00  0.00  0.00
3  0.00   30  255  1.00  0.00  0.03  0.04  0.03  0.01  0.00  0.01
4  0.09  255  255  1.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

1 Model building

[60]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

[61]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state = 3)
```

[62]:
```
X_train.shape, X_test.shape
```

[62]:
```
((17634, 41), (7558, 41))
```

[63]:
```
y_train.shape, y_test.shape
```

[63]:
```
((17634,), (7558,))
```

[64]:
```
from sklearn.ensemble import RandomForestClassifier
```

[65]:
```
rf_model = RandomForestClassifier(n_estimators = 150, random_state = 3)
```

[66]:
```
rf_model.fit(X_train, y_train)
```

[66]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=False, random_state=3, verbose=0,
                       warm_start=False)
```

[67]:
```
rf_prediction = rf_model.predict(X_test)
```

[68]:
```
print(classification_report(y_test, rf_prediction))
```

```
              precision    recall  f1-score   support
```

```
           0       1.00      0.96      0.98        77
           1       1.00      1.00      1.00         1
           2       0.00      0.00      0.00         1
           3       1.00      1.00      1.00         5
           4       1.00      0.67      0.80         3
           5       0.99      0.99      0.99       198
           9       1.00      1.00      1.00      2509
          10       0.98      0.95      0.96        98
          11       0.99      1.00      1.00      3976
          12       0.00      0.00      0.00         1
          13       1.00      1.00      1.00         7
          14       0.99      0.99      0.99       188
          16       1.00      0.96      0.98       222
          17       1.00      1.00      1.00       170
          18       0.00      0.00      0.00         1
          19       1.00      1.00      1.00        53
          20       1.00      0.94      0.97        47
          21       0.00      0.00      0.00         1

    accuracy                           1.00      7558
   macro avg       0.77      0.75      0.76      7558
weighted avg       1.00      1.00      1.00      7558


/opt/conda/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

[91]:
```
ps = metrics.precision_score
```

[69]:
```
X.columns
```

[69]:
```
Index(['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'a11',
       'a12', 'a13', 'a14', 'a15', 'a16', 'a17', 'a18', 'a19', 'a20', 'a21',
       'a22', 'a23', 'a24', 'a25', 'a26', 'a27', 'a28', 'a29', 'a30', 'a31',
       'a32', 'a33', 'a34', 'a35', 'a36', 'a37', 'a38', 'a39', 'a40', 'a41'],
      dtype='object')
```

[70]:
```
# Recall(Sensitivity) == It is the ratio of correctly predicted positive␣
 ↪observations to all the observation in actual class
# Accuracy == ratio of correctly predicted observation to the total␣
 ↪observations
# Precision ==  Precision is the ratio of correctly predicted positive␣
 ↪observations to the total predicted positive observations
# F1 score - F1 Score is the weighted average of Precision and Recall
```

[88]:
```
print(confusion_matrix(y_test, rf_prediction))
```

```
[[  74    0    0    0    0    0    0    0    3    0    0    0    0    0
     0    0    0    0]
 [   0    1    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    5    0    0    0    0    0    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    2    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0  196    0    1    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0 2509    0    0    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    2    0   93    3    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    1 3973    0    0    1    1    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    7    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    2    0    0  186    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    8    0    0    1  213    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0  170
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0   53    0    0]
 [   0    0    0    0    0    0    0    0    3    0    0    0    0    0
     0    0   44    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]]
```

[95]:
```
y_test.value_counts()
```

[95]:
```
11    3976
9     2509
16     222
5      198
14     188
17     170
10      98
0       77
19      53
```

```
20      47
13       7
3        5
4        3
21       1
2        1
18       1
1        1
12       1
Name: a42, dtype: int64
```

[72]:
```python
from sklearn import metrics
```

[73]:
```python
print("Accuracy:",metrics.accuracy_score(y_test, rf_prediction))
```

```
Accuracy: 0.9957660756813972
```

[74]:
```python
X_list = list(X.columns)

