Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings(action= 'ignore')
```

Importing Dataset & Some Visualization Using Correlation HeatMap,Box Plot, DistPlot etc., and finding the Skewness, to know the details in the Data

In [2]:
```
dataset=pd.read_csv("C:\\Users\\mishr\\OneDrive\\Desktop\\PROJECT\\Dataset_1.csv")
dataset
```

Out[2]:

	URL	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleId.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

In [3]:
```
dataset.isna().any().any()  # There is no missing data in this dataset .
# To Check the missing values in a particular row or column       pd.isna(dataset["R
```

Out[3]:
```
False
```

In [4]:
```
data=dataset.columns
data
```

Out[4]:
```
Index(['                                                                  UR
L',
       'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at',
       'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore', 'nb_tilde',
       'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma',
       'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com',
       'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',
       'ratio_digits_host', 'punycode', 'port', 'tld_in_path',
       'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',
       'prefix_suffix', 'random_domain', 'shortening_service',
```

```
       'path_extension', 'nb_redirection', 'nb_external_redirection',
       'length_words_raw', 'char_repeat', 'shortest_words_raw',
       'shortest_word_host', 'shortest_word_path', 'longest_words_raw',
       'longest_word_host', 'longest_word_path', 'avg_words_raw',
       'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',
       'brand_in_subdomain', 'brand_in_path', 'suspecious_tld',
       'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks',
       'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS',
       'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors',
       'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags',
       'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe',
       'popup_window', 'safe_anchor', 'onmouseover', 'right_clic',
       'empty_title', 'domain_in_title', 'domain_with_copyright',
       'whois_registered_domain', 'domain_registration_length', 'domain_age',
       'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status'],
      dtype='object')
```

In [5]:

```python
for i in data:
    print(i)
```

```
                                                                        URL
length_url
length_hostname
ip
nb_dots
nb_hyphens
nb_at
nb_qm
nb_and
nb_or
nb_eq
nb_underscore
nb_tilde
nb_percent
nb_slash
nb_star
nb_colon
nb_comma
nb_semicolumn
nb_dollar
nb_space
nb_www
nb_com
nb_dslash
http_in_path
https_token
ratio_digits_url
ratio_digits_host
punycode
port
tld_in_path
tld_in_subdomain
abnormal_subdomain
nb_subdomains
prefix_suffix
random_domain
shortening_service
path_extension
nb_redirection
nb_external_redirection
length_words_raw
char_repeat
shortest_words_raw
shortest_word_host
shortest_word_path
longest_words_raw
longest_word_host
longest_word_path
avg_words_raw
```

```
avg_word_host
avg_word_path
phish_hints
domain_in_brand
brand_in_subdomain
brand_in_path
suspecious_tld
statistical_report
nb_hyperlinks
ratio_intHyperlinks
ratio_extHyperlinks
ratio_nullHyperlinks
nb_extCSS
ratio_intRedirection
ratio_extRedirection
ratio_intErrors
ratio_extErrors
login_form
external_favicon
links_in_tags
submit_email
ratio_intMedia
ratio_extMedia
sfh
iframe
popup_window
safe_anchor
onmouseover
right_clic
empty_title
domain_in_title
domain_with_copyright
whois_registered_domain
domain_registration_length
domain_age
web_traffic
dns_record
google_index
page_rank
status
```

In [6]:
```
dataset.corr()
```

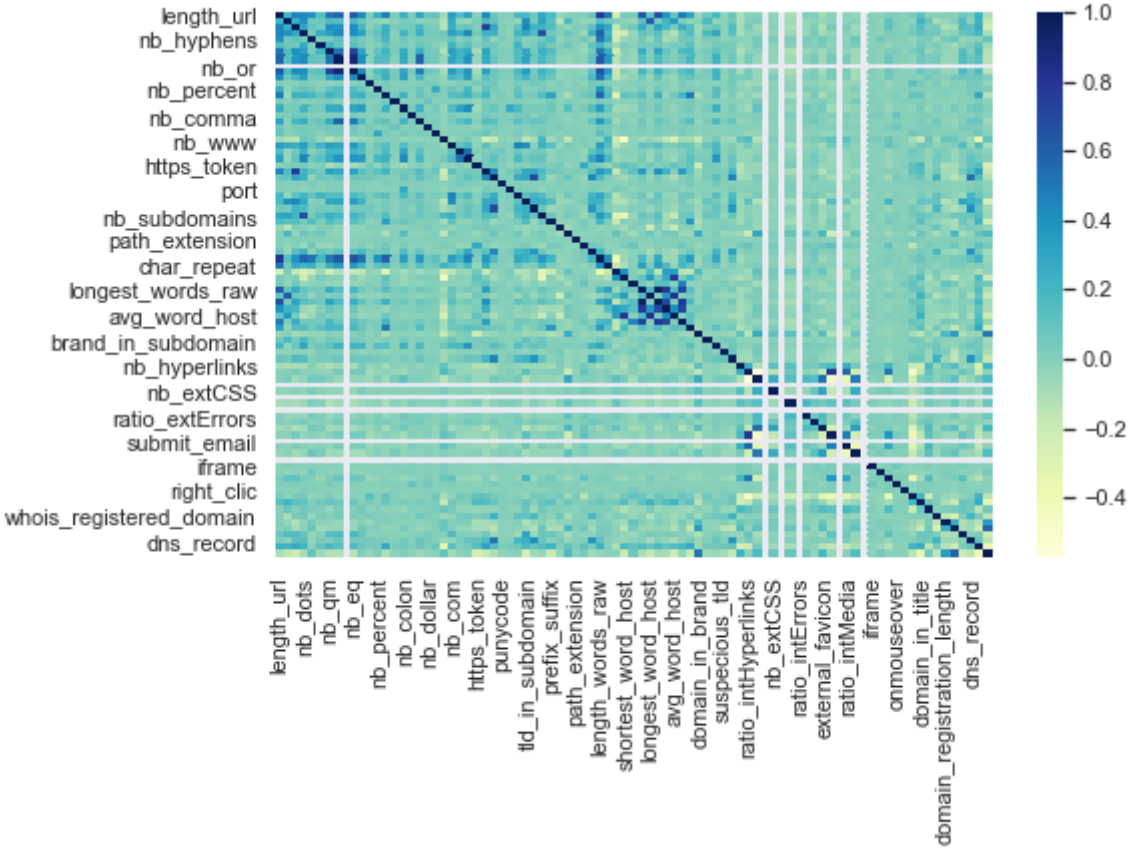
Out[6]:

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_q
length_url	1.000000	0.223025	0.453961	0.443589	0.399564	0.150739	0.5209
length_hostname	0.223025	1.000000	0.252013	0.408956	0.057702	0.071793	0.1624
ip	0.453961	0.252013	1.000000	0.288398	0.109860	0.059401	0.4054
nb_dots	0.443589	0.408956	0.288398	1.000000	0.045099	0.263283	0.3474
nb_hyphens	0.399564	0.057702	0.109860	0.045099	1.000000	0.018770	0.0368
...	...	...	...	...	...	...	
domain_age	-0.006798	0.013854	-0.077020	-0.007818	0.080104	-0.067334	-0.0456
web_traffic	0.072205	0.163238	0.167930	0.087969	-0.041464	-0.009459	0.1437
dns_record	0.023357	-0.023344	0.127823	0.126659	-0.031477	0.031611	0.0094
google_index	0.236395	0.213990	0.270743	0.209616	-0.018828	0.113217	0.2012
page_rank	-0.102582	-0.159342	-0.218968	-0.097312	0.104341	-0.066356	-0.1238

87 rows × 87 columns

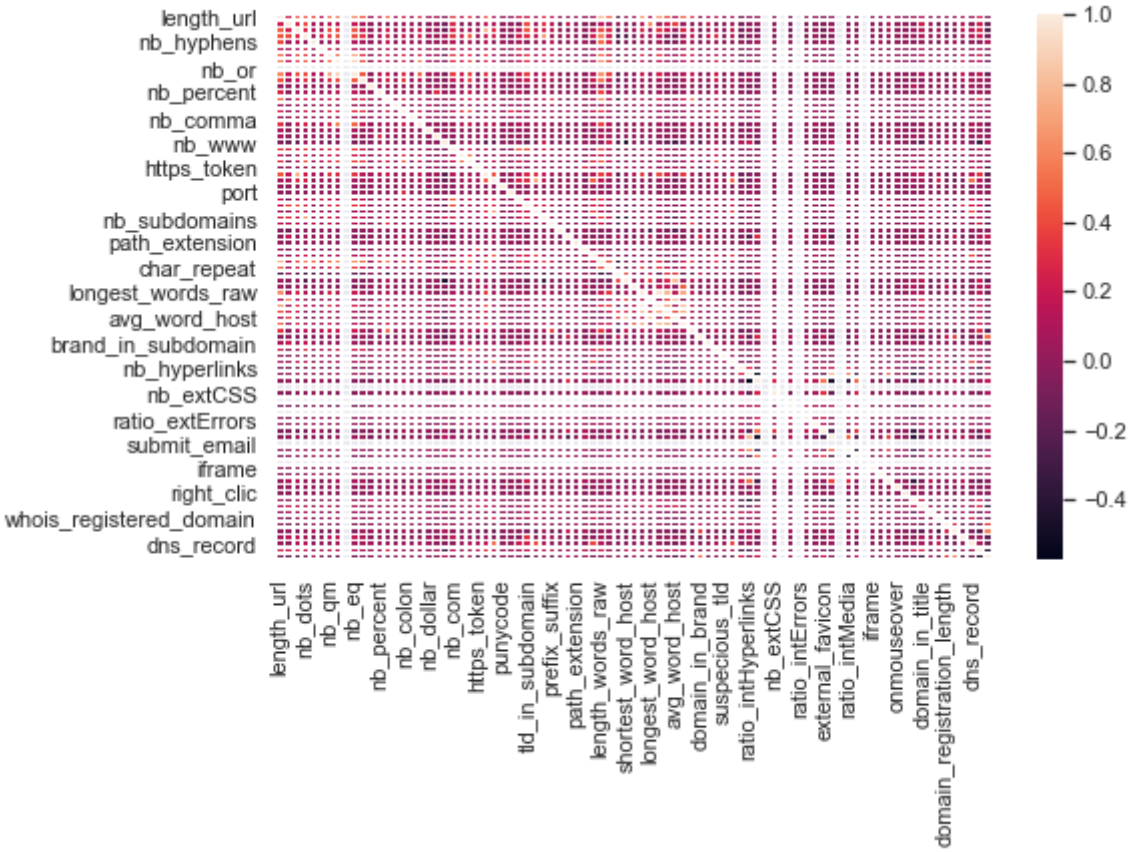
In [7]:
```
sns.set(rc={"figure.figsize":(8,5)})
sns.heatmap(dataset.corr(),cmap="YlGnBu")
```

Out[7]: <AxesSubplot:>



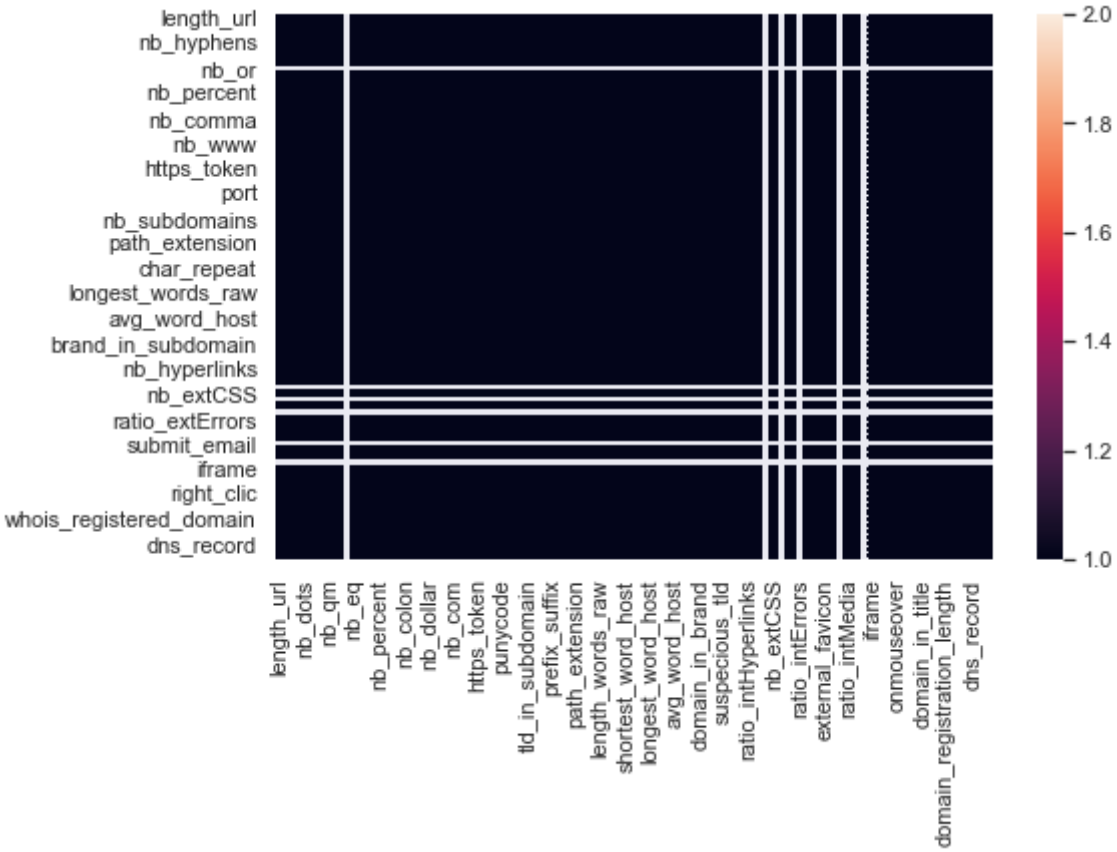
In [8]:
```
sns.heatmap(dataset.corr(),linewidths=0.7)
```

Out[8]: <AxesSubplot:>



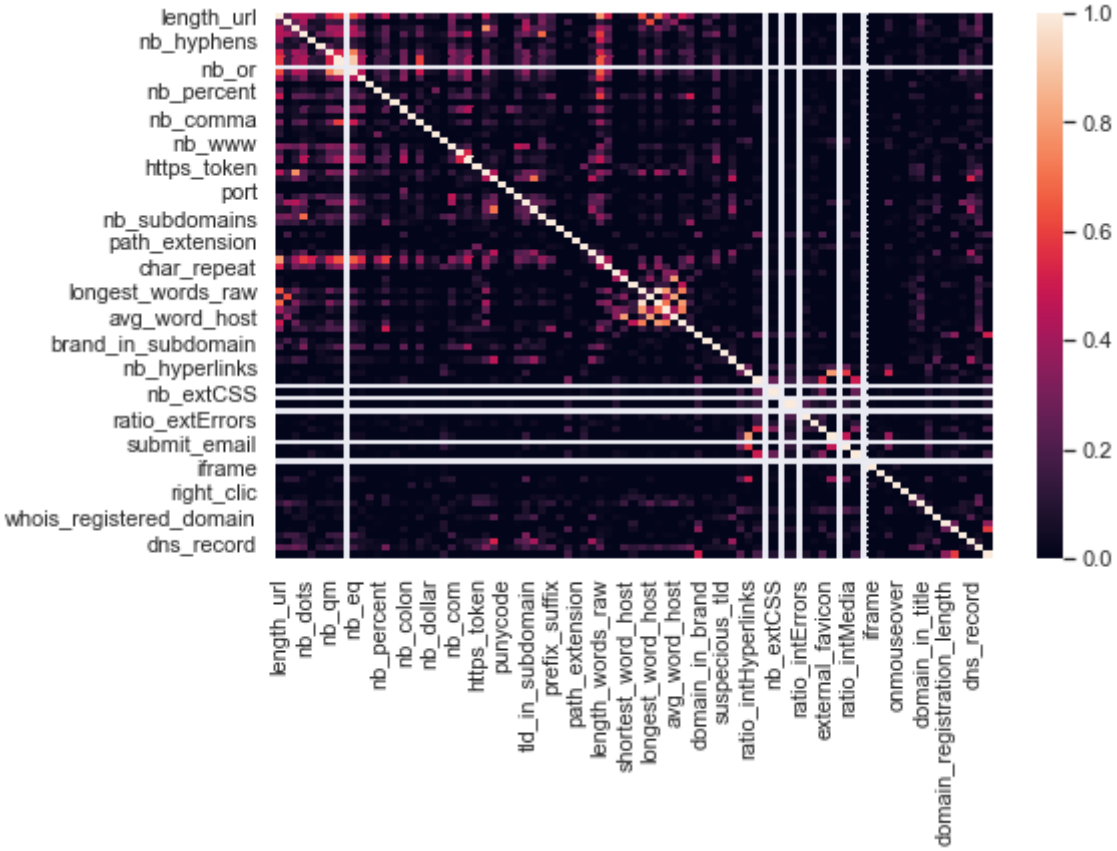
In [9]:
```
sns.heatmap(dataset.corr(),vmin=1,vmax=2)
```

Out[9]: <AxesSubplot:>



In [10]:
```
sns.heatmap(dataset.corr(),vmin=0,vmax=1)
```

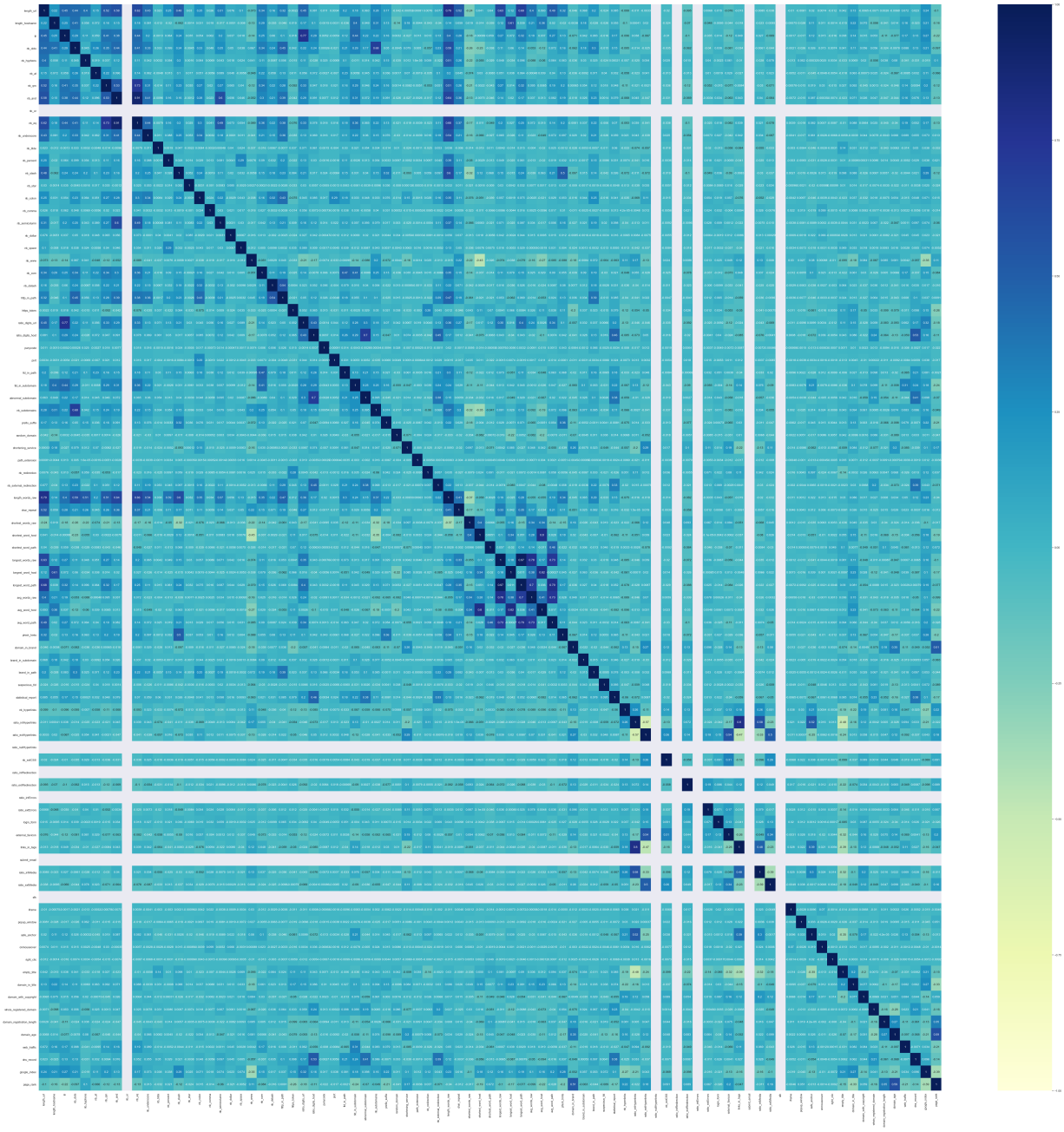
Out[10]: <AxesSubplot:>



In [11]:
```
sns.set(rc={"figure.figsize":(80,79)})
```

```
sns.heatmap(dataset.corr(),cmap="YlGnBu",annot=True,vmin=-1,vmax=1)
```

Out[11]: <AxesSubplot:>



In [12]:
```
new_col=dataset.iloc[:,1:-1]
new_col
```

Out[12]:

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	nb_e
0	37	19	0	3	0	0	0	0	0	
1	77	23	1	1	0	0	0	0	0	
2	126	50	1	4	1	0	1	2	0	
3	18	11	0	2	0	0	0	0	0	
4	55	15	0	2	2	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
11425	45	17	0	2	0	0	0	0	0	
11426	84	18	0	5	0	1	1	0	0	

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	nb_e
11427	105	16	1	2	6	0	1	0	0	
11428	38	30	0	2	0	0	0	0	0	
11429	477	14	1	24	0	1	1	9	0	

11430 rows × 87 columns

In [13]:
```
new_col.columns
```

Out[13]:
```
Index(['length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at',
       'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore', 'nb_tilde',
       'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma',
       'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com',
       'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',
       'ratio_digits_host', 'punycode', 'port', 'tld_in_path',
       'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',
       'prefix_suffix', 'random_domain', 'shortening_service',
       'path_extension', 'nb_redirection', 'nb_external_redirection',
       'length_words_raw', 'char_repeat', 'shortest_words_raw',
       'shortest_word_host', 'shortest_word_path', 'longest_words_raw',
       'longest_word_host', 'longest_word_path', 'avg_words_raw',
       'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',
       'brand_in_subdomain', 'brand_in_path', 'suspecious_tld',
       'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks',
       'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS',
       'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors',
       'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags',
       'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe',
       'popup_window', 'safe_anchor', 'onmouseover', 'right_clic',
       'empty_title', 'domain_in_title', 'domain_with_copyright',
       'whois_registered_domain', 'domain_registration_length', 'domain_age',
       'web_traffic', 'dns_record', 'google_index', 'page_rank'],
      dtype='object')
```

In [14]:
```
for col in dataset:
    print(col)
```

```
                                                                          URL
length_url
length_hostname
ip
nb_dots
nb_hyphens
nb_at
nb_qm
nb_and
nb_or
nb_eq
nb_underscore
nb_tilde
nb_percent
nb_slash
nb_star
nb_colon
nb_comma
nb_semicolumn
nb_dollar
nb_space
nb_www
nb_com
nb_dslash
http_in_path
https_token
```

```
ratio_digits_url
ratio_digits_host
punycode
port
tld_in_path
tld_in_subdomain
abnormal_subdomain
nb_subdomains
prefix_suffix
random_domain
shortening_service
path_extension
nb_redirection
nb_external_redirection
length_words_raw
char_repeat
shortest_words_raw
shortest_word_host
shortest_word_path
longest_words_raw
longest_word_host
longest_word_path
avg_words_raw
avg_word_host
avg_word_path
phish_hints
domain_in_brand
brand_in_subdomain
brand_in_path
suspecious_tld
statistical_report
nb_hyperlinks
ratio_intHyperlinks
ratio_extHyperlinks
ratio_nullHyperlinks
nb_extCSS
ratio_intRedirection
ratio_extRedirection
ratio_intErrors
ratio_extErrors
login_form
external_favicon
links_in_tags
submit_email
ratio_intMedia
ratio_extMedia
sfh
iframe
popup_window
safe_anchor
onmouseover
right_clic
empty_title
domain_in_title
domain_with_copyright
whois_registered_domain
domain_registration_length
domain_age
web_traffic
dns_record
google_index
page_rank
status
```

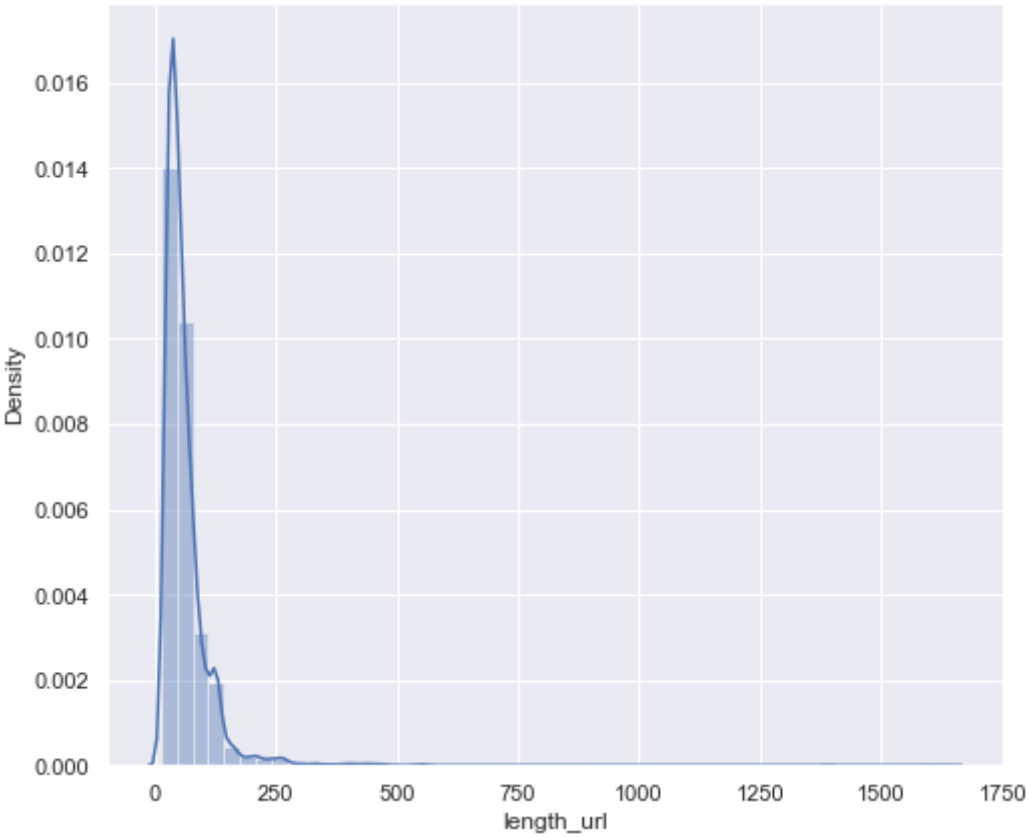
In [15]:

```
# skewness
sns.set(rc={"figure.figsize":(8,7)})
from scipy.stats import skew
for col in new_col:
    print(col)
```

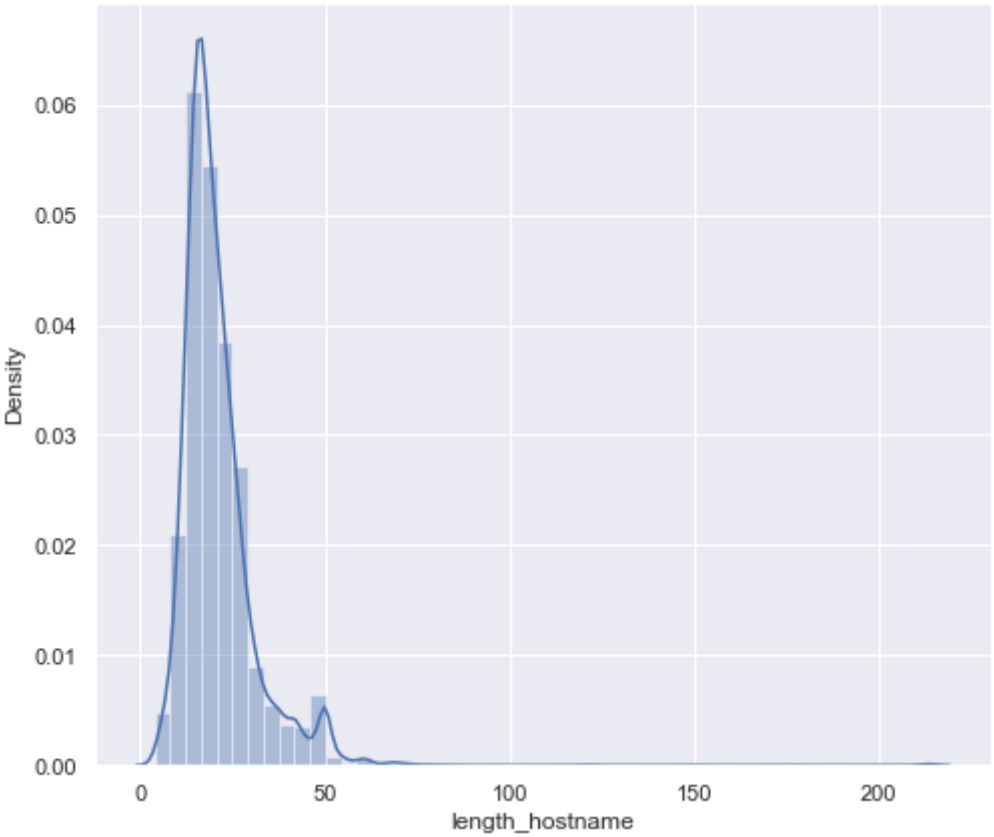
```
    print(skew(new_col[col]))

    plt.figure()
    sns.distplot(new_col[col])
    plt.show()
```

```
length_url
8.084128701285604
```

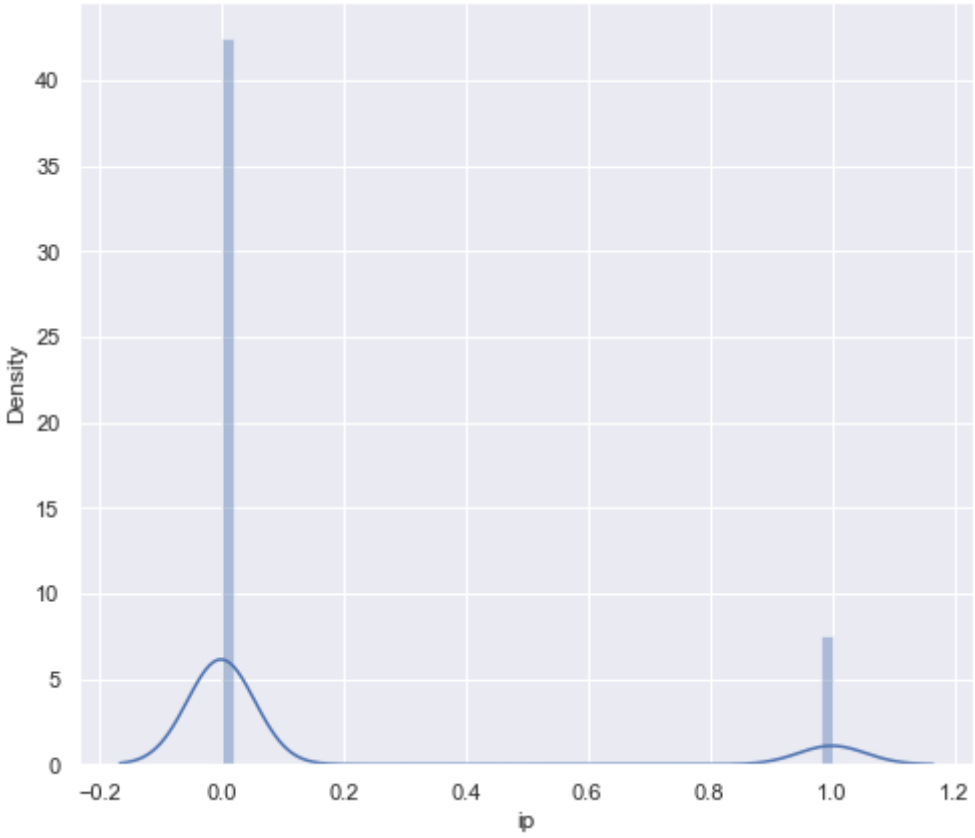



```
length_hostname
5.159400636003864
```

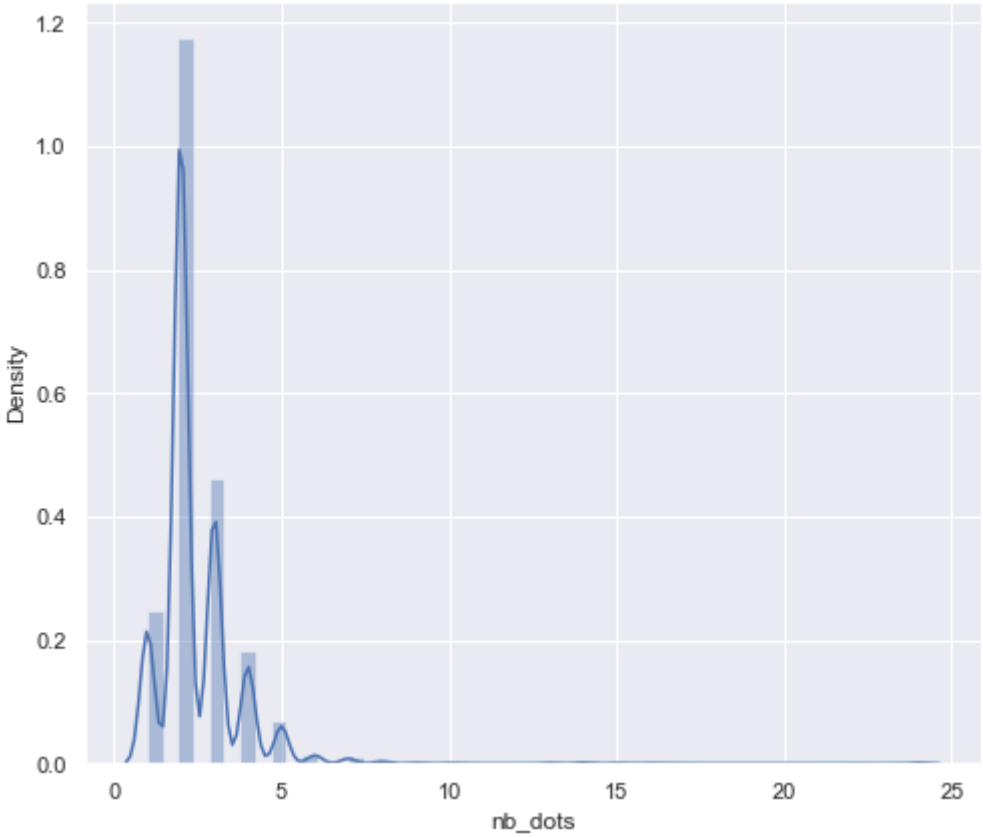



```
ip
```

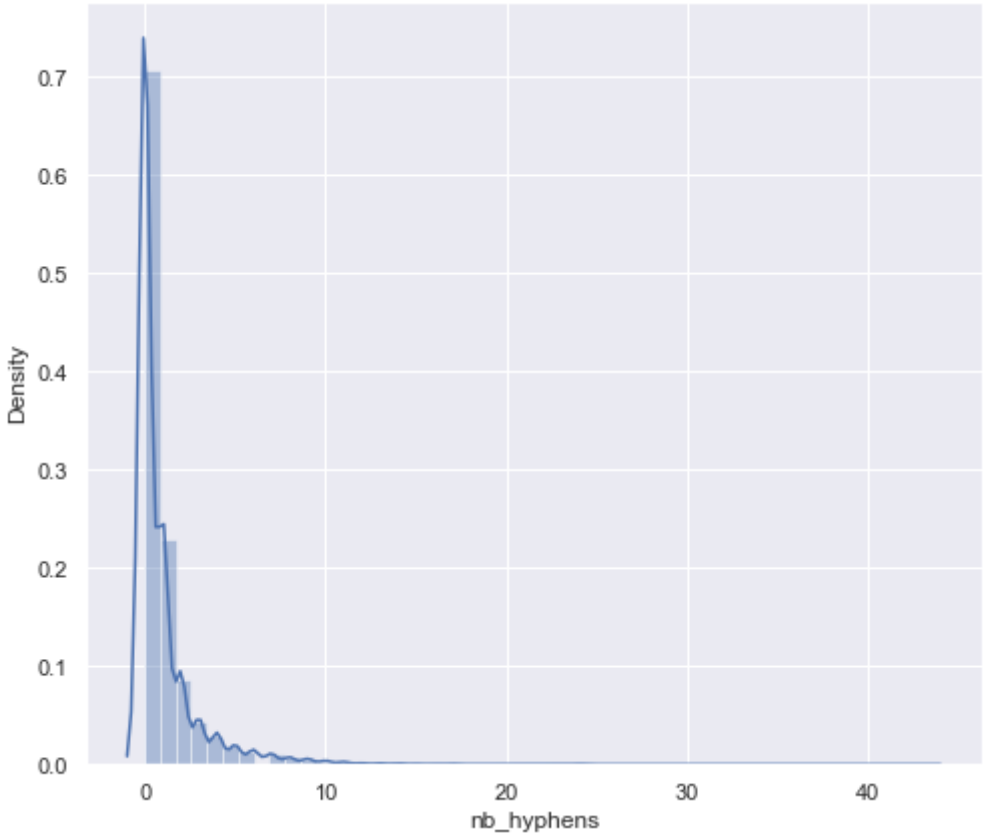
1.954161129817969



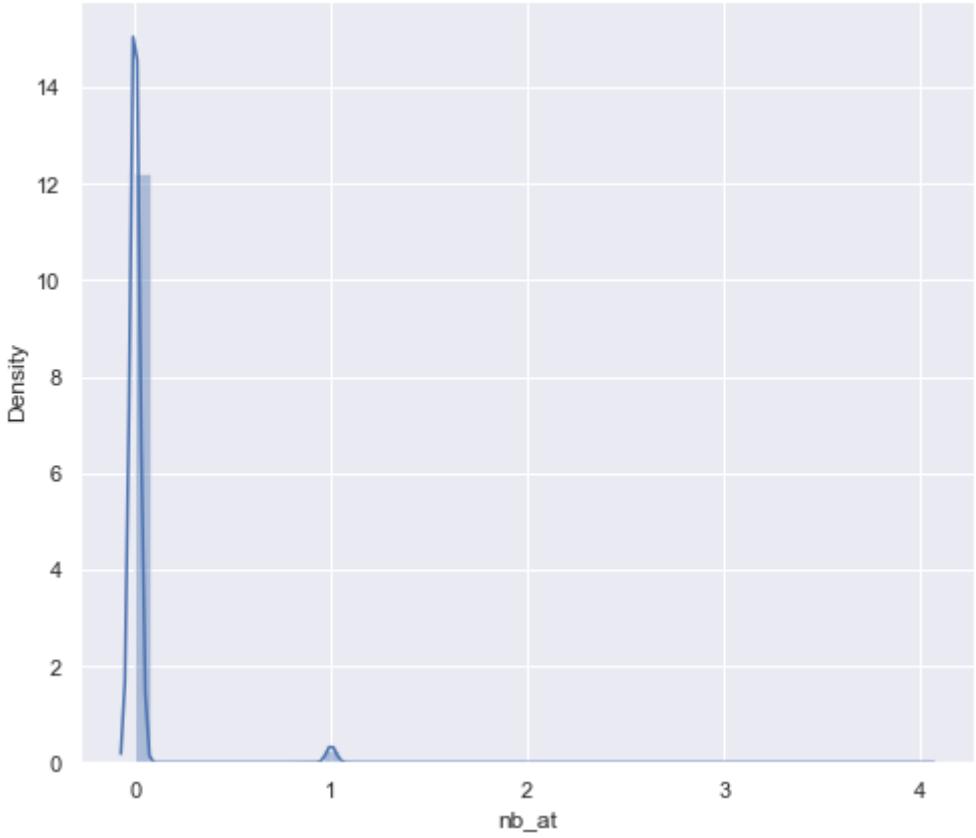
nb_dots
5.717366272299363



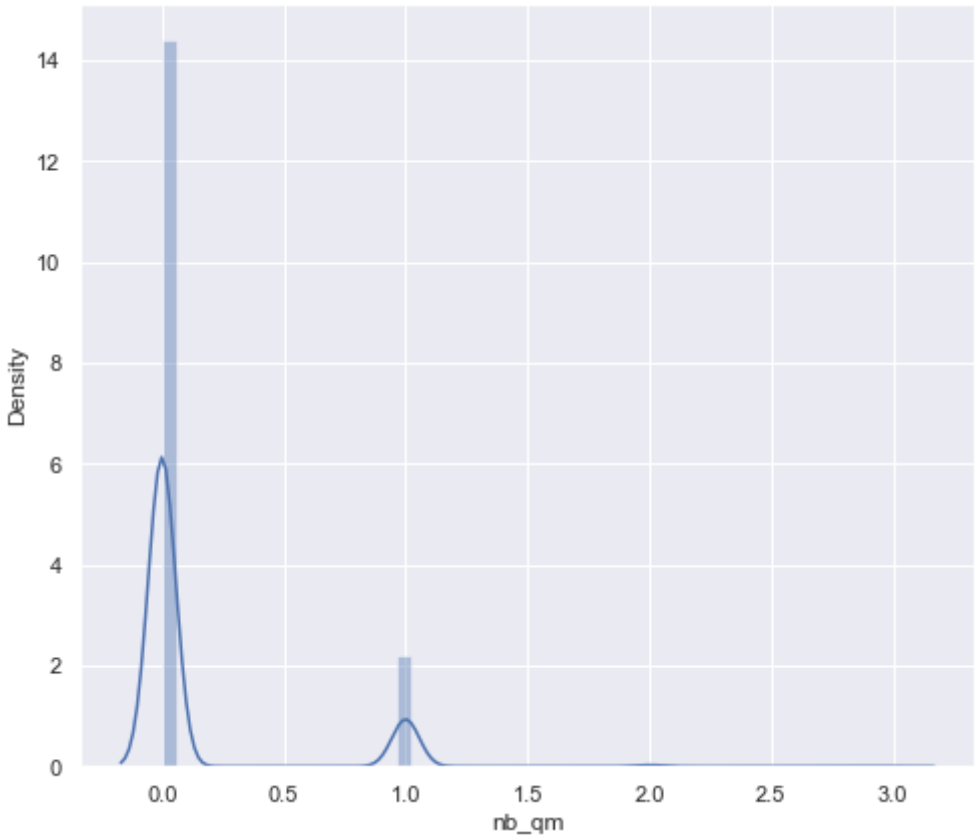
nb_hyphens
4.694623150416177



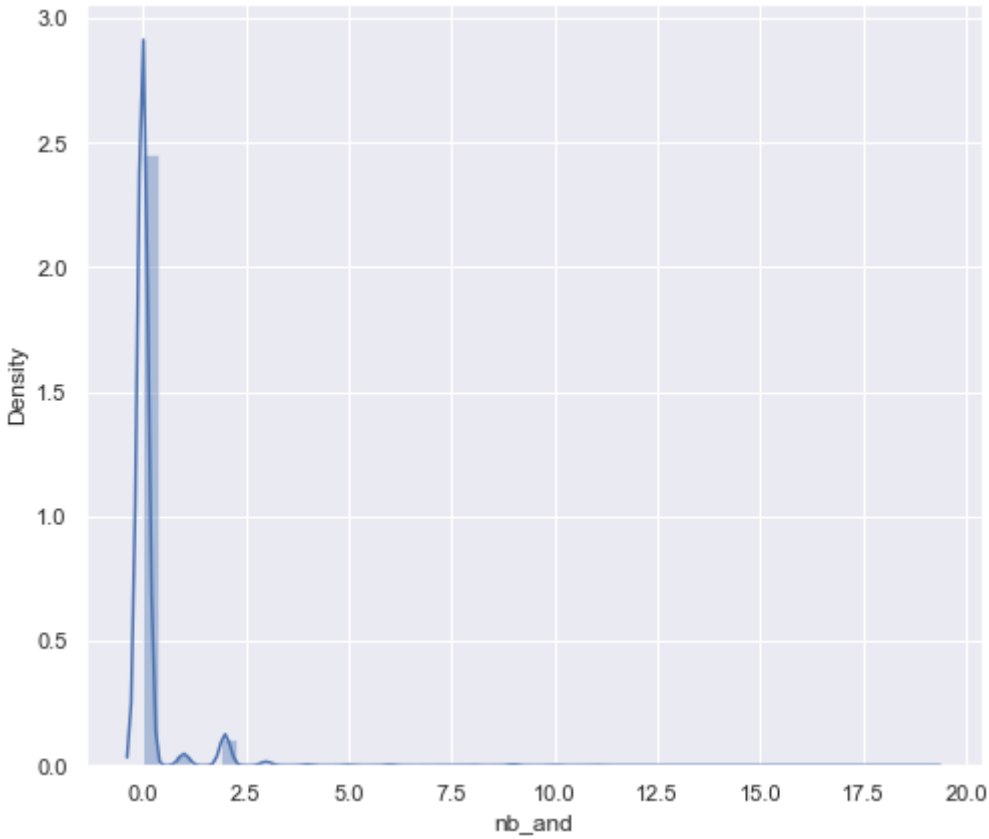
nb_at
8.271806951687719



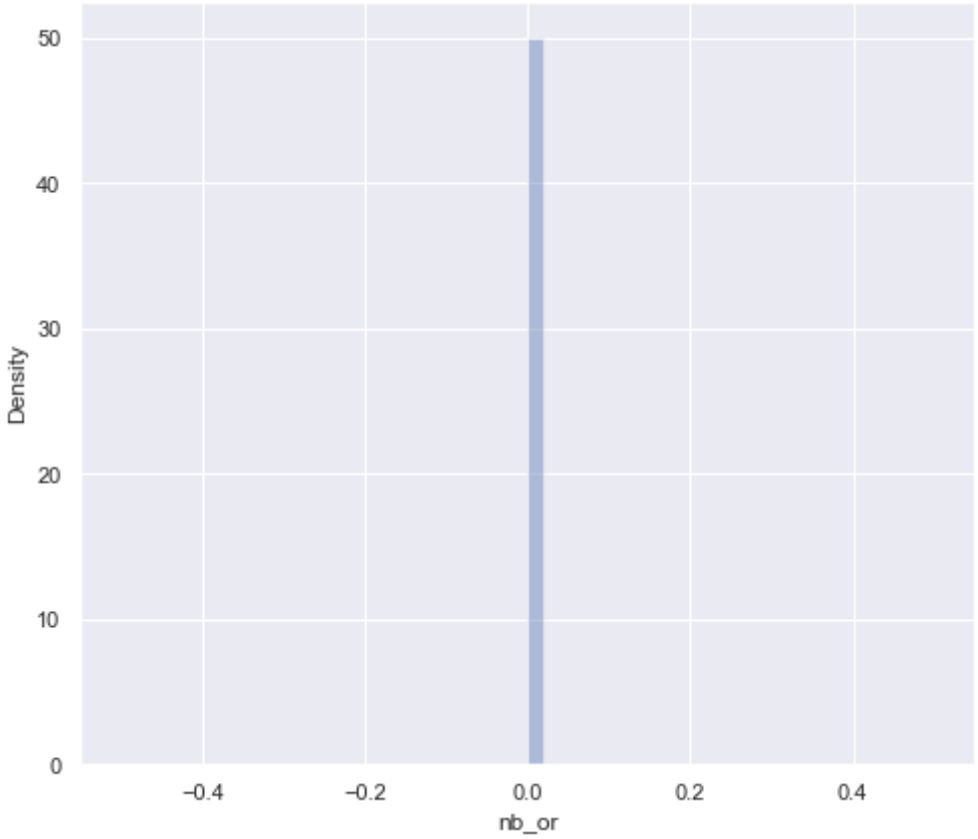
nb_qm
2.4884108813674986



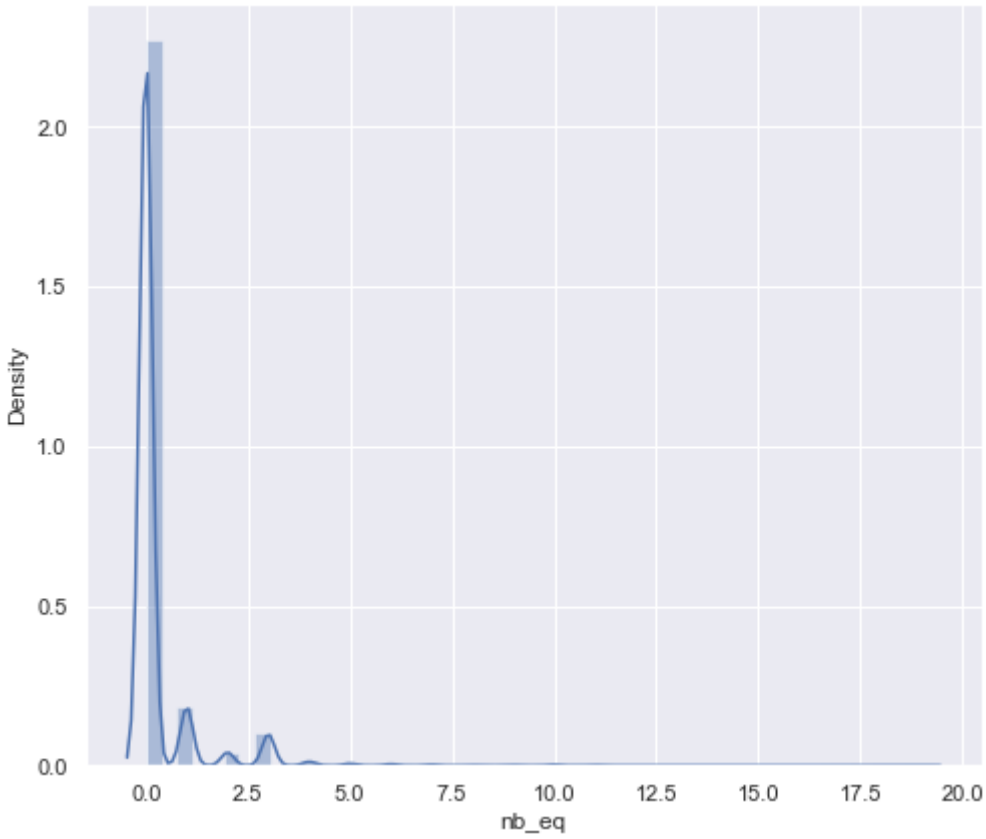

```
nb_and
9.724018326611633
```



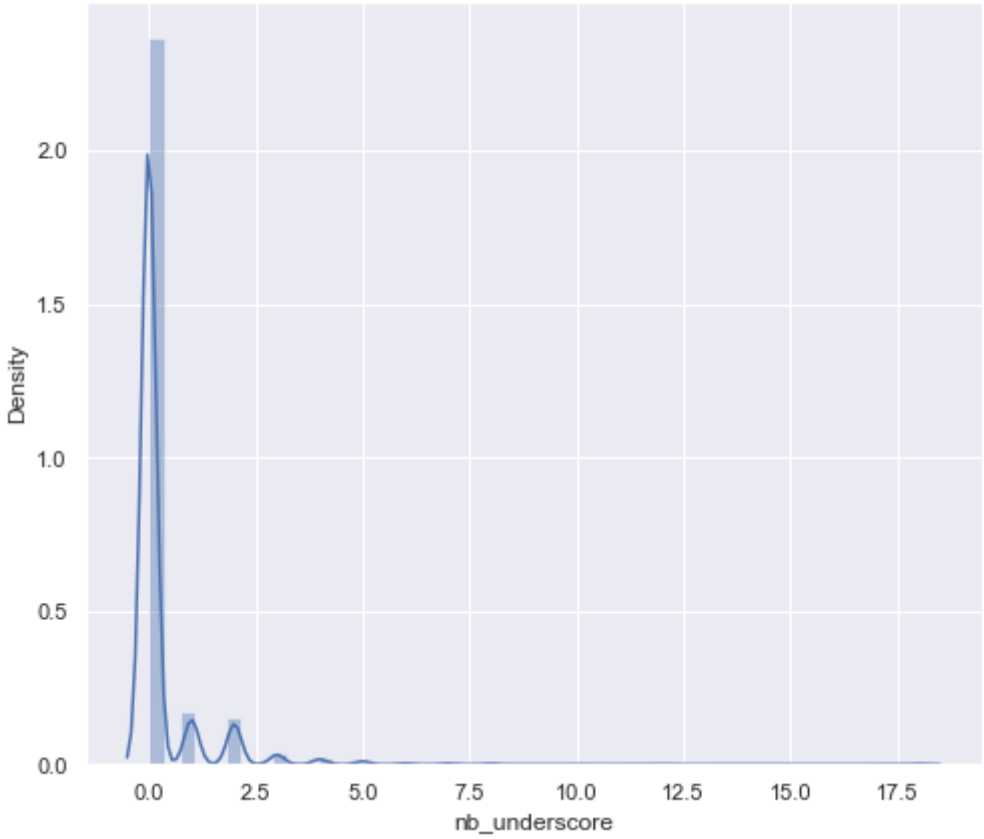

```
nb_or
0.0
```



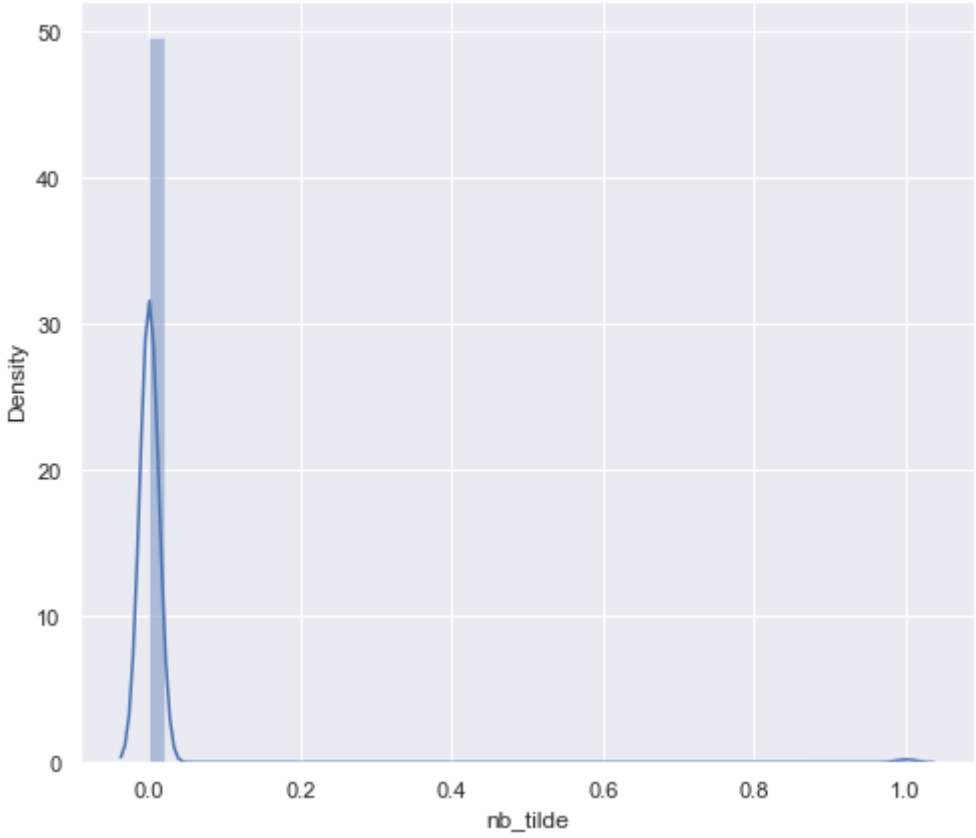
nb_eq
6.52917880406454



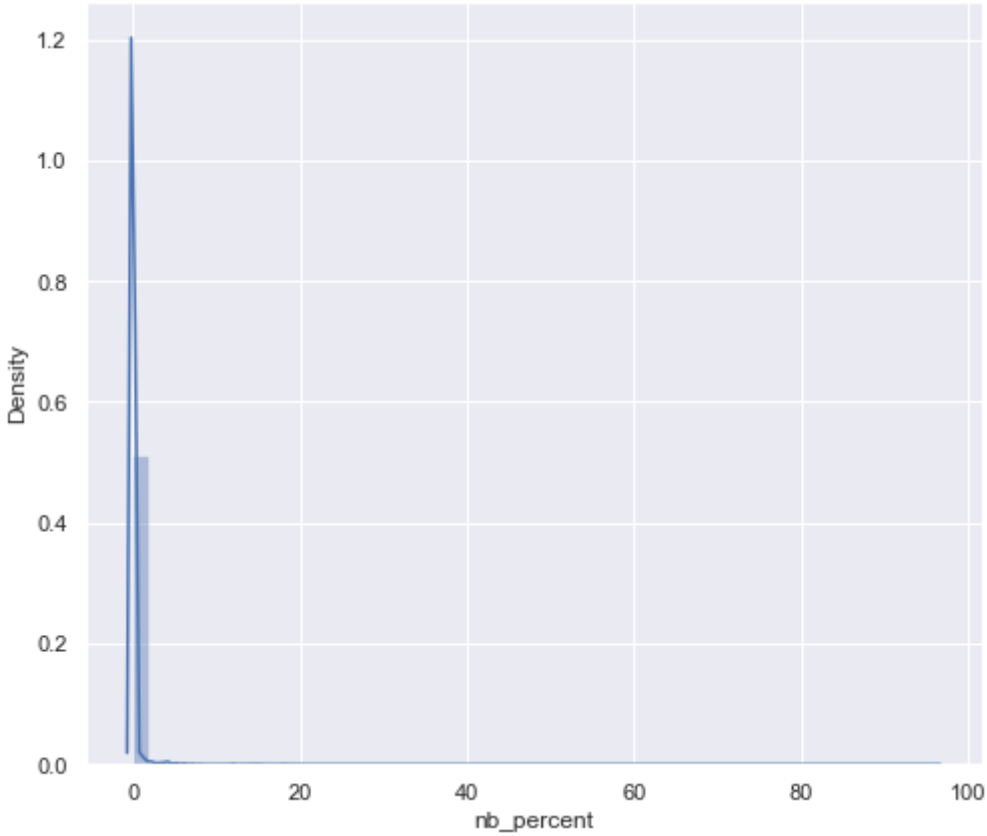
nb_underscore
7.2649716358768615



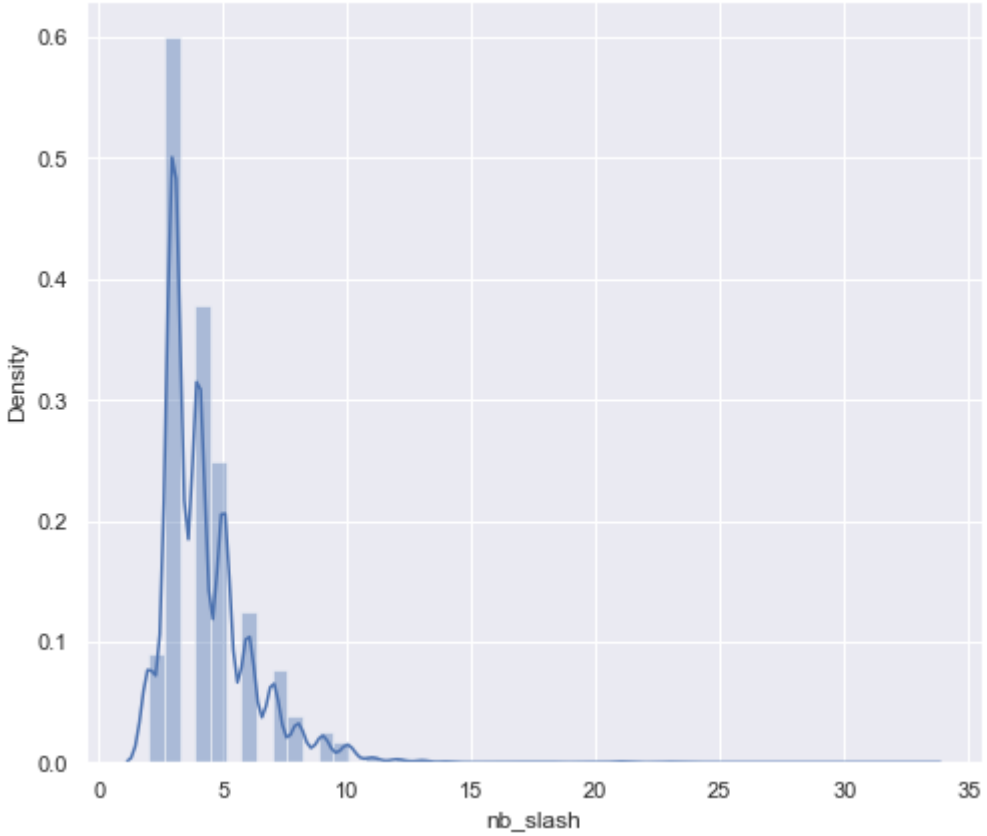
nb_tilde
12.140899081960224



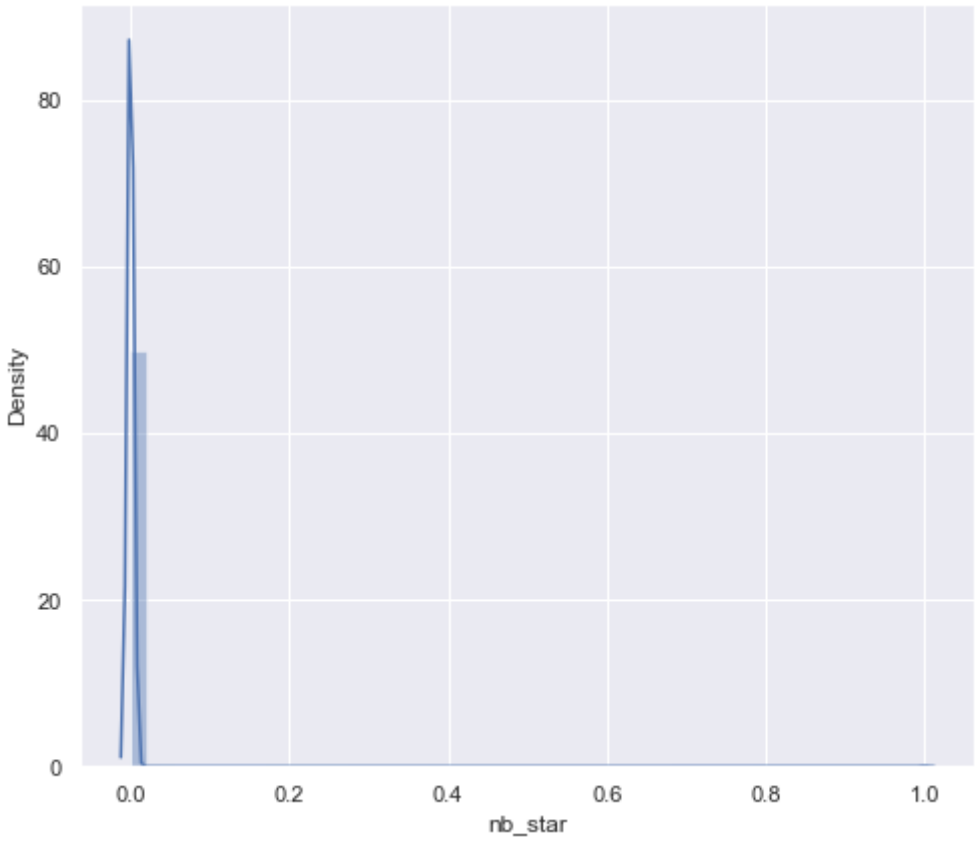
nb_percent
35.419469856992016



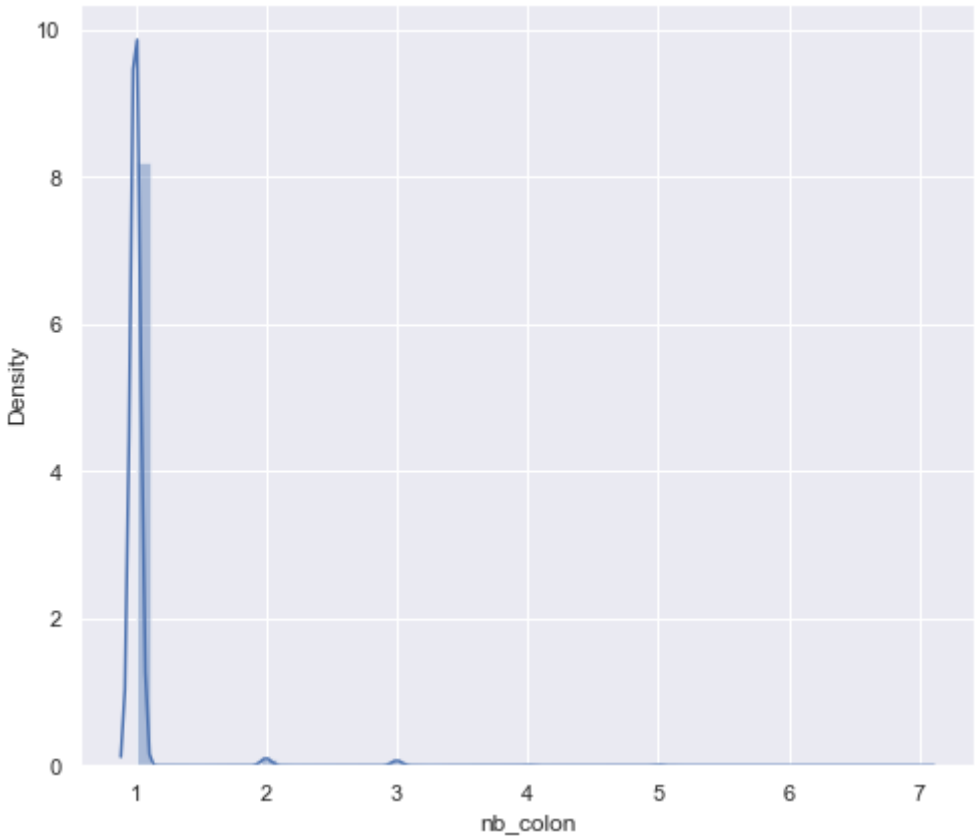
nb_slash
2.7309534049522513



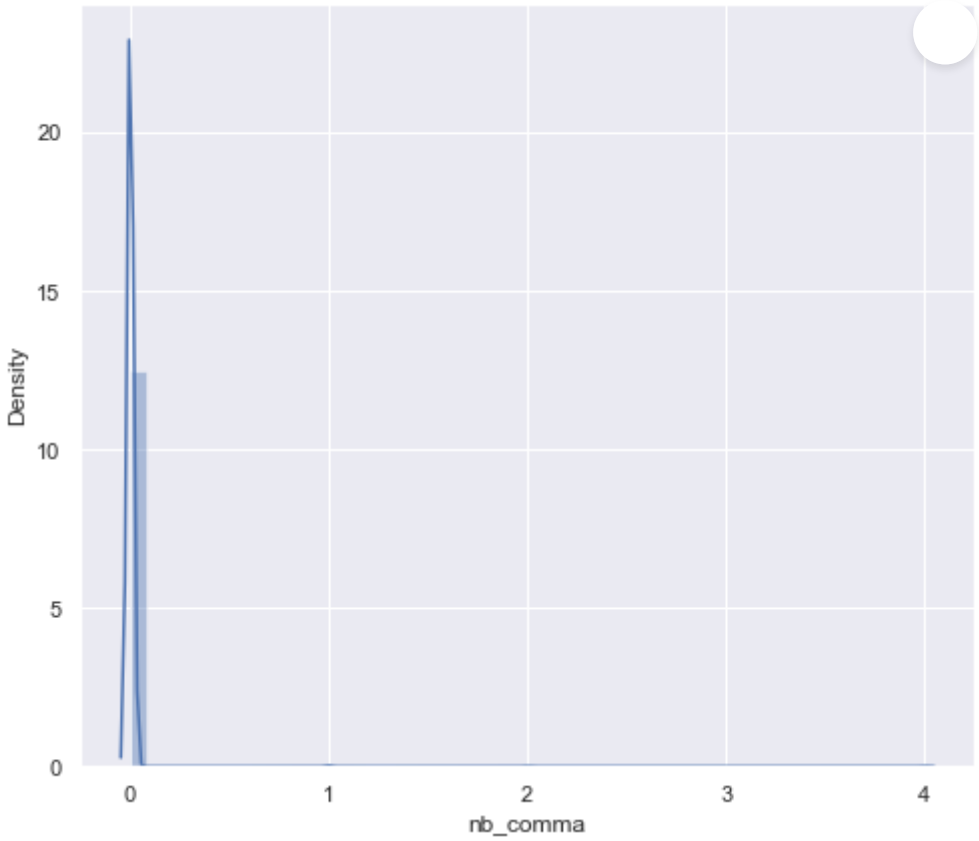
nb_star
37.759114136890595



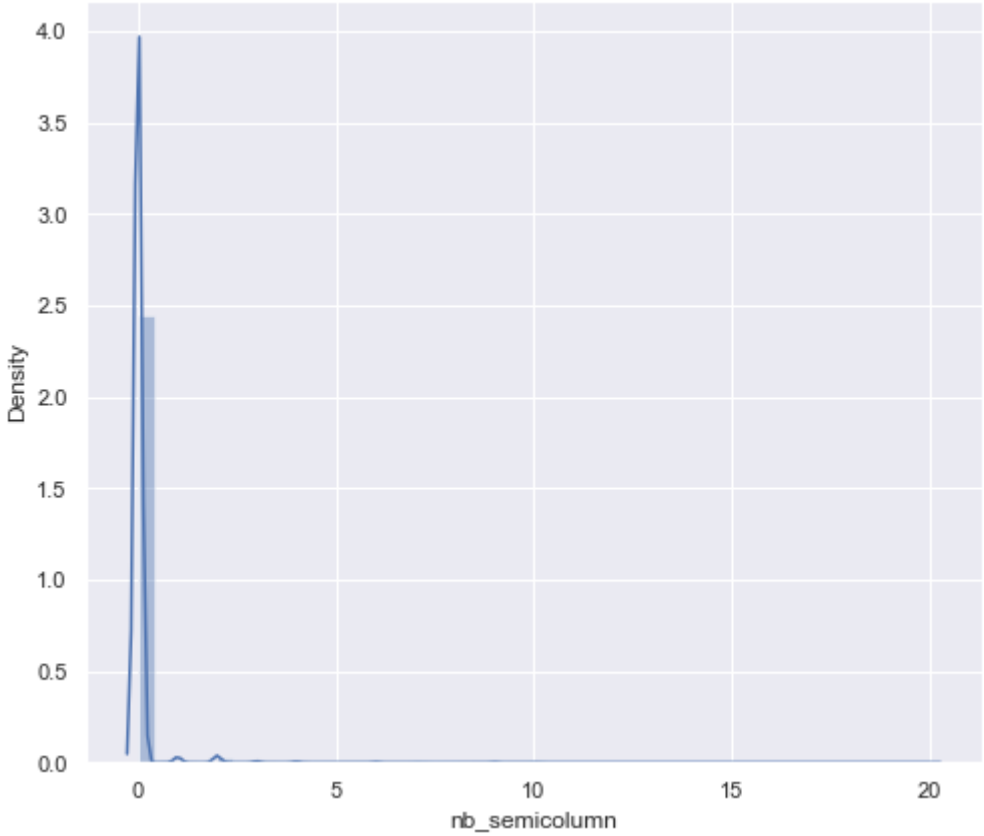
nb_colon
11.381643808000376



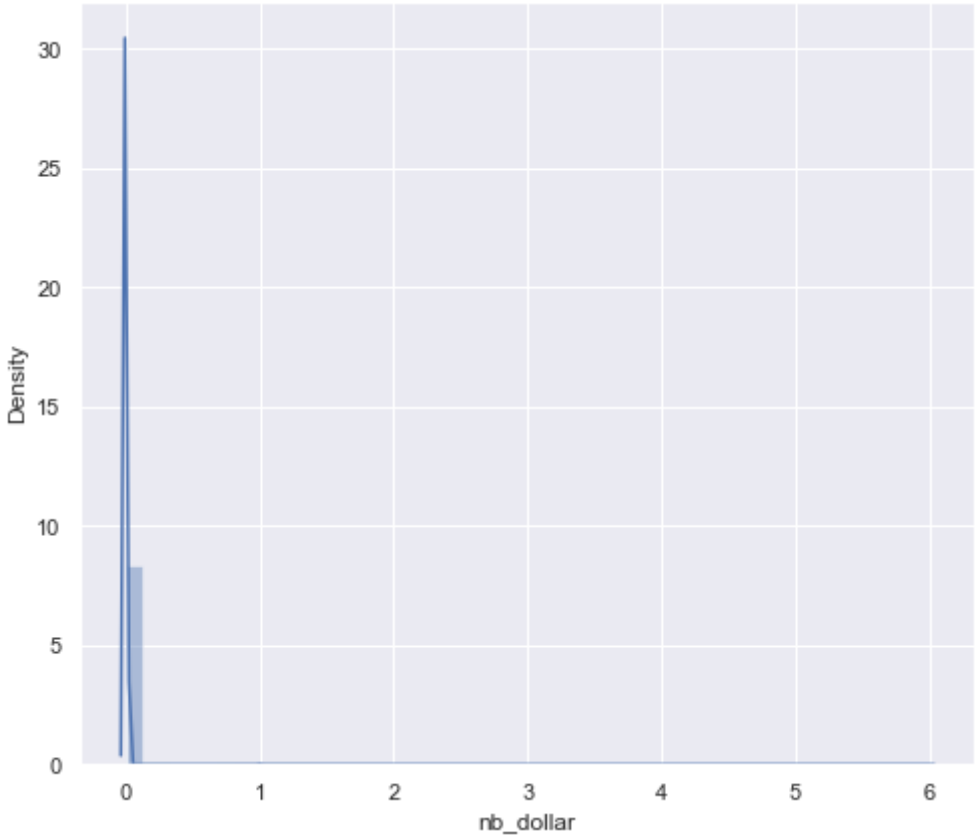
nb_comma
31.690715910692905



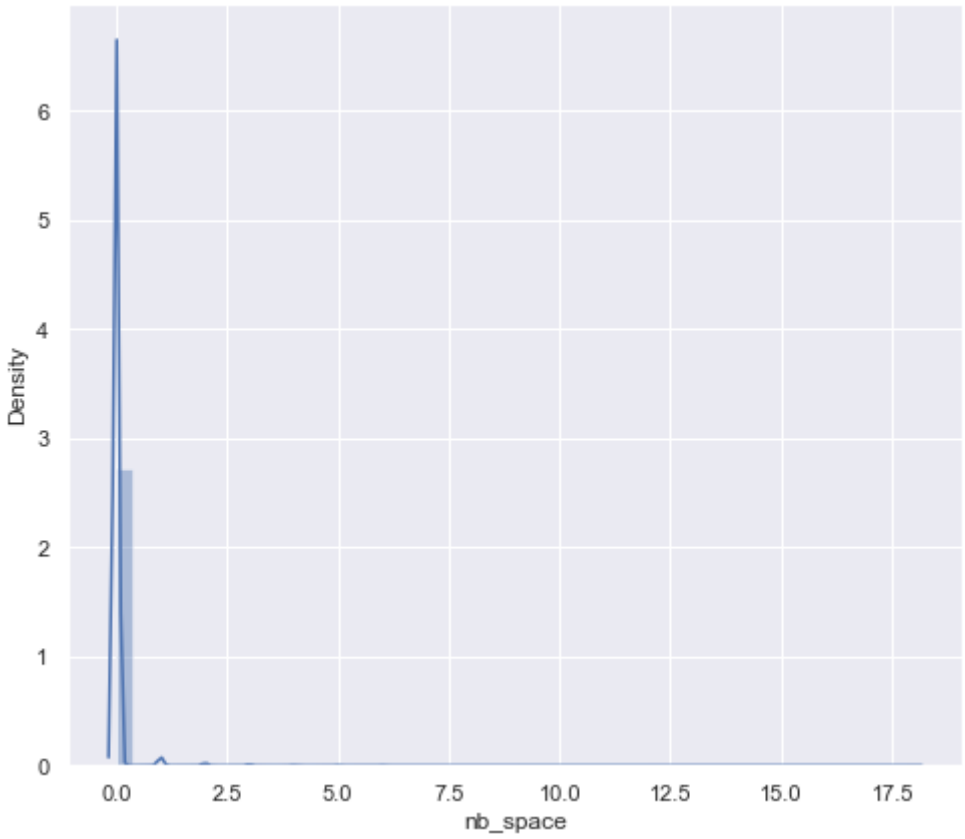
nb_semicolumn
16.15800867350721



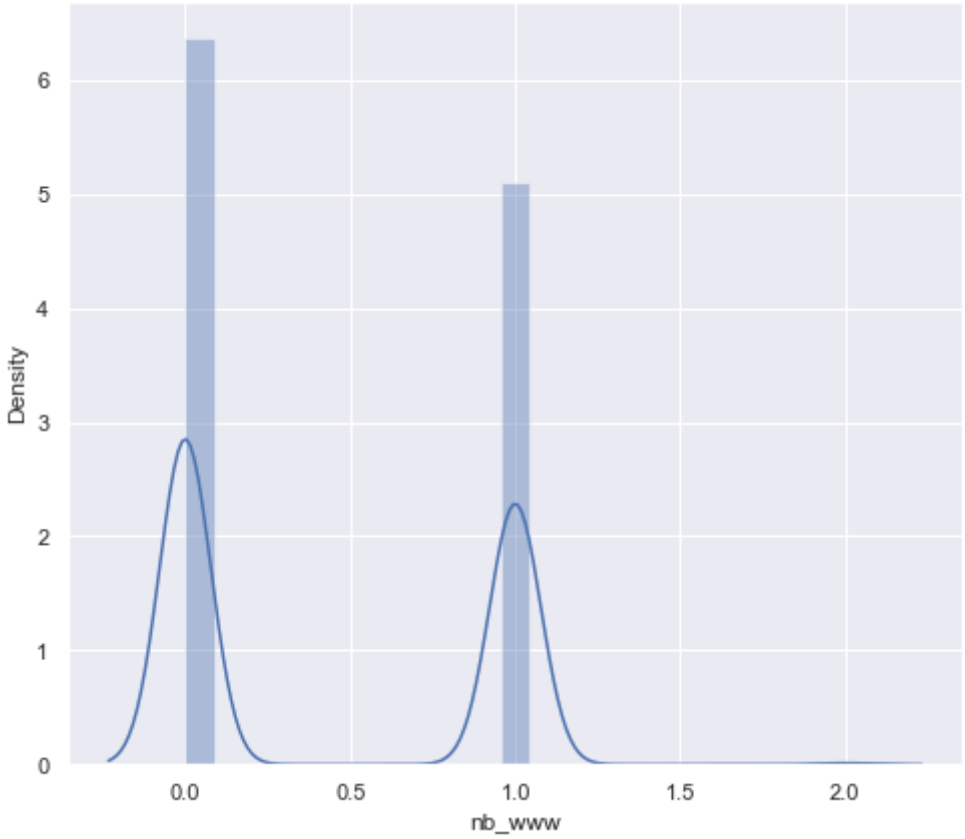
nb_dollar
55.649881448934664



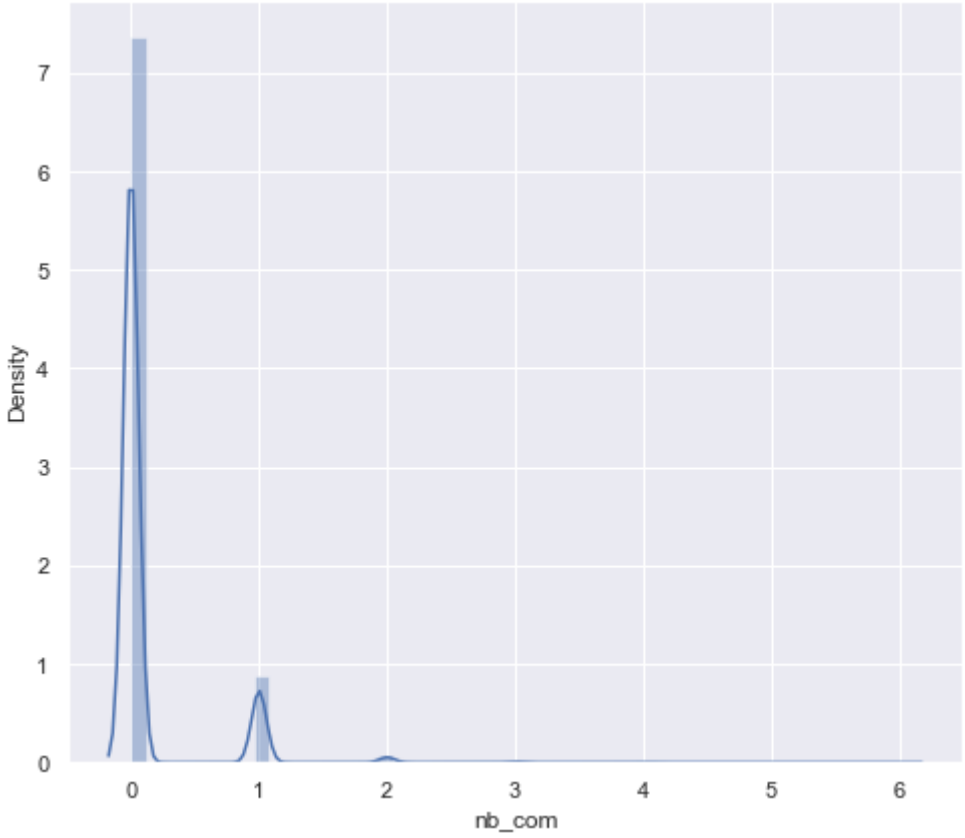