# Get numerical feature importances
importances = list(rf_model.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(X, round(importance, 2)) for X, importance in␣
 ↪zip(X_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse␣
 ↪= True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in␣
 ↪feature_importances]
```

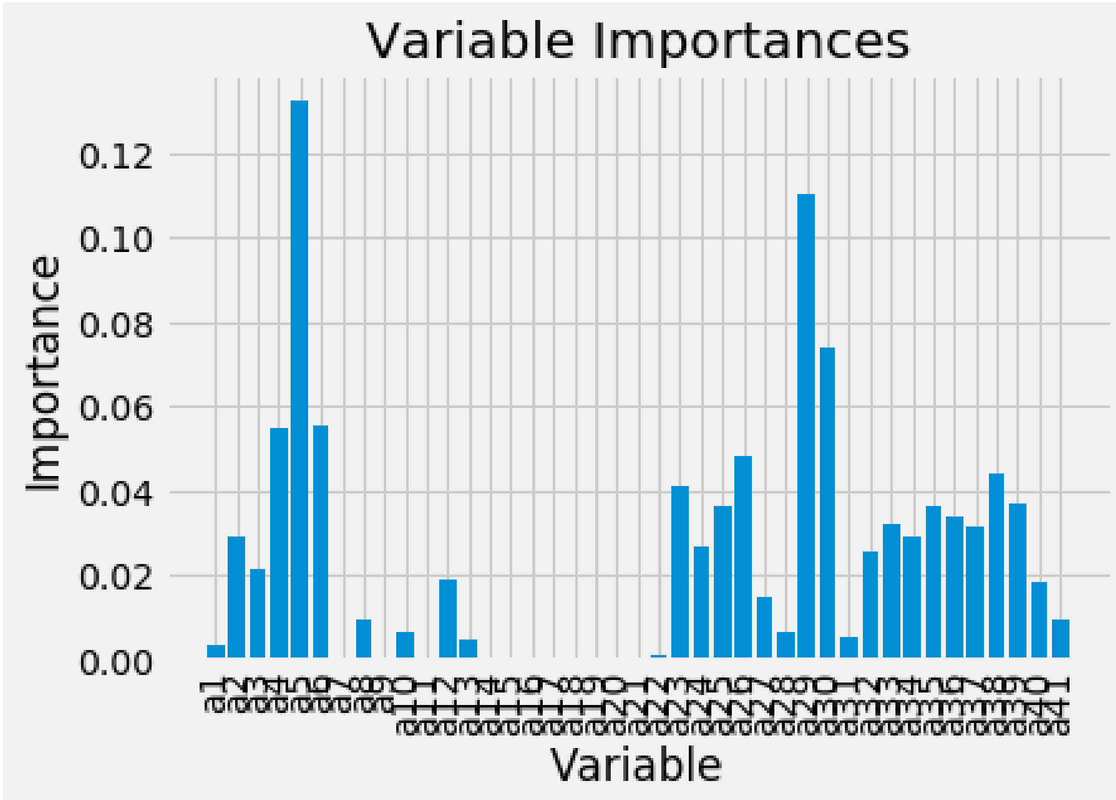
```
Variable: a5                   Importance: 0.13
Variable: a29                  Importance: 0.11
Variable: a30                  Importance: 0.07
Variable: a6                   Importance: 0.06
Variable: a4                   Importance: 0.05
Variable: a26                  Importance: 0.05
Variable: a23                  Importance: 0.04
Variable: a25                  Importance: 0.04
Variable: a35                  Importance: 0.04
Variable: a38                  Importance: 0.04
Variable: a39                  Importance: 0.04
Variable: a2                   Importance: 0.03
Variable: a24                  Importance: 0.03
Variable: a32                  Importance: 0.03
Variable: a33                  Importance: 0.03
Variable: a34                  Importance: 0.03
Variable: a36                  Importance: 0.03
```

```
Variable: a37                  Importance: 0.03
Variable: a3                   Importance: 0.02
Variable: a12                  Importance: 0.02
Variable: a40                  Importance: 0.02
Variable: a8                   Importance: 0.01
Variable: a10                  Importance: 0.01
Variable: a27                  Importance: 0.01
Variable: a28                  Importance: 0.01
Variable: a31                  Importance: 0.01
Variable: a41                  Importance: 0.01
Variable: a1                   Importance: 0.0
Variable: a7                   Importance: 0.0
Variable: a9                   Importance: 0.0
Variable: a11                  Importance: 0.0
Variable: a13                  Importance: 0.0
Variable: a14                  Importance: 0.0
Variable: a15                  Importance: 0.0
Variable: a16                  Importance: 0.0
Variable: a17                  Importance: 0.0
Variable: a18                  Importance: 0.0
Variable: a19                  Importance: 0.0
Variable: a20                  Importance: 0.0
Variable: a21                  Importance: 0.0
Variable: a22                  Importance: 0.0
```

[74]:
```
[None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
```