```
nb_space
25.42075052287114
```



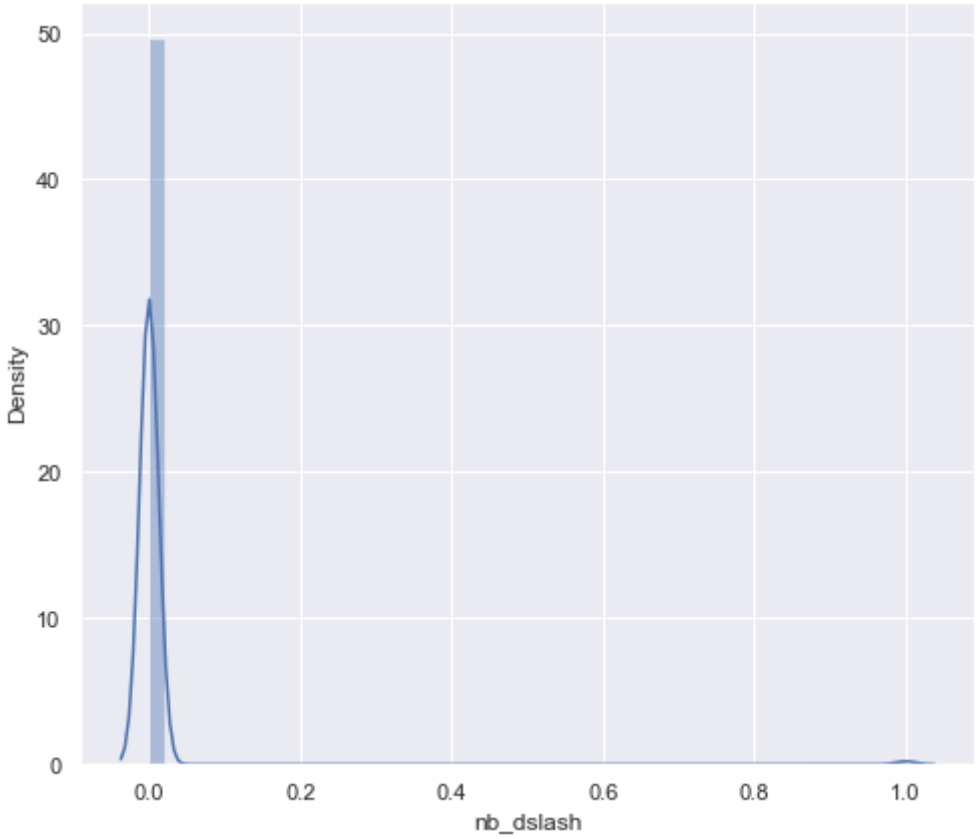

```
nb_www
0.26118084406590486
```



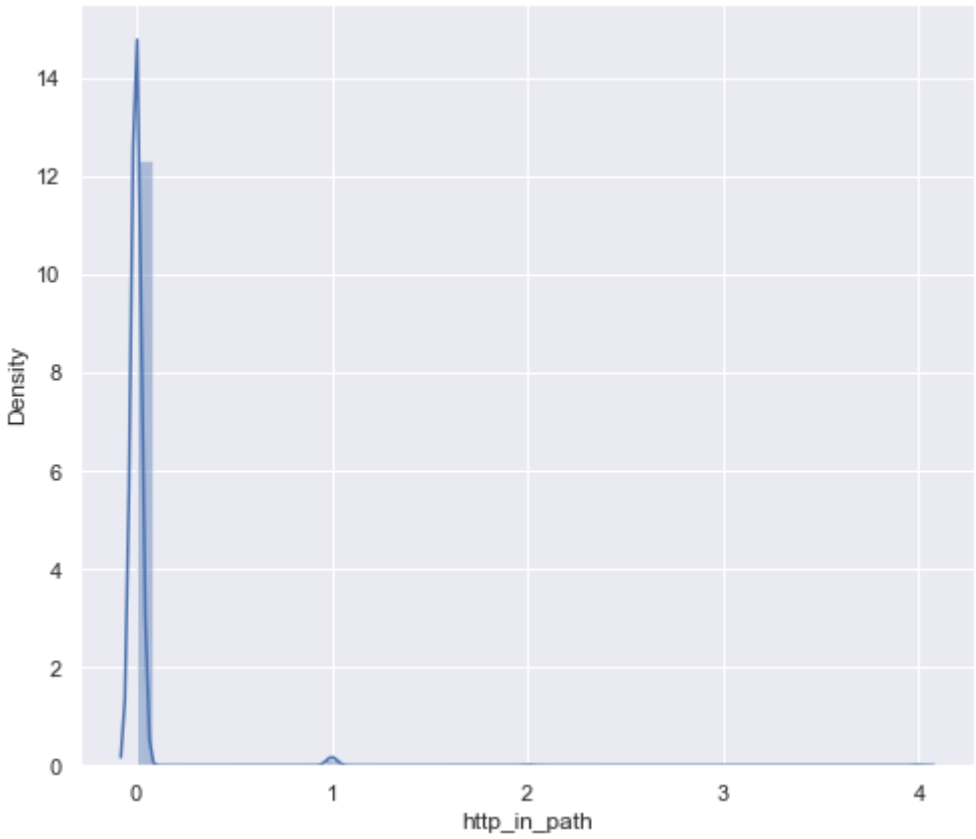
nb_com
3.7778834783625963



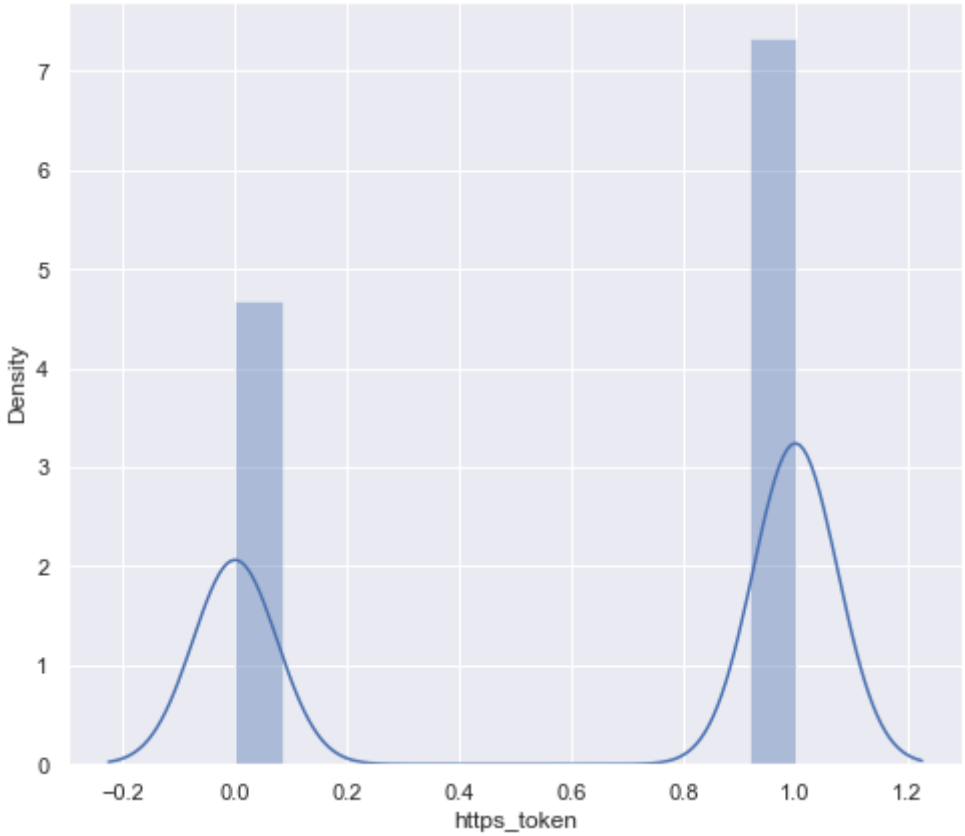
nb_dslash
12.223199459217504



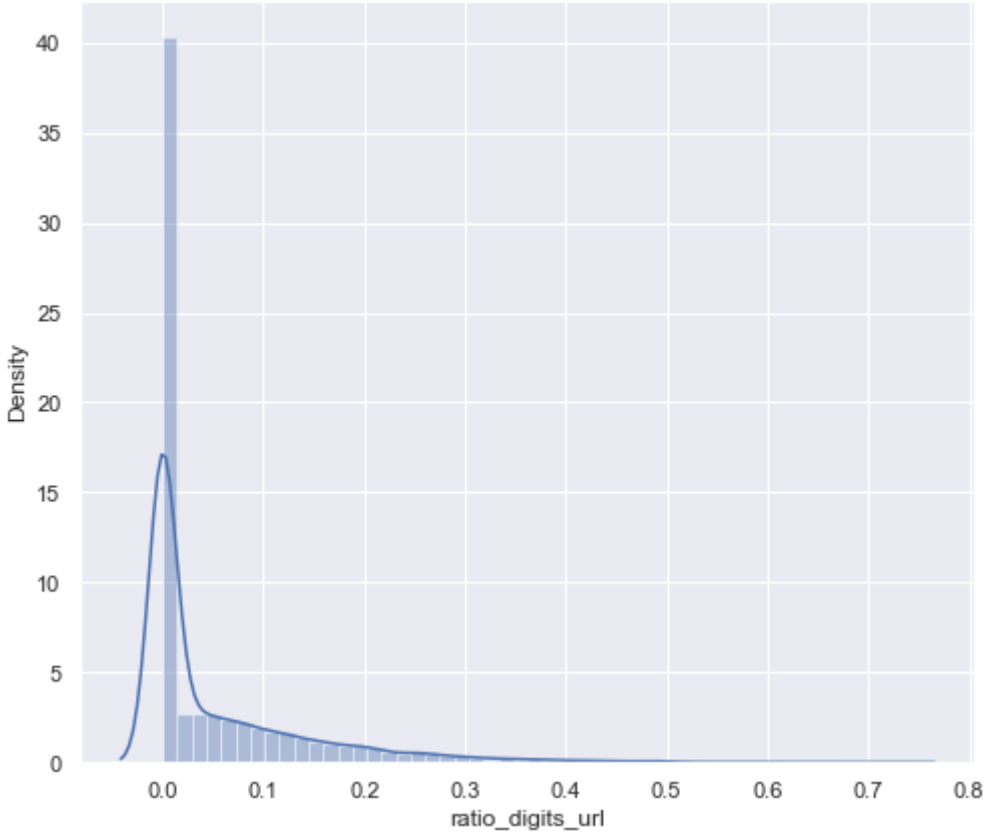
http_in_path
14.816173894258103



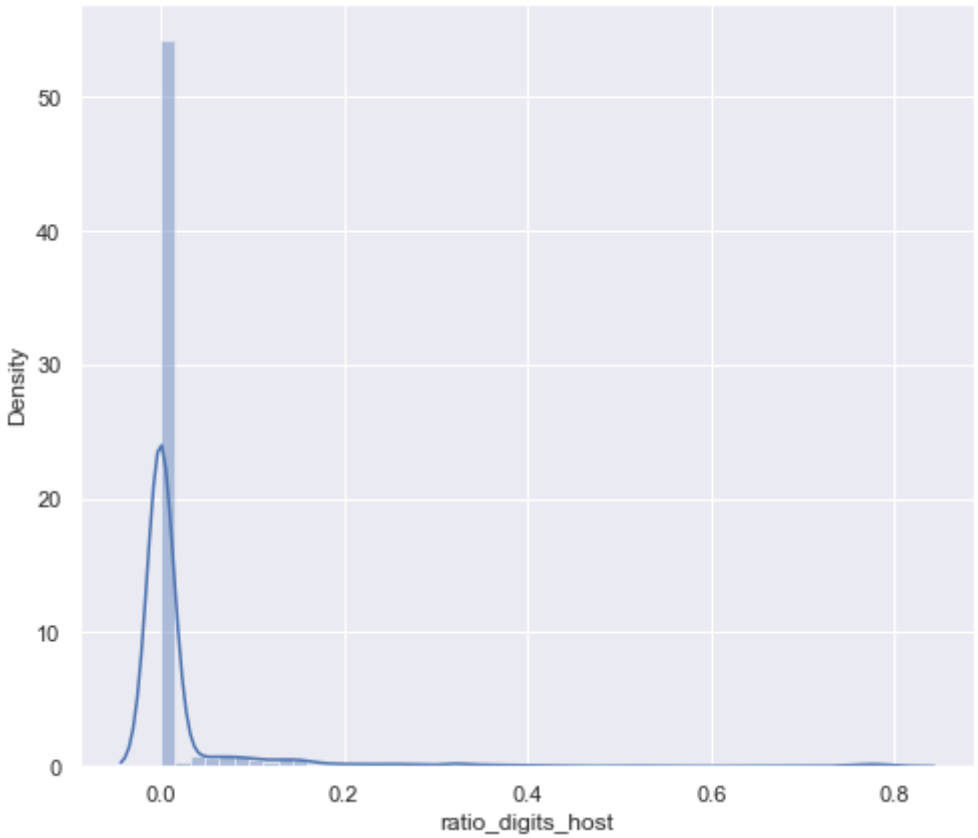
https_token
-0.4550872450174506



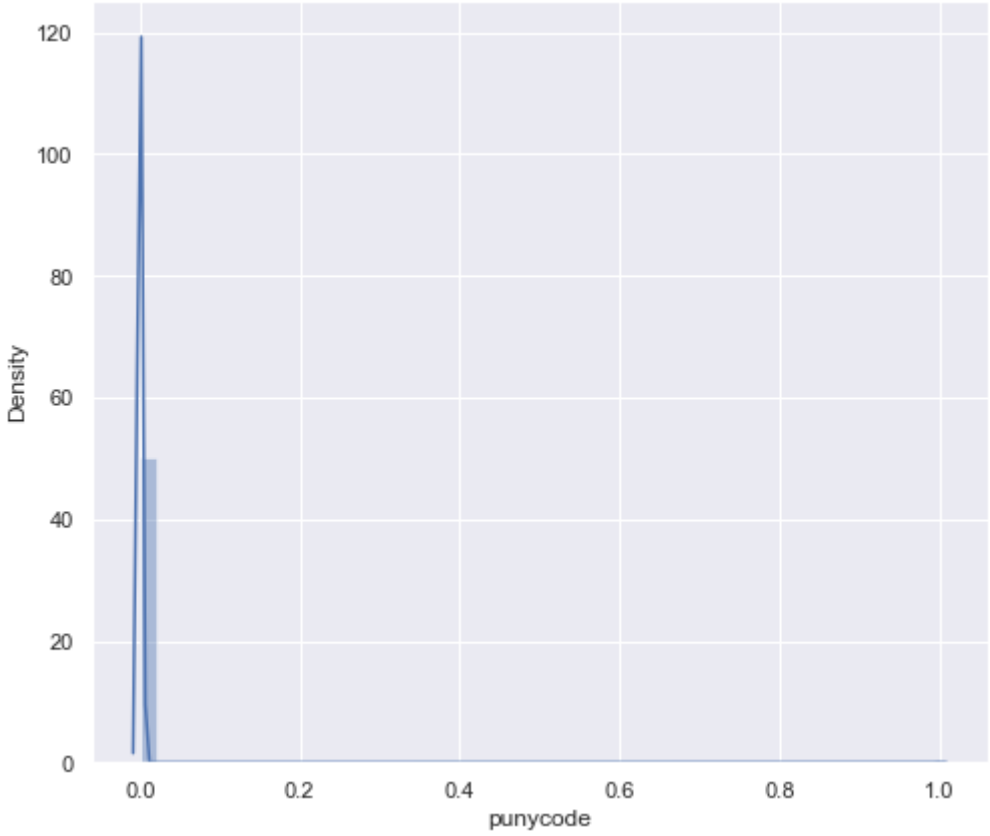
ratio_digits_url
2.1852483801509814



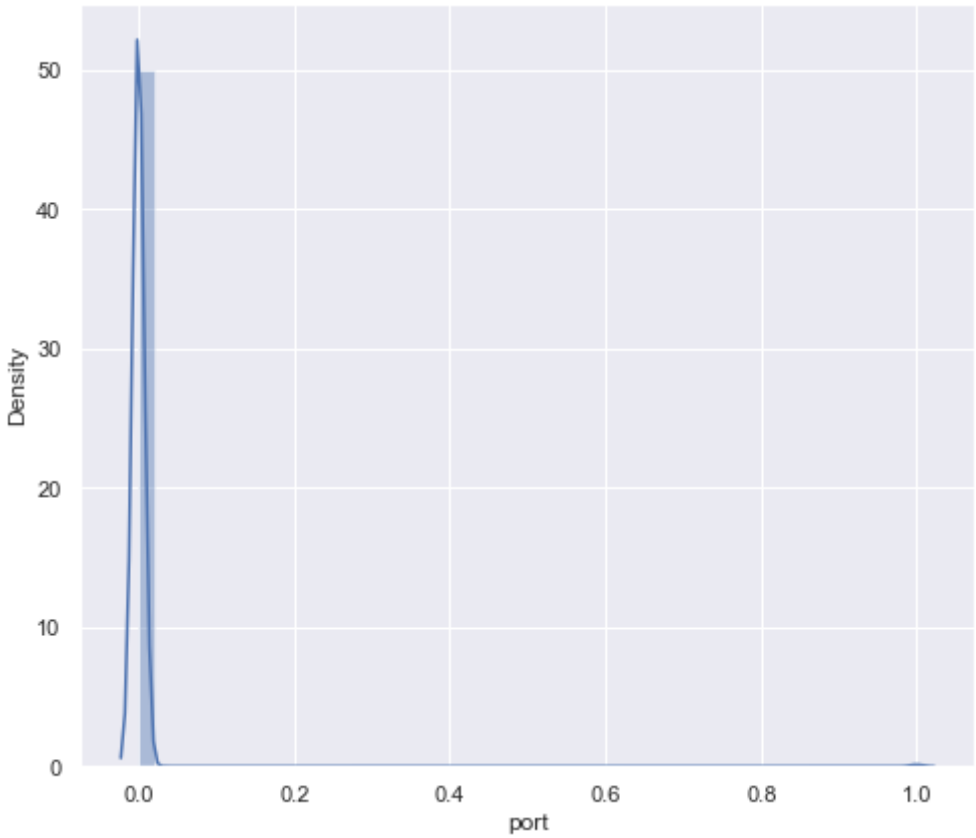
ratio_digits_host
5.590464319761369



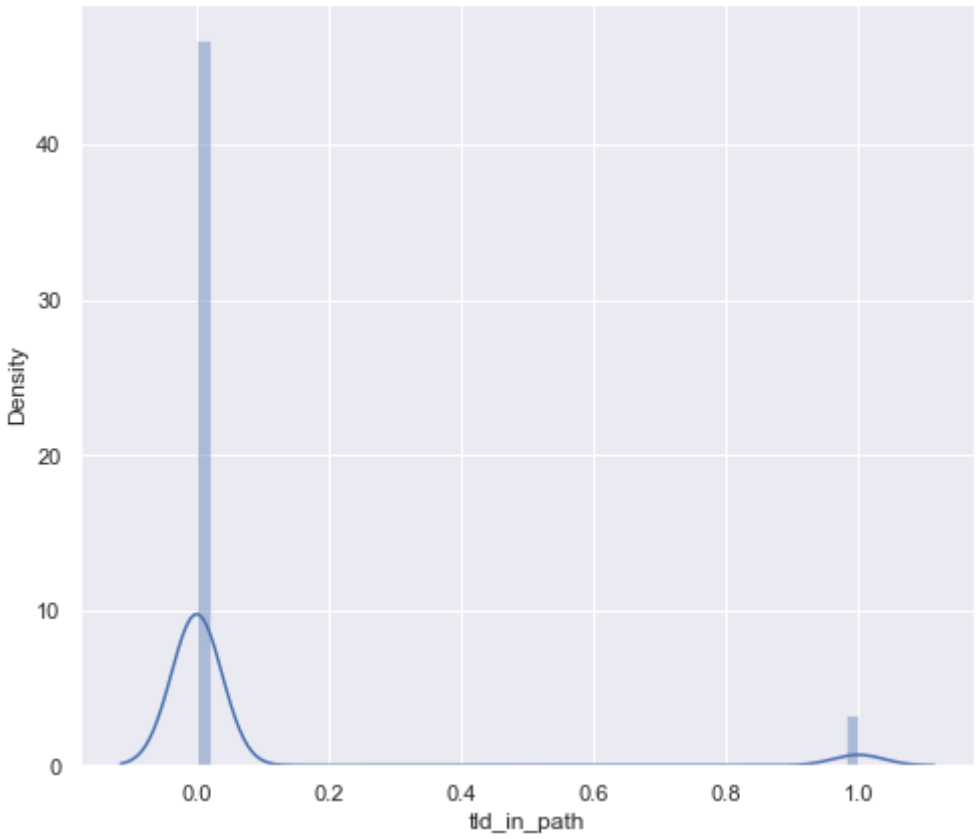
punycode
53.42752427427988



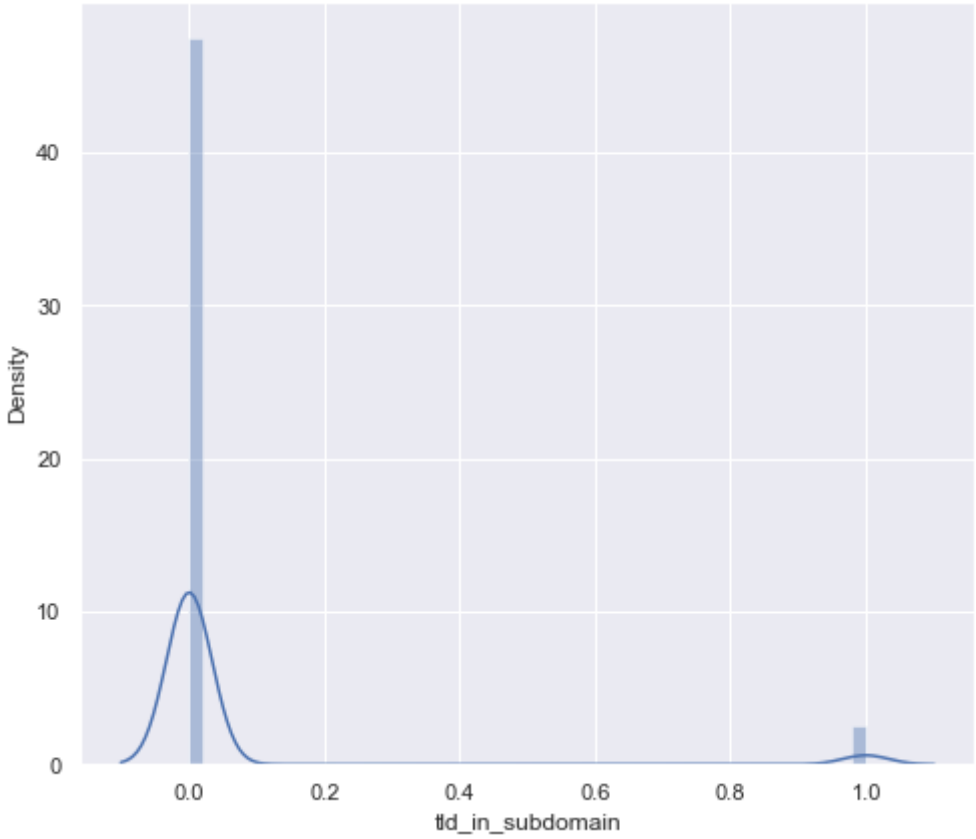
port
20.502090164938817



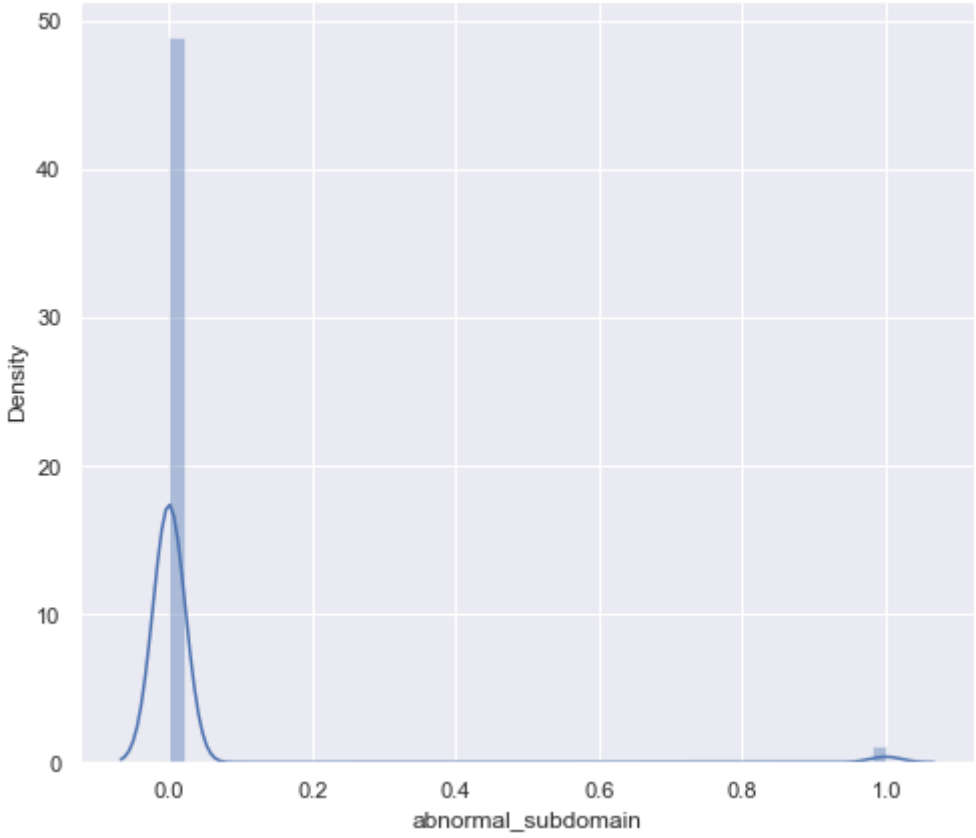

```
tld_in_path
3.5085929828210505
```



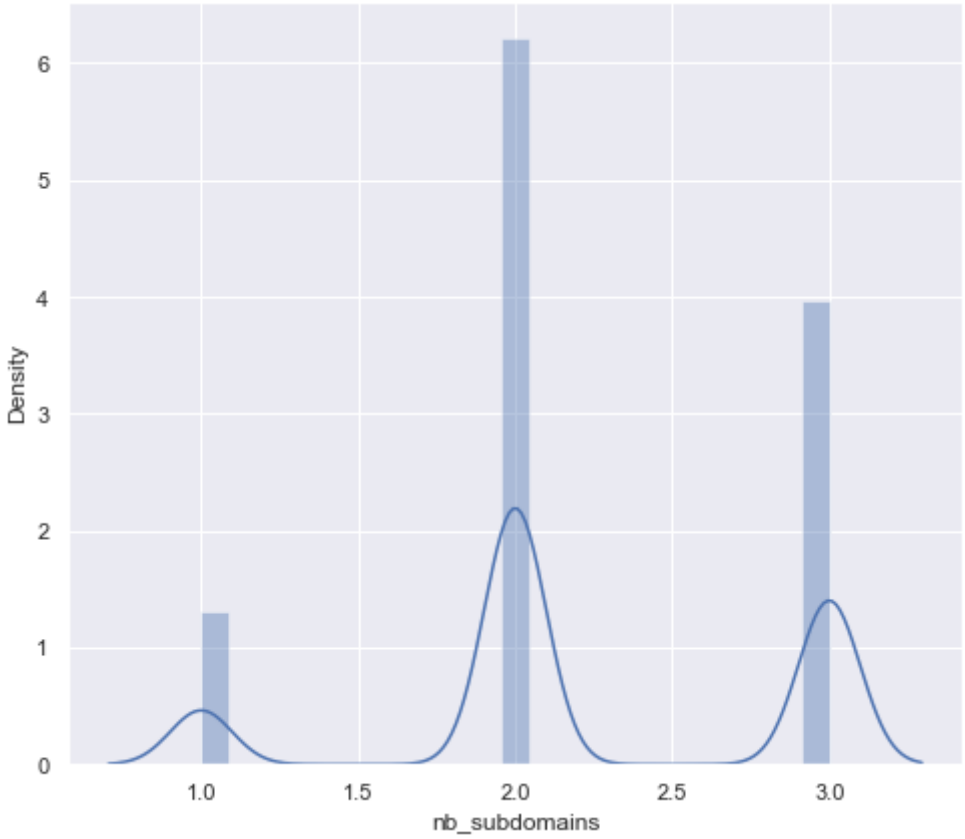

```
tld_in_subdomain
4.123156677750848
```



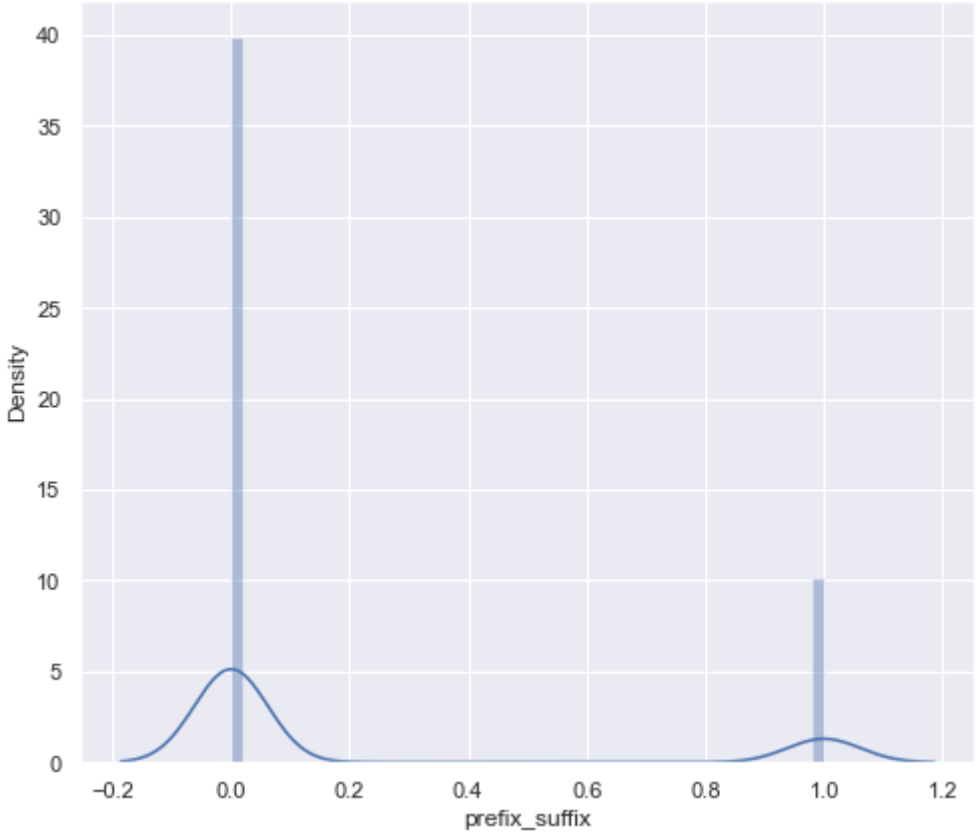

```
abnormal_subdomain
6.580075283779847
```



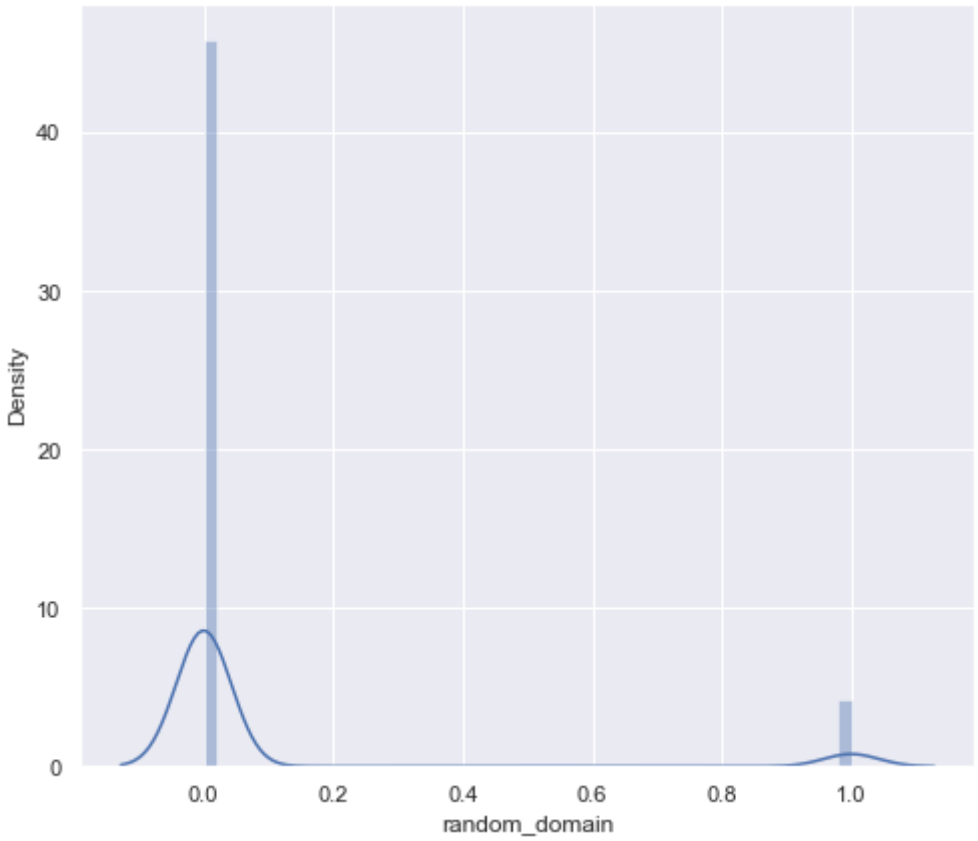

```
nb_subdomains
-0.2429713161792032
```



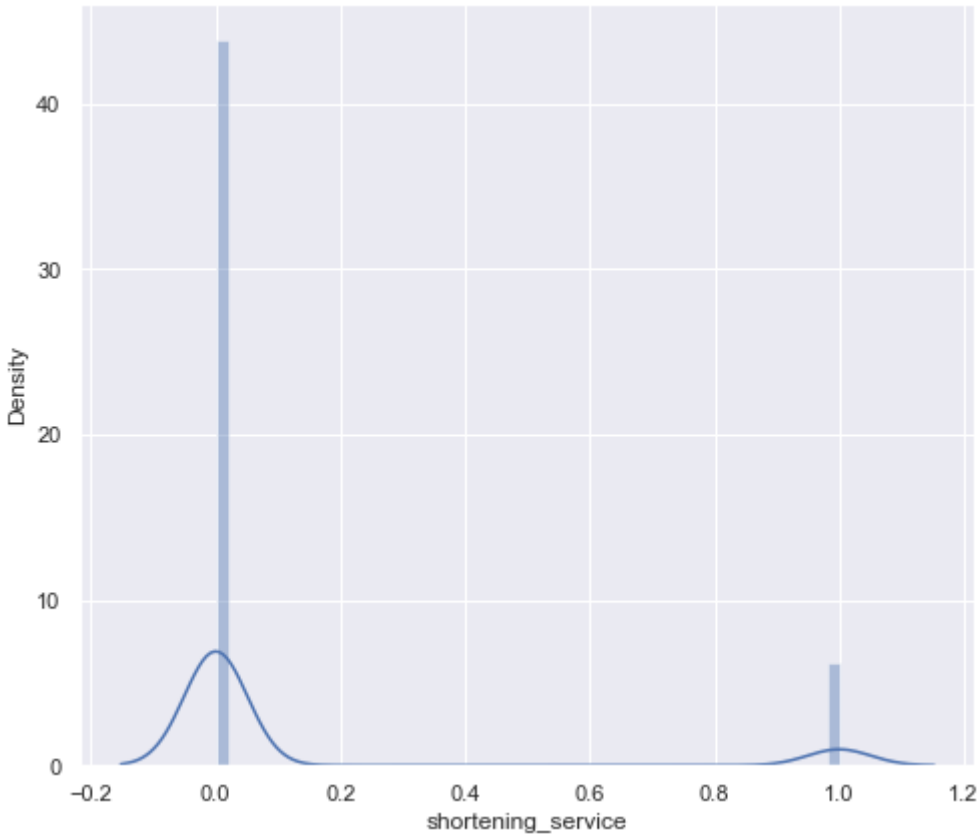
prefix_suffix
1.4809922710097732



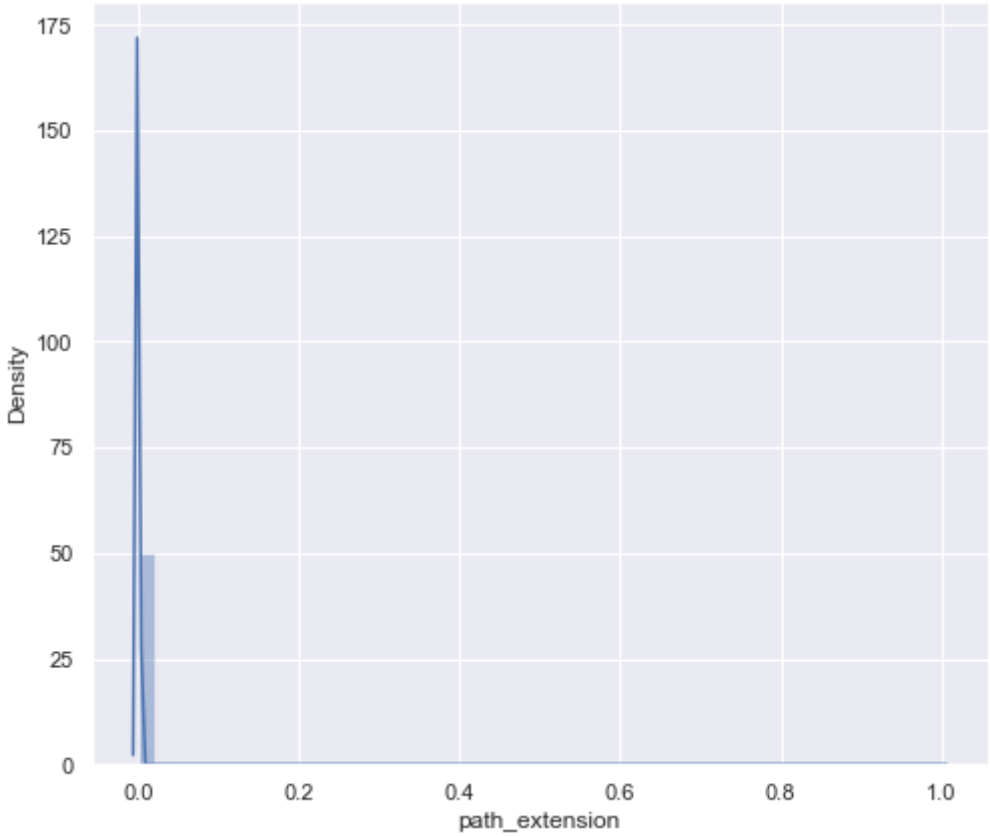
random_domain
3.016149789846897



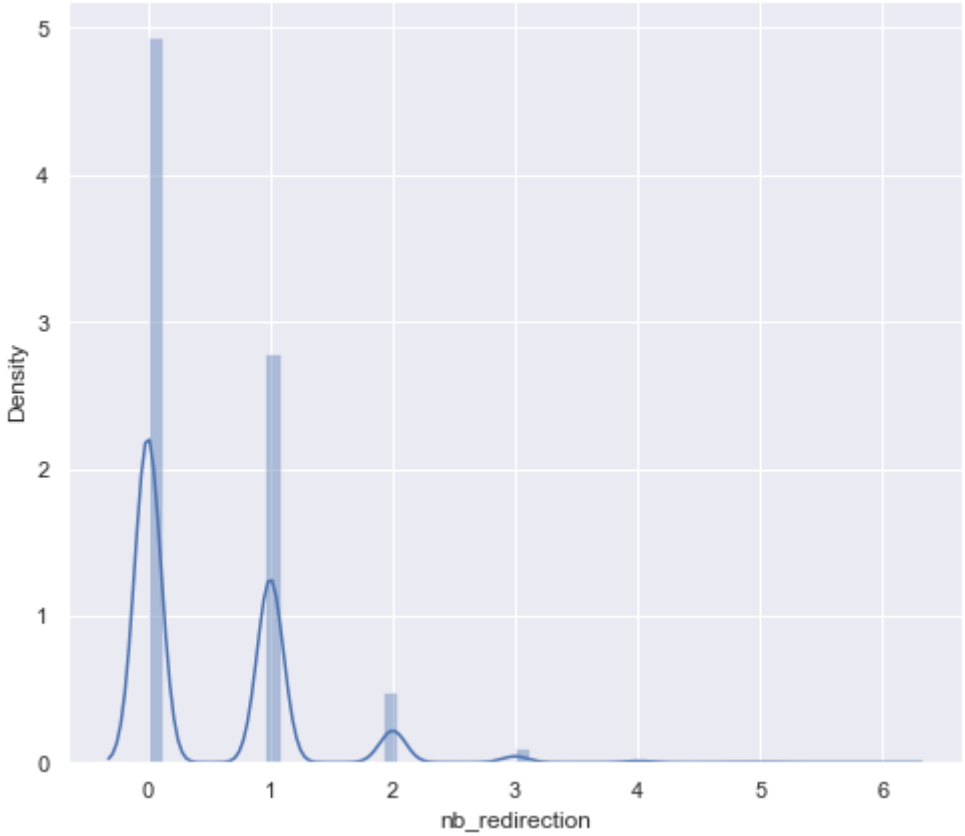
shortening_service
2.2894257498125326



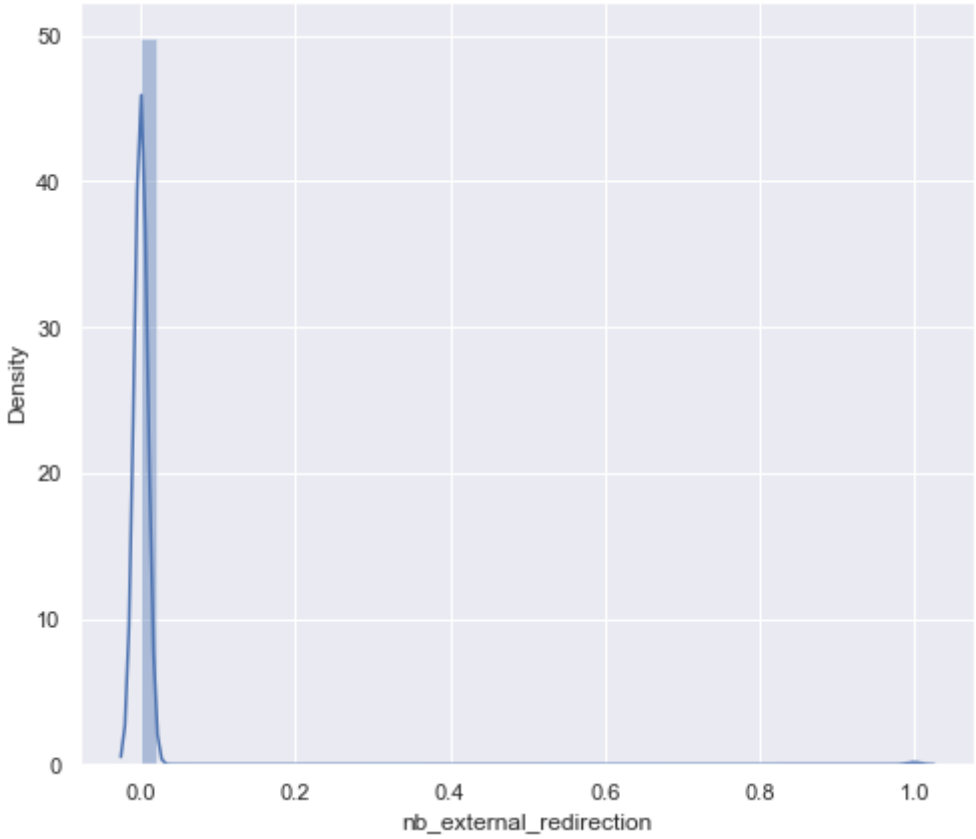
path_extension
75.57777566857037



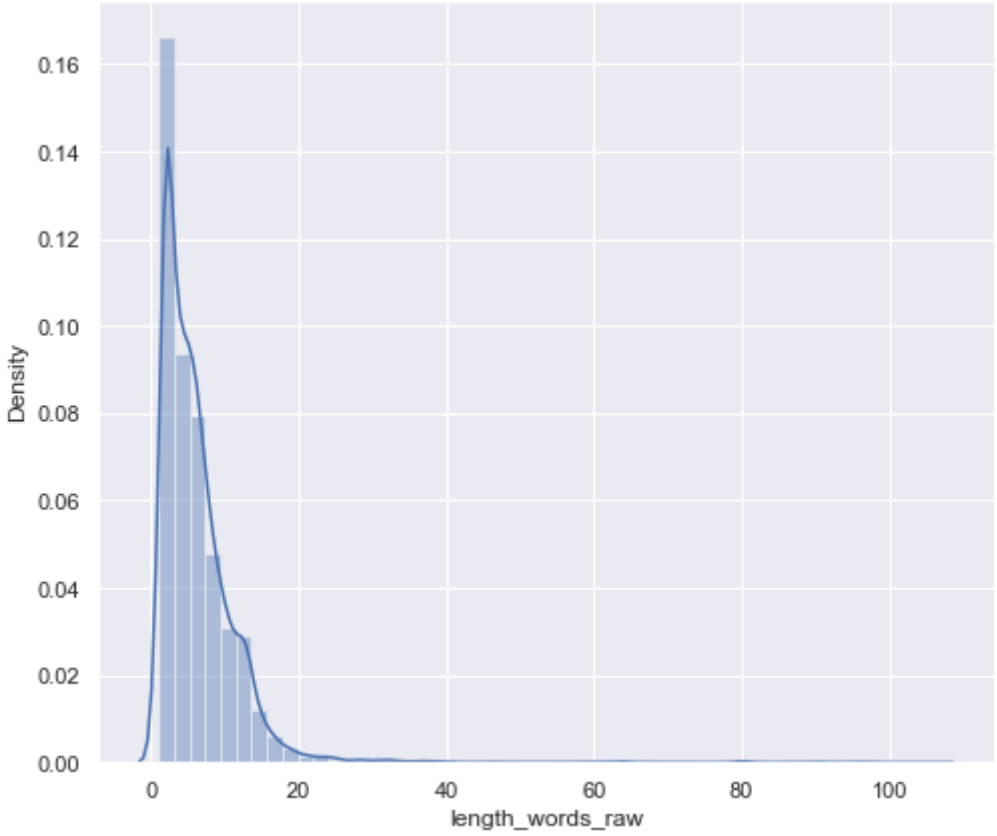
nb_redirection
1.5682466351899065



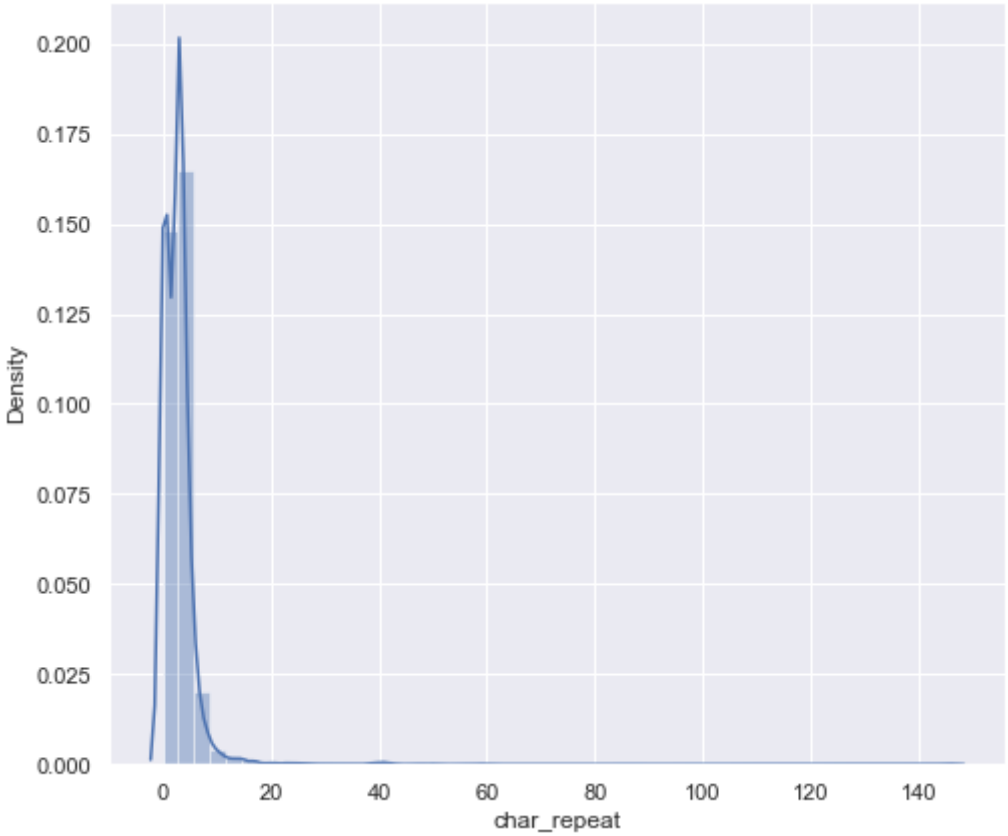
nb_external_redirection
17.734236931925274



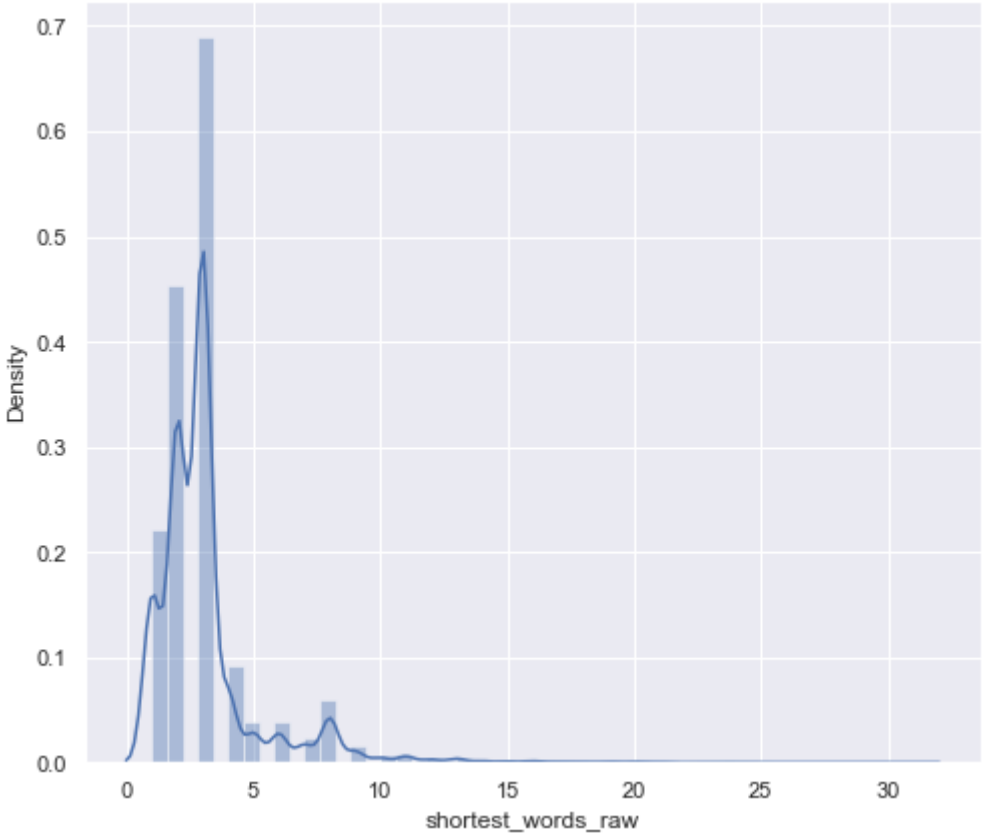
length_words_raw
5.366645557780265



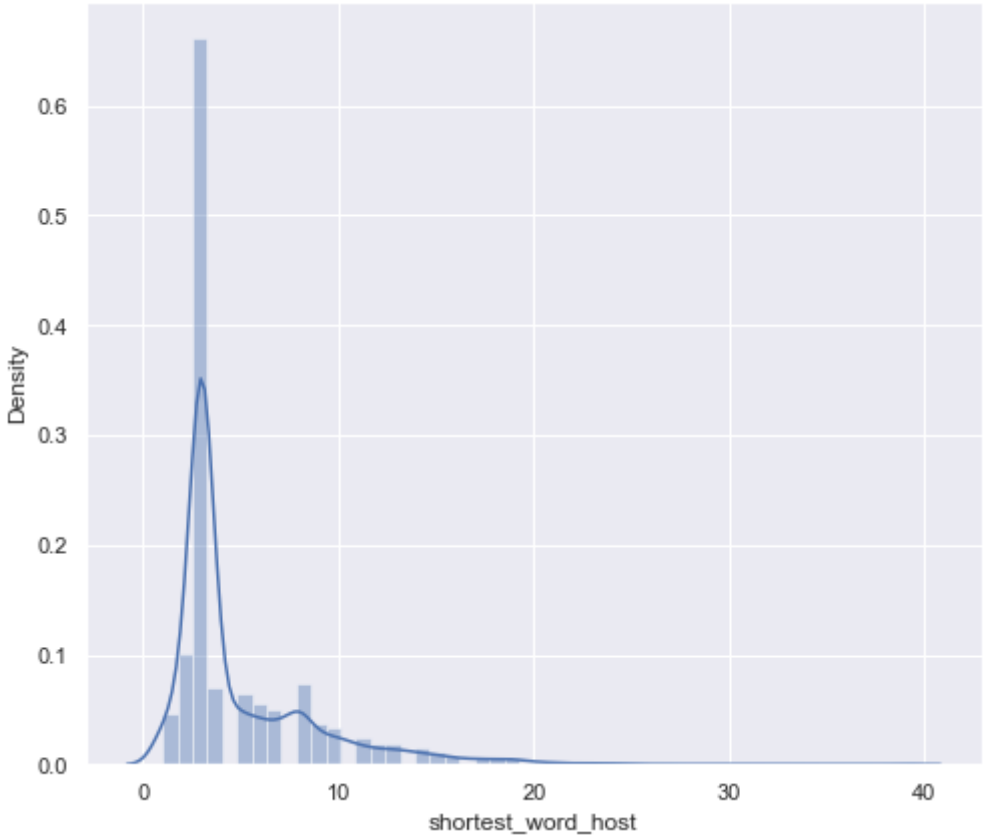
char_repeat
15.754713240663138



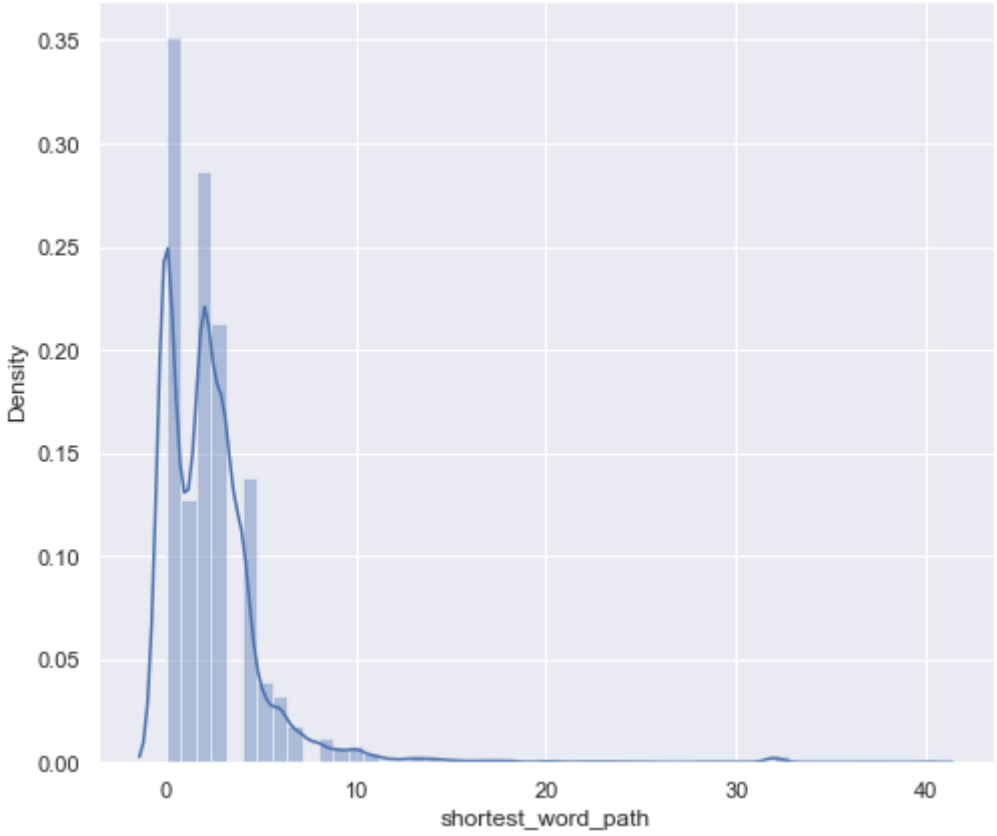
shortest_words_raw
3.1562439104641475



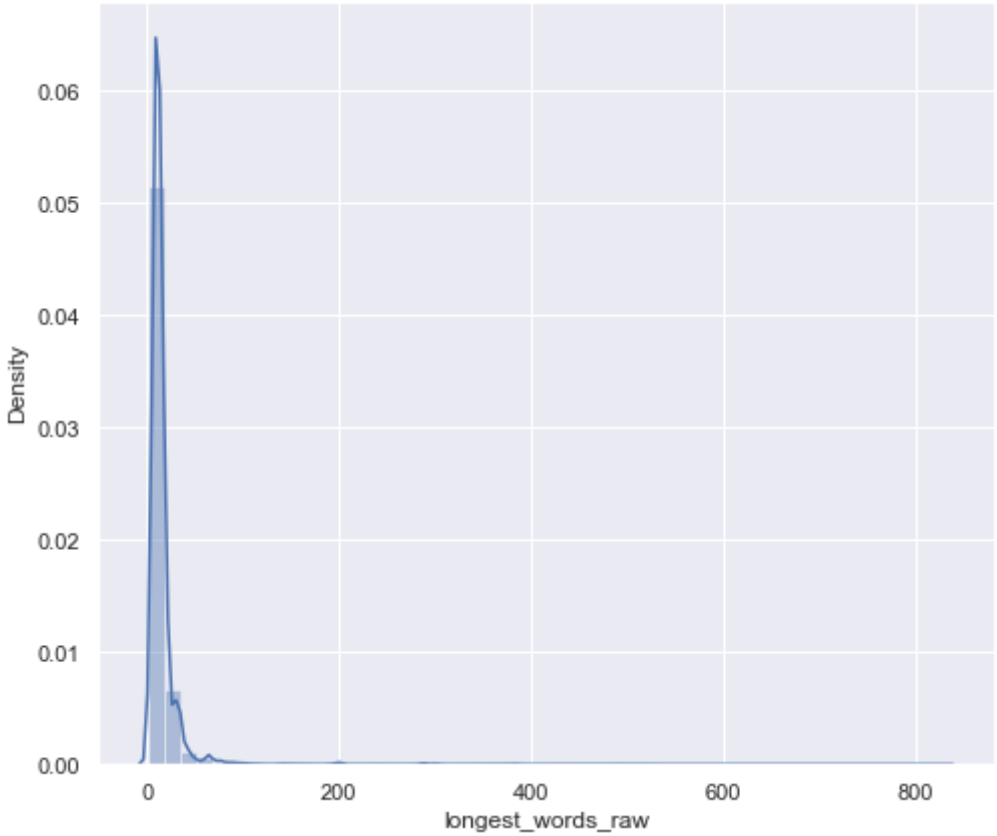
shortest_word_host
2.267638466973383



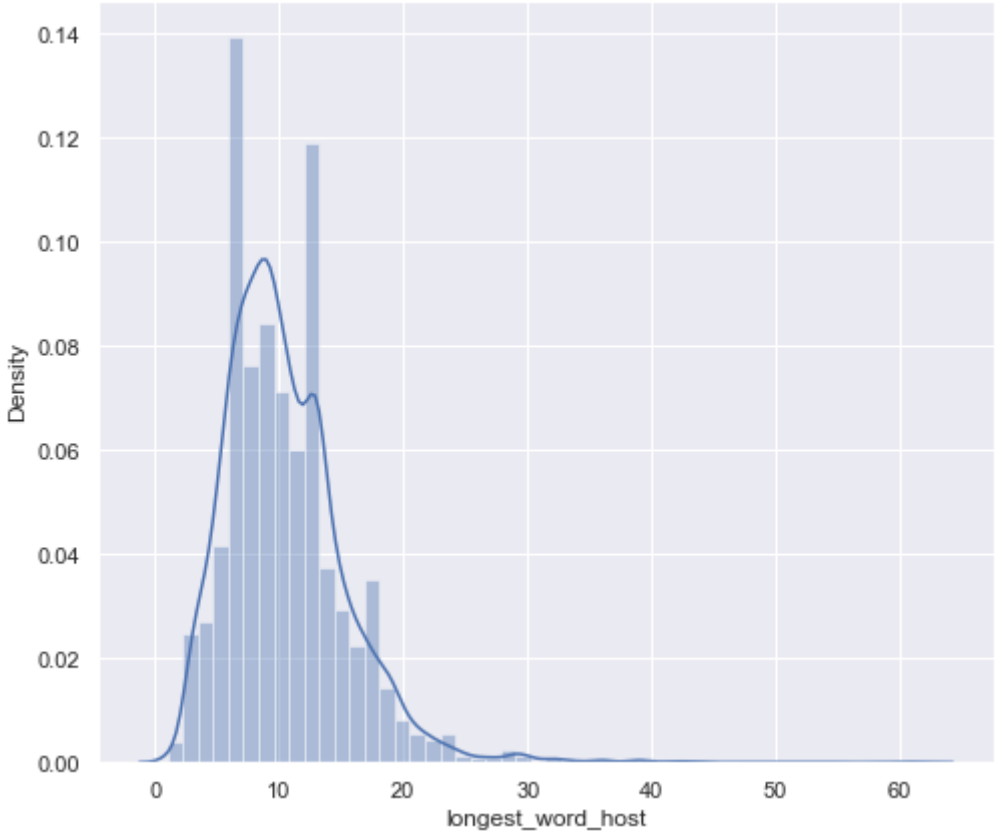
shortest_word_path
4.686988442210764



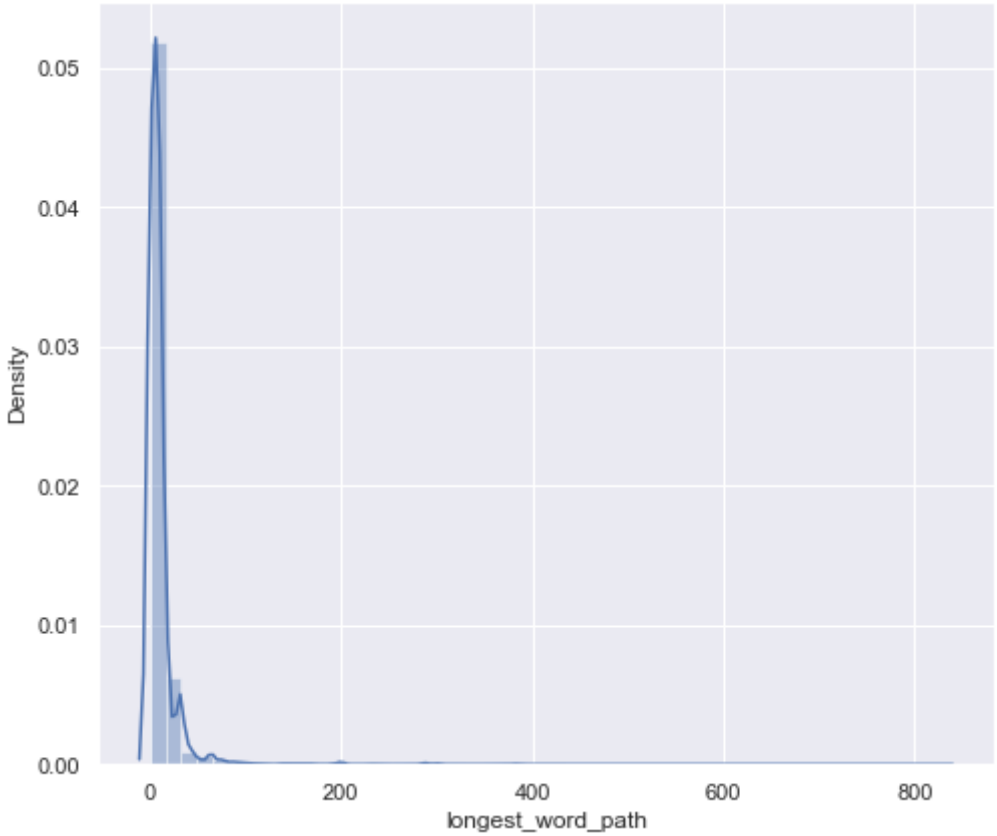
longest_words_raw
13.529247747051324



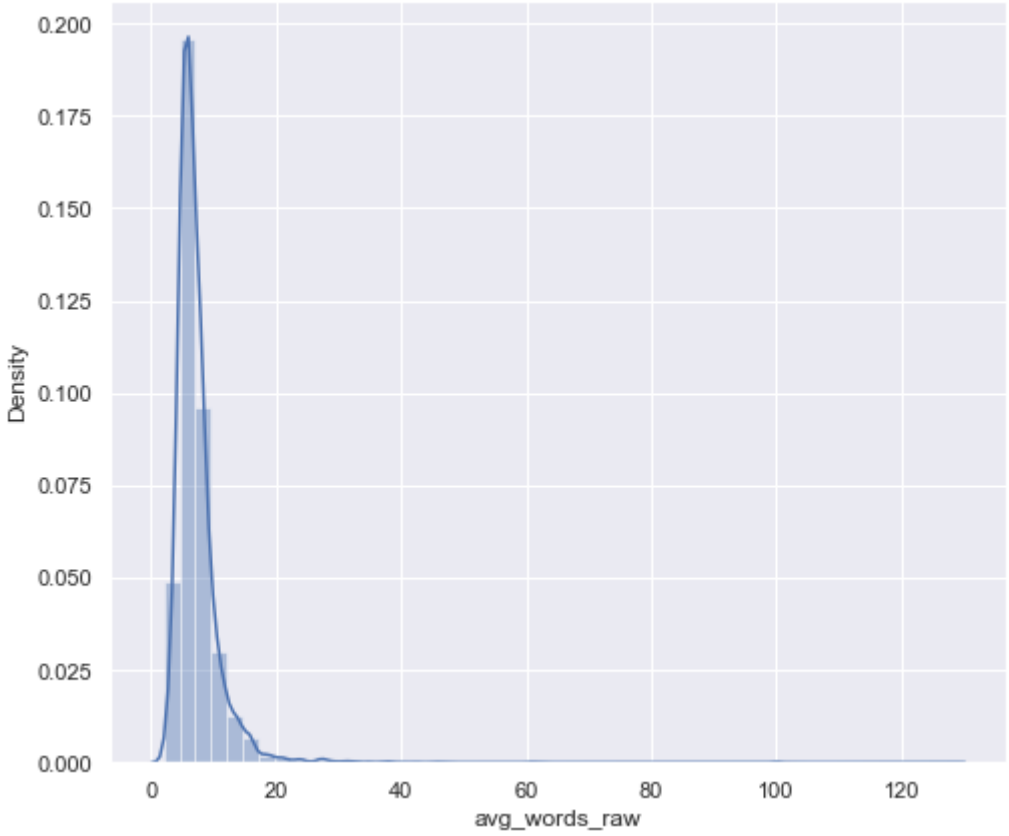
longest_word_host
1.6308622654965566



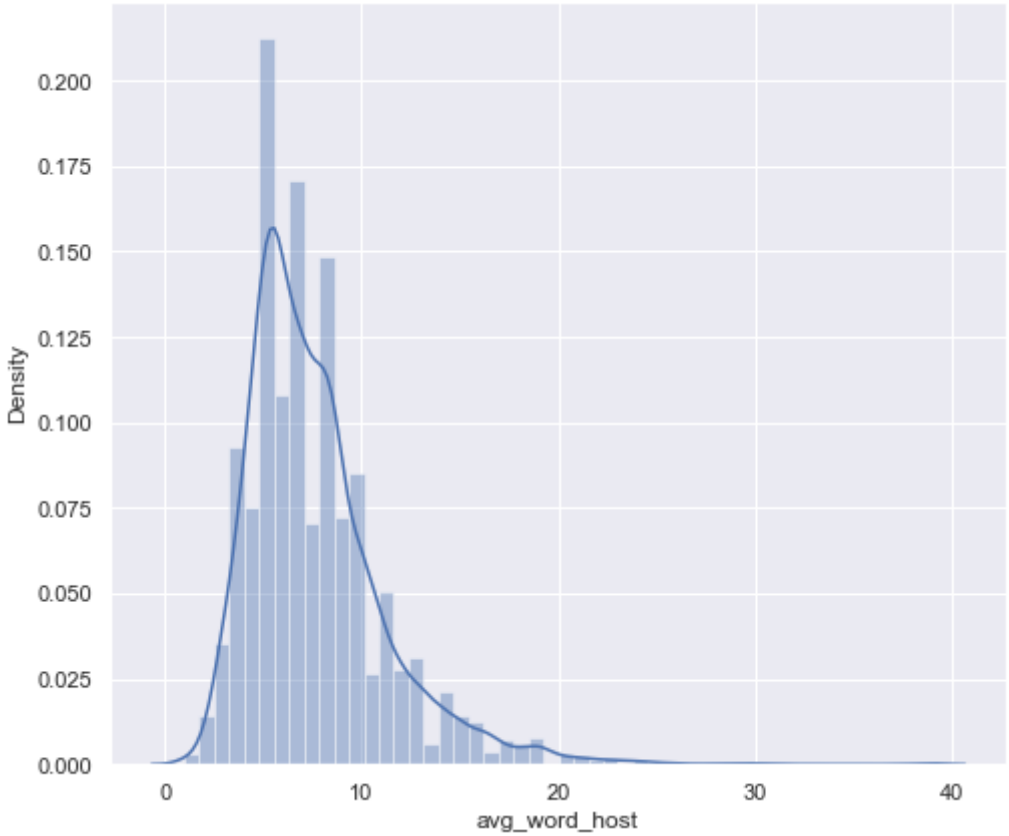
longest_word_path
12.39410296007728



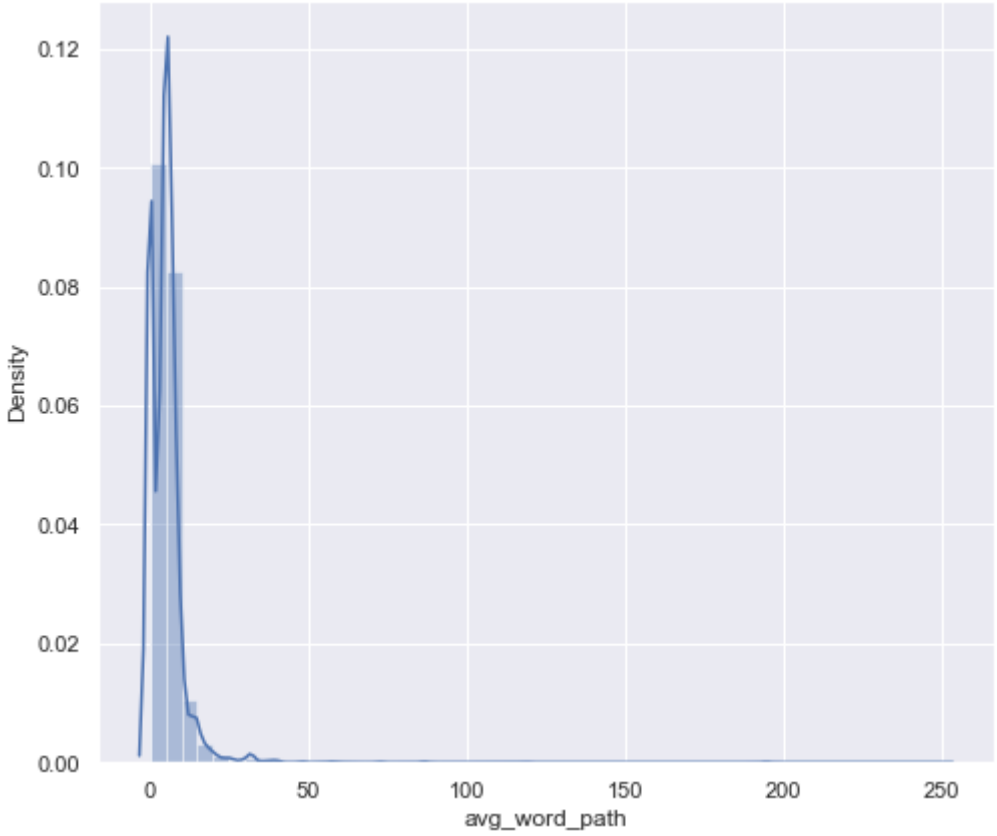
avg_words_raw
9.54746858606169



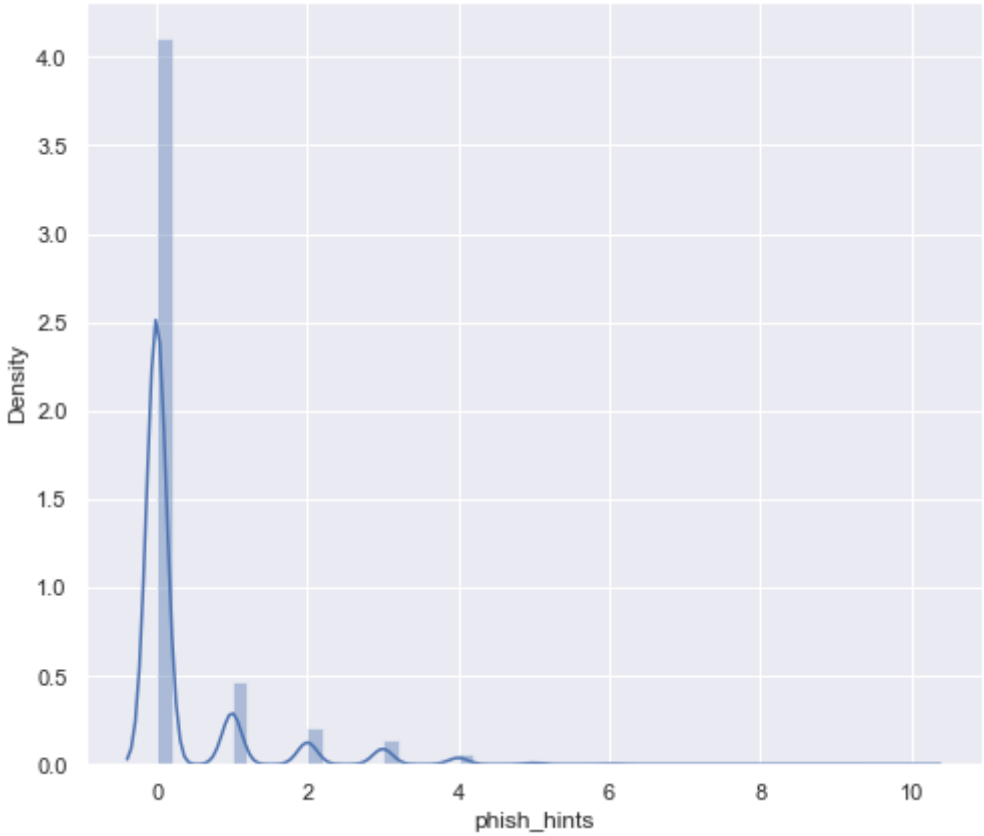
avg_word_host
1.7458504096089944



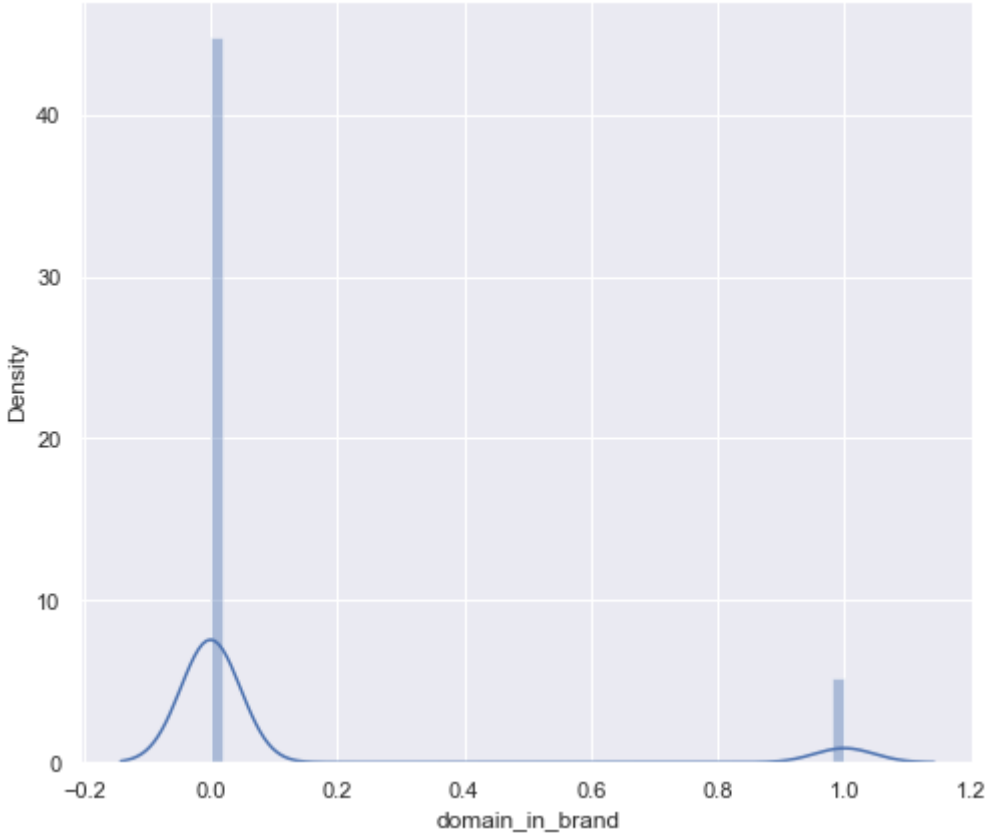
avg_word_path
13.444976644200679



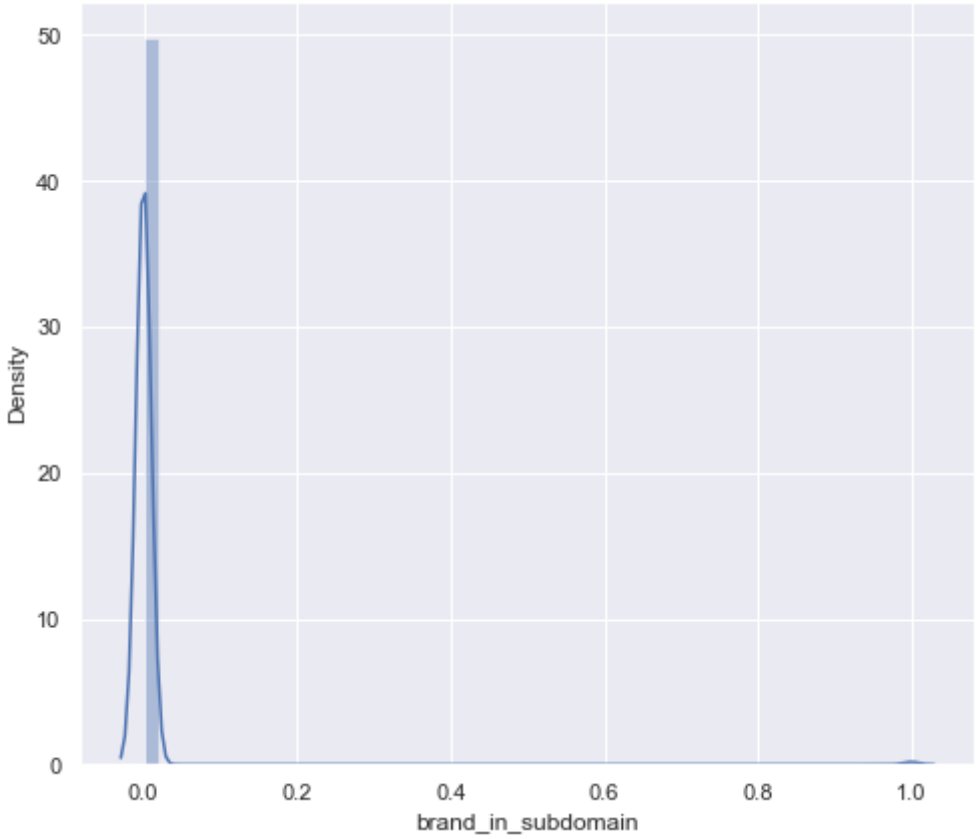
phish_hints
3.216062270808012



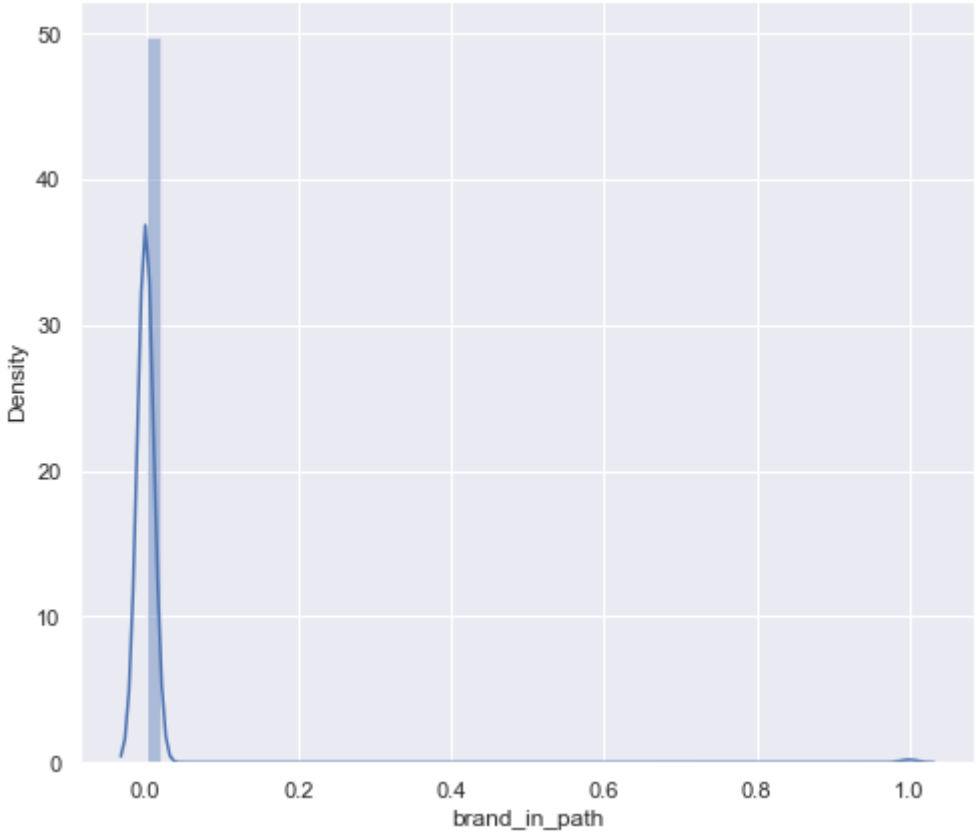

```
domain_in_brand
2.5910031422395523
```



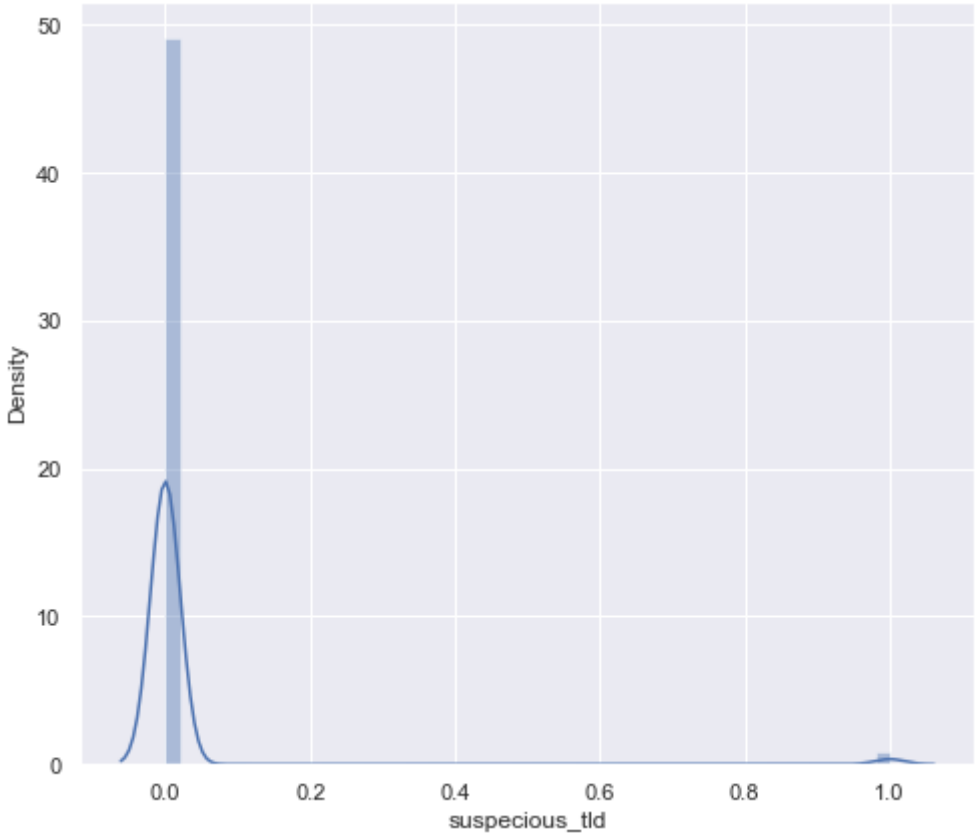