```
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None]
```

[75]:
```
plt.style.use("fivethirtyeight")

x_values = list(range(len(importances)))

plt.bar(x_values, importances, orientation = 'vertical')
plt.xticks(x_values, X_list, rotation = 'vertical')
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable␣
 ↪Importances')
plt.show()
```



[76]:
```python
X_new = version1[['a5', 'a8', 'a10', 'a27', 'a28', 'a31', 'a41', 'a29', 'a30',
    'a6', 'a4', 'a26', 'a23', 'a25', 'a35', 'a39', 'a38', 'a39', 'a2', 'a24',
    'a32', 'a33', 'a34', 'a36', 'a37']]
X_new.head()
```

[76]:

	a5	a8	a10	a27	a28	a31	a41	a29	a30	a6	a4	a26	a23	a25	\
0	491	0	0	0.0	0.0	0.00	0.00	1.00	0.00	0	9	0.0	2	0.0	
1	146	0	0	0.0	0.0	0.00	0.00	0.08	0.15	0	9	0.0	13	0.0	
2	0	0	0	0.0	0.0	0.00	0.00	0.05	0.07	0	5	1.0	123	1.0	
3	232	0	0	0.0	0.0	0.00	0.01	1.00	0.00	8153	9	0.2	5	0.2	
4	199	0	0	0.0	0.0	0.09	0.00	1.00	0.00	420	9	0.0	30	0.0	

	a35	a39	a38	a39	a2	a24	a32	a33	a34	a36	a37
0	0.03	0.00	0.00	0.00	1	2	150	25	0.17	0.17	0.00
1	0.60	0.00	0.00	0.00	2	1	255	1	0.00	0.88	0.00
2	0.05	1.00	1.00	1.00	1	6	255	26	0.10	0.00	0.00
3	0.00	0.01	0.03	0.01	1	5	30	255	1.00	0.03	0.04
4	0.00	0.00	0.00	0.00	1	32	255	255	1.00	0.00	0.00

[132]:
```python
X_newtrain, X_newtest, y_newtrain, y_newtest = train_test_split(X_new, y,
    test_size=0.3, random_state = 3)
```

[133]:
```python
rf_model_new = RandomForestClassifier(n_estimators = 150, oob_score = True,
    random_state = 3)
```

[134]:
```python
rf_model_new.fit(X_newtrain, y_train)
```

[134]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=True, random_state=3, verbose=0,
                       warm_start=False)
```

[135]:
```python
rf_newprediction = rf_model_new.predict(X_newtest)
```

[136]:
```python
print("Accuracy:",metrics.accuracy_score(y_test, rf_newprediction))
```

```
Accuracy: 0.9955014554114845
```

[137]:
```python
print(confusion_matrix(y_test, rf_newprediction))
```

```
[[  75    0    0    0    0    0    0    0    2    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    5    0    0    0    0    0    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    1    0    0    0    2    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0  196    0    1    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0 2509    0    0    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    2    0   92    4    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0 3974    0    0    1    1    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    7    0    0    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    2    0    0  185    1    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    9    0    0    1  212    0
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0  170
     0    0    0    0]
 [   0    0    0    0    0    0    0    0    1    0    0    0    0    0
     0    0    0    0]
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0   53    0    0]
[   0    0    0    0    0    0    0    0    2    0    0    0    0    0
    0    0   45    0]
[   0    0    0    0    0    0    0    0    1    0    0    0    0    0
    0    0    0    0]]
```

[138]:
```python
print(classification_report(y_test, rf_newprediction))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        77
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         1
           3       1.00      1.00      1.00         5
           4       1.00      0.33      0.50         3
           5       0.99      0.99      0.99       198
           9       1.00      1.00      1.00      2509
          10       0.99      0.94      0.96        98
          11       0.99      1.00      1.00      3976
          12       0.00      0.00      0.00         1
          13       1.00      1.00      1.00         7
          14       0.99      0.98      0.99       188
          16       0.99      0.95      0.97       222
          17       1.00      1.00      1.00       170
          18       0.00      0.00      0.00         1
          19       1.00      1.00      1.00        53
          20       1.00      0.96      0.98        47
          21       0.00      0.00      0.00         1

    accuracy                           1.00      7558
   macro avg       0.72      0.67      0.69      7558
weighted avg       0.99      1.00      1.00      7558


/opt/conda/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Out of bag error

[140]:
```python
print('Score: ', rf_model_new.score(X_newtrain, y_train))
```

```
Score:  1.0
```

[142]:
```python
print('Score: ', rf_model_new.score(X_newtest, y_test))
```

```
Score:  0.9955014554114845
```

[ ]:

```

```