```
brand_in_subdomain
15.498245653168201
```



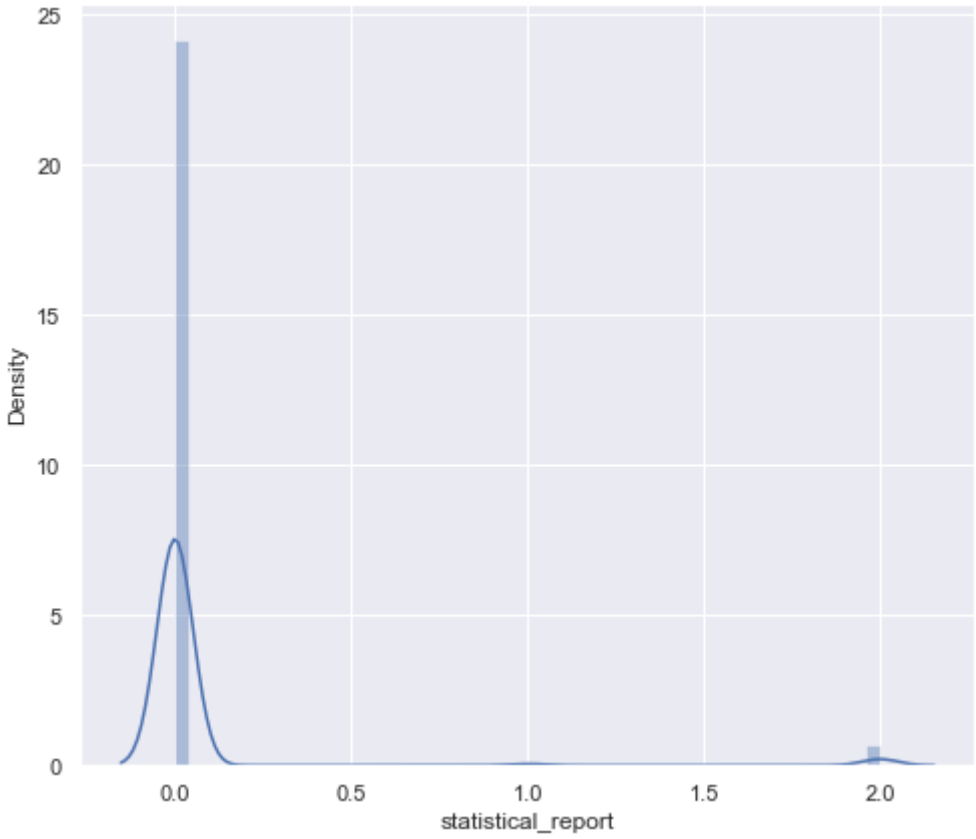
brand_in_path
14.181398604048276



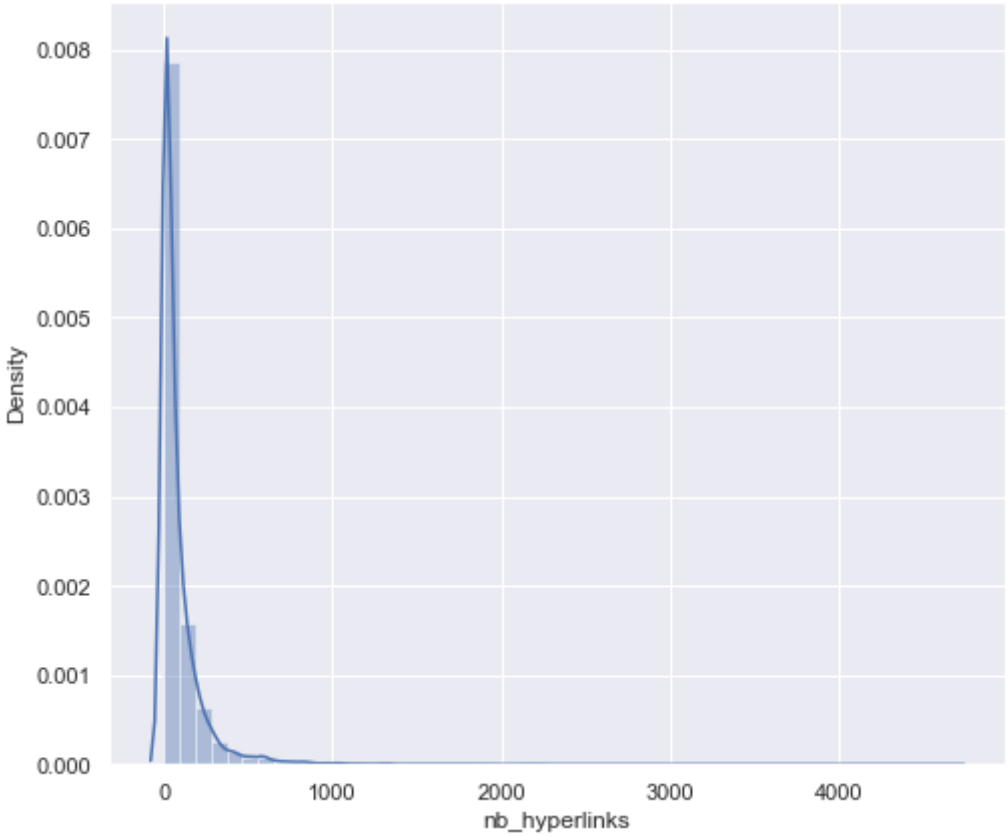
suspecious_tld
7.264596366434388



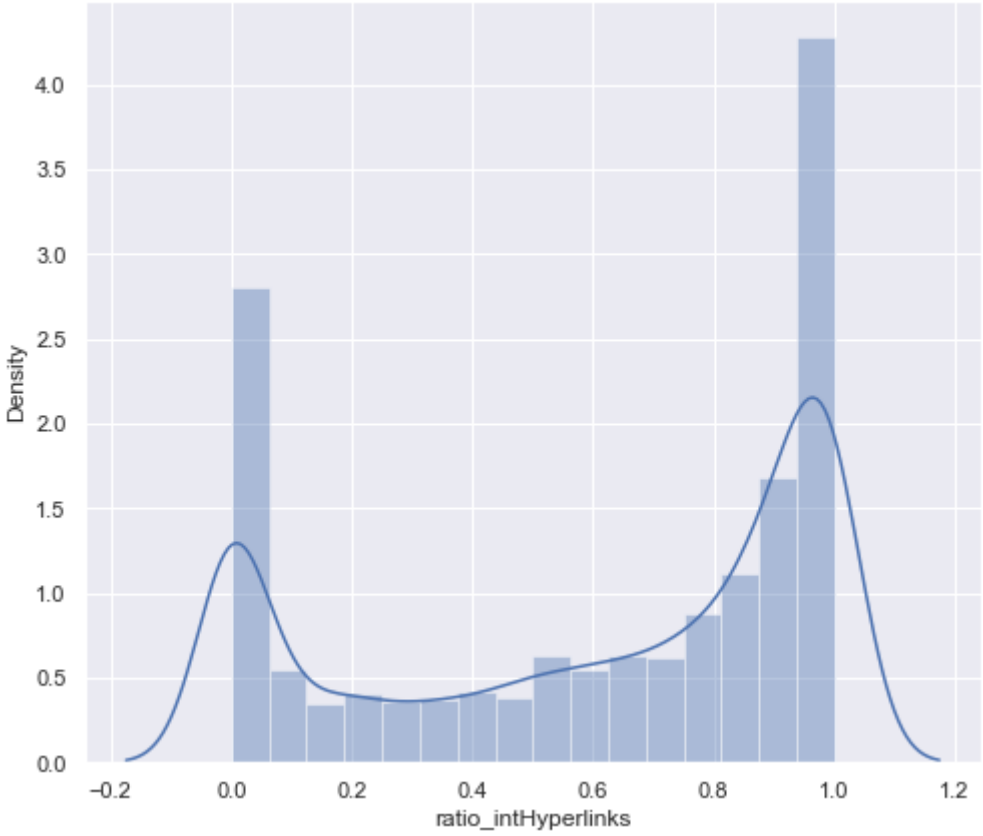
statistical_report
5.516235695445257



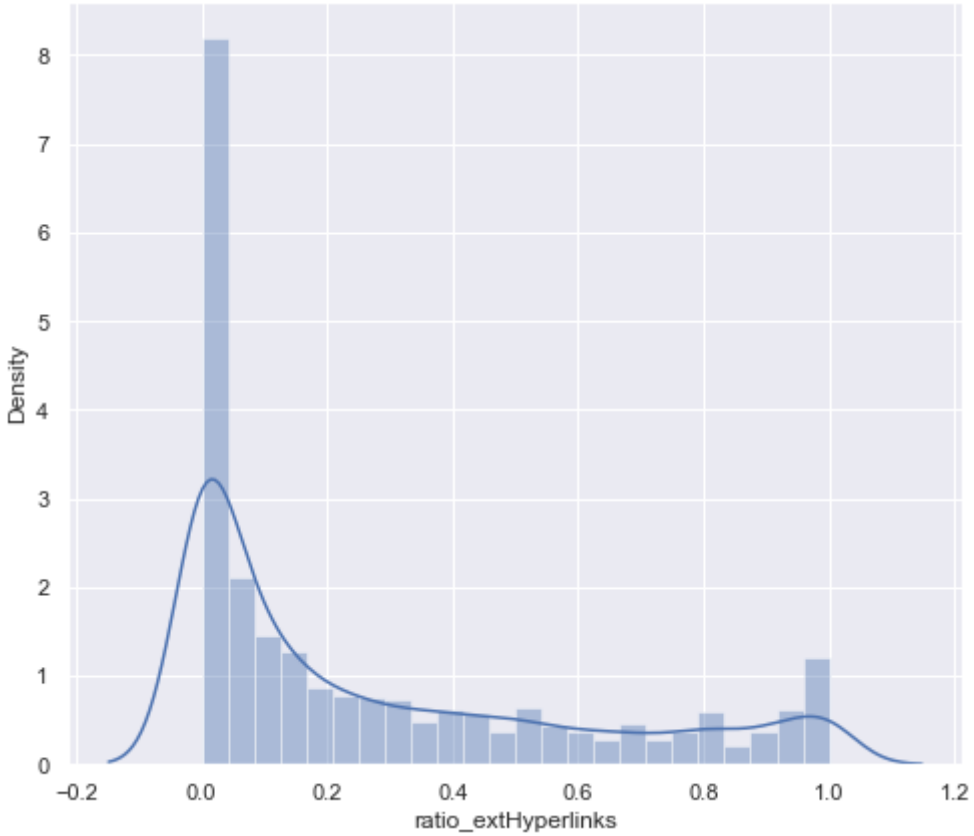
nb_hyperlinks
7.674053392500226



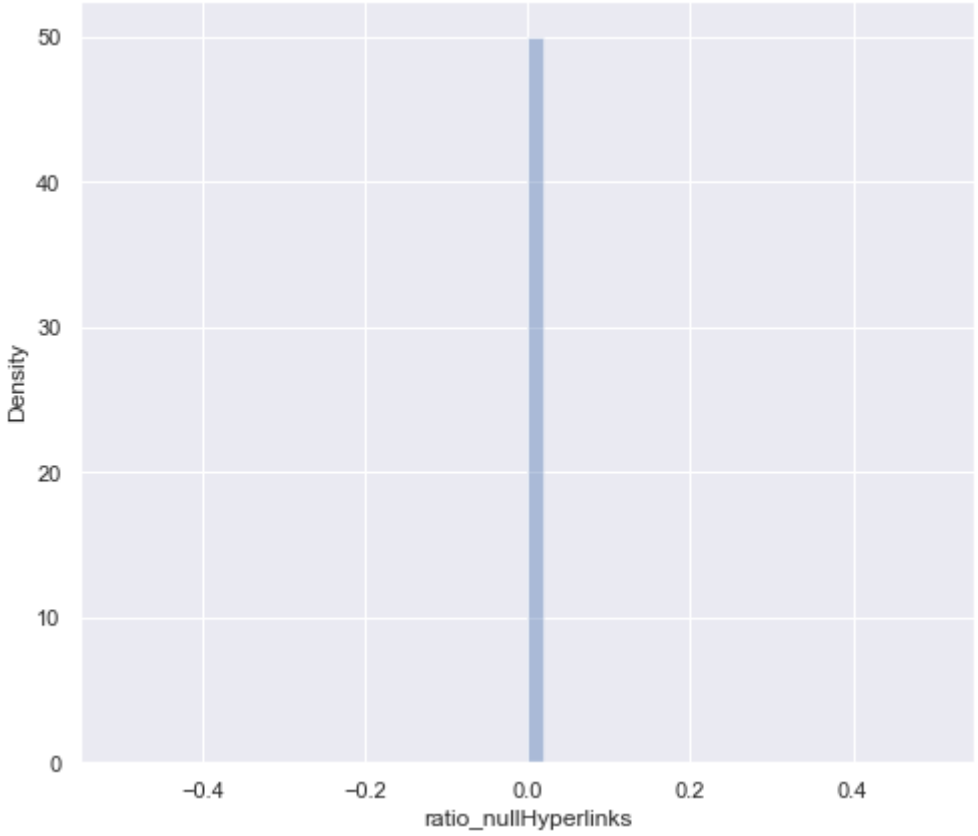
ratio_intHyperlinks
-0.5279398443363689



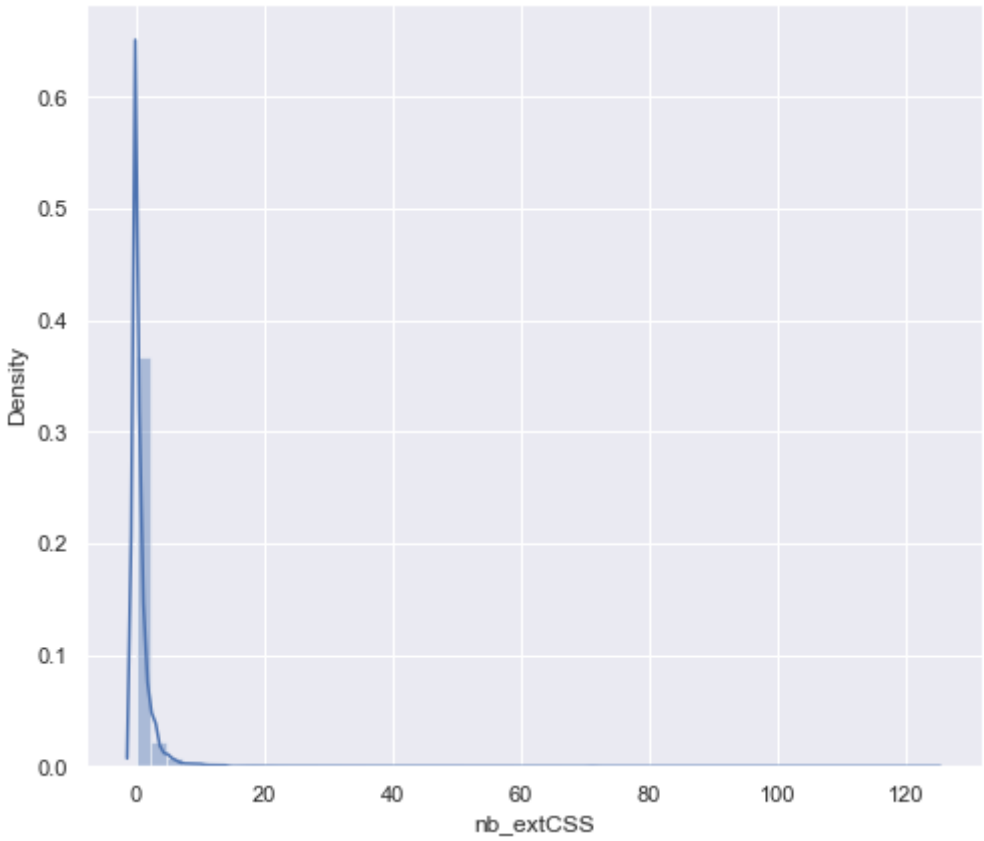
ratio_extHyperlinks
1.0083230592062746



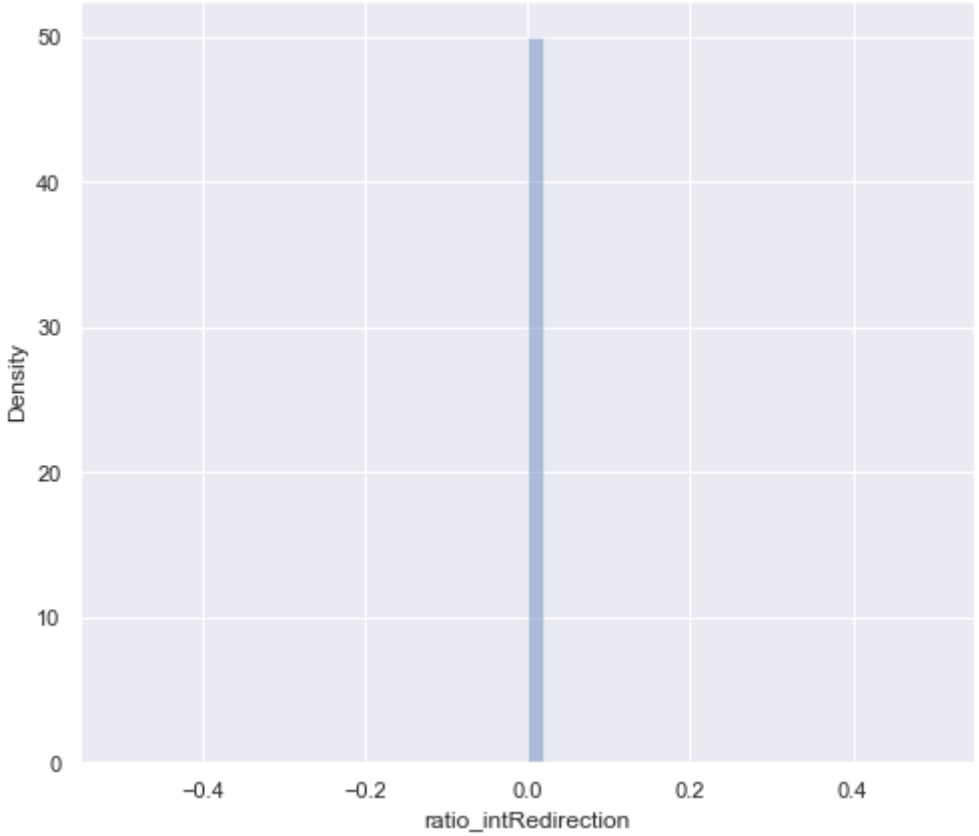

```
ratio_nullHyperlinks
0.0
```



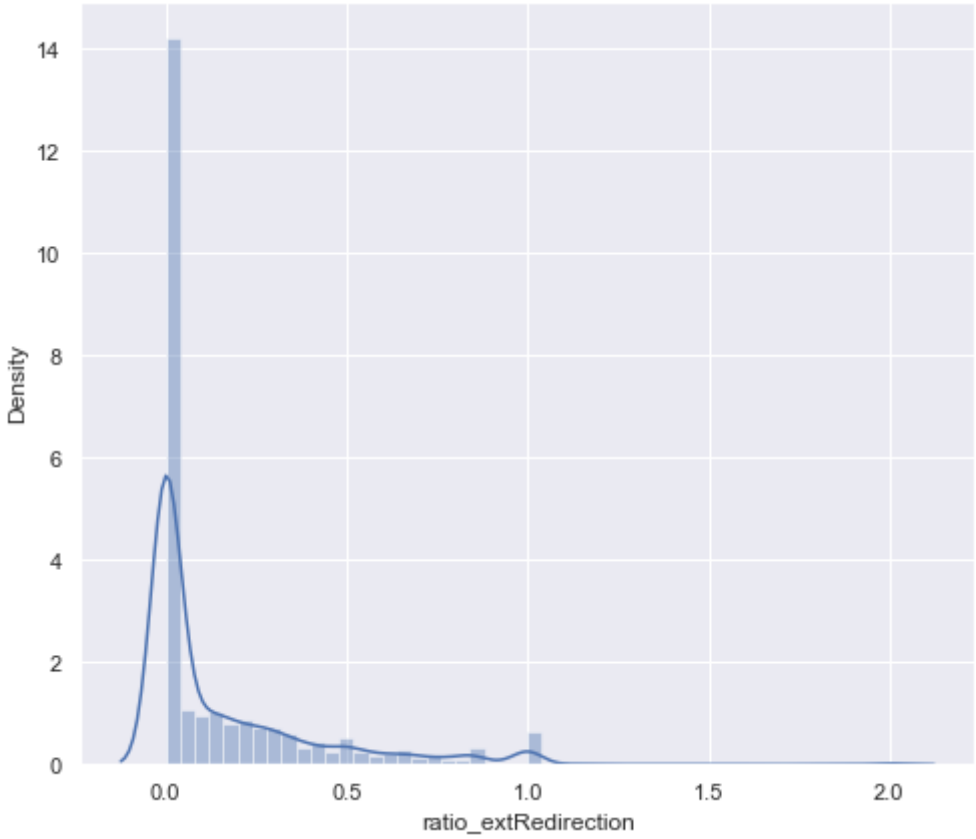

```
nb_extCSS
23.492395598390996
```



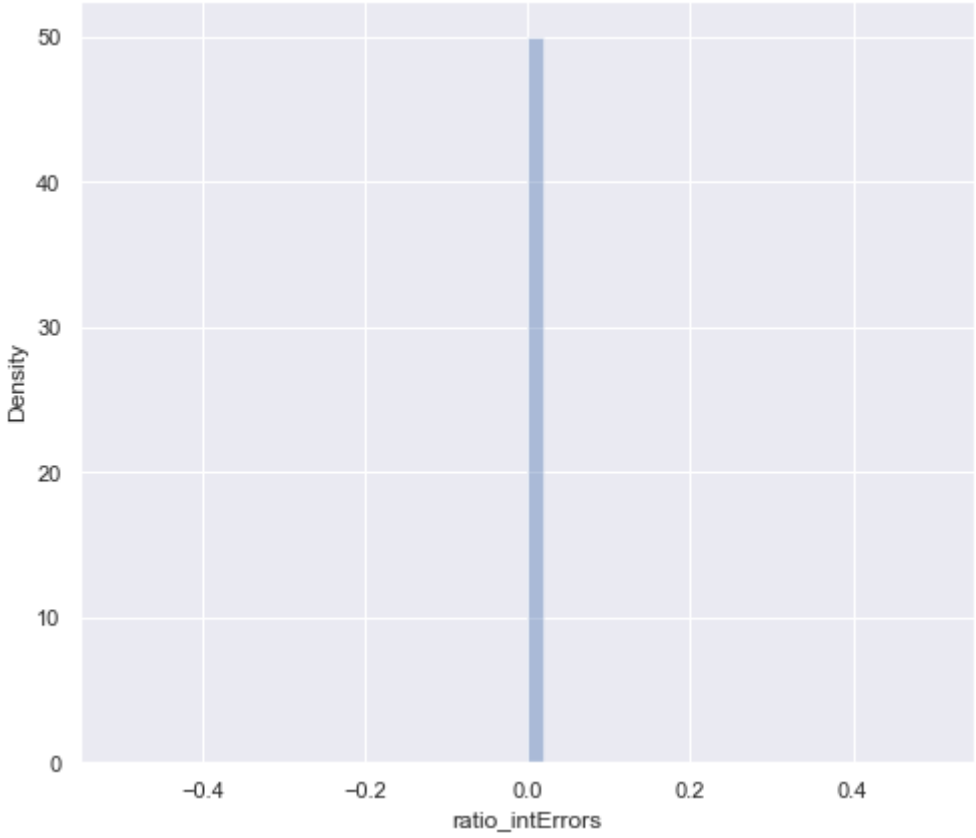

```
ratio_intRedirection
0.0
```



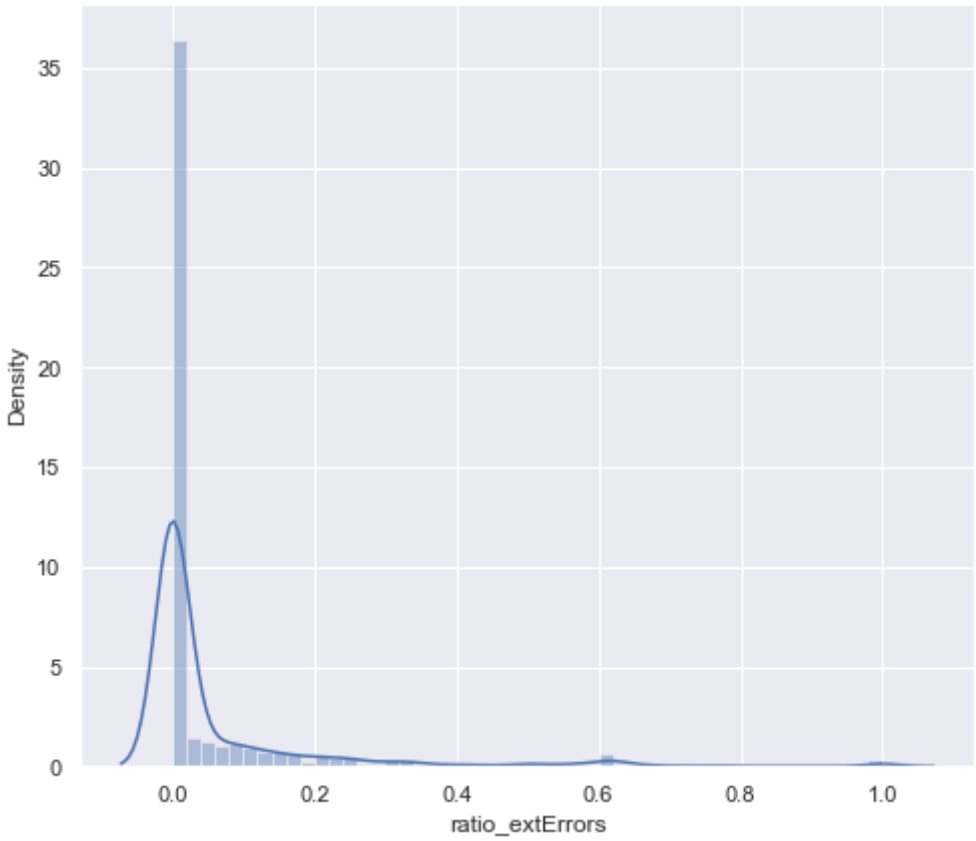

```
ratio_extRedirection
2.2965086895659725
```



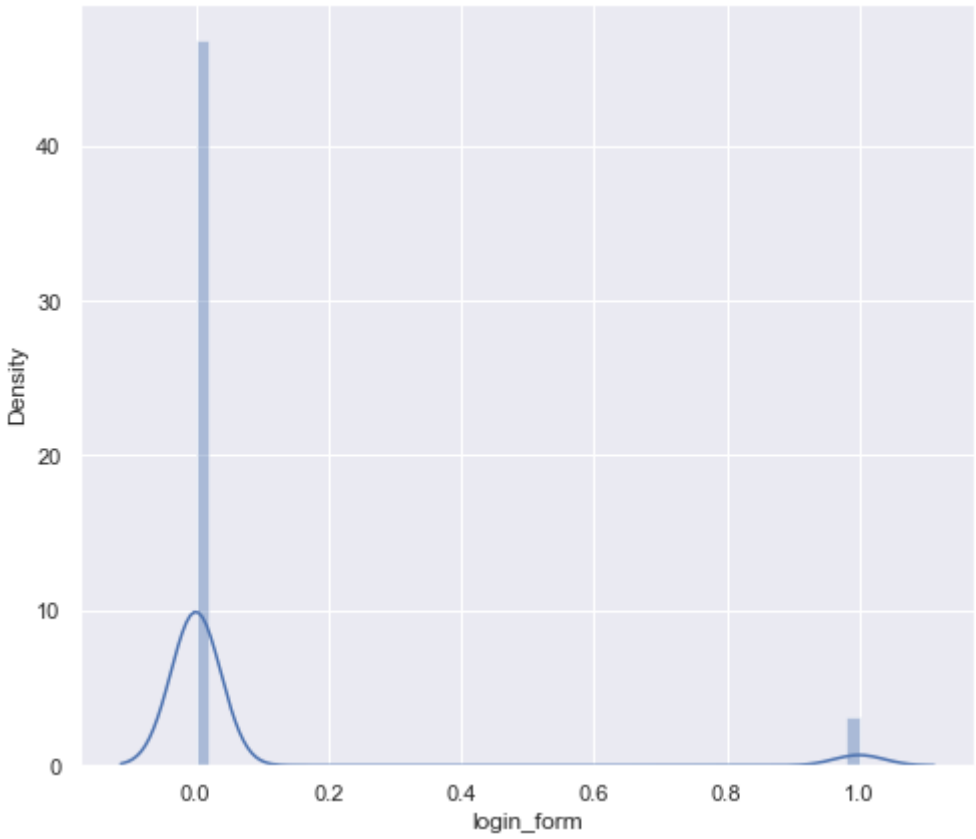

```
ratio_intErrors
0.0
```



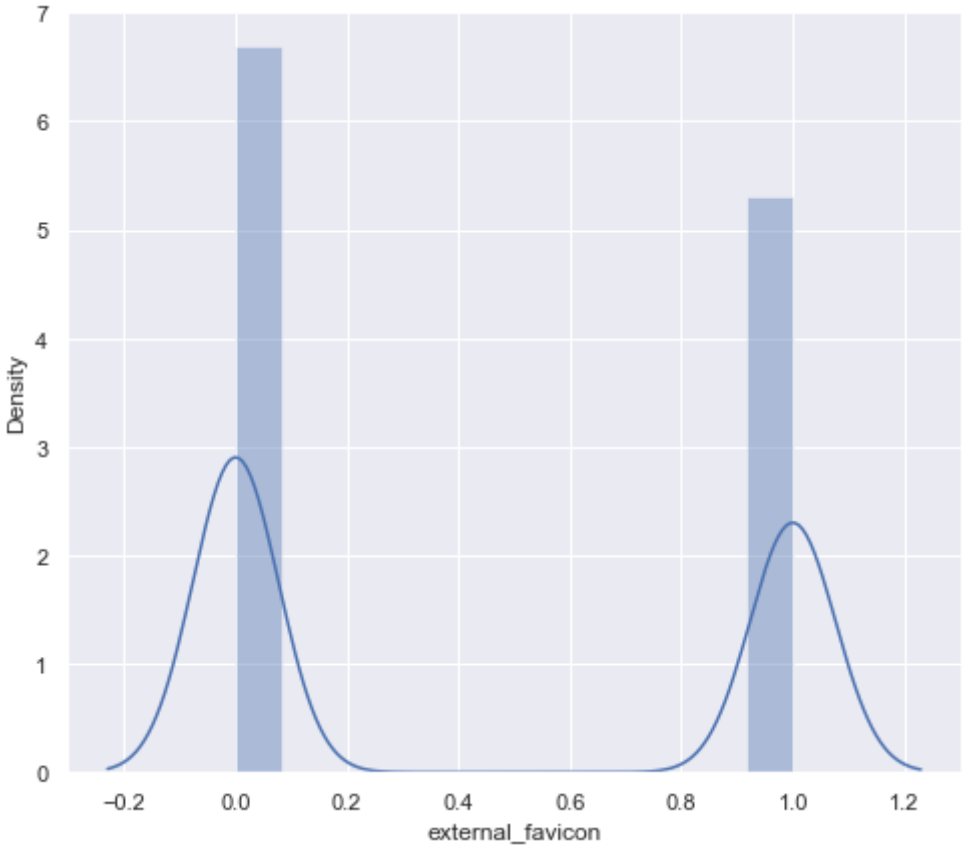

```
ratio_extErrors
3.510336776082494
```



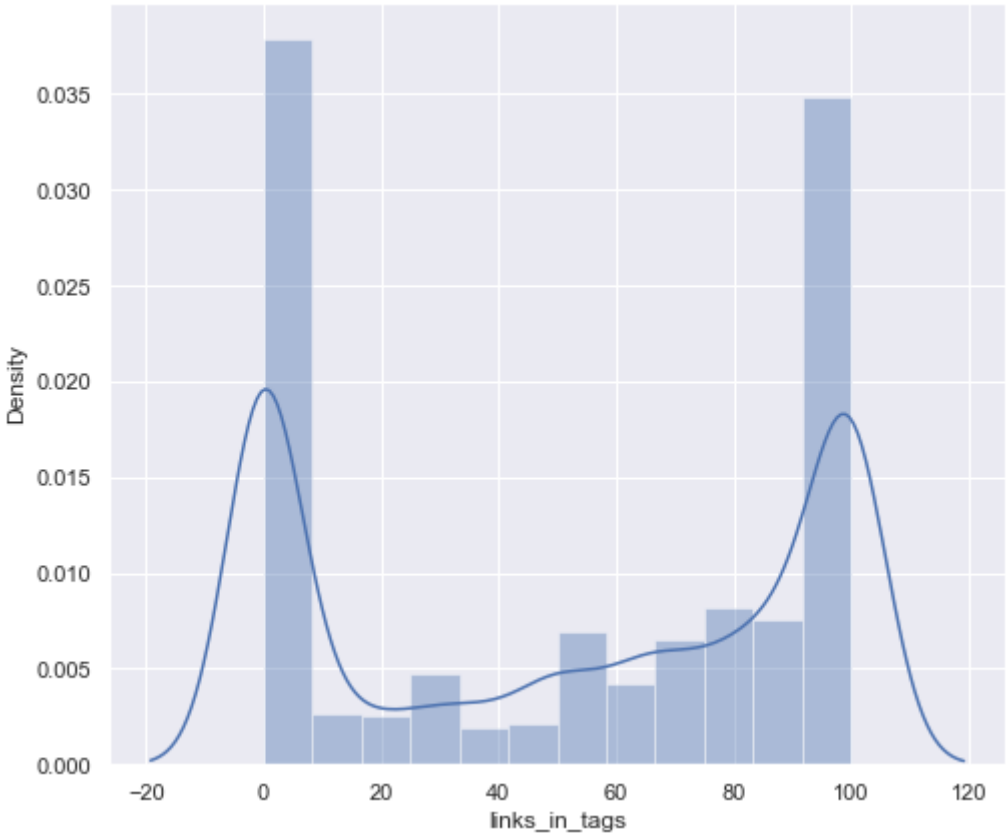
login_form
3.5763208309030783



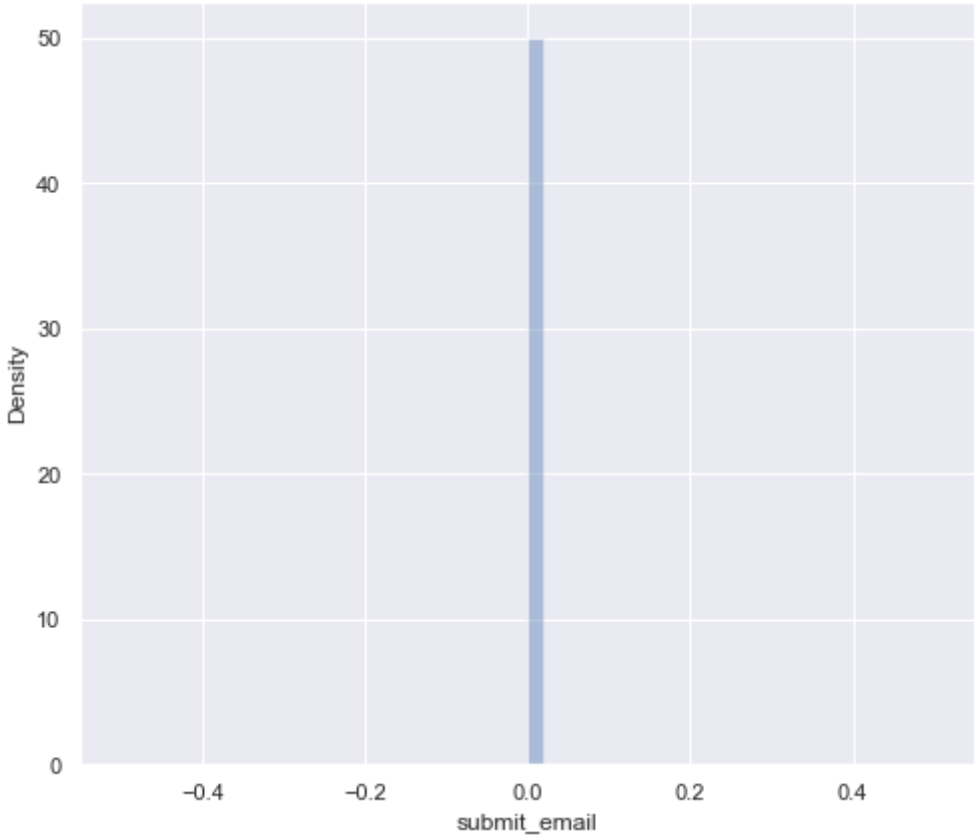
external_favicon
0.23288401622426355



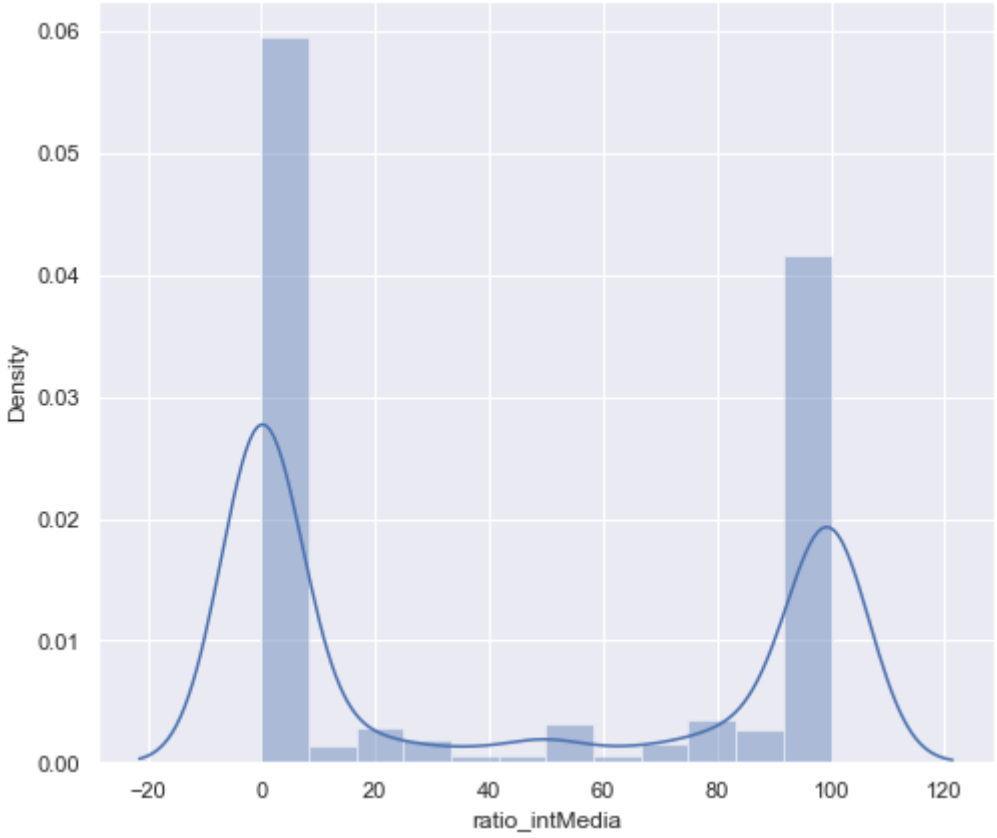
links_in_tags
-0.15074995414185735



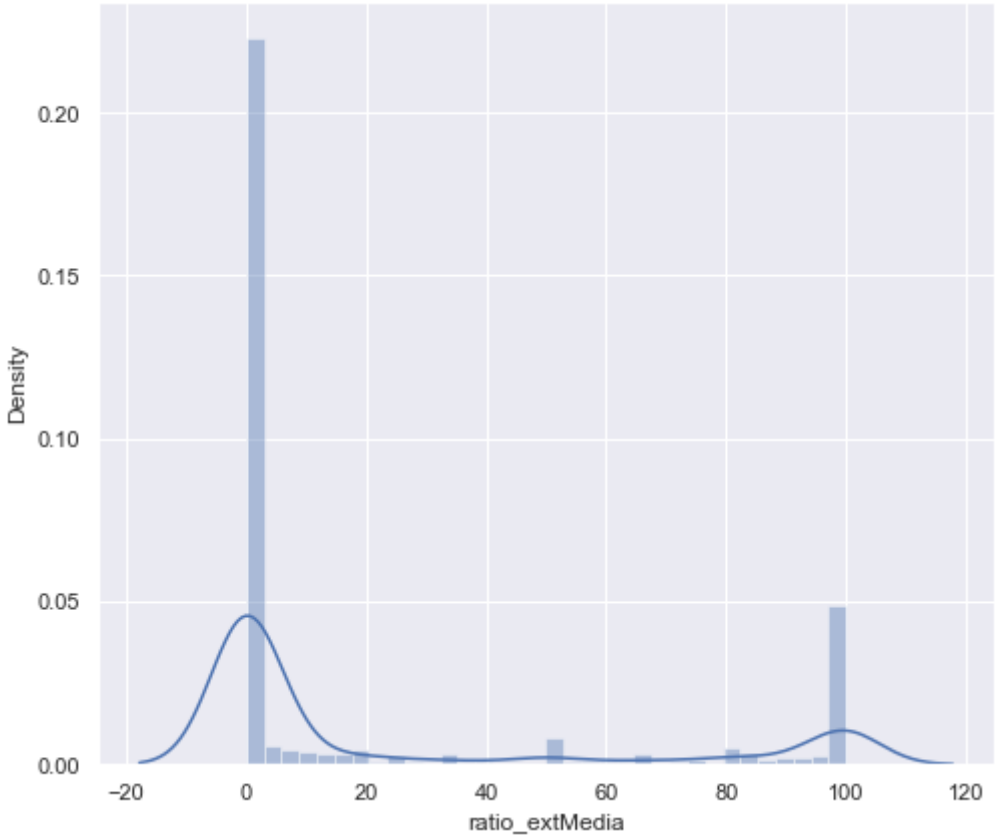
submit_email
0.0



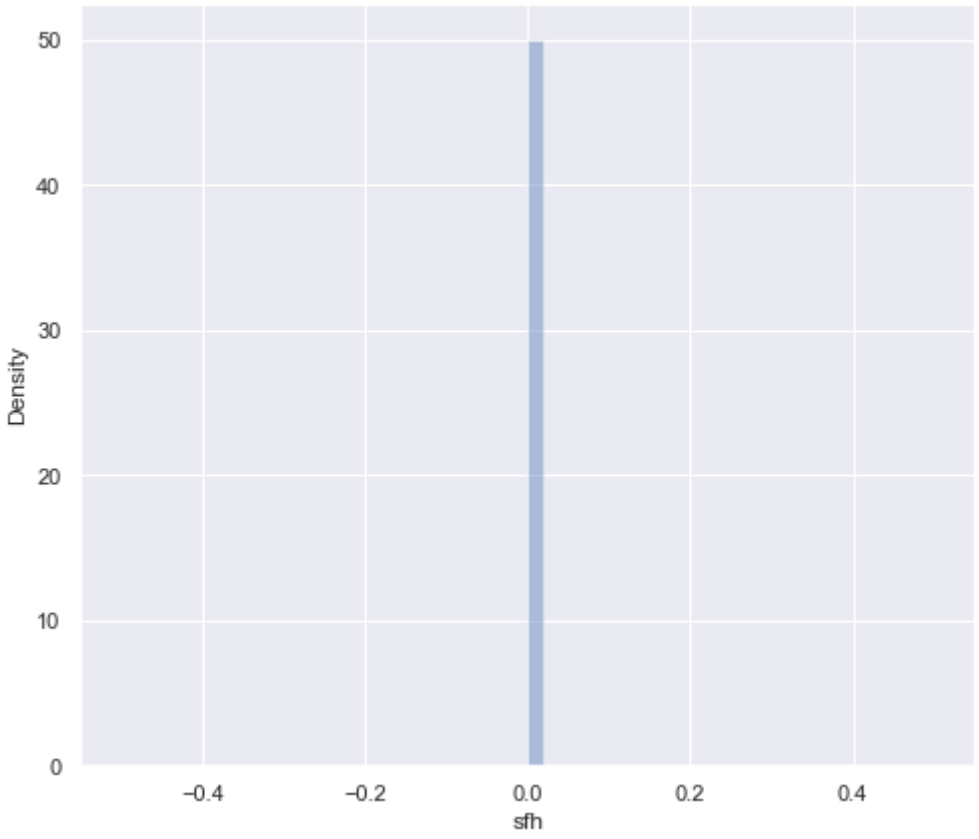

```
ratio_intMedia
0.27577677154052044
```



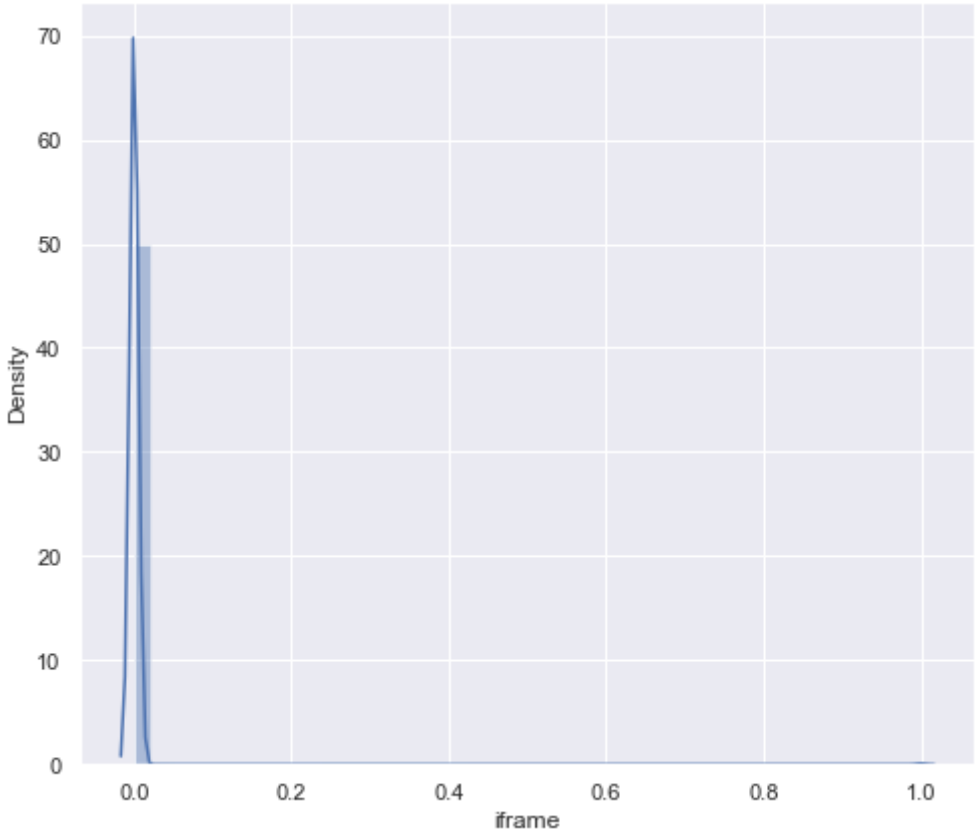

```
ratio_extMedia
1.265449187806516
```



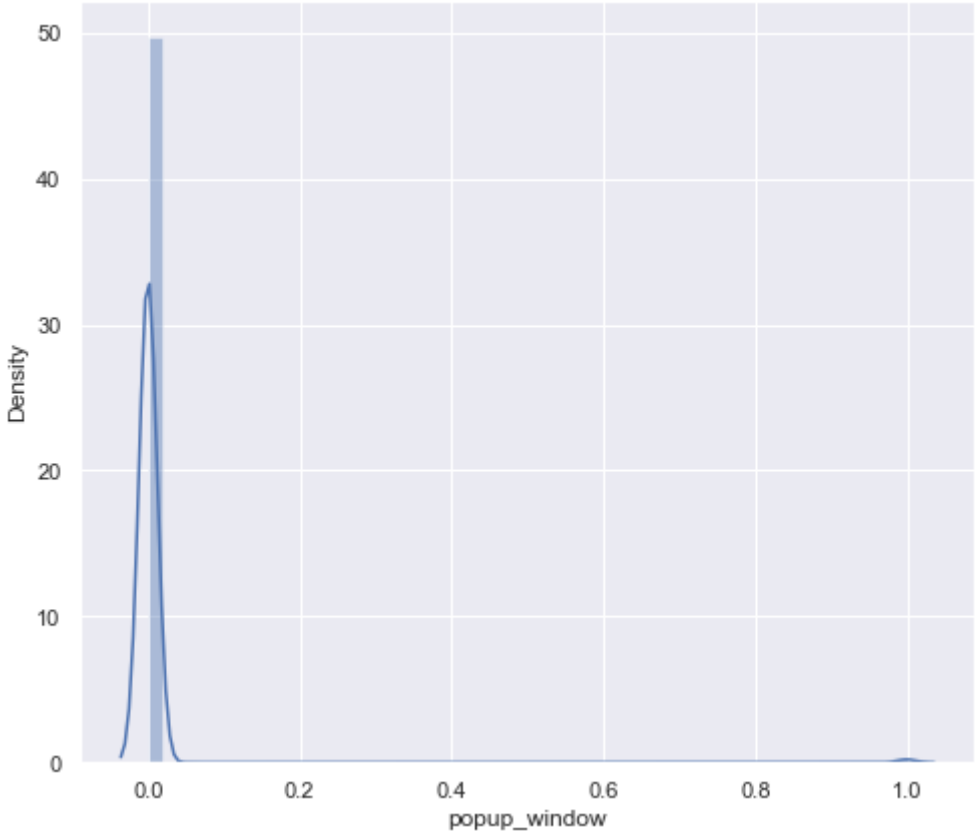
sfh
0.0



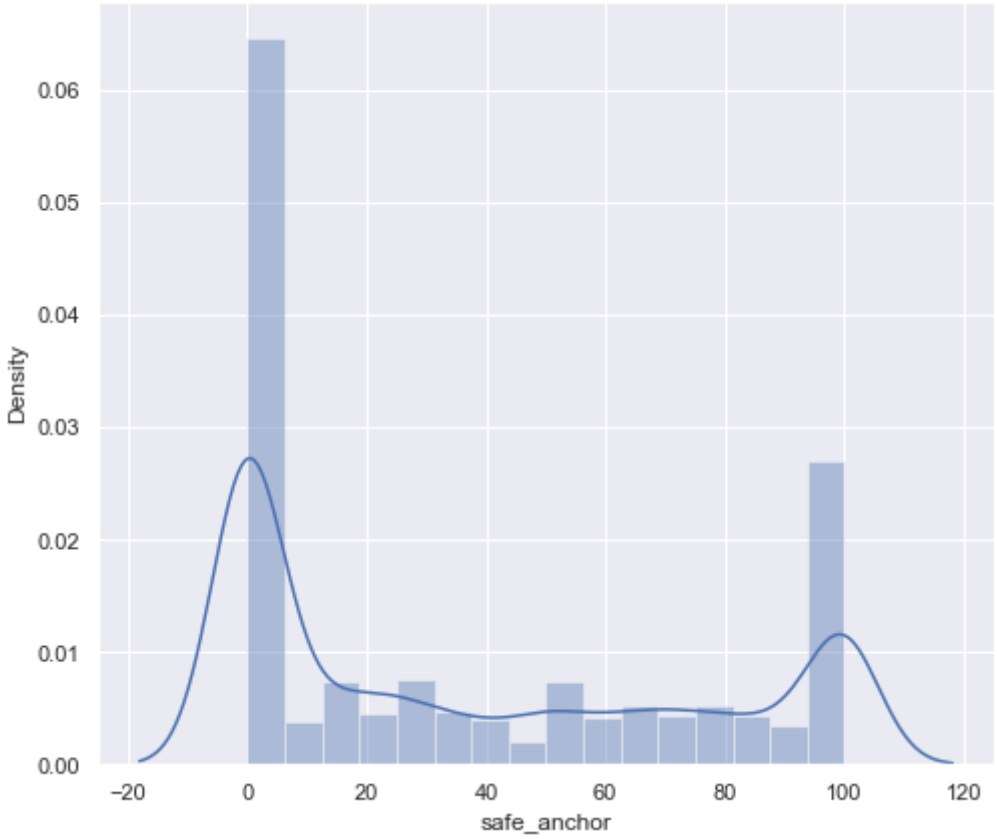
iframe
27.54997847658774



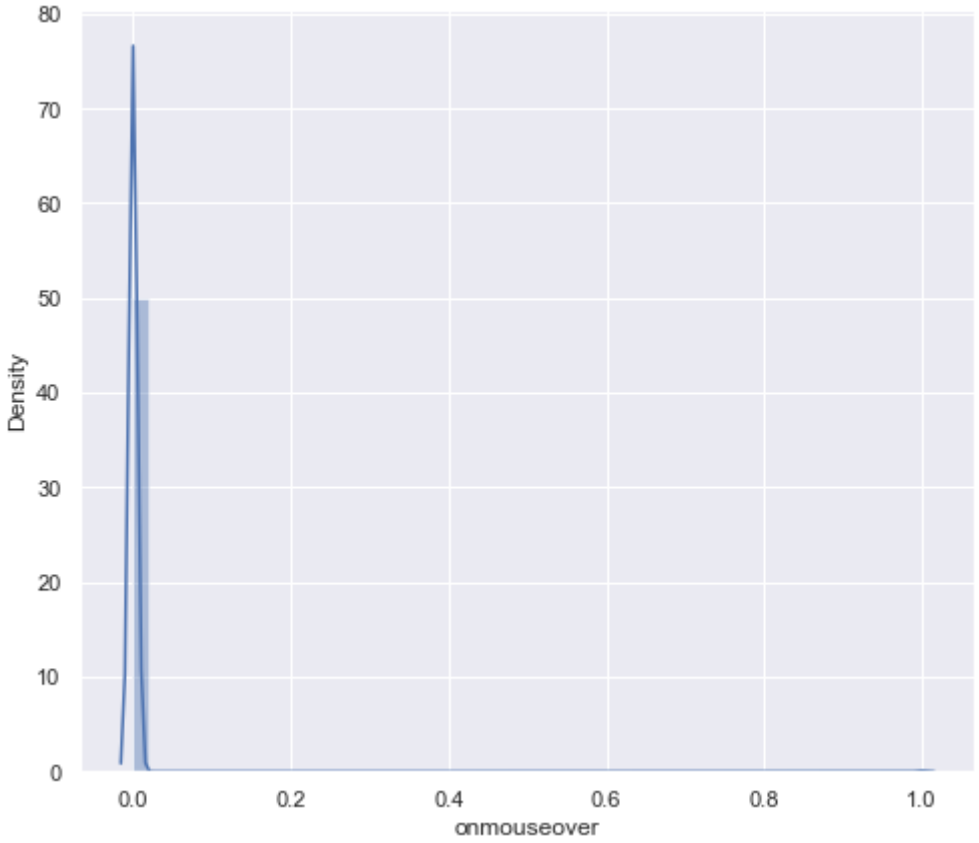
popup_window
12.753754244224499



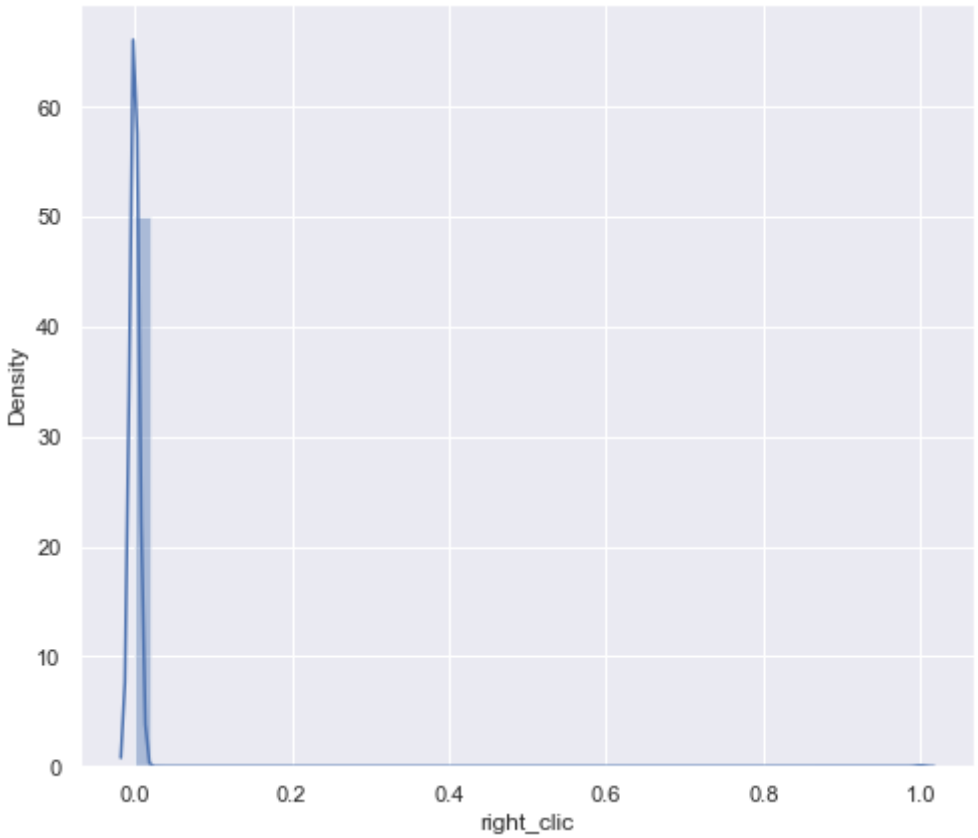
safe_anchor
0.5130078830338682



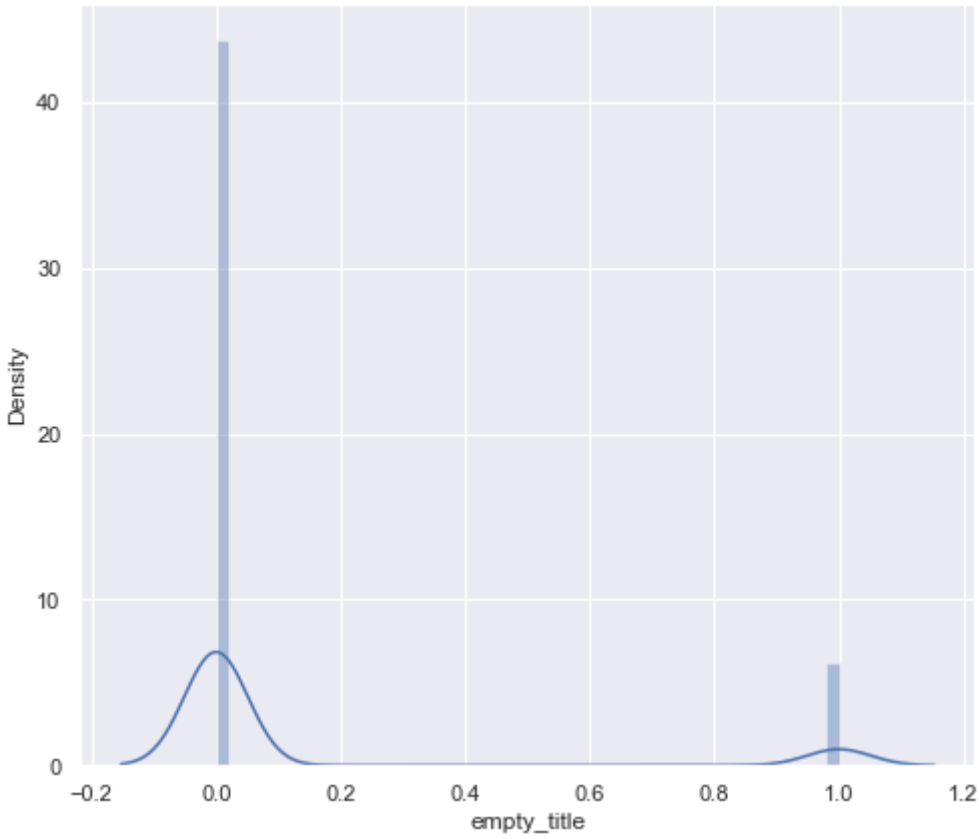
onmouseover
29.60121463527562



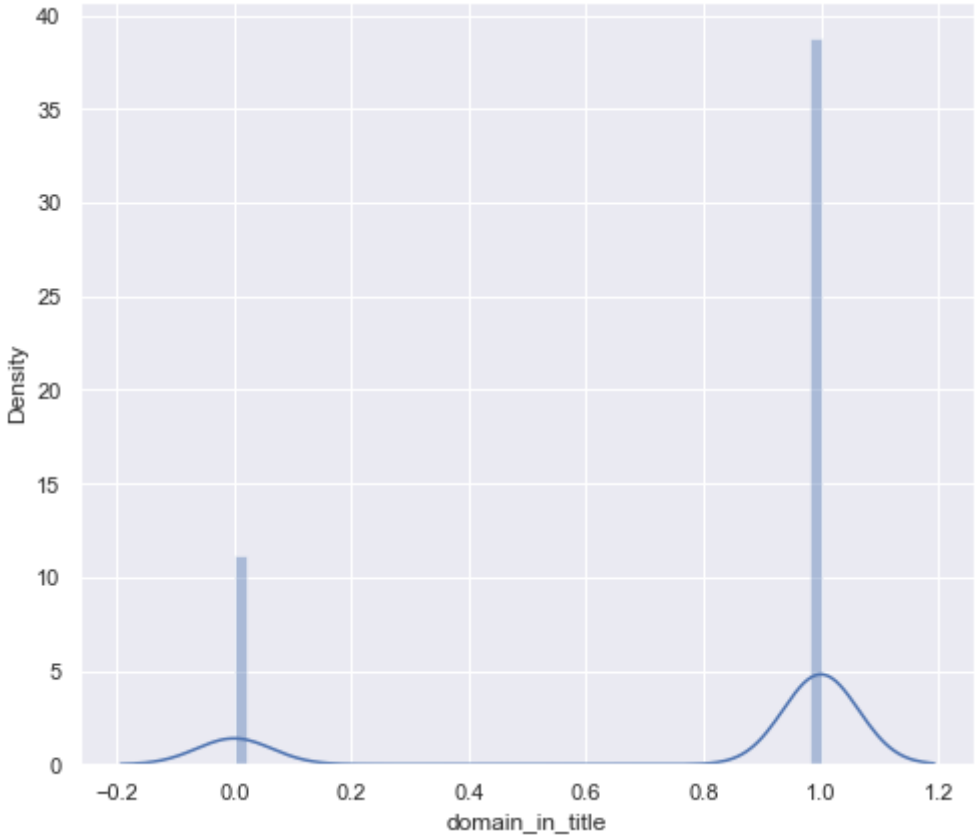
right_clic
26.67164040300631



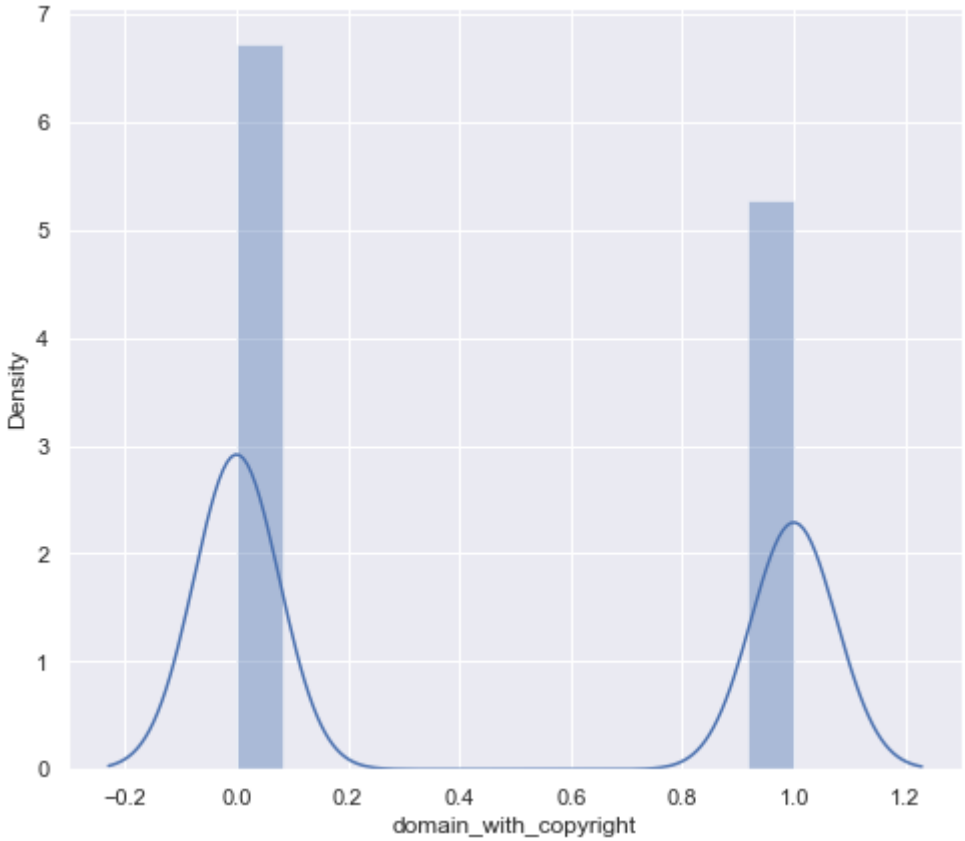

```
empty_title
2.2711166312615583
```



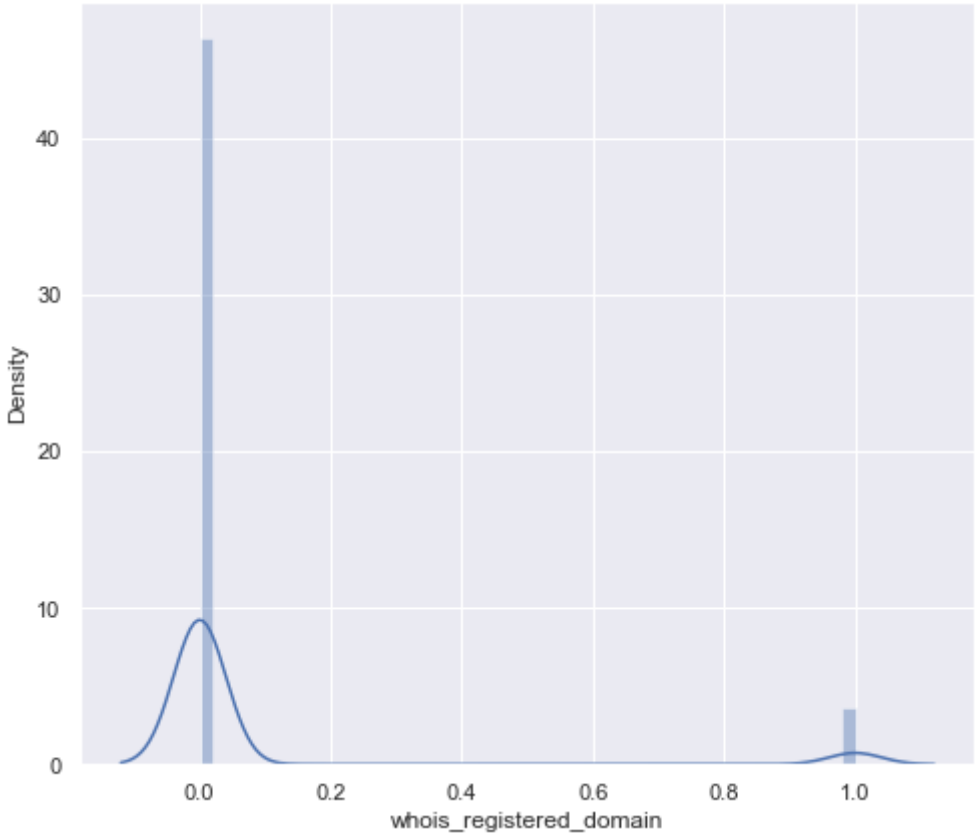

```
domain_in_title
-1.3229747684817161
```



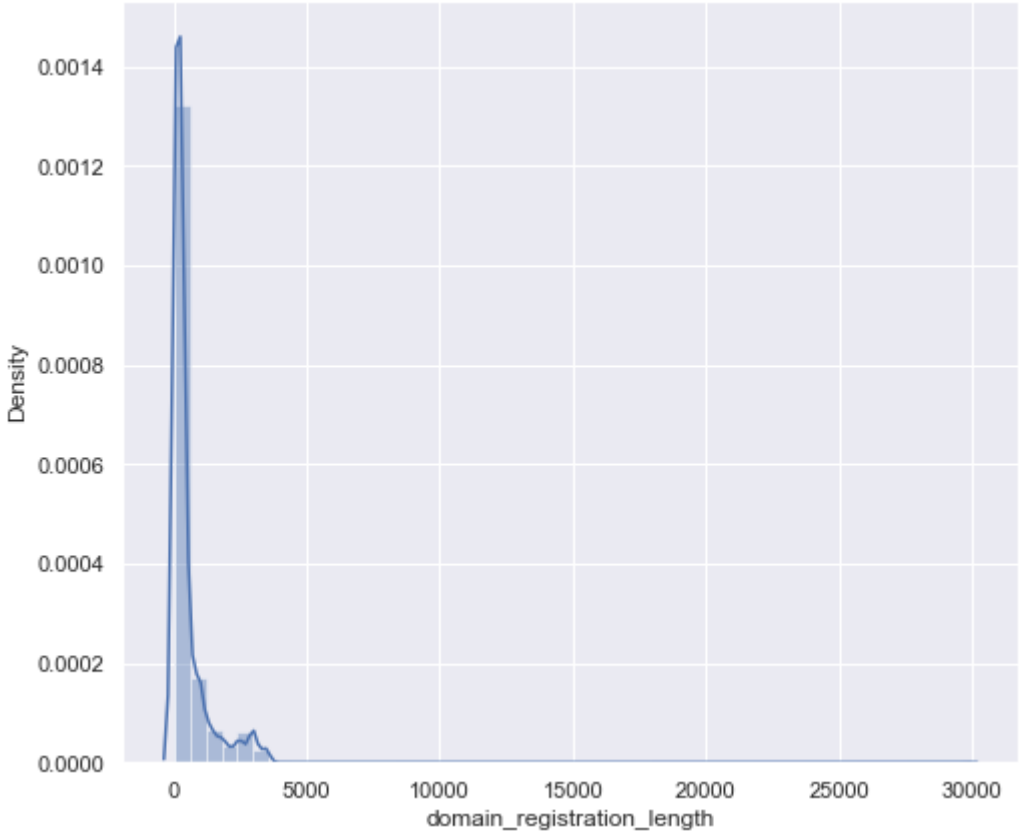
domain_with_copyright
0.24360699831816565



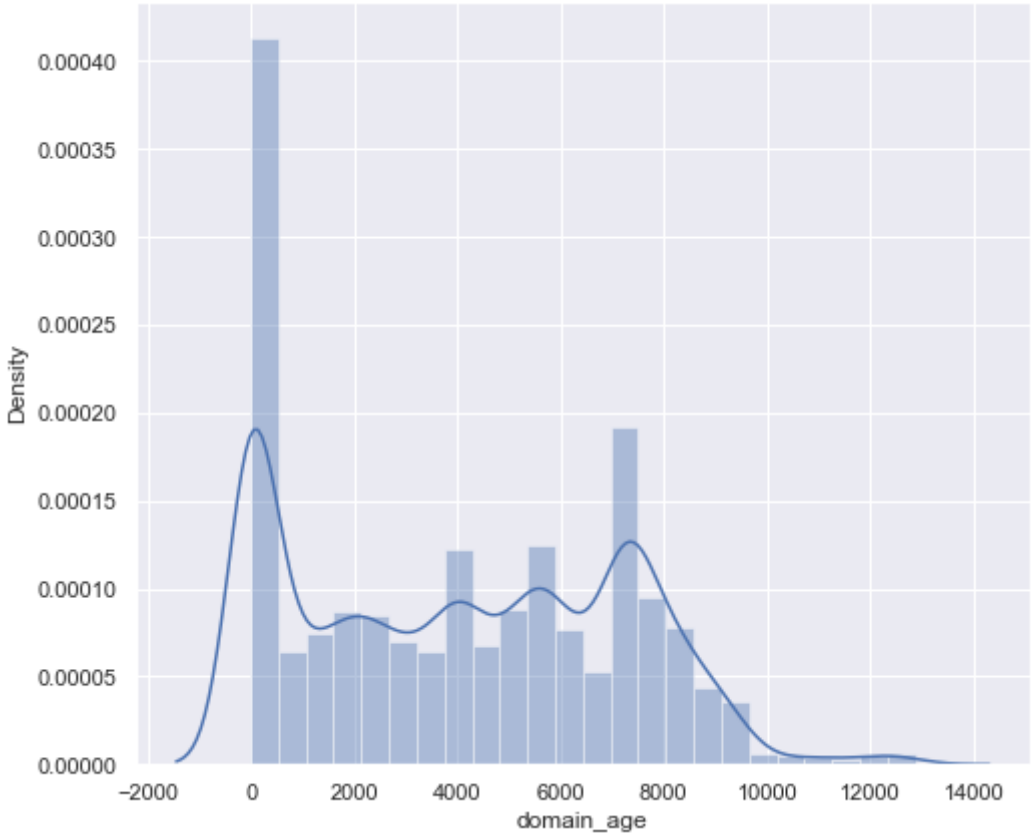
whois_registered_domain
3.2863499126547655



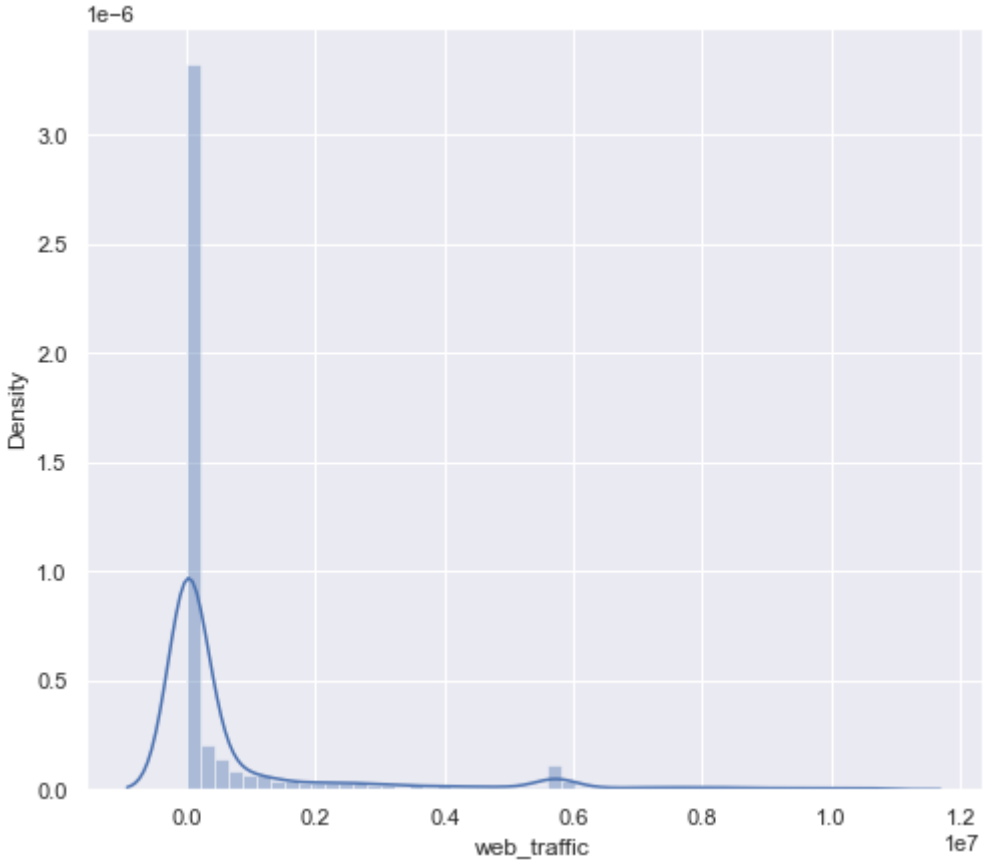
domain_registration_length
9.818318735154634



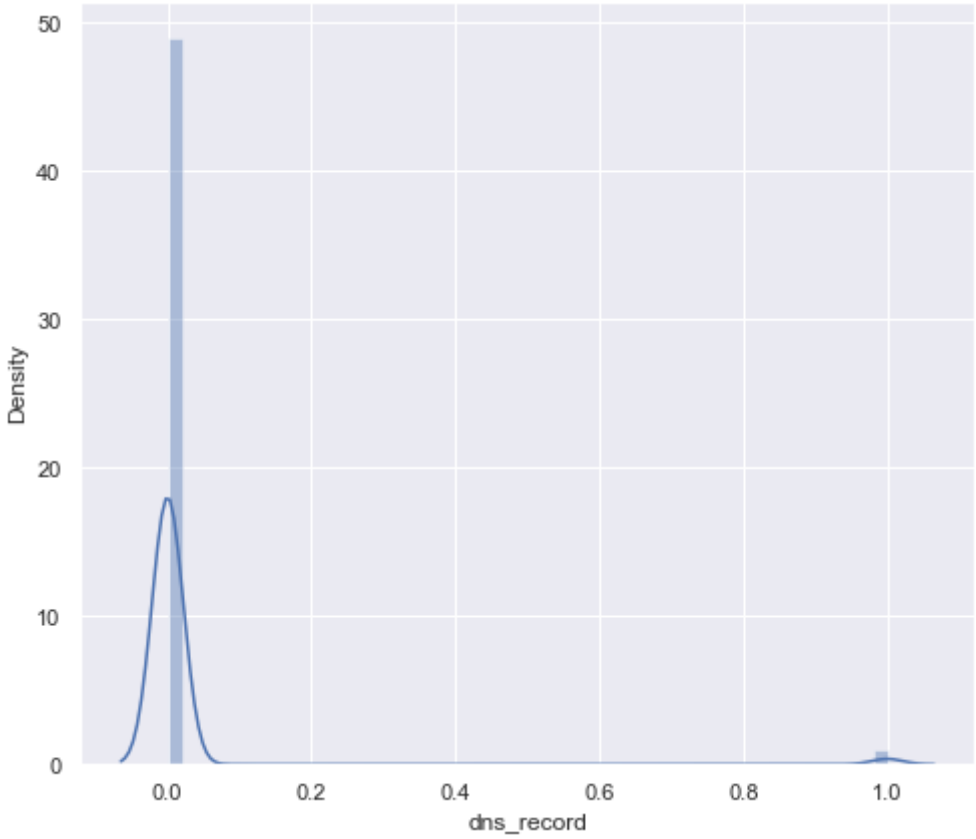
domain_age
0.1641651913111619



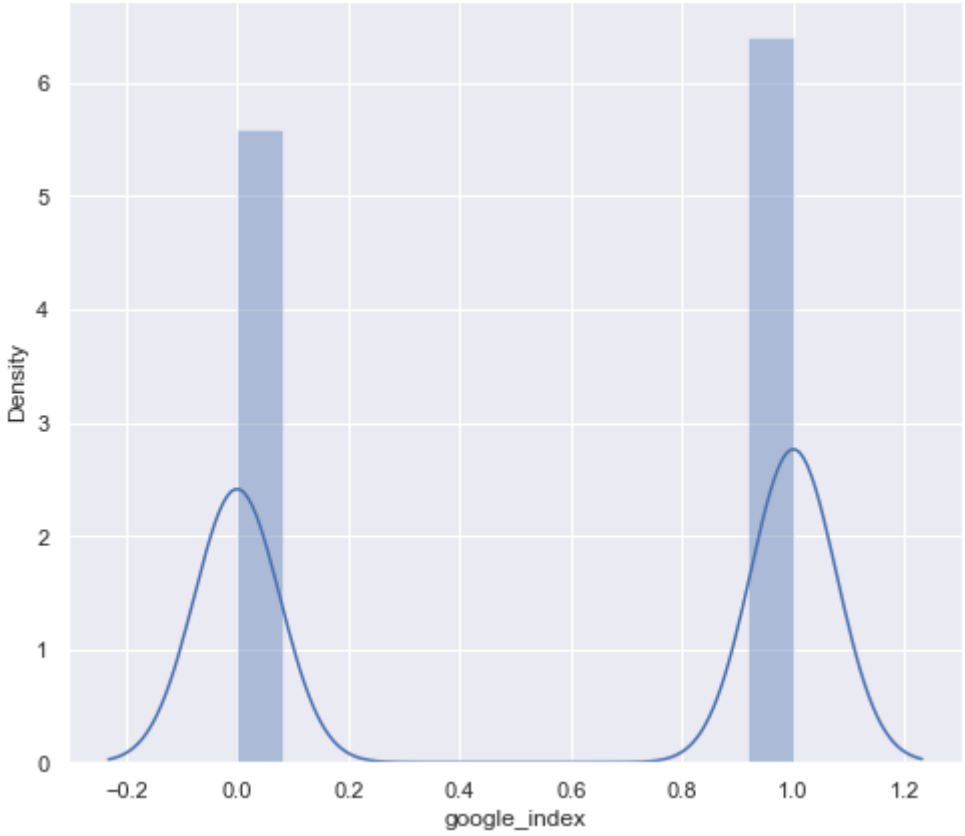
web_traffic
2.7789045210110026



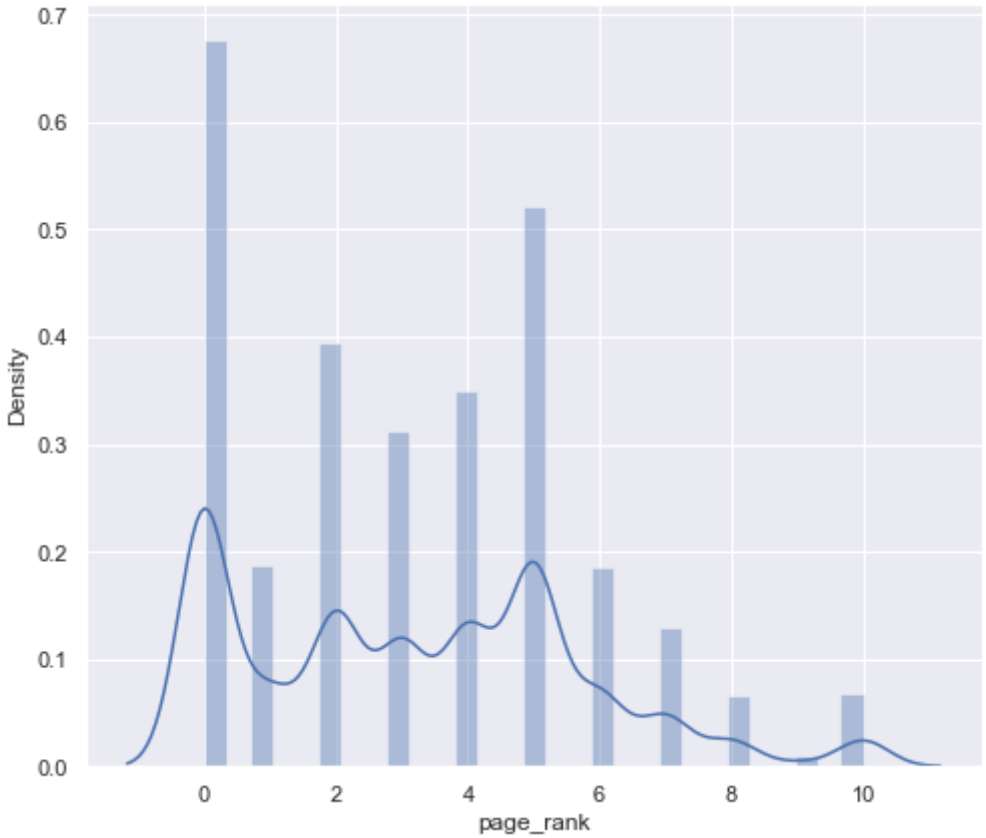
dns_record
6.834924131853895



google_index
-0.1360970425488085

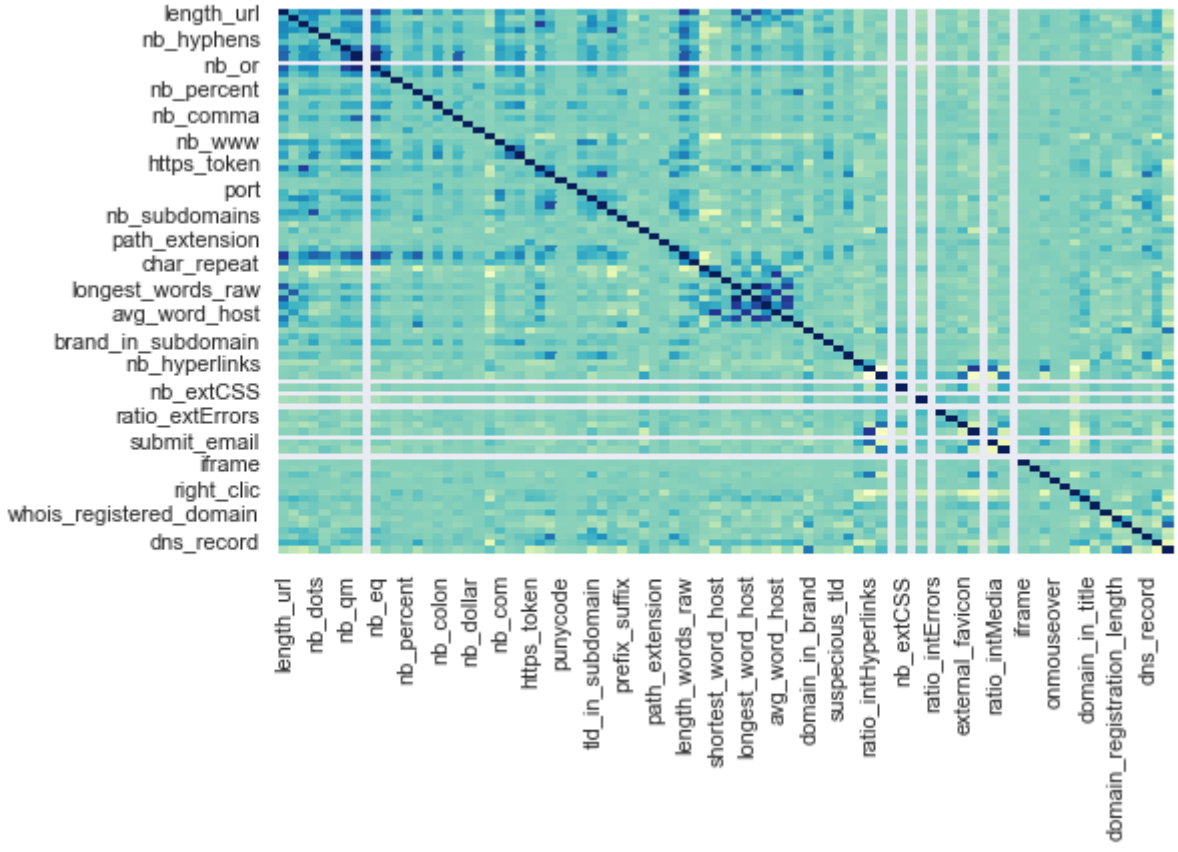


page_rank
0.4459724928934958



In [16]:
```
sns.set(rc={"figure.figsize":(8,5)})
sns.heatmap(dataset.corr(),cmap="YlGnBu",cbar=False)
```

Out[16]: <AxesSubplot:>



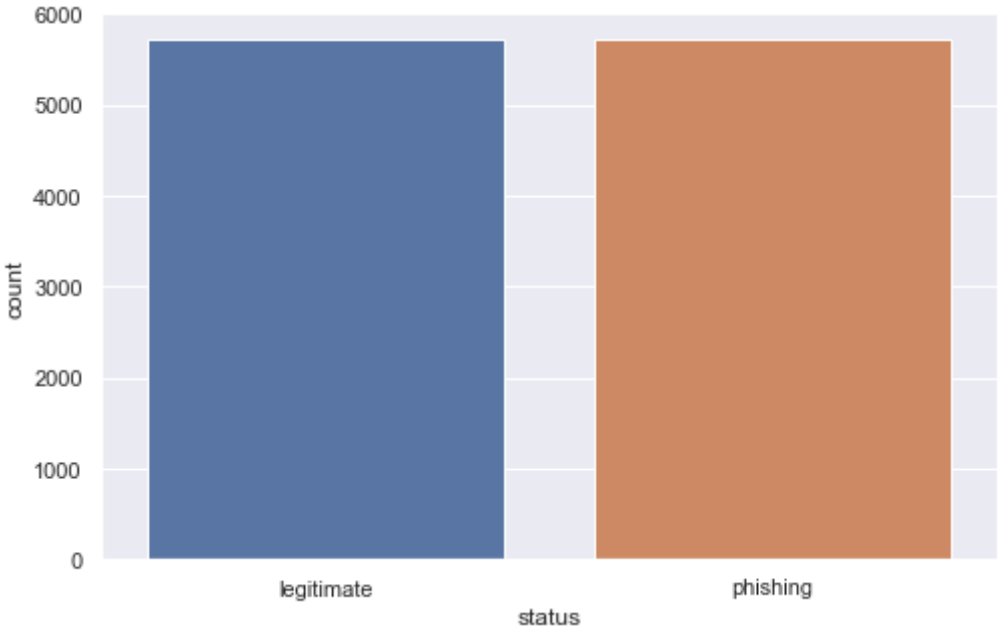
In [17]:
```
dataset["status"].value_counts()
```

Out[17]:
```
legitimate    5715
phishing      5715
```

```
Name: status, dtype: int64
```

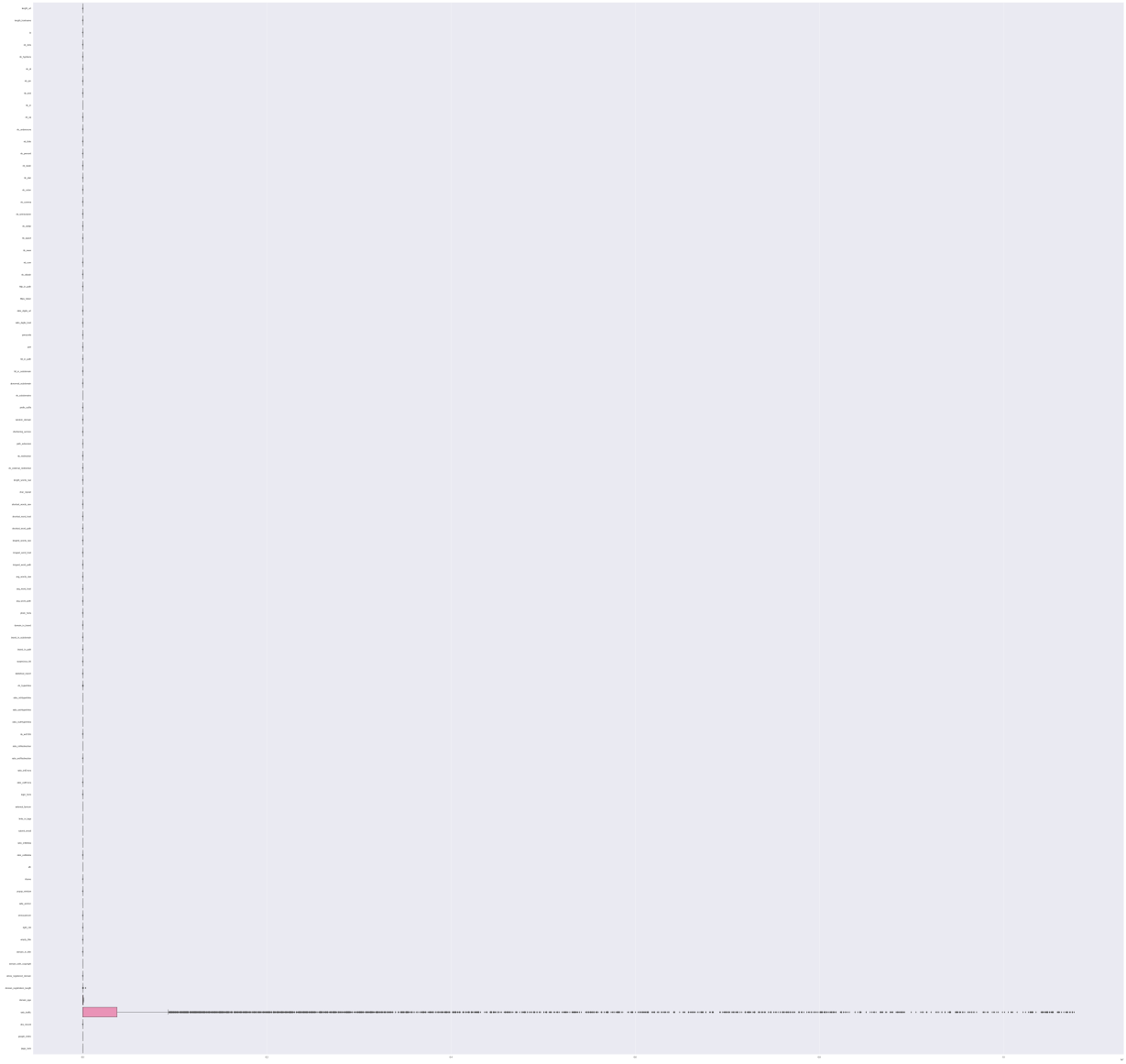
In [18]:
```
sns.countplot(x="status",data=dataset)
```

Out[18]:
```
<AxesSubplot:xlabel='status', ylabel='count'>
```



In [19]:
```
sns.set(rc={"figure.figsize":(100,99)})
sns.boxplot(data=dataset,orient="h")
```

Out[19]:
```
<AxesSubplot:>
```



In [20]:
```
dataset
```

Out[20]:

	URL	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

In [21]:
```
dataset["length_url"].value_counts()
```

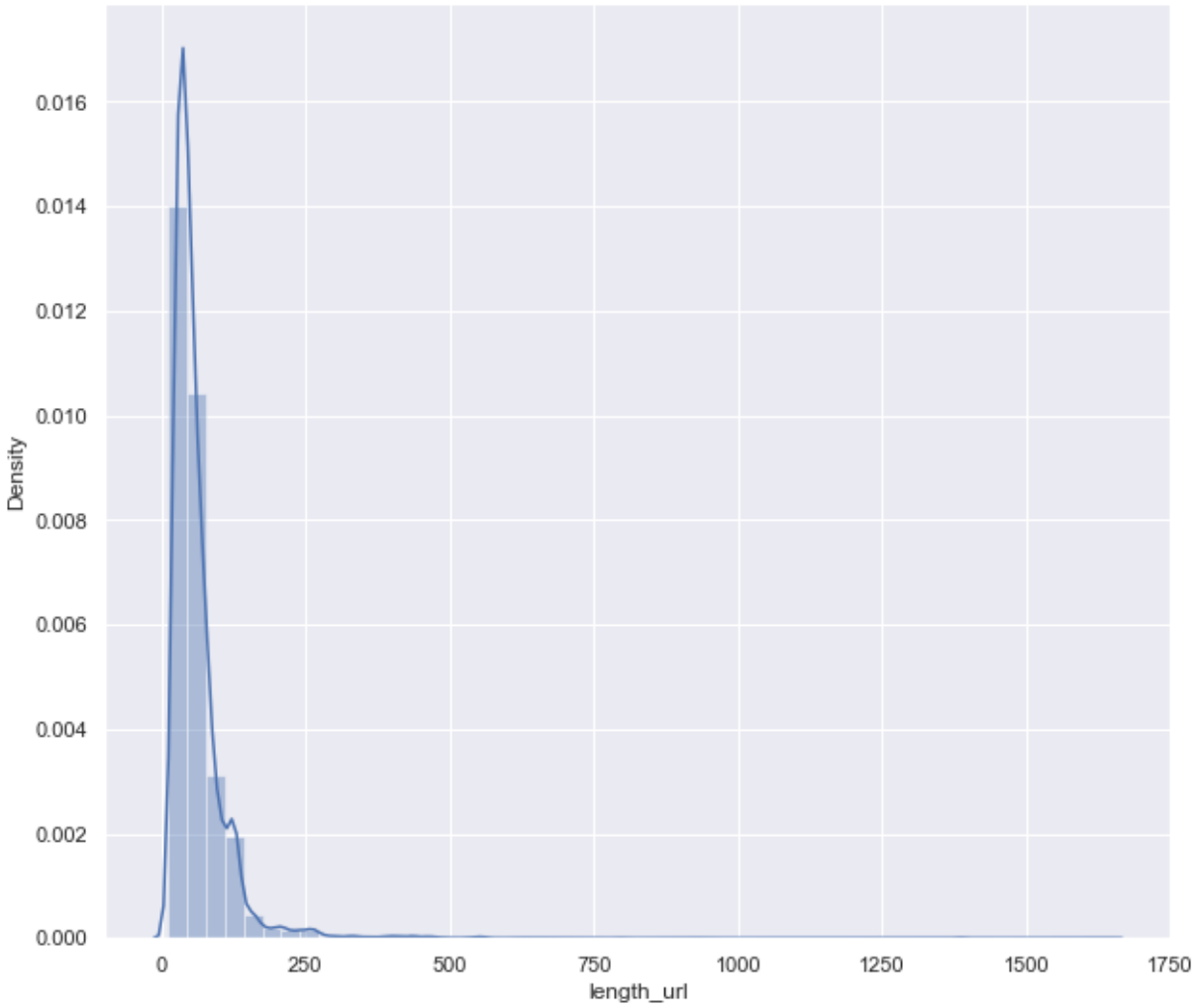
Out[21]:
```
26     251
29     250
32     250
33     230
27     230
      ...
403      1
395      1
339      1
315      1
907      1
Name: length_url, Length: 324, dtype: int64
```

In [22]:
```
dataset["length_hostname"].value_counts()
```

Out[22]:
```
16     956
15     754
18     731
17     725
14     702
      ...
75       1
179      1
211      1
87       1
95       1
Name: length_hostname, Length: 83, dtype: int64
```

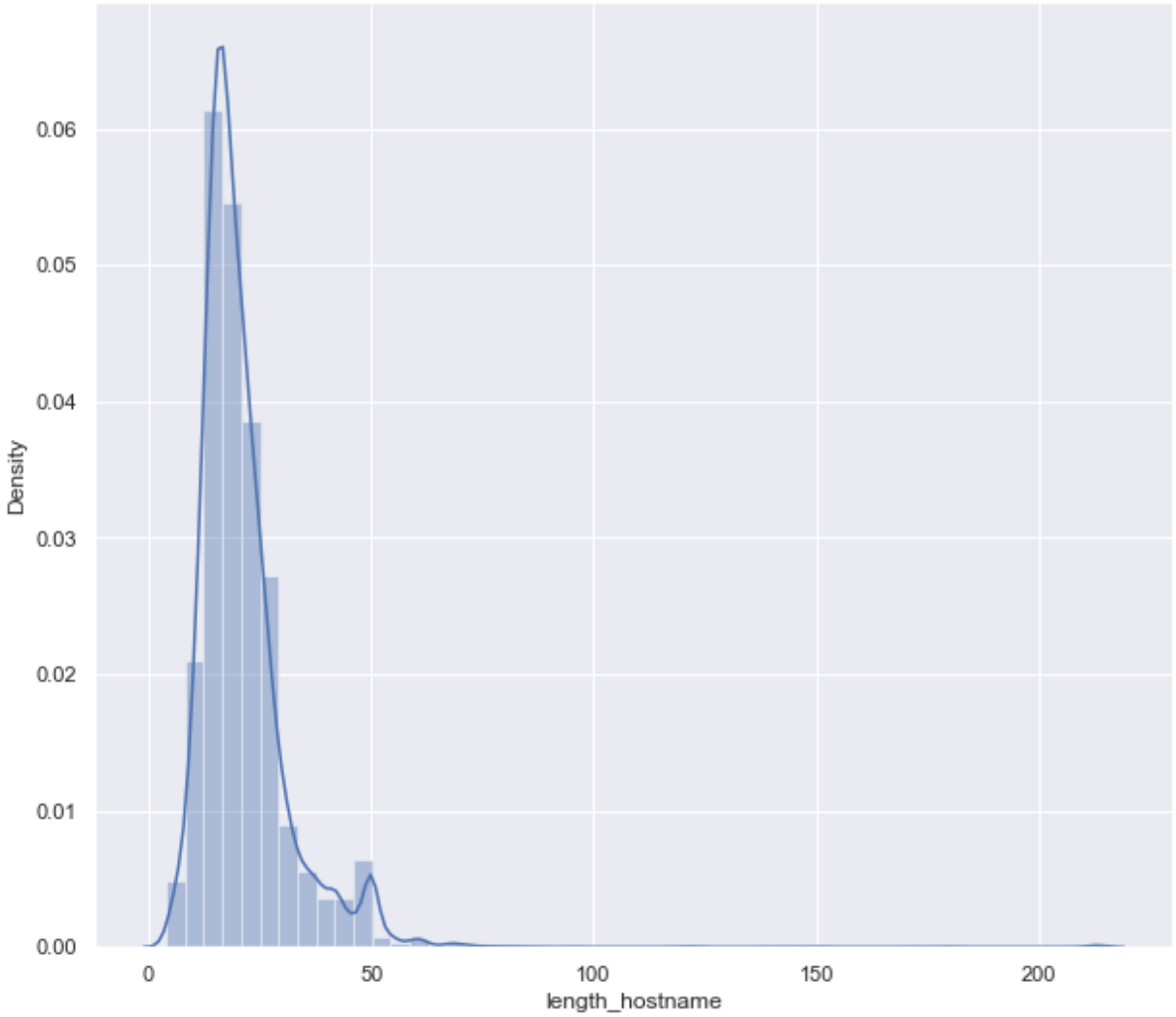
In [23]:
```
sns.set(rc={"figure.figsize":(10,9)})
sns.distplot(dataset.length_url)
```

Out[23]:
```
<AxesSubplot:xlabel='length_url', ylabel='Density'>
```



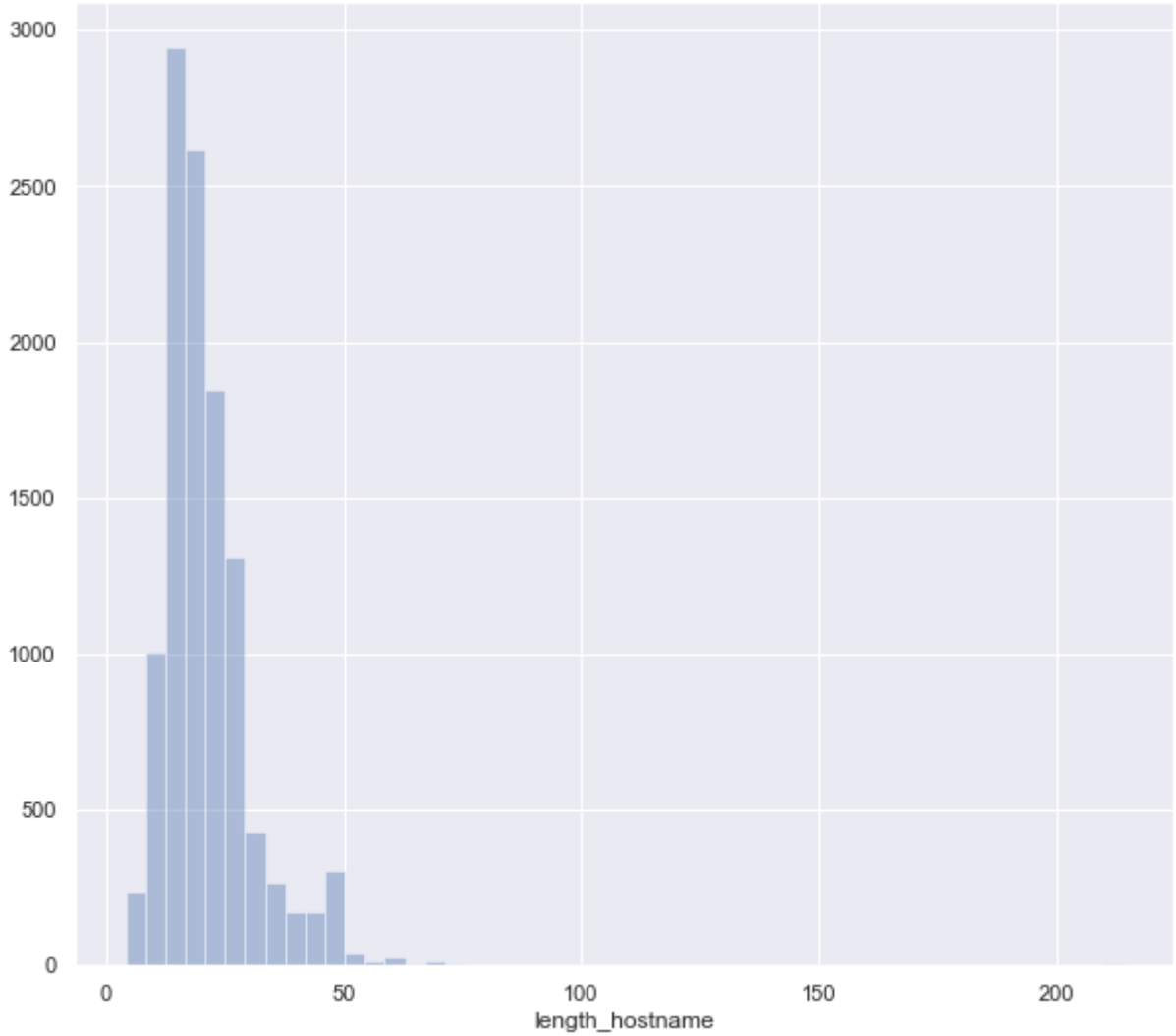
In [24]:
```
sns.distplot(dataset.length_hostname)
```

Out[24]:
```
<AxesSubplot:xlabel='length_hostname', ylabel='Density'>
```



In [25]:
```
sns.distplot(dataset["length_hostname"],kde=False)
```

Out[25]:
```
<AxesSubplot:xlabel='length_hostname'>
```



In [26]:
```
dataset
```

Out[26]:

	URL	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

In [27]:
```
data
```

Out[27]:
```
Index(['                                                                  UR
L',
       'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at',
       'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore', 'nb_tilde',
       'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma',
       'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com',
       'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url',
       'ratio_digits_host', 'punycode', 'port', 'tld_in_path',
       'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains',
       'prefix_suffix', 'random_domain', 'shortening_service',
       'path_extension', 'nb_redirection', 'nb_external_redirection',
       'length_words_raw', 'char_repeat', 'shortest_words_raw',
       'shortest_word_host', 'shortest_word_path', 'longest_words_raw',
       'longest_word_host', 'longest_word_path', 'avg_words_raw',
       'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand',
       'brand_in_subdomain', 'brand_in_path', 'suspecious_tld',
       'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks',
       'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS',
       'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors',
       'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags',
       'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe',
       'popup_window', 'safe_anchor', 'onmouseover', 'right_clic',
       'empty_title', 'domain_in_title', 'domain_with_copyright',
       'whois_registered_domain', 'domain_registration_length', 'domain_age',
       'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status'],
      dtype='object')
```

In [28]:
```
dataset.describe()
```

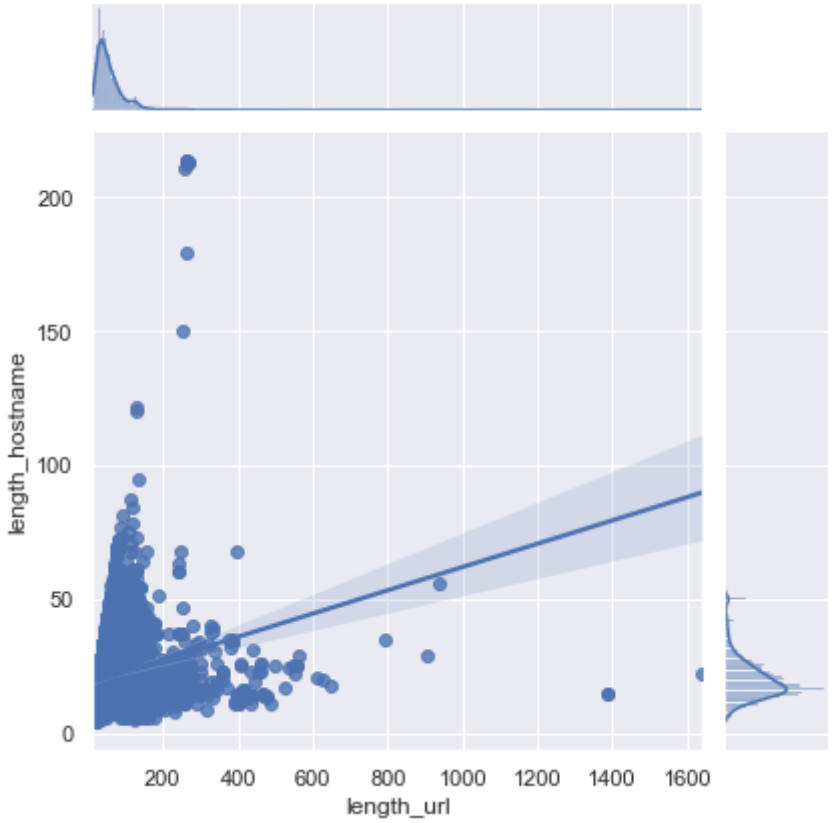
Out[28]:

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	
count	11430.000000	11430.000000	11430.000000	11430.000000	11430.000000	11430.000000	114
mean	61.126684	21.090289	0.150569	2.480752	0.997550	0.022222	
std	55.297318	10.777171	0.357644	1.369686	2.087087	0.155500	
min	12.000000	4.000000	0.000000	1.000000	0.000000	0.000000	
25%	33.000000	15.000000	0.000000	2.000000	0.000000	0.000000	
50%	47.000000	19.000000	0.000000	2.000000	0.000000	0.000000	
75%	71.000000	24.000000	0.000000	3.000000	1.000000	0.000000	
max	1641.000000	214.000000	1.000000	24.000000	43.000000	4.000000	

8 rows × 87 columns

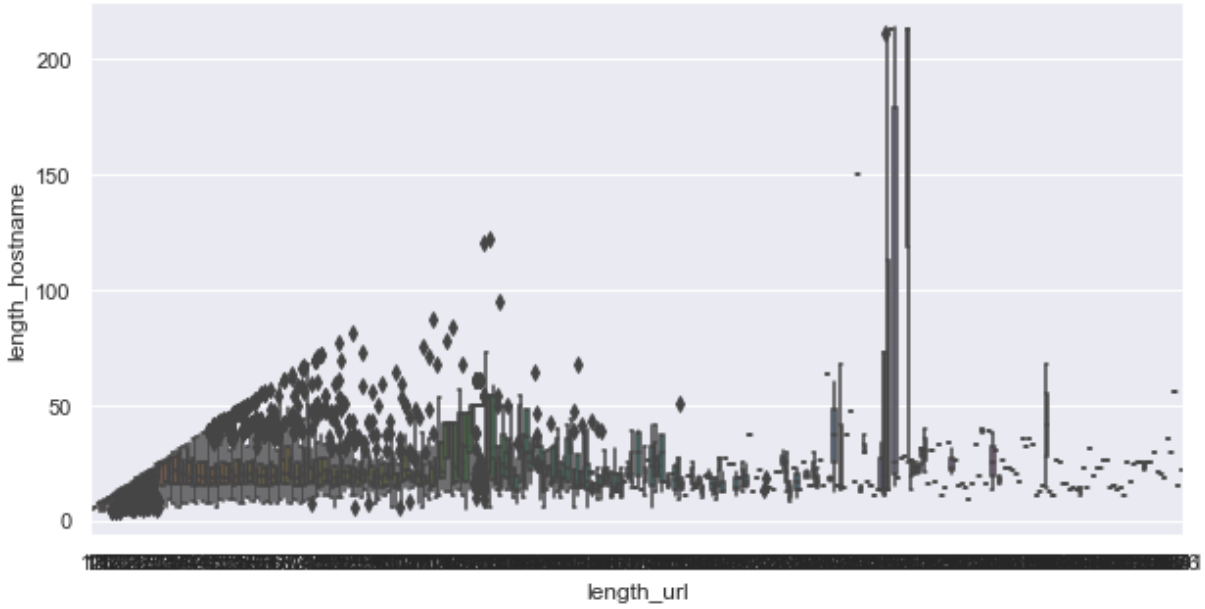
In [29]:
```
sns.jointplot(x="length_url",y="length_hostname",data=dataset,kind="reg")
```

Out[29]:
```
<seaborn.axisgrid.JointGrid at 0x1b56c30ab80>
```



In [30]:
```
sns.set(rc={"figure.figsize":(10,5)})
sns.boxplot(x="length_url",y="length_hostname",data=dataset)
```

Out[30]:
```
<AxesSubplot:xlabel='length_url', ylabel='length_hostname'>
```



In [31]:
```
new_dataset=dataset.drop(["nb_or","sfh","submit_email","ratio_intErrors","ratio_intR
```

In [32]:
```
new_dataset.columns
```

Out[32]:
```
Index(['                                                                UR
L',
       'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at',
       'nb_qm', 'nb_and', 'nb_eq', 'nb_underscore', 'nb_tilde', 'nb_percent',
       'nb_slash', 'nb_star', 'nb_colon', 'nb_comma', 'nb_semicolumn',
       'nb_dollar', 'nb_space', 'nb_www', 'nb_com', 'nb_dslash',
       'http_in_path', 'https_token', 'ratio_digits_url', 'ratio_digits_host',
       'punycode', 'port', 'tld_in_path', 'tld_in_subdomain',
```

```
       'abnormal_subdomain', 'nb_subdomains', 'prefix_suffix', 'random_domain',
       'shortening_service', 'path_extension', 'nb_redirection',
       'nb_external_redirection', 'length_words_raw', 'char_repeat',
       'shortest_words_raw', 'shortest_word_host', 'shortest_word_path',
       'longest_words_raw', 'longest_word_host', 'longest_word_path',
       'avg_words_raw', 'avg_word_host', 'avg_word_path', 'phish_hints',
       'domain_in_brand', 'brand_in_subdomain', 'brand_in_path',
       'suspecious_tld', 'statistical_report', 'nb_hyperlinks',
       'ratio_intHyperlinks', 'ratio_extHyperlinks', 'nb_extCSS',
       'ratio_extRedirection', 'ratio_extErrors', 'login_form',
       'external_favicon', 'links_in_tags', 'ratio_intMedia', 'ratio_extMedia',
       'iframe', 'popup_window', 'safe_anchor', 'onmouseover', 'right_clic',
       'empty_title', 'domain_in_title', 'domain_with_copyright',
       'whois_registered_domain', 'domain_registration_length', 'domain_age',
       'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status'],
      dtype='object')
```

In [33]:
```
new_dataset=new_dataset.rename(columns={"
```

In [34]:
```
new_dataset
```

Out[34]:

	Url	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleId.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 83 columns

In [35]:
```
x=new_dataset.iloc[:,1:-1].values
```

In [36]:
```
print(x)
```

```
[[ 37.  19.   0. ...   1.   1.   4.]
 [ 77.  23.   1. ...   0.   1.   2.]
 [126.  50.   1. ...   0.   1.   0.]
 ...
 [105.  16.   1. ...   0.   1.  10.]
 [ 38.  30.   0. ...   0.   0.   4.]
 [477.  14.   1. ...   1.   1.   0.]]
```

In [37]:
```
y=new_dataset.iloc[:,-1].values
```

In [38]:
```
print(y)
```

```
['legitimate' 'phishing' 'phishing' ... 'legitimate' 'legitimate'
 'phishing']
```

In [39]:
```
dataset
```

Out[39]:

	URL	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleId.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

Encoding Categorical Data

In [40]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(sparse=False),[0])],rema
x= np.array(ct.fit_transform(x))
print(x)
```

```
[[ 0.  0.  0. ...  1.  1.  4.]
 [ 0.  0.  0. ...  0.  1.  2.]
 [ 0.  0.  0. ...  0.  1.  0.]
 ...
 [ 0.  0.  0. ...  0.  1. 10.]
 [ 0.  0.  0. ...  0.  0.  4.]
 [ 0.  0.  0. ...  1.  1.  0.]]
```

In [41]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=le.fit_transform(y)
print(y)

#Legitimate=0
#phising=1
```

```
[0 1 1 ... 0 0 1]
```

In [42]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [43]:
```
print(x_train)
```

```
[[0. 0. 0. ... 0. 0. 4.]
 [0. 0. 0. ... 0. 1. 6.]
 [0. 0. 0. ... 0. 0. 4.]
 ...
 [0. 0. 0. ... 0. 1. 1.]
```

```
 [0. 0. 0. ... 0. 1. 3.]
 [0. 0. 0. ... 0. 1. 2.]]
```

In [44]:
```
print(x_test)
```

```
[[0. 0. 0. ... 0. 0. 4.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 5.]
 [0. 0. 0. ... 0. 1. 2.]
 [0. 0. 0. ... 0. 0. 5.]]
```

In [45]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [46]:
```
print(y_test)
```

```
[0 1 1 ... 0 1 0]
```

Feature Scaling

In [47]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train[:,:]=sc.fit_transform(x_train[:,:])
x_test[:,:]=sc.transform(x_test[:,:])
```

In [48]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [49]:
```
print(x_test)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -1.25571207]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
  -1.25571207]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.70796401]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.70796401]]
```

Using Logistic Regression

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete

values for a given set of features (or inputs), X

In [50]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

Out[50]:
```
LogisticRegression(random_state=0)
```

In [51]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [1 1]
 ...
 [0 0]
 [1 1]
 [0 0]]
```

In [52]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[1614  110]
 [  76 1629]]
```

Out[52]:
```
0.9457567804024497
```

Sensitivity (true positive rate) refers to the probability of a positive test, conditioned on truly being positive.

Specificity (true negative rate) refers to the probability of a negative test, conditioned on truly being negative.

In [53]:
```
print((1614/(1614+110)))    #Sensivity_Y_pred
print((1629/(1629+76)))     #Specificity_Y_test
```

```
0.9361948955916474
0.955425219941349
```

Using K Cross Validation

Lets take the scenario of 5-Fold cross validation (K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set.

In [54]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 94.45 %
Standard Deviation: 0.54 %
```

Using Decision Tree Classifier

Decision tree classifiers are supervised machine learning models. This means that they use prelabelled data in order to train an algorithm that can be used to make a prediction.

In [55]:
```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[55]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [56]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [57]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [58]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [1 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [59]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[1574  116]
 [ 116 1623]]
```

Out[59]:
```
0.9323417906095072
```

In [60]:
```
print((1574/(1574+116)))     #Sensivity_Y_pred
print((1623/(1623+116)))     #Specificity_Y_test
```

```
0.9313609467455621
0.9332949971247844
```

In [61]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 93.49 %
Standard Deviation: 0.71 %
```

Using Random Forest Classifier

A random forest classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

In [62]:
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[62]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [63]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [64]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [65]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [1 1]
 ...
 [0 0]
 [1 1]
 [0 0]]
```

In [66]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[1638   64]
 [  52 1675]]
```

Out[66]:
```
0.9661708953047535
```

In [67]:
```
print((1638/(1638+64)))     #Sensivity_Y_pred
print((1675/(1675+52)))     #Specificity_Y_test
```

```
0.9623971797884842
0.9698899826288361
```

In [68]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 96.56 %
Standard Deviation: 0.80 %
```

Using KNN Classifier

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

In [69]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier()            # Study the parameters in a better way
classifier.fit(x_train,y_train)
```

Out[69]:
```
KNeighborsClassifier()
```

In [70]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [71]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [72]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [1 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [73]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1555  381]
 [ 135 1358]]
```

Out[73]:
```
0.8495188101487314
```

In [74]:
```
print((1555/(1555+135)))    #Sensivity_Y_pred
print((1358/(1358+381)))     #Specificity_Y_test
```

```
0.9201183431952663
0.7809085681426107
```

In [75]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 84.66 %
```

```
Standard Deviation: 1.09 %
```

Using SVM Classifier

A support vector machine is a supervised machine learning algorithm that can be used for both classification and regression tasks. The Support vector machine classifier works by finding the hyperplane that maximizes the margin between the two classes. The Support vector machine algorithm is also known as a max-margin classifier.

In [76]:
```
from sklearn.svm import SVC
classifier=SVC(kernel="linear",random_state=0)       # for linear model ----> kernel
classifier.fit(x_train,y_train)
```

Out[76]:
```
SVC(kernel='linear', random_state=0)
```

In [77]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [78]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [79]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [1 1]
 ...
 [0 0]
 [1 1]
 [0 0]]
```

In [80]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1616  111]
 [  74 1628]]
```

Out[80]:
```
0.9460484106153397
```

In [81]:
```
print((1616/(1616+74)))    #Sensivity_Y_pred
print((1628/(1628+111)))     #Specificity_Y_test
```

```
0.9562130177514793
0.9361702127659575
```

In [82]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 94.58 %
Standard Deviation: 0.56 %
```

Using Naive Bayes Classifier

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

In [83]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[83]:
```
GaussianNB()
```

In [84]:
```
print(x_train)
```

```
[[ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
   1.10069922]
 [ 0.         -0.01936734  0.         ... -0.14009045 -1.06711431
   0.31522879]
 ...
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.86297686]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.07750642]
 [ 0.         -0.01936734  0.         ... -0.14009045  0.93710673
  -0.47024164]]
```

In [85]:
```
print(y_train)
```

```
[0 0 0 ... 1 1 1]
```

In [86]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [0 1]
 [0 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [87]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1672 1445]
 [  18  294]]
```

Out[87]:
```
0.573344998541849
```

In [88]:
```
print((1672/(1672+18)))    #Sensivity_Y_pred
```

```
print((294/(294+1445)))     #Specificity_Y_test
```

```
0.9893491124260355
0.16906267970097758
```

In [89]:
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 58.39 %
Standard Deviation: 0.75 %
```

Accuracy_Score_DataFrame

In [131…
```
Score={"Classifier":["Logistic","Decision Tree","Random Forest","SVM","KNN","Naive B
       "Accuracy_Score(%)":[94.45,93.49,96.56,94.58,84.66,58.39],
       "Standard_Deviation(%)":[0.54,0.71,0.80,0.56,1.09,0.75],"True_Positive_Rate(S
      "True_Negative_Rate(Specificity)":[95.5,93.3,96.9,93.6,78.0,16.9]}
```

In [132…
```
Score
```

Out[132…
```
{'Classifier': ['Logistic',
  'Decision Tree',
  'Random Forest',
  'SVM',
  'KNN',
  'Naive Bayes'],
 'Accuracy_Score(%)': [94.45, 93.49, 96.56, 94.58, 84.66, 58.39],
 'Standard_Deviation(%)': [0.54, 0.71, 0.8, 0.56, 1.09, 0.75],
 'True_Positive_Rate(Sensivity)': [93.6, 93.1, 96.2, 95.6, 92.0, 98.9],
 'True_Negative_Rate(Specificity)': [95.5, 93.3, 96.9, 93.6, 78.0, 16.9]}
```

Accuracy_Score

In [133…
```
Accuracy_Score=pd.DataFrame(Score)
```

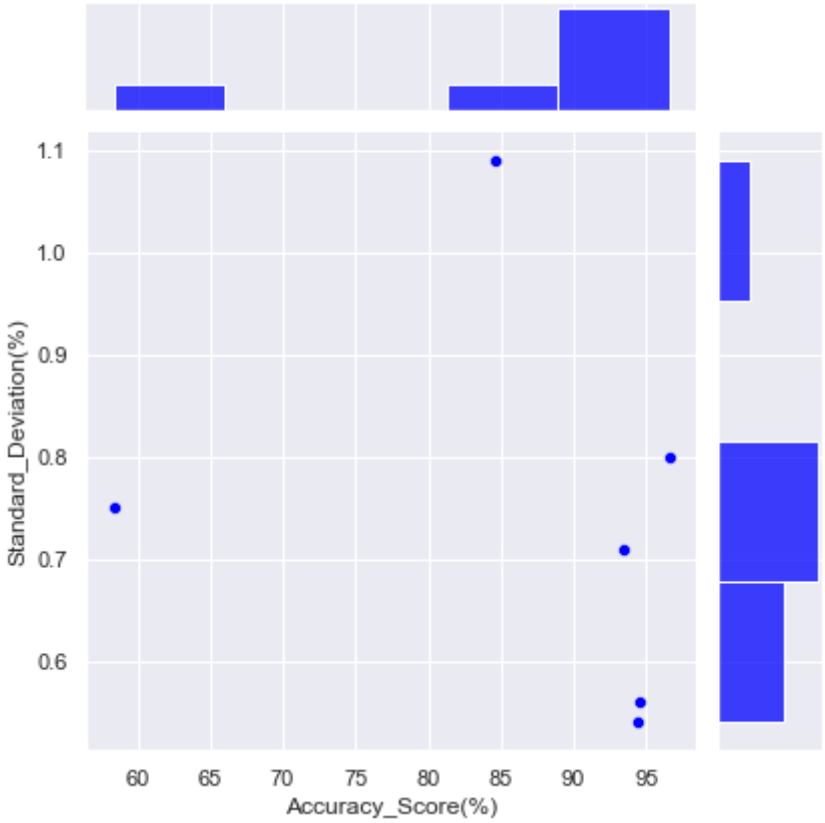
In [134…
```
Accuracy_Score
```

Out[134…

	Classifier	Accuracy_Score(%)	Standard_Deviation(%)	True_Positive_Rate(Sensivity)	True_Negative_I
0	Logistic	94.45	0.54	93.6	
1	Decision Tree	93.49	0.71	93.1	
2	Random Forest	96.56	0.80	96.2	
3	SVM	94.58	0.56	95.6	
4	KNN	84.66	1.09	92.0	
5	Naive Bayes	58.39	0.75	98.9	

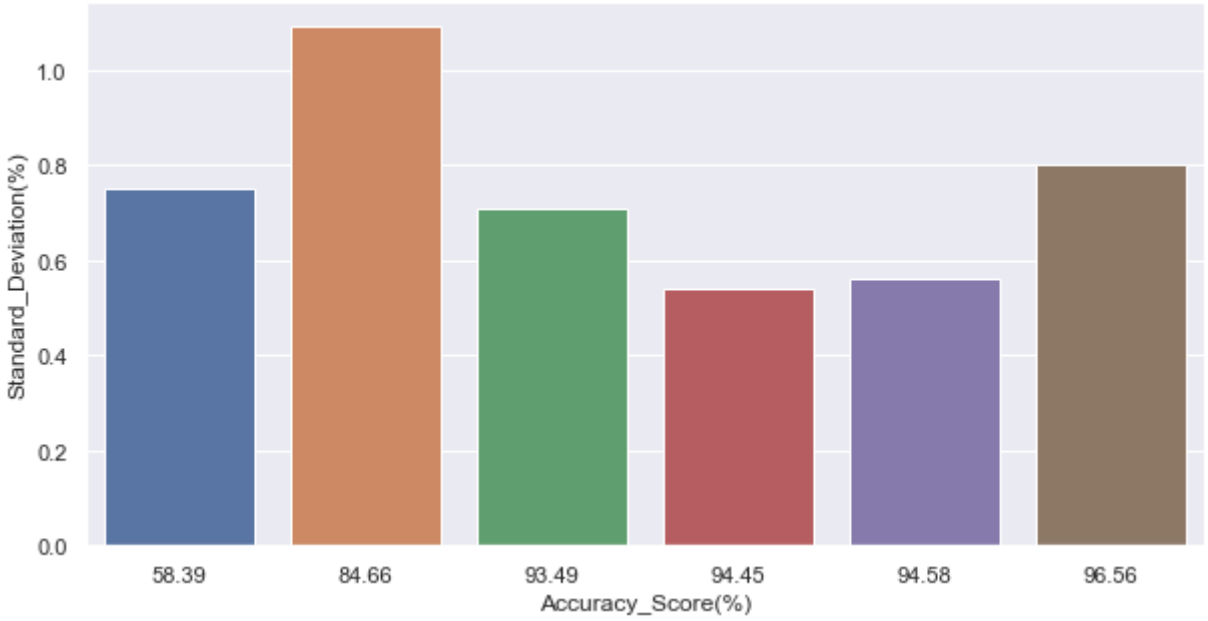
In [94]:
```
sns.jointplot(x="Accuracy_Score(%)",y="Standard_Deviation(%)",data=Accuracy_Score,ki
```

Out[94]:
```
<seaborn.axisgrid.JointGrid at 0x1b57028db20>
```



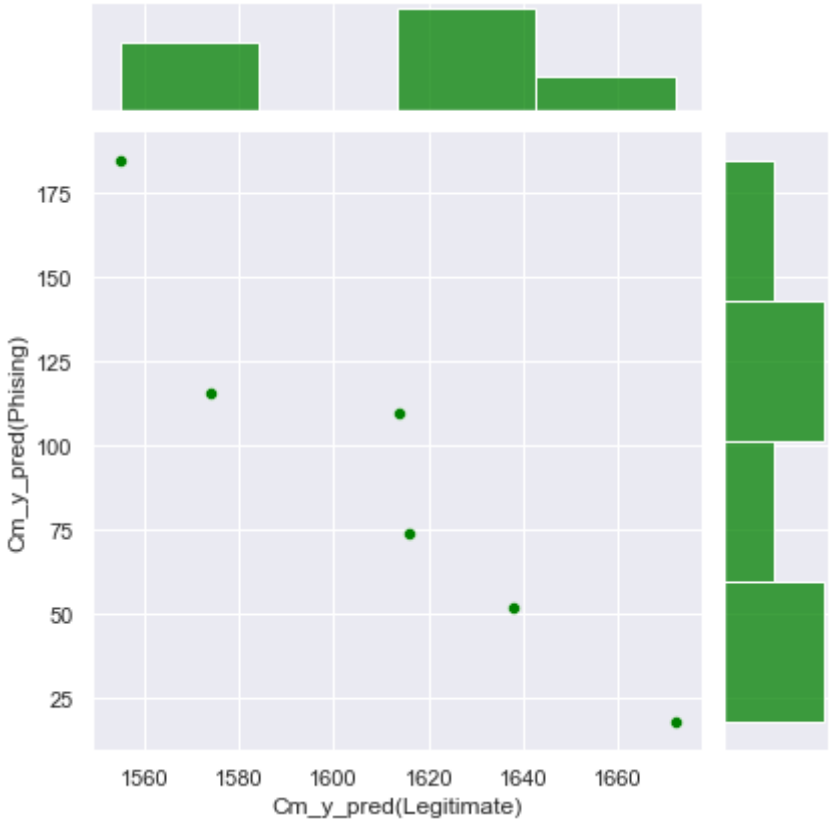
In [150…
```
sns.barplot(x="Accuracy_Score(%)",y="Standard_Deviation(%)",data=Accuracy_Score)
```

Out[150…
```
<AxesSubplot:xlabel='Accuracy_Score(%)', ylabel='Standard_Deviation(%)'>
```



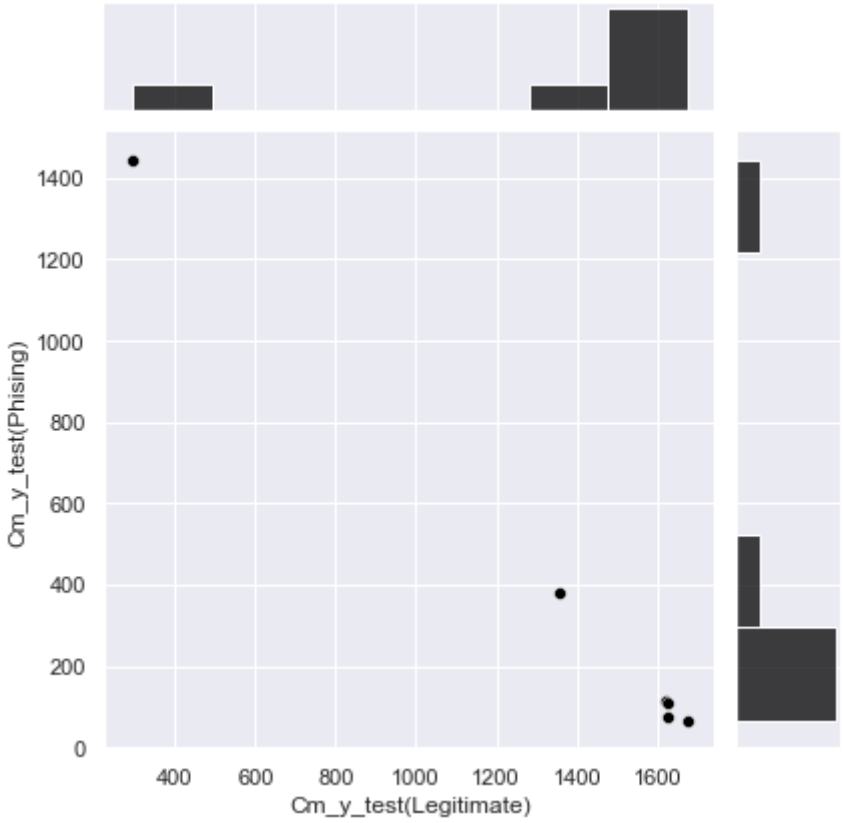
In [95]:
```
sns.jointplot(x="Cm_y_pred(Legitimate)",y="Cm_y_pred(Phising)",data=Accuracy_Score,k
```

Out[95]:
```
<seaborn.axisgrid.JointGrid at 0x1b5078e6f10>
```



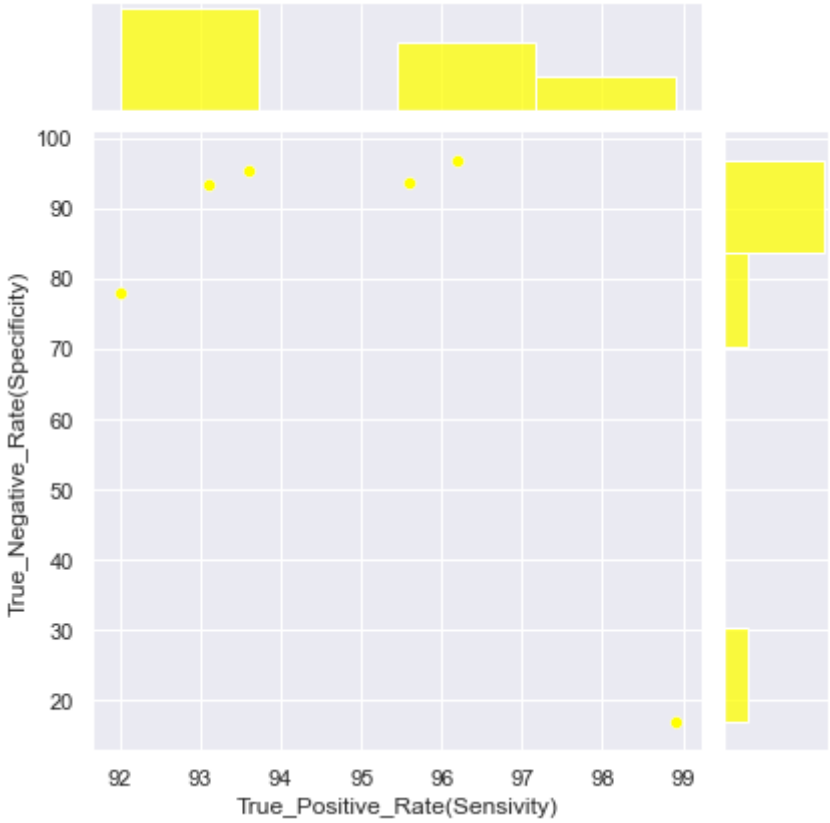
In [96]:
```
sns.jointplot(x="Cm_y_test(Legitimate)",y="Cm_y_test(Phising)",data=Accuracy_Score,k
```

Out[96]: <seaborn.axisgrid.JointGrid at 0x1b507a3f7c0>



In [97]:
```
sns.jointplot(x="True_Positive_Rate(Sensivity)",y="True_Negative_Rate(Specificity)",
```

Out[97]: <seaborn.axisgrid.JointGrid at 0x1b507b47880>



Finding The Accuracy Score by Reducing the number of features : Using PCA

In [98]:
```
dataset
```

Out[98]:

	URL	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleId.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

Importing PCA

Principal Component Analysis (PCA) Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models.

In [99]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
x_train=pca.fit_transform(x_train)
x_test=pca.fit_transform(x_test)
```

Accuracy Score of Logistic Regression Using PCA

In [100…
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

Out[100… LogisticRegression(random_state=0)

In [101…
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [0 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [102…
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1358  650]
 [ 332 1089]]
```

Out[102… 0.7136191309419656

In [103…
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 81.35 %
Standard Deviation: 1.26 %
```

Accuracy Score is quite reduced when computed using PCA i.e by reducing the dimensions

---

Accuracy Score of Random Forest Using PCA

In [104…
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[104… RandomForestClassifier(criterion='entropy', random_state=0)

In [105…
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [0 1]
 [1 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [106…
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1169  552]
 [ 521 1187]]
```

Out[106… 0.6870807815689706

In [107…
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 80.74 %
Standard Deviation: 1.26 %
```

Accuracy Score is quite decreased when computed using PCA i.e by reducing the dimensions

---

Accuracy Score of Naive Bayes Using PCA

In [108…
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[108… GaussianNB()

In [109…
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [0 1]
 [0 1]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

In [110…
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1636 1170]
 [  54  569]]
```

Out[110… 0.6430446194225722

In [111…
```
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
```

```
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 73.47 %
Standard Deviation: 1.16 %
```

Accuracy Score is well increased when computed using PCA i.e by reducing the dimensions

---

In [112…

```
new_dataset
```

Out[112…

	Url	length_url	length_hostname	ip	nb_dots	n
0	http://www.crestonwood.com/router.php	37	19	0	3	
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1	
2	https://support-appleld.com.secureupdate.duila...	126	50	1	4	
3	http://rgipt.ac.in	18	11	0	2	
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2	
...	...	...	...	...	...	
11425	http://www.fontspace.com/category/blackletter	45	17	0	2	
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 83 columns

Mutual Information Classification

It checks the dependency between the two variables and if the two variables are independent then there correlation is cosiderered as zero. Higher the values of the features , more is the Dependency between the variables. Mutual Information always gives either Positive Value for Dependency or Zero Value for Independency

---

In [113…

```
import pandas as pd
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
from sklearn.feature_selection import mutual_info_classif
mutual_info=mutual_info_classif(x_train,y_train)
mutual_info
```

Out[113…

```
array([0.00000000e+00, 1.46054458e-02, 0.00000000e+00, 7.33257088e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 3.08220674e-03, 0.00000000e+00, 6.13091206e-03,
       8.16662930e-03, 6.18550955e-03, 1.13400147e-02, 3.76564757e-03,
       7.10447672e-03, 8.87769848e-04, 2.76268325e-03, 0.00000000e+00,
```

0.00000000e+00, 0.00000000e+00, 8.34594447e-04, 0.00000000e+00,
0.00000000e+00, 5.75117184e-04, 0.00000000e+00, 7.13100013e-03,
0.00000000e+00, 6.51558616e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.84361905e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 8.22838776e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 1.62881524e-03, 0.00000000e+00,
7.40514242e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 5.42196585e-04, 0.00000000e+00, 7.72547643e-03,
0.00000000e+00, 3.42114984e-03, 0.00000000e+00, 2.40382454e-03,
6.33890562e-04, 0.00000000e+00, 0.00000000e+00, 3.82801935e-03,
1.66334143e-03, 4.29747612e-03, 0.00000000e+00, 1.73700925e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.79368485e-03, 0.00000000e+00, 4.61192996e-04,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 7.33645170e-03,
0.00000000e+00, 1.29660647e-03, 6.61581314e-03, 0.00000000e+00,
0.00000000e+00, 4.84241265e-03, 3.42739121e-03, 0.00000000e+00,
0.00000000e+00, 9.45631767e-03, 0.00000000e+00, 5.43891142e-03,
2.29579734e-03, 3.90956144e-03, 0.00000000e+00, 0.00000000e+00,
6.13419046e-03, 6.81680140e-03, 1.49286192e-03, 6.90936973e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.04783173e-03,
8.07852037e-03, 6.71880517e-03, 5.88802182e-03, 2.42210534e-03,
0.00000000e+00, 2.25030439e-03, 6.18321021e-03, 0.00000000e+00,
1.08771394e-03, 2.67883411e-03, 0.00000000e+00, 0.00000000e+00,
9.99793755e-03, 0.00000000e+00, 3.69793460e-03, 2.97479640e-03,
3.09487893e-03, 5.50801399e-03, 0.00000000e+00, 2.77740613e-04,
1.88507022e-04, 4.27063083e-04, 0.00000000e+00, 0.00000000e+00,
2.58942381e-04, 2.05524906e-03, 0.00000000e+00, 2.63633597e-03,
0.00000000e+00, 2.05521707e-04, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 8.00600642e-05, 1.04160068e-03, 0.00000000e+00,
4.09185611e-03, 2.46069283e-03, 6.30125856e-05, 7.88984390e-04,
3.46091784e-03, 0.00000000e+00, 2.44009024e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 8.01150971e-03,
0.00000000e+00, 0.00000000e+00, 6.34281382e-03, 1.78510251e-03,
3.37471081e-03, 1.41675999e-03, 7.76471529e-03, 2.41815326e-03,
8.46569679e-03, 4.35221357e-03, 1.77476416e-04, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 2.30366230e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 5.10715794e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 7.28643214e-03, 1.24286340e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.47650627e-03,
0.00000000e+00, 7.31742021e-04, 2.19287434e-03, 1.17887699e-02,
0.00000000e+00, 4.94125413e-03, 0.00000000e+00, 0.00000000e+00,
6.09431546e-03, 0.00000000e+00, 6.37571317e-03, 0.00000000e+00,
0.00000000e+00, 2.48958691e-03, 7.31883425e-03, 6.14366798e-03,
0.00000000e+00, 2.32995314e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 6.95382637e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 5.21081240e-03, 6.79522246e-03,
7.66017379e-03, 2.67253369e-03, 1.04421065e-03, 0.00000000e+00,
8.40420220e-03, 1.23698916e-02, 2.71565608e-03, 5.49731672e-03,
8.46644484e-03, 0.00000000e+00, 9.38913278e-03, 7.66361927e-03,
1.12347243e-02, 0.00000000e+00, 2.34552670e-04, 0.00000000e+00,
4.24345417e-04, 1.32590707e-02, 0.00000000e+00, 2.80729079e-03,
1.74237135e-03, 6.29335154e-03, 1.01253066e-02, 2.21772802e-03,
0.00000000e+00, 1.19440953e-02, 1.27191967e-03, 0.00000000e+00,
4.57669543e-03, 0.00000000e+00, 7.90871985e-04, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.49113148e-04, 0.00000000e+00, 0.00000000e+00,
6.03235040e-03, 5.15847002e-03, 0.00000000e+00, 0.00000000e+00,
9.03759209e-04, 0.00000000e+00, 0.00000000e+00, 5.10144305e-04,
0.00000000e+00, 9.32863959e-04, 2.56153347e-03, 9.05497623e-03,
0.00000000e+00, 1.48984888e-02, 2.81799448e-04, 1.01769632e-02,
0.00000000e+00, 8.59408064e-03, 0.00000000e+00, 2.10428237e-03,
8.69731379e-04, 0.00000000e+00, 4.44844324e-03, 8.17910388e-04,
0.00000000e+00, 3.34317111e-03, 0.00000000e+00, 5.63924533e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
8.22470521e-03, 0.00000000e+00, 5.68215151e-03, 0.00000000e+00,
5.56988048e-03, 6.57023562e-03, 0.00000000e+00, 0.00000000e+00,
3.85093333e-03, 8.69393900e-03, 4.09445771e-04, 0.00000000e+00,

```
       1.39699333e-03, 0.00000000e+00, 0.00000000e+00, 5.03745025e-03,
       4.37390610e-04, 0.00000000e+00, 5.83471580e-03, 6.39535577e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       2.74395528e-04, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 5.03850358e-03, 0.00000000e+00,
       1.41994959e-03, 2.34036649e-03, 2.88287762e-03, 0.00000000e+00,
       3.69177953e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       5.85741347e-02, 6.62751591e-02, 5.55750131e-02, 3.31625067e-02,
       2.22283815e-02, 5.71500501e-02, 3.20611198e-02, 4.67760613e-02,
       1.16803222e-02, 0.00000000e+00, 4.29527352e-03, 2.65429596e-02,
       3.09719824e-03, 1.01734992e-02, 0.00000000e+00, 1.54781624e-02,
       5.85543899e-03, 0.00000000e+00, 1.09979211e-01, 1.04478696e-02,
       2.46980442e-03, 7.60321004e-03, 9.62056297e-03, 9.72112085e-02,
       5.08573701e-02, 0.00000000e+00, 0.00000000e+00, 1.18408210e-02,
       1.94429244e-02, 9.08446513e-03, 2.83616023e-02, 3.41009324e-02,
       0.00000000e+00, 1.60331845e-02, 0.00000000e+00, 0.00000000e+00,
       3.68869055e-03, 4.08784169e-02, 8.17517978e-02, 5.35204245e-02,
       7.98443182e-02, 3.70812709e-02, 7.71392748e-02, 2.76924539e-02,
       1.13578833e-01, 6.75843423e-02, 8.67791170e-02, 8.05861121e-02,
       7.74984451e-02, 4.46667427e-03, 3.38617695e-04, 0.00000000e+00,
       3.51416344e-03, 1.79276701e-02, 2.28849668e-01, 2.31346014e-01,
       2.32233602e-01, 1.04813503e-02, 1.19845090e-01, 7.99274909e-02,
       0.00000000e+00, 1.11107899e-02, 1.19126590e-01, 6.97847242e-02,
       6.17000744e-02, 0.00000000e+00, 0.00000000e+00, 1.71295188e-01,
       0.00000000e+00, 0.00000000e+00, 3.11811266e-02, 6.10429675e-02,
       9.11550593e-03, 0.00000000e+00, 1.49031446e-01, 2.72786355e-01,
       2.87175526e-01, 1.09180249e-02, 2.99703422e-01, 2.19333835e-01])
```

In [114…
```
x_train
```

Out[114…
```
array([[0., 0., 0., ..., 0., 0., 4.],
       [0., 0., 0., ..., 0., 1., 6.],
       [0., 0., 0., ..., 0., 0., 4.],
       ...,
       [0., 0., 0., ..., 0., 1., 1.],
       [0., 0., 0., ..., 0., 1., 3.],
       [0., 0., 0., ..., 0., 1., 2.]])
```

In [115…
```
mutual_info_classif(x_train,y_train)    # mutual information always gives either pos
```

Out[115…
```
array([0.00000000e+00, 0.00000000e+00, 6.56685596e-03, 6.70260160e-03,
       1.36766870e-03, 0.00000000e+00, 0.00000000e+00, 7.31492936e-03,
       2.49983741e-03, 0.00000000e+00, 5.97584290e-05, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 3.73820569e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 1.98810736e-03, 5.42494765e-03,
       6.47324198e-03, 0.00000000e+00, 0.00000000e+00, 3.11632035e-03,
       5.81066359e-03, 0.00000000e+00, 0.00000000e+00, 2.99554384e-03,
       2.29094966e-03, 0.00000000e+00, 1.97485612e-04, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 6.73439002e-03,
       0.00000000e+00, 0.00000000e+00, 1.97677169e-03, 0.00000000e+00,
       0.00000000e+00, 4.41328207e-03, 1.02542316e-02, 4.22666997e-03,
       9.27056886e-05, 4.36968848e-03, 5.12319240e-03, 2.97727958e-03,
       0.00000000e+00, 0.00000000e+00, 7.70256893e-04, 0.00000000e+00,
       9.94262820e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 3.62645830e-04, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 1.70085879e-03, 3.88222444e-03,
       1.57251159e-02, 2.99134213e-03, 0.00000000e+00, 3.12893461e-03,
       0.00000000e+00, 0.00000000e+00, 7.58746549e-04, 9.21030796e-04,
       1.01502933e-02, 0.00000000e+00, 2.19036930e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.79968025e-03, 0.00000000e+00, 6.32617953e-03,
       9.15346948e-03, 0.00000000e+00, 4.04822583e-03, 0.00000000e+00,
       1.40489043e-04, 1.07188728e-02, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 9.32774551e-03, 3.36533971e-03, 0.00000000e+00,
       7.18144647e-03, 1.62125291e-03, 5.43482455e-03, 4.01933038e-03,
       1.39298234e-02, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 2.47759900e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 4.84560688e-03, 2.45957322e-04,
       7.32495914e-03, 0.00000000e+00, 1.31378854e-02, 3.75843986e-03,
```

3.94575579e-03, 0.00000000e+00, 1.33685141e-04, 0.00000000e+00,
0.00000000e+00, 1.41550765e-03, 0.00000000e+00, 9.69343412e-03,
0.00000000e+00, 0.00000000e+00, 2.48909831e-03, 0.00000000e+00,
2.21369841e-03, 1.14330458e-03, 1.13416826e-02, 9.03101236e-03,
0.00000000e+00, 0.00000000e+00, 9.92488900e-03, 8.42685880e-03,
1.70140387e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 1.66949448e-03, 5.54340951e-03,
1.77152699e-03, 0.00000000e+00, 0.00000000e+00, 7.76386350e-03,
0.00000000e+00, 6.02433164e-03, 2.46071655e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 8.22030144e-03, 1.32272997e-03,
0.00000000e+00, 6.91616603e-03, 4.62623948e-03, 6.71428977e-03,
0.00000000e+00, 0.00000000e+00, 4.12892548e-03, 3.82063003e-03,
3.56406691e-03, 1.02536550e-02, 6.40030201e-04, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 4.03649057e-03, 0.00000000e+00,
1.09570613e-02, 5.87659333e-03, 0.00000000e+00, 0.00000000e+00,
1.32859932e-03, 0.00000000e+00, 0.00000000e+00, 3.15552743e-03,
3.79192180e-04, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 5.30827514e-04,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.62626470e-03,
0.00000000e+00, 2.10123687e-03, 1.19787205e-02, 3.12289977e-03,
0.00000000e+00, 4.39583051e-03, 0.00000000e+00, 4.52674263e-03,
0.00000000e+00, 3.57566921e-03, 0.00000000e+00, 0.00000000e+00,
3.95427907e-03, 6.32440849e-03, 2.86370809e-03, 3.65000930e-05,
0.00000000e+00, 1.02293431e-02, 0.00000000e+00, 0.00000000e+00,
2.32498970e-05, 2.36717165e-03, 0.00000000e+00, 0.00000000e+00,
3.49673013e-03, 7.09175698e-05, 0.00000000e+00, 0.00000000e+00,
2.87836813e-03, 7.03988192e-03, 6.50480496e-03, 0.00000000e+00,
3.16809111e-04, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
2.06174950e-03, 1.33419521e-03, 3.14848537e-03, 3.88146280e-03,
3.68386193e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
8.51570117e-03, 0.00000000e+00, 4.65625865e-03, 1.01123111e-02,
0.00000000e+00, 0.00000000e+00, 4.77078255e-05, 0.00000000e+00,
0.00000000e+00, 1.85037944e-03, 1.79949682e-04, 4.10645994e-03,
7.68154164e-03, 6.80683936e-04, 0.00000000e+00, 0.00000000e+00,
1.07587334e-03, 1.40150793e-03, 0.00000000e+00, 3.99919809e-03,
0.00000000e+00, 6.26210519e-03, 0.00000000e+00, 0.00000000e+00,
3.68102154e-03, 7.10268050e-04, 0.00000000e+00, 0.00000000e+00,
2.71937794e-05, 5.47498753e-03, 9.43045330e-03, 3.55136076e-04,
7.11690835e-03, 2.23515732e-03, 0.00000000e+00, 0.00000000e+00,
7.39036225e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
2.62346748e-03, 0.00000000e+00, 1.58082949e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 9.68055510e-03,
1.93124069e-03, 5.66052556e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 3.20377539e-03, 3.90746226e-03, 1.91438822e-03,
0.00000000e+00, 2.25985779e-04, 0.00000000e+00, 1.59219006e-03,
8.87636632e-04, 0.00000000e+00, 0.00000000e+00, 7.34699381e-03,
0.00000000e+00, 5.36802447e-04, 0.00000000e+00, 3.47879452e-03,
6.60822001e-04, 4.25521038e-04, 0.00000000e+00, 1.04529783e-02,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.71078970e-03,
0.00000000e+00, 2.94641896e-03, 4.64084779e-03, 0.00000000e+00,
0.00000000e+00, 5.00112487e-03, 0.00000000e+00, 8.40172323e-03,
7.46547181e-02, 4.72808463e-02, 6.69899160e-02, 3.18340271e-02,
1.59115611e-02, 4.77325945e-02, 3.45058116e-02, 4.49258876e-02,
0.00000000e+00, 0.00000000e+00, 8.01618129e-03, 4.04357738e-02,
7.66655542e-03, 6.98492761e-03, 0.00000000e+00, 4.18169916e-03,
5.31425654e-03, 0.00000000e+00, 1.13365146e-01, 1.34320753e-02,
3.97551882e-03, 1.19315970e-02, 3.63460553e-03, 9.65477921e-02,
5.62430922e-02, 0.00000000e+00, 4.52235711e-03, 0.00000000e+00,
3.04807094e-02, 5.84234180e-03, 3.58305596e-02, 2.12184137e-02,
0.00000000e+00, 1.03633250e-02, 0.00000000e+00, 0.00000000e+00,
1.05513288e-02, 3.77043548e-02, 9.23512581e-02, 5.64354064e-02,
6.85794503e-02, 3.45289299e-02, 6.92071842e-02, 2.55536301e-02,
1.22244929e-01, 6.85084995e-02, 8.17558635e-02, 8.37693783e-02,
7.79775584e-02, 4.31466021e-03, 0.00000000e+00, 6.27396390e-03,
3.68412550e-03, 5.70882739e-03, 2.29459660e-01, 2.33480479e-01,
2.28376418e-01, 8.43629854e-03, 1.17298756e-01, 7.66348332e-02,
3.88601234e-03, 1.44934441e-02, 1.18807528e-01, 7.68996784e-02,
6.53784442e-02, 1.10196812e-02, 5.41261976e-03, 1.83225950e-01,

```
       7.36641934e-03, 5.60431305e-03, 2.89735753e-02, 6.13167294e-02,
       6.16058787e-03, 2.87892536e-03, 1.46318678e-01, 2.73827179e-01,
       2.84539728e-01, 1.31983008e-02, 3.02773953e-01, 2.14167098e-01])
```

In [116…
```
y_train
```

Out[116…
```
array([0, 0, 0, ..., 1, 1, 1])
```

In [117…
```
import pandas as pd
mutual_data=pd.Series(mutual_info)
mutual_data
```

Out[117…
```
0      0.000000
1      0.014605
2      0.000000
3      0.007333
4      0.000000
         ...
399    0.272786
400    0.287176
401    0.010918
402    0.299703
403    0.219334
Length: 404, dtype: float64
```

In [118…
```
pd.Series(mutual_info).sort_values(ascending=False)
```

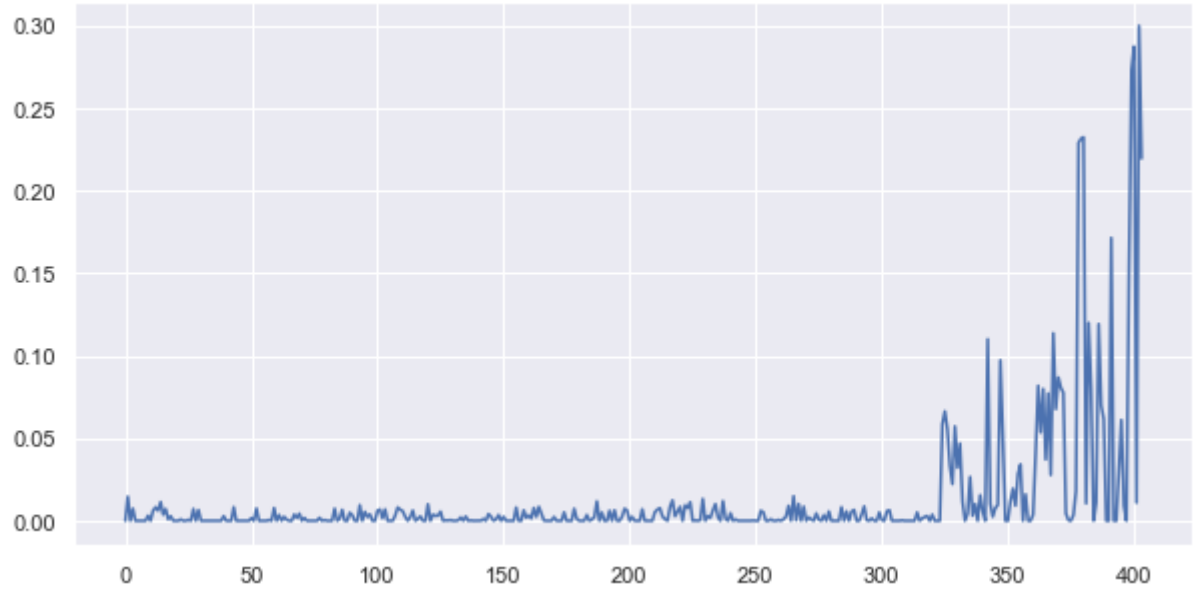
Out[118…
```
402    0.299703
400    0.287176
399    0.272786
380    0.232234
379    0.231346
         ...
131    0.000000
130    0.000000
126    0.000000
287    0.000000
0      0.000000
Length: 404, dtype: float64
```

In [119…
```
mutual_data.nlargest(10)    # Features with top 10 Largest Score
```

Out[119…
```
402    0.299703
400    0.287176
399    0.272786
380    0.232234
379    0.231346
378    0.228850
403    0.219334
391    0.171295
398    0.149031
382    0.119845
dtype: float64
```

In [120…
```
import matplotlib.pyplot as plt
mutual_data.plot()
```

Out[120…
```
<AxesSubplot:>
```



Conclusion: There are total 404 Scores Observed while doing Mutual Information and the Highest Dependency of a feature is 0.298131 among all the features.

In [ ]:

Feature Importance :

This Technique gives the Scores of all features individually, The Higher the Score More Relevant it is.

In [ ]:

In [121…
```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model=ExtraTreesClassifier()
model.fit(x_train,y_train)
```

Out[121… ExtraTreesClassifier()

In [122…
```
print(model.feature_importances_)
```

```
[0.00000000e+00 3.20707187e-05 0.00000000e+00 1.59510233e-04
 2.86656903e-04 3.14829065e-04 5.17944604e-04 4.68151321e-04
 6.10877041e-04 7.22357617e-04 5.79339499e-04 1.00986492e-03
 1.18182360e-03 1.06020284e-03 1.10295863e-03 1.53589127e-03
 1.40419646e-03 1.14075773e-03 8.73954475e-04 9.02900524e-04
 1.04164103e-03 1.24943512e-03 9.93680339e-04 9.79516519e-04
 9.38540571e-04 1.14167273e-03 9.20686367e-04 1.13035777e-03
 6.72340509e-04 9.84239452e-04 7.44151463e-04 1.24764266e-03
 9.85575084e-04 1.06083526e-03 6.54156475e-04 1.02229127e-03
 9.45219880e-04 1.23913172e-03 6.57362826e-04 8.00382002e-04
 6.32442549e-04 7.14172501e-04 6.46036981e-04 5.63635940e-04
 6.72330027e-04 6.25421749e-04 6.79338825e-04 5.64489851e-04
 6.92135836e-04 5.17485069e-04 4.26361937e-04 5.38188011e-04
 5.73459225e-04 2.70560678e-04 4.17962281e-04 5.61475958e-04
 5.28018741e-04 2.16771296e-04 4.64512950e-04 3.39761238e-04
 3.83009630e-04 2.70272340e-04 4.54106792e-04 4.98302770e-04
 2.65667768e-04 6.24458194e-04 2.89555701e-04 3.26430990e-04
 5.68647438e-04 7.06455064e-04 2.18731009e-04 2.99330052e-04
 5.68376224e-04 3.01933631e-04 1.93323770e-04 2.01404711e-04
 2.29798651e-04 3.56671423e-04 9.94775111e-05 1.66605104e-04
```

1.39811041e-04 2.10118810e-04 1.12715752e-04 9.68040894e-05
2.66769787e-04 1.14300932e-04 1.35493128e-04 3.54960784e-04
1.29043440e-04 2.85679333e-04 7.17528363e-05 1.51019058e-04
8.09455000e-05 1.42240652e-04 1.25304423e-04 9.10381617e-05
1.14724349e-04 3.54665271e-05 1.41817222e-04 6.23789963e-05
1.05780465e-04 1.93601859e-04 3.80111535e-05 1.87777461e-04
6.86221659e-05 8.53881383e-05 3.07578640e-05 7.08850155e-05
6.17101124e-05 8.49576629e-05 3.93535303e-05 1.86350203e-04
9.31656317e-05 4.01898903e-05 5.81794606e-04 3.72128782e-04
3.00843554e-04 1.86447512e-05 3.61259848e-05 9.70082534e-06
2.89182439e-04 3.44678508e-05 3.21348462e-07 5.64493251e-05
6.48408072e-06 3.74408931e-05 9.46623256e-05 2.97593011e-07
2.93823704e-05 1.30343437e-05 3.25860414e-05 5.98514462e-05
5.54646755e-06 1.15642451e-05 2.78899647e-05 1.51138025e-05
1.79844474e-05 4.21335649e-05 1.09605956e-05 4.16397546e-06
8.31336599e-07 1.52474564e-05 5.85357381e-06 5.43206011e-05
8.92779034e-08 2.08315108e-07 1.44119044e-06 5.91326747e-05
1.75855523e-05 0.00000000e+00 4.27899635e-06 2.75116599e-06
2.67359994e-06 3.86450174e-05 3.00971706e-06 3.21718258e-05
8.51903794e-06 4.34312366e-05 1.75820477e-06 8.33260432e-07
0.00000000e+00 1.27216484e-05 1.09375381e-06 0.00000000e+00
1.62158217e-06 2.83277714e-06 2.73812208e-06 1.24134404e-05
1.73019193e-06 3.03025880e-05 0.00000000e+00 1.67560458e-05
0.00000000e+00 6.40095514e-07 0.00000000e+00 8.20708112e-06
6.51511315e-06 1.21446865e-07 5.42305873e-05 3.14317677e-06
3.10975261e-06 3.67842464e-06 0.00000000e+00 1.82320231e-06
3.31983150e-06 4.77791835e-05 2.40340661e-05 9.51915519e-06
3.40232820e-06 1.90459527e-06 8.33260432e-07 3.45724979e-05
1.75459991e-06 0.00000000e+00 8.13179701e-06 0.00000000e+00
0.00000000e+00 0.00000000e+00 4.15817115e-06 3.39167260e-06
2.65999681e-06 0.00000000e+00 3.09462442e-05 0.00000000e+00
2.49978130e-06 2.23774426e-06 3.59090137e-05 2.84960349e-05
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 8.26122593e-06
1.15141442e-06 0.00000000e+00 0.00000000e+00 1.57566759e-05
5.81981469e-06 0.00000000e+00 8.83893159e-07 3.01469691e-06
0.00000000e+00 6.72210097e-08 1.03983971e-05 3.53054291e-08
1.02936890e-05 3.61710779e-07 1.92190978e-06 0.00000000e+00
0.00000000e+00 0.00000000e+00 2.50638453e-06 2.57330428e-06
3.63061950e-06 2.64441681e-06 2.80365806e-05 3.13024576e-06
0.00000000e+00 0.00000000e+00 5.29685324e-06 1.48309508e-05
1.61266379e-06 3.16588310e-08 1.39018557e-07 4.16630216e-07
7.92816904e-06 1.98080812e-06 0.00000000e+00 0.00000000e+00
3.60722365e-06 0.00000000e+00 0.00000000e+00 1.27369809e-07
0.00000000e+00 0.00000000e+00 0.00000000e+00 3.33304173e-06
1.16334304e-05 7.31146201e-06 4.66730226e-06 0.00000000e+00
3.36764188e-06 1.45679828e-05 1.18974255e-08 2.54556484e-06
6.94383693e-08 4.33444165e-06 0.00000000e+00 0.00000000e+00
0.00000000e+00 3.83183866e-06 1.02555130e-06 0.00000000e+00
0.00000000e+00 0.00000000e+00 3.70200858e-09 0.00000000e+00
0.00000000e+00 0.00000000e+00 1.86833653e-05 0.00000000e+00
0.00000000e+00 4.58293238e-06 0.00000000e+00 6.66608346e-07
8.09417834e-06 2.62139009e-06 0.00000000e+00 0.00000000e+00
1.28011560e-07 9.05717861e-09 0.00000000e+00 1.27897819e-08
8.64967615e-06 2.55240827e-06 0.00000000e+00 0.00000000e+00
0.00000000e+00 1.66652086e-06 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 4.58666036e-06
2.58946379e-05 5.52404480e-06 8.80645213e-06 2.15127924e-05
1.18197642e-05 1.37925501e-05 1.80663033e-05 0.00000000e+00
1.92111240e-06 0.00000000e+00 3.83489176e-07 4.00068518e-06
0.00000000e+00 3.07665390e-08 4.46328728e-06 0.00000000e+00
1.33663176e-02 2.00679410e-02 1.21310083e-02 8.70165386e-03
3.38299479e-03 1.10501633e-02 3.68412596e-03 6.69630413e-03
3.65886753e-03 4.41114557e-04 1.71092886e-03 1.28649867e-02
2.10342600e-05 1.49067598e-03 1.62066433e-04 1.19414208e-03
8.42759777e-06 1.57753350e-03 4.90274830e-02 2.87178181e-03
9.57238326e-04 1.40161733e-03 6.64456017e-03 1.58481981e-02
9.70246202e-03 2.50951695e-05 3.30173509e-04 1.88232953e-03
5.88814320e-03 3.07388418e-03 1.19219500e-02 1.02426841e-02

```
2.46857093e-03 6.58515599e-03 6.80729904e-05 7.47203457e-03
9.62001406e-05 9.55921486e-03 9.67314237e-03 8.57821668e-03
1.16234766e-02 8.74018185e-03 8.56687347e-03 8.07150238e-03
1.16193115e-02 7.28702976e-03 8.82180924e-03 1.14144224e-02
1.91958996e-02 8.61705165e-03 3.51815413e-04 3.29872765e-04
2.90087884e-03 4.20412709e-03 1.77243625e-02 2.45931815e-02
2.00754950e-02 5.19000842e-03 9.30284067e-03 7.15197307e-03
2.76194194e-03 9.66239754e-03 1.92534301e-02 1.25811728e-02
1.20543922e-02 9.80402054e-05 5.43893230e-04 2.31471284e-02
6.17832517e-05 2.63220696e-04 8.77075660e-03 3.15626282e-02
1.66210591e-02 2.45420882e-03 1.05919987e-02 2.87483660e-02
8.23891087e-03 3.75195121e-03 1.99879111e-01 8.83413991e-02]
```

In [123…
```
model.feature_importances_.shape
```

Out[123…
```
(404,)
```

In [124…
```
Feature_imp=pd.Series(model.feature_importances_)
Feature_imp
```

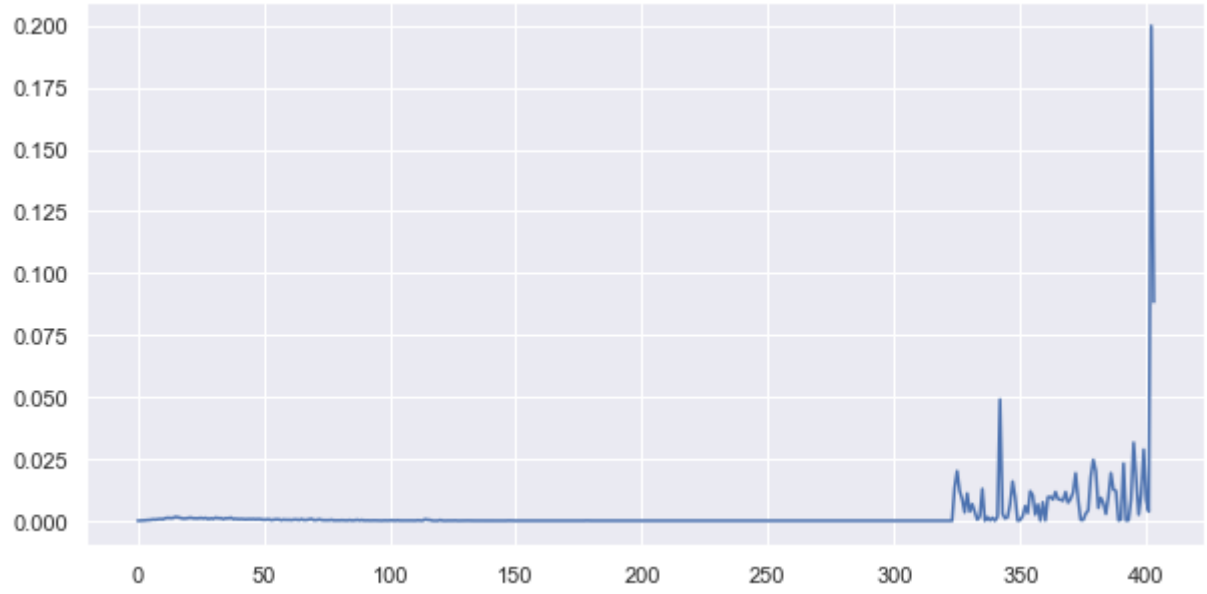
Out[124…
```
0      0.000000
1      0.000032
2      0.000000
3      0.000160
4      0.000287
         ...
399    0.028748
400    0.008239
401    0.003752
402    0.199879
403    0.088341
Length: 404, dtype: float64
```

In [125…
```
pd.Series(model.feature_importances_).nlargest(10)
```

Out[125…
```
402    0.199879
403    0.088341
342    0.049027
395    0.031563
399    0.028748
379    0.024593
391    0.023147
380    0.020075
325    0.020068
386    0.019253
dtype: float64
```

In [126…
```
Feature_imp.plot()
```

Out[126…
```
<AxesSubplot:>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: