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., 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-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 [3]:
```
dataset.isna().any().any()                    # There is no missing data in this dataset
```

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]:
```
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()
```

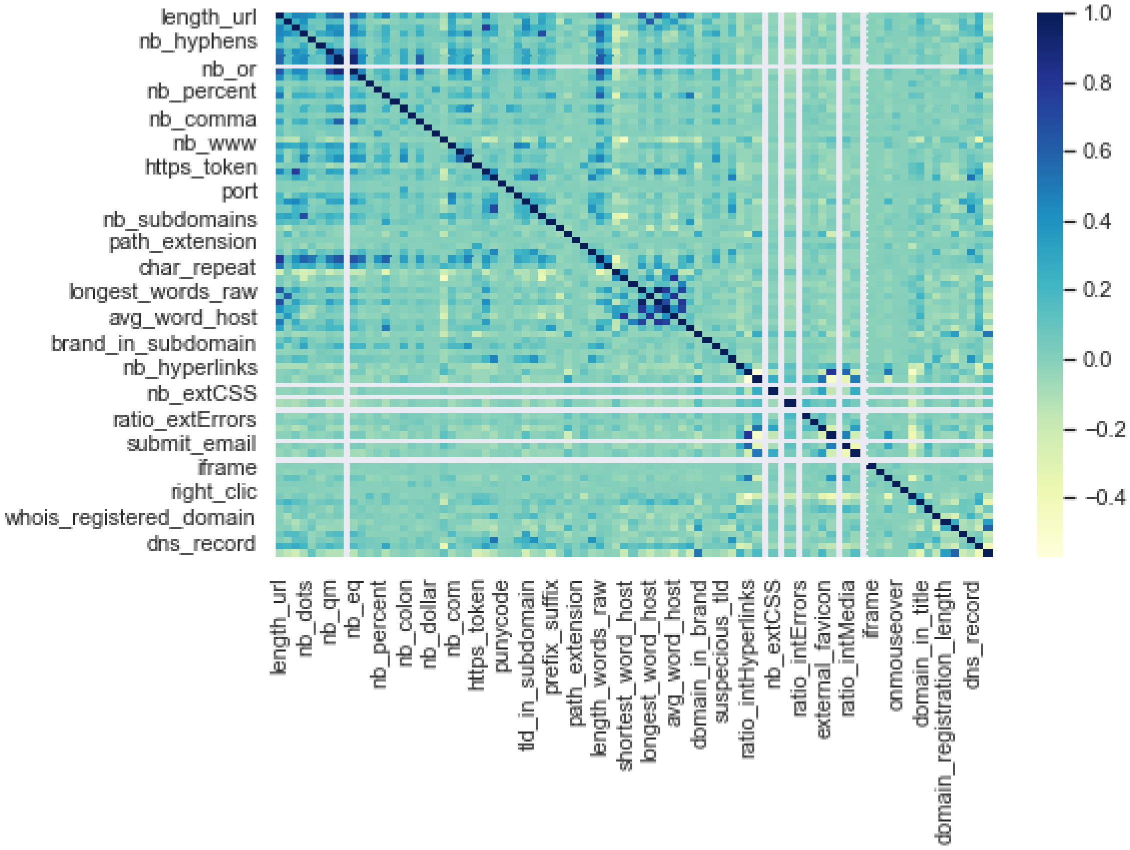
Out[6]:

	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

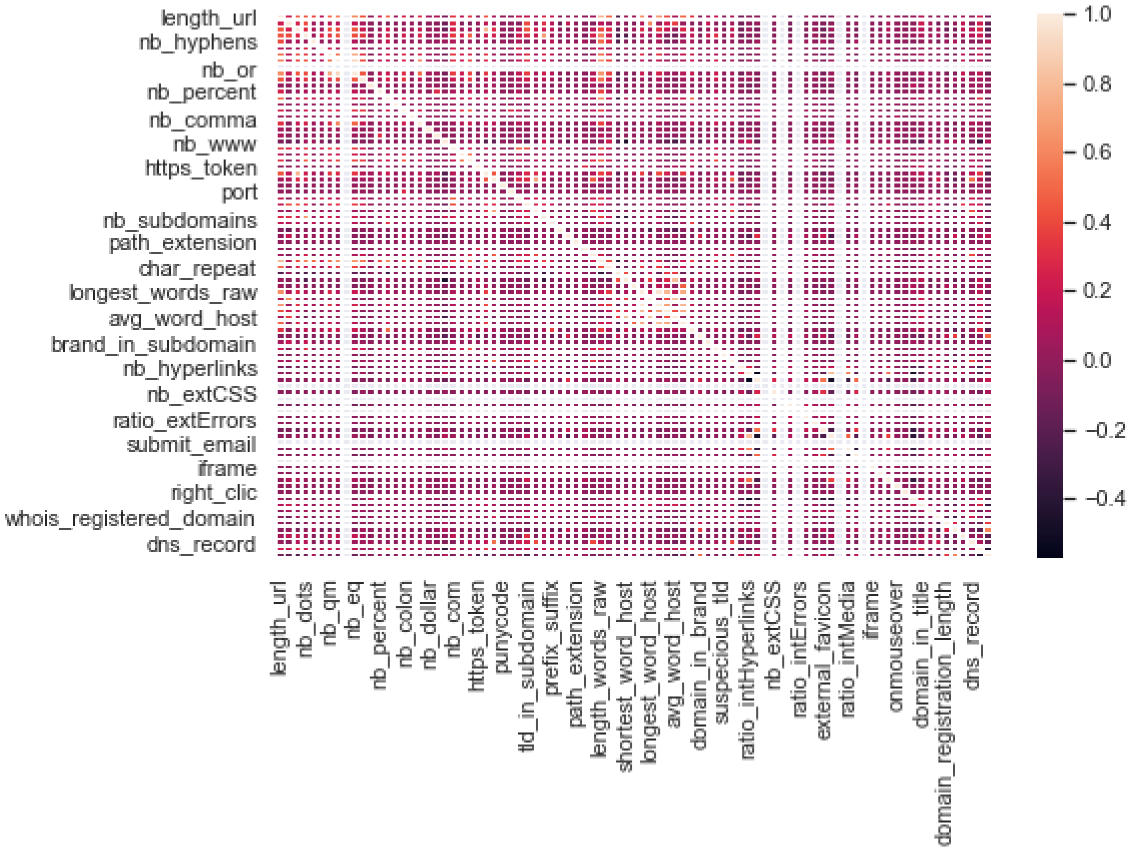
In [7]:
```
sns.set(rc={"figure.figsize":(8,5)})
sns.heatmap(dataset.corr(),cmap="YlGnBu")
```

Out[7]: <AxesSubplot:>



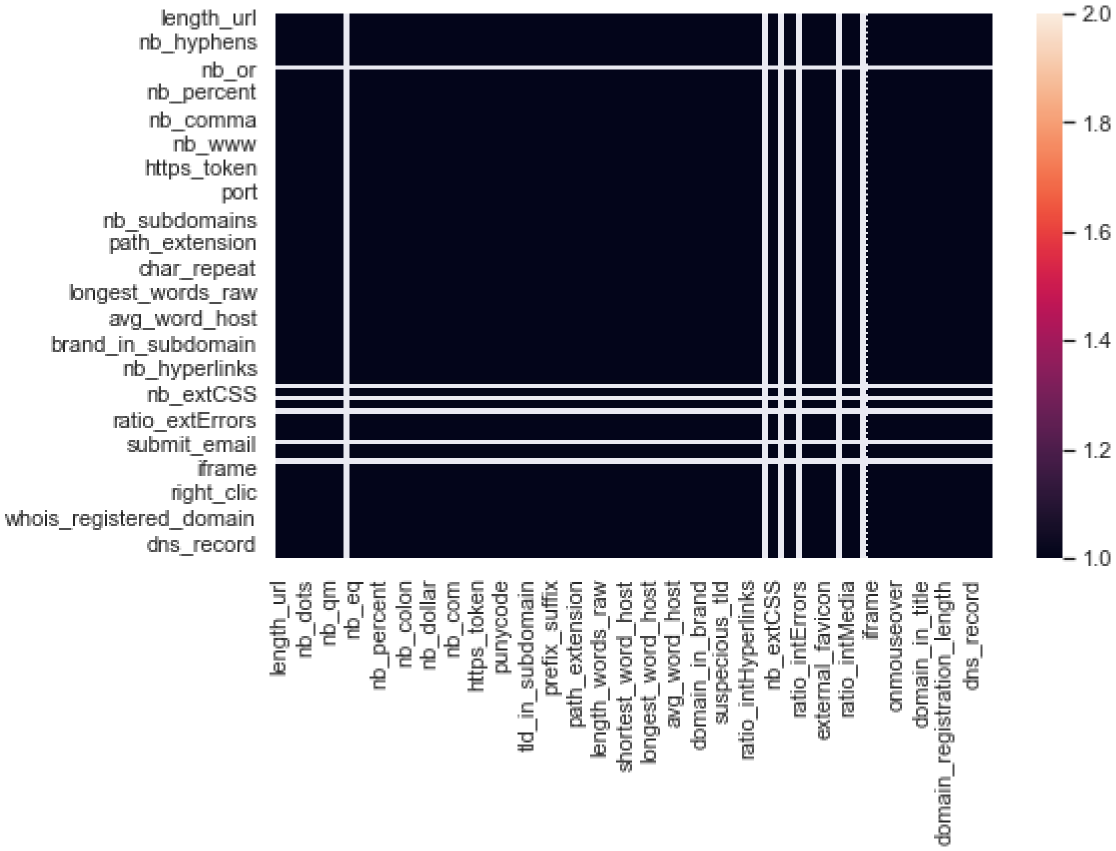
In [8]:
```
sns.heatmap(dataset.corr(),linewidths=0.7)
```

Out[8]: <AxesSubplot:>



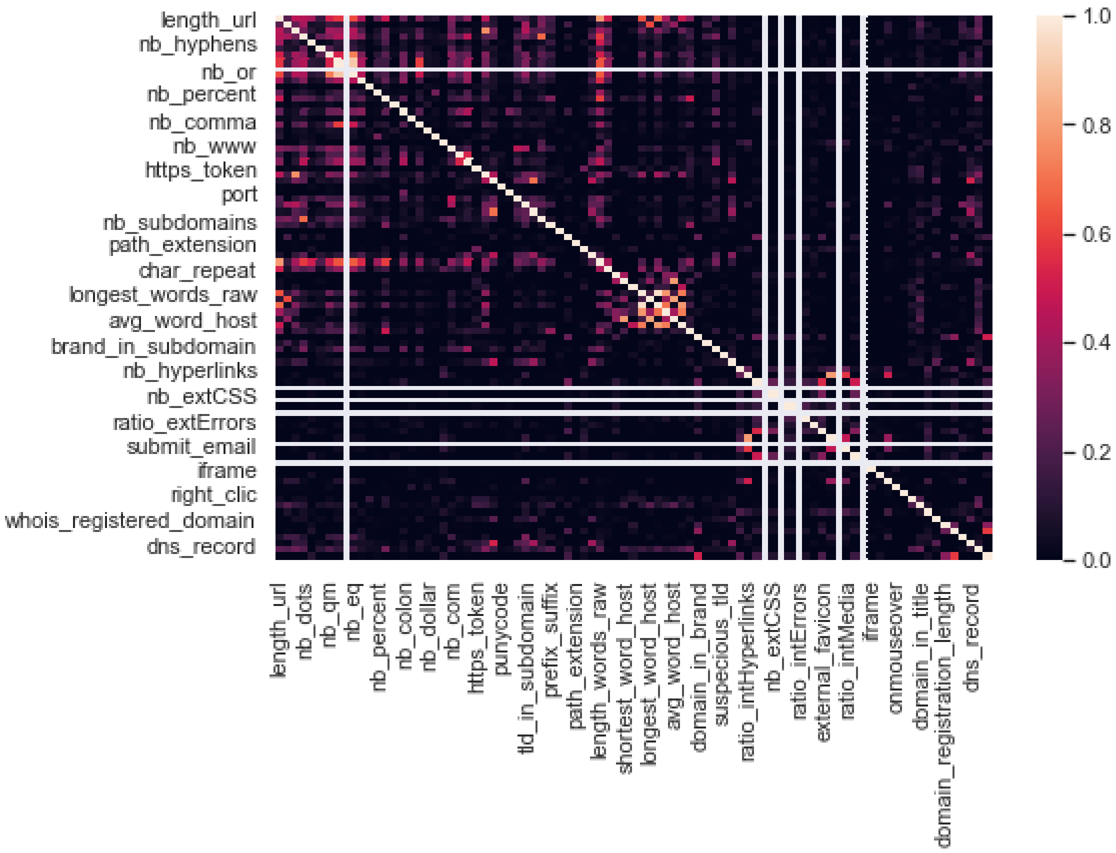
In [9]:
```
sns.heatmap(dataset.corr(),vmin=1,vmax=2)
```

Out[9]: <AxesSubplot:>



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

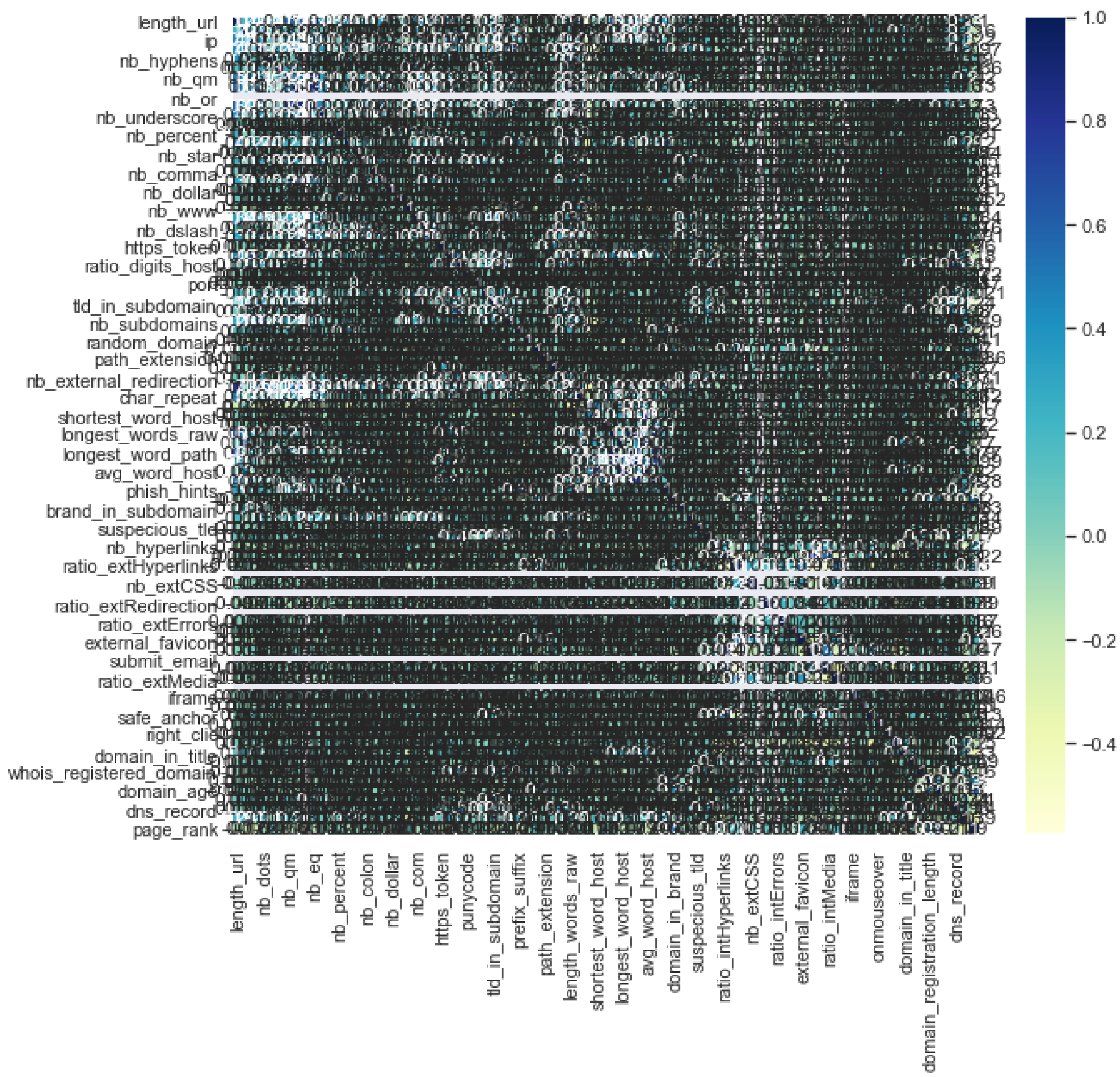
Out[10]: <AxesSubplot:>



In [11]:
```
sns.set(rc={"figure.figsize":(10,9)})
```

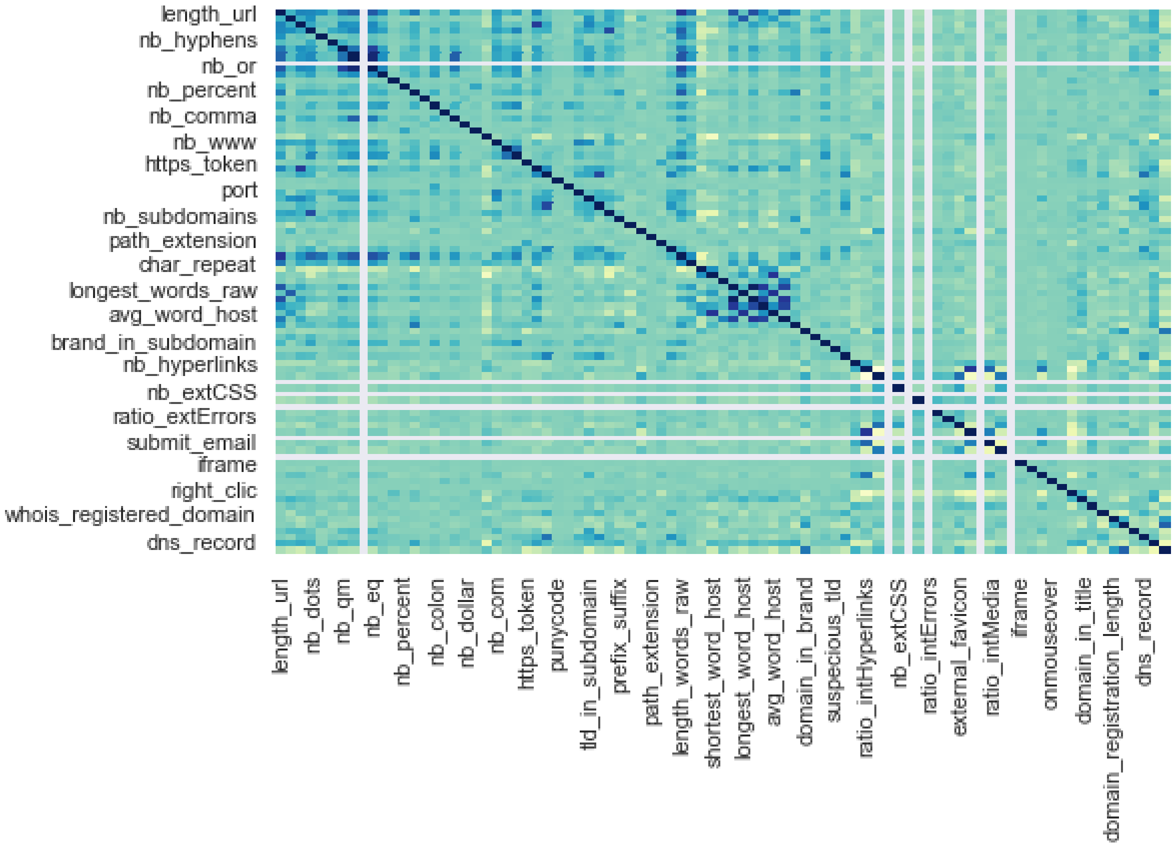
```python
sns.heatmap(dataset.corr(),cmap="YlGnBu",annot=True)
```

Out[11]: <AxesSubplot:>



In [12]:
```python
sns.set(rc={"figure.figsize":(8,5)})
sns.heatmap(dataset.corr(),cmap="YlGnBu",cbar=False)
```

Out[12]: <AxesSubplot:>



In [13]:
```
sns.barplot(data=dataset)
```

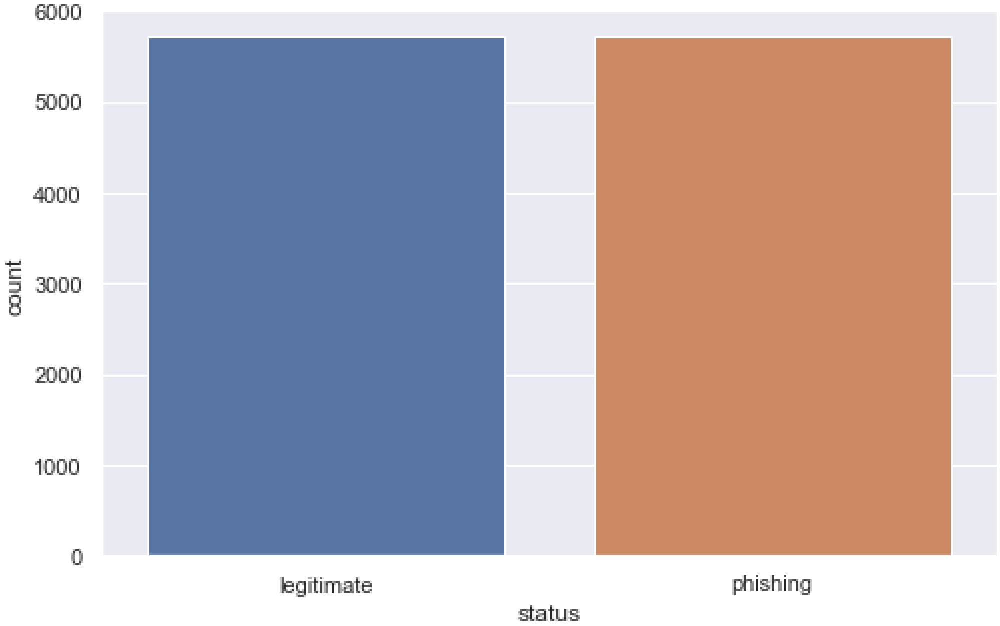
Out[13]: <AxesSubplot:>

In [14]:
```
dataset["status"].value_counts()
```

Out[14]:
```
phishing      5715
legitimate    5715
Name: status, dtype: int64
```

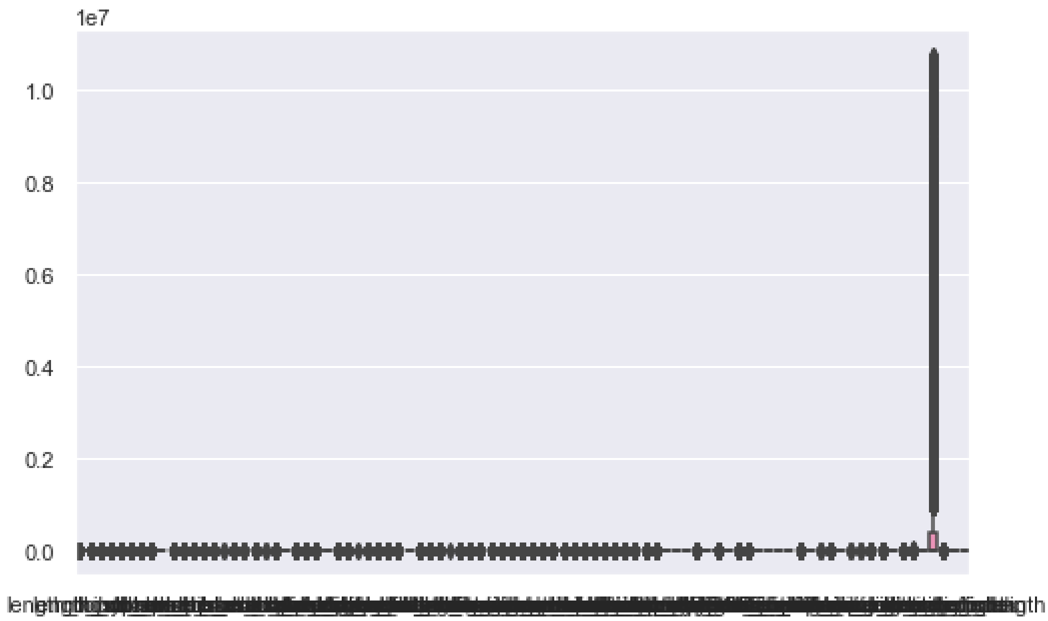
In [15]:
```
sns.countplot(x="status",data=dataset)
```

Out[15]: <AxesSubplot:xlabel='status', ylabel='count'>



In [16]:
```
sns.boxplot(data=dataset)
```

Out[16]: <AxesSubplot:>



In [17]:
```
dataset
```

Out[17]:

	URL	length_url	length_hostname	ip	nb_dots
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

	URL	length_url	length_hostname	ip	nb_dots	n
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 [18]:
```
dataset["length_url"].value_counts()
```

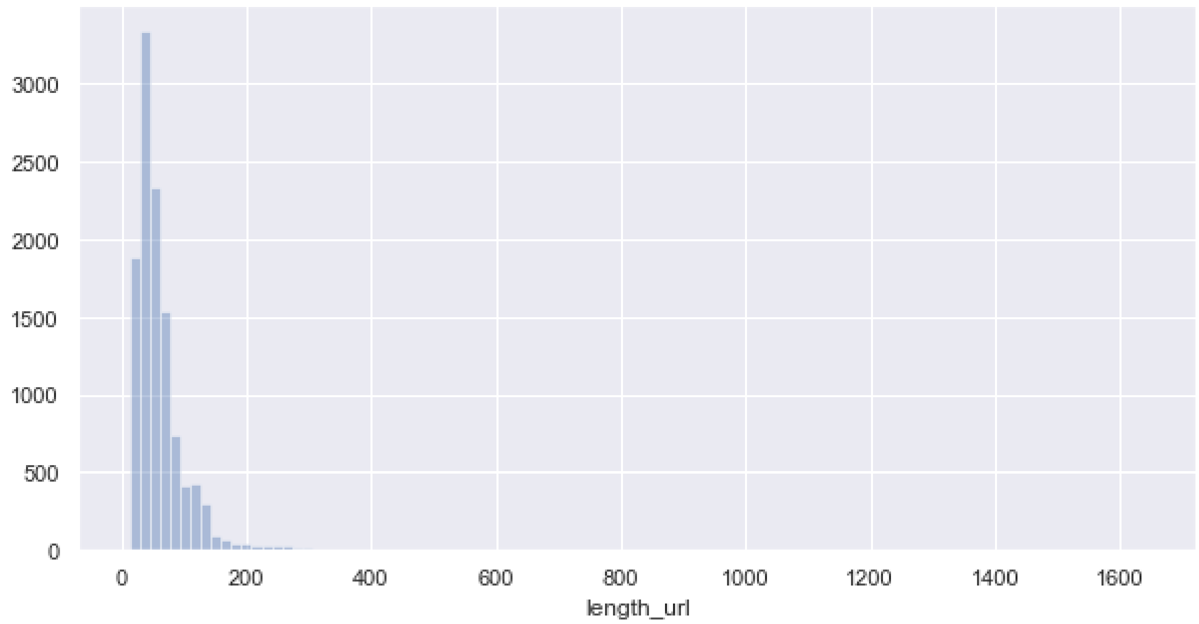
Out[18]:
```
26     251
29     250
32     250
33     230
27     230
      ...
403      1
395      1
339      1
315      1
907      1
Name: length_url, Length: 324, dtype: int64
```

In [19]:
```
dataset["length_hostname"].value_counts()
```

Out[19]:
```
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
```

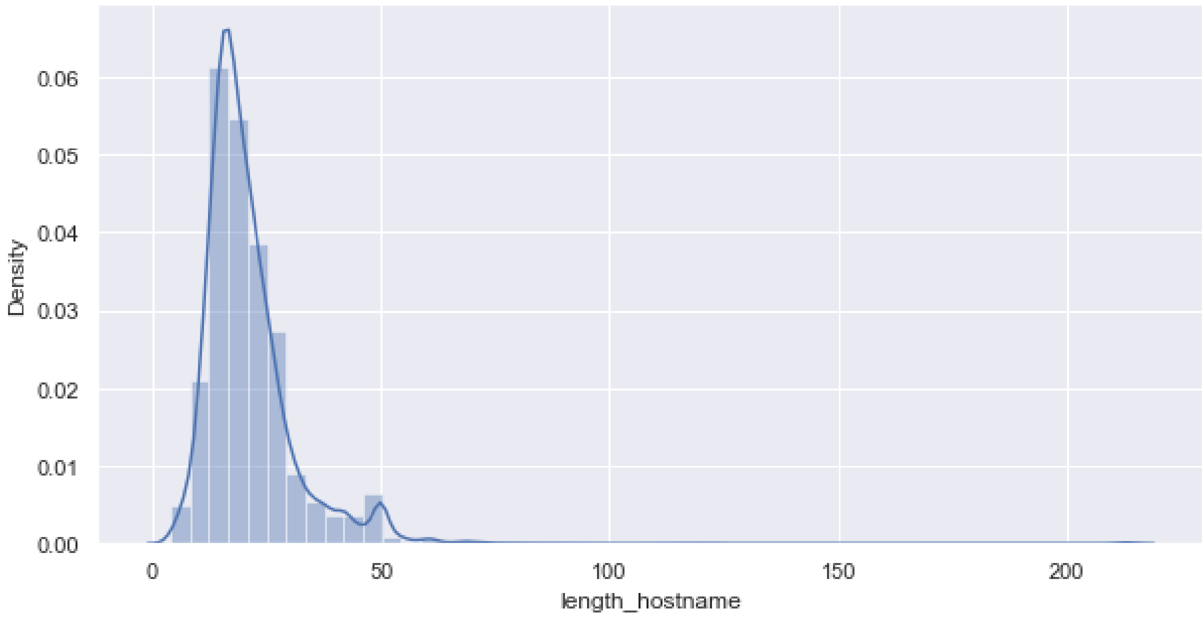
In [20]:
```
sns.set(rc={"figure.figsize":(10,5)})
sns.distplot(dataset.length_url,kde=False,bins=100)
```

Out[20]: <AxesSubplot:xlabel='length_url'>



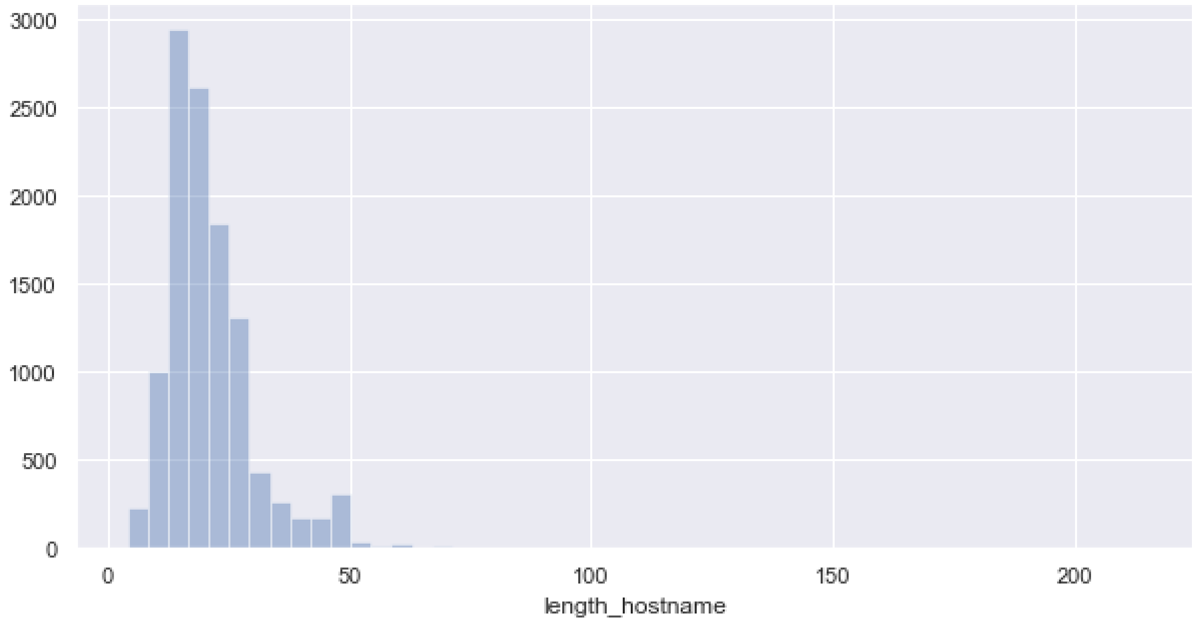
In [21]:
```
sns.distplot(dataset.length_hostname)
```

Out[21]: <AxesSubplot:xlabel='length_hostname', ylabel='Density'>



In [22]:
```
sns.distplot(dataset["length_hostname"],kde=False)
```

Out[22]: <AxesSubplot:xlabel='length_hostname'>



In [23]:
```
dataset
```

Out[23]:

	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	

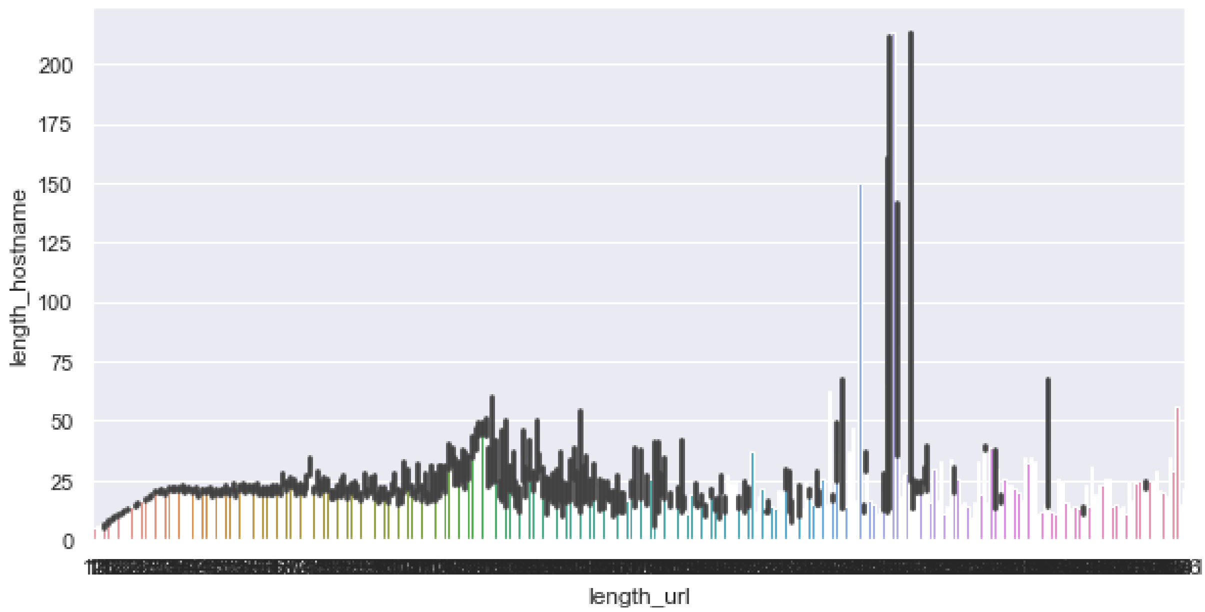
	URL	length_url	length_hostname	ip	nb_dots	n
11426	http://www.budgetbots.com/server.php/Server%20...	84	18	0	5	
11427	https://www.facebook.com/Interactive-Televisio...	105	16	1	2	
11428	http://www.mypublicdomainpictures.com/	38	30	0	2	
11429	http://174.139.46.123/ap/signin?openid.pape.ma...	477	14	1	24	

11430 rows × 89 columns

In [24]:

```
sns.set(rc={"figure.figsize":(10,5)})
sns.barplot(x="length_url",y="length_hostname",data=dataset)
```

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



In [25]:

```
data
```

Out[25]:

```
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 [26]:
```
dataset.describe()
```

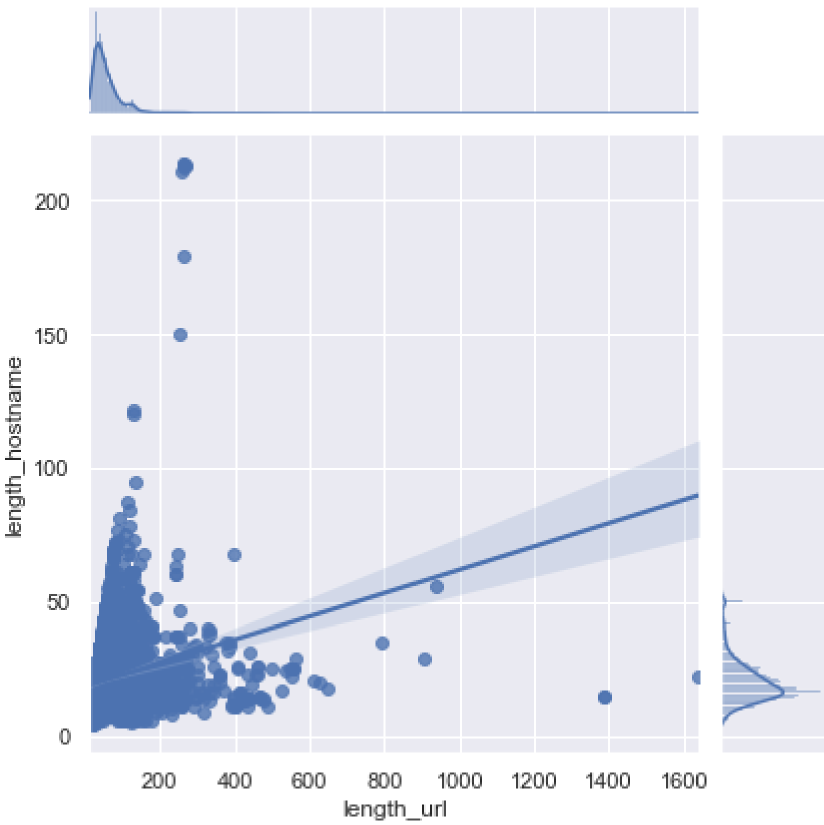
Out[26]:

	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

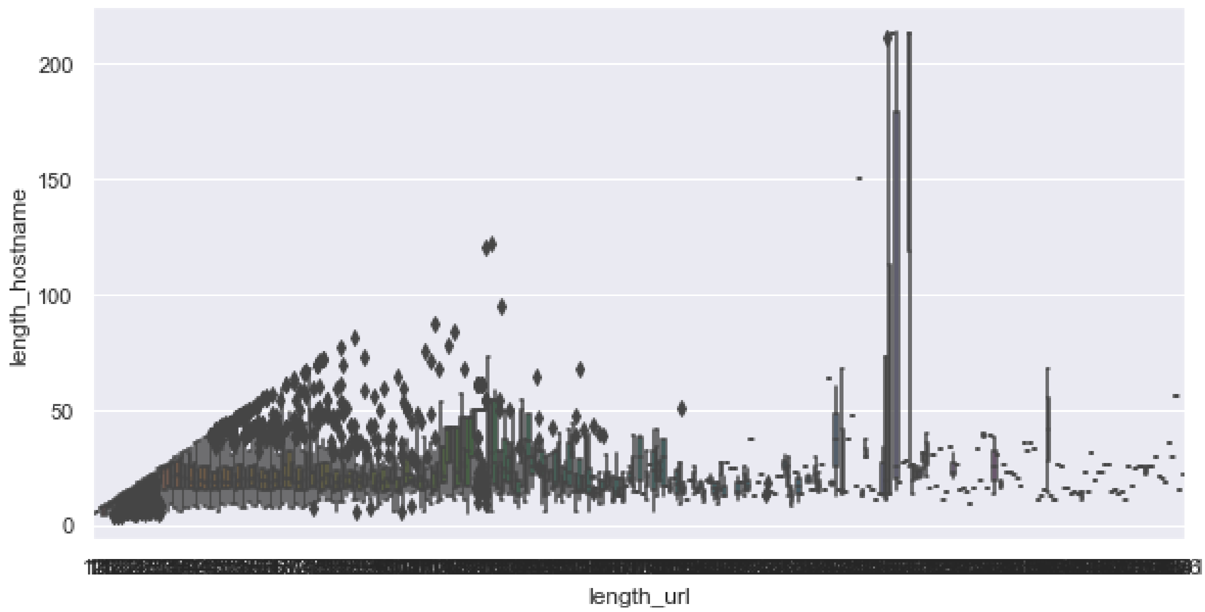
In [27]:
```
sns.jointplot(x="length_url",y="length_hostname",data=dataset,kind="reg")
```

Out[27]: <seaborn.axisgrid.JointGrid at 0x2744c362e80>



In [28]:
```
sns.set(rc={"figure.figsize":(10,5)})
sns.boxplot(x="length_url",y="length_hostname",data=dataset)
```

Out[28]: <AxesSubplot:xlabel='length_url', ylabel='length_hostname'>



In [29]:
```
new_dataset=dataset.drop(["nb_or","sfh","submit_email","ratio_intErrors","ratio_intR
```

In [30]:
```
new_dataset.columns
```

Out[30]:
```
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 [31]:
```
new_dataset=new_dataset.rename(columns={"
```

In [32]:
```
new_dataset
```

Out[32]:

	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	
...	...	...	...	...	...	

	Url	length_url	length_hostname	ip	nb_dots	n
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 [33]:
```
x=new_dataset.iloc[:,1:-1].values
```

In [34]:
```
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 [35]:
```
y=new_dataset.iloc[:,-1].values
```

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

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

In [37]:
```
dataset
```

Out[37]:

	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

Encoding Categorical Data

In [38]:
```
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 [39]:
```
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 [40]:
```
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 [41]:
```
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 [42]:
```
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 [43]:
```
print(y_train)
```

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

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

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

Feature Scaling

In [45]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train[:,:]=sc.fit_transform(x_train[:,:])
x_test[:,:]=sc.transform(x_test[:,:])
```

In [46]:
```
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 [47]:
```
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

In [48]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

Out[48]:
```
LogisticRegression(random_state=0)
```

In [49]:
```
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 [50]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

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

Out[50]:
```
0.9457567804024497
```

In [51]:
```
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

In [52]:
```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[52]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [53]:
```
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 [54]:
```
print(y_train)
```

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

In [55]:
```
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 [56]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

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

Out[56]:
```
0.9323417906095072
```

In [57]:
```
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

In [58]:
```python
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[58]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [59]:
```python
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 [60]:
```python
print(y_train)
```

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

In [61]:
```python
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 [62]:
```python
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

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

Out[62]:
```
0.9661708953047535
```

In [63]:
```python
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

In [64]:
```python
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier()            # Study the parameters in a better way
classifier.fit(x_train,y_train)
```

Out[64]:
```
KNeighborsClassifier()
```

In [65]:
```
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 [66]:
```
print(y_train)
```

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

In [67]:
```
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 [68]:
```
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[68]: 0.8495188101487314

In [69]:
```
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

In [70]:
```
from sklearn.svm import SVC
classifier=SVC(kernel="linear",random_state=0)       # for linear model ----> kernel
classifier.fit(x_train,y_train)
```

Out[70]: SVC(kernel='linear', random_state=0)

In [71]:
```
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 [72]:
```
print(y_train)
```

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

In [73]:
```
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 [74]:
```
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[74]:
```
0.9460484106153397
```

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: 94.58 %
Standard Deviation: 0.56 %
```

Using Naive Bayes Classifier

In [76]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[76]:
```
GaussianNB()
```

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

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

Out[80]:
```
0.573344998541849
```

In [81]:
```
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 [82]:
```
Score={"Classfier":["Logistic","Decision Tree","Random Forest","SVM","KNN","Naive Ba
       "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],"Cm_y_pred(Legitimate
       "Cm_y_test(Legitimate)":[1629,1623,1675,1628,1358,294],"Cm_y_pred(Phising)":[
       ,"Cm_y_test(Phising)":[76,116,64,111,381,1445]}
```

In [83]:
```
Score
```

Out[83]:
```
{'Classfier': ['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],
 'Cm_y_pred(Legitimate)': [1614, 1574, 1638, 1616, 1555, 1672],
 'Cm_y_test(Legitimate)': [1629, 1623, 1675, 1628, 1358, 294],
 'Cm_y_pred(Phising)': [110, 116, 52, 74, 185, 18],
 'Cm_y_test(Phising)': [76, 116, 64, 111, 381, 1445]}
```

Accuracy_Score

In [84]:
```
Accuracy_Score=pd.DataFrame(Score)
```

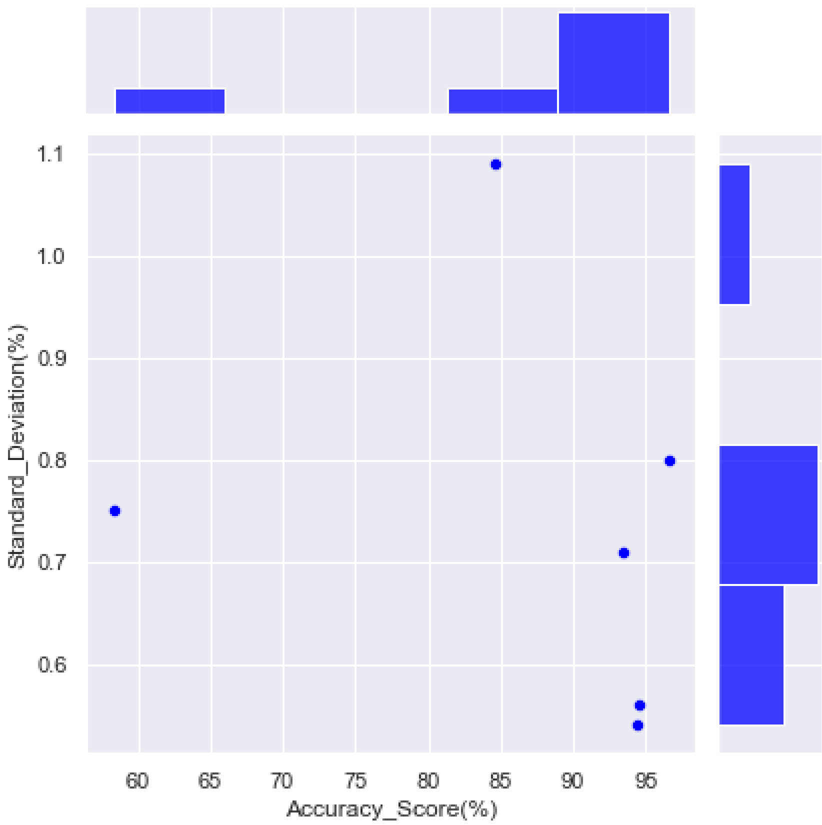
In [85]:
```
Accuracy_Score
```

Out[85]:

	Classfier	Accuracy_Score(%)	Standard_Deviation(%)	Cm_y_pred(Legitimate)	Cm_y_test(Legitimate)
0	Logistic	94.45	0.54	1614	1629
1	Decision Tree	93.49	0.71	1574	1623
2	Random Forest	96.56	0.80	1638	1675
3	SVM	94.58	0.56	1616	1628
4	KNN	84.66	1.09	1555	1358
5	Naive Bayes	58.39	0.75	1672	294

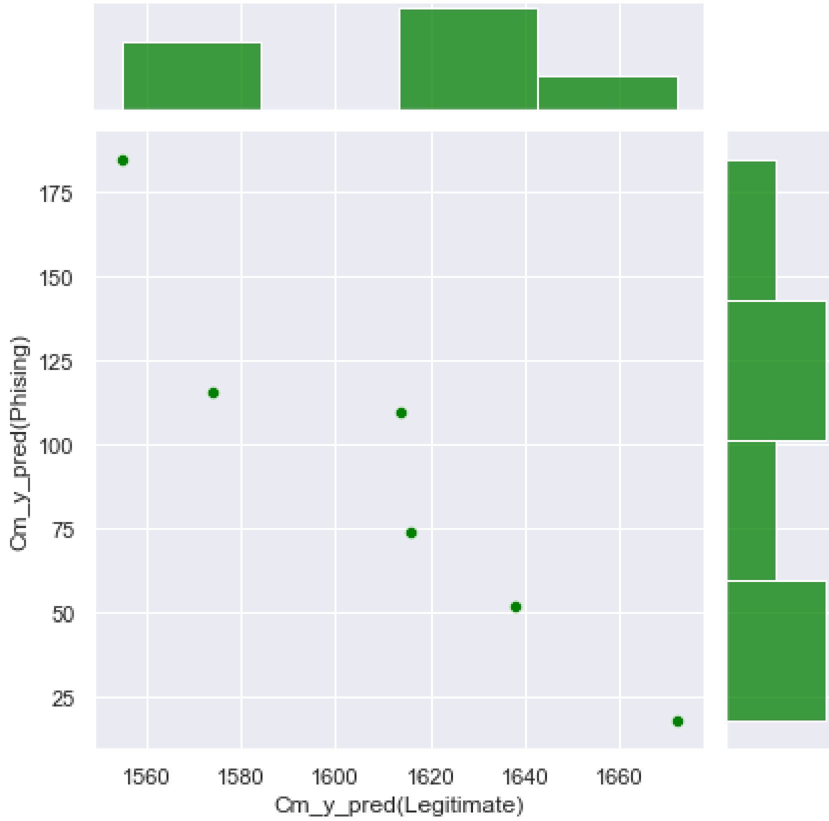
In [86]:
```
sns.jointplot(x="Accuracy_Score(%)",y="Standard_Deviation(%)",data=Accuracy_Score,ki
```

Out[86]:
```
<seaborn.axisgrid.JointGrid at 0x2744fc19d60>
```



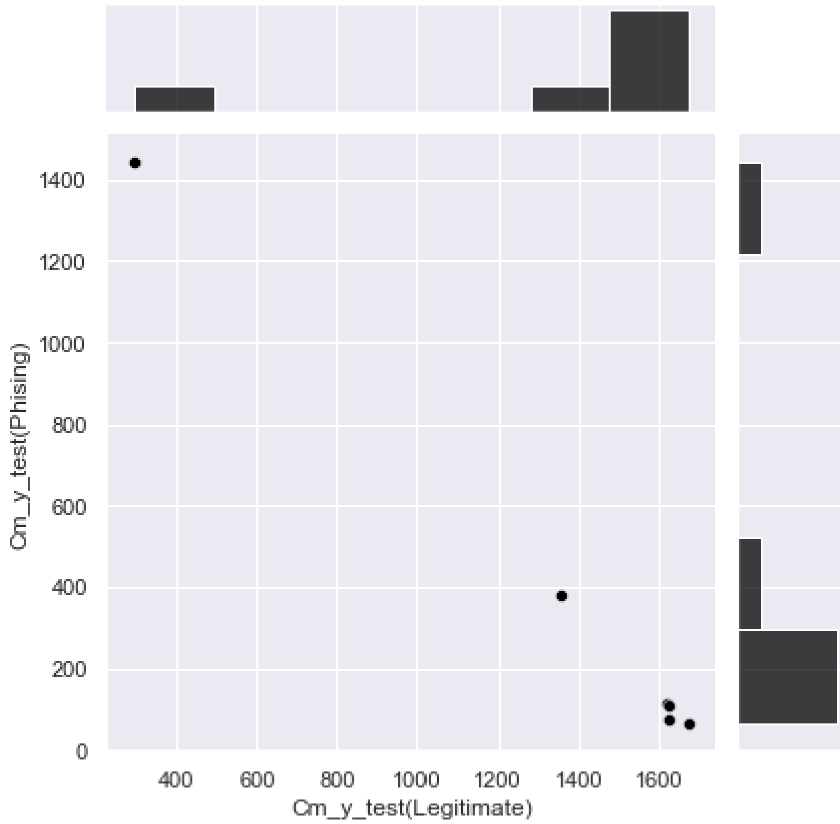
In [87]:
```
sns.jointplot(x="Cm_y_pred(Legitimate)",y="Cm_y_pred(Phising)",data=Accuracy_Score,k
```

Out[87]:
```
<seaborn.axisgrid.JointGrid at 0x274554dd7f0>
```



In [88]:
```
sns.jointplot(x="Cm_y_test(Legitimate)",y="Cm_y_test(Phising)",data=Accuracy_Score,k
```

Out[88]: <seaborn.axisgrid.JointGrid at 0x27455630b50>



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

In [89]:
```
dataset
```

Out[89]:

	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

Importing PCA

In [90]:
```
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 [91]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

Out[91]:
```
LogisticRegression(random_state=0)
```

In [92]:
```
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 [93]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1355  660]
 [ 335 1079]]
```

Out[93]:
```
0.7098279381743948
```

In [94]:
```
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 [95]:
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(criterion="entropy",random_state=0)
classifier.fit(x_train,y_train)
```

Out[95]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [96]:
```
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 [97]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1185  563]
 [ 505 1176]]
```

Out[97]:
```
0.6885389326334208
```

In [98]:
```
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: 80.78 %
Standard Deviation: 1.10 %
```

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

---

Accuracy Score of Naive Bayes Using PCA

In [99]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[99]: GaussianNB()

In [100…
```
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 [101…
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_pred,y_test)
print(cm)
accuracy_score(y_pred,y_test)
```

```
[[1638 1171]
 [  52  568]]
```

Out[101… 0.6433362496354622

In [102…
```
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: 1.14 %
```

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

---

In [103…
```
new_dataset
```

Out[103…

	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

In [ ]:

In [104…

```
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[104…

```
array([1.53109169e-03, 0.00000000e+00, 0.00000000e+00, 8.08940242e-03,
       3.20699399e-03, 5.31501337e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 5.46369295e-03, 3.83429950e-03,
       2.98391631e-03, 6.98627155e-03, 8.98617878e-04, 9.93634603e-04,
       0.00000000e+00, 0.00000000e+00, 4.13529255e-05, 0.00000000e+00,
       2.79492057e-03, 8.02714549e-03, 6.68615546e-03, 2.79836781e-04,
       4.19168114e-04, 0.00000000e+00, 9.33997975e-03, 3.21223303e-03,
       1.22462660e-02, 0.00000000e+00, 0.00000000e+00, 1.83594719e-03,
       2.46870657e-03, 6.54882285e-03, 0.00000000e+00, 0.00000000e+00,
       6.09098287e-03, 0.00000000e+00, 9.88071636e-04, 6.02282598e-03,
       0.00000000e+00, 1.40528396e-02, 1.02134431e-04, 0.00000000e+00,
       7.54153924e-03, 5.53992667e-03, 0.00000000e+00, 1.27781310e-03,
       0.00000000e+00, 1.69076382e-03, 2.72775127e-03, 0.00000000e+00,
       0.00000000e+00, 6.26237681e-03, 3.44139008e-03, 0.00000000e+00,
       1.47878727e-03, 0.00000000e+00, 4.93360752e-03, 1.90713080e-03,
       3.46505093e-03, 6.91247231e-03, 2.34562332e-03, 0.00000000e+00,
       5.49238596e-03, 3.77252870e-03, 3.97791426e-03, 0.00000000e+00,
       3.48898998e-04, 0.00000000e+00, 4.78289049e-03, 0.00000000e+00,
       9.40601991e-03, 0.00000000e+00, 0.00000000e+00, 8.33305353e-03,
       4.17626651e-03, 1.37808810e-02, 0.00000000e+00, 6.08252985e-04,
       0.00000000e+00, 0.00000000e+00, 1.68164600e-03, 3.45411550e-03,
       0.00000000e+00, 4.00184067e-03, 3.52187192e-03, 3.28559638e-03,
       0.00000000e+00, 4.28170847e-03, 8.79679096e-03, 7.17316603e-03,
       0.00000000e+00, 9.35131086e-03, 1.27396764e-03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 2.66783474e-03, 0.00000000e+00, 5.59447300e-03,
       0.00000000e+00, 5.02535842e-03, 6.50786491e-04, 1.36013811e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.75162450e-03,
       3.01567211e-03, 0.00000000e+00, 5.41163752e-03, 2.29668817e-03,
       1.85196728e-03, 1.19257973e-02, 7.08840031e-03, 2.07628857e-03,
       4.32206869e-03, 1.03952919e-03, 0.00000000e+00, 2.86294472e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 9.81453054e-03,
       0.00000000e+00, 0.00000000e+00, 9.71035998e-03, 4.43338606e-03,
       3.15207322e-03, 3.27739669e-03, 8.65712331e-03, 2.86885317e-03,
       0.00000000e+00, 0.00000000e+00, 8.38461066e-03, 3.13211871e-03,
       0.00000000e+00, 3.45842353e-03, 1.33470664e-02, 0.00000000e+00,
       8.26287395e-03, 6.38741837e-03, 2.94674069e-03, 2.84491171e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 4.40840254e-03,
       1.70121295e-03, 9.86277239e-03, 1.01775587e-03, 0.00000000e+00,
       0.00000000e+00, 7.46686927e-04, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 2.38321793e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.08424940e-02, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 1.47481641e-03, 2.10123952e-03,
       9.82802319e-04, 7.65124669e-03, 0.00000000e+00, 6.19462460e-04,
       2.37986758e-04, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       1.00664045e-02, 1.36472311e-02, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 2.45138219e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       4.79279895e-03, 3.13123622e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 5.90211872e-03, 0.00000000e+00, 2.80762392e-03,
       0.00000000e+00, 2.53112283e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 5.18288856e-03, 1.42624815e-03,
       6.73265423e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       1.51967984e-04, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       9.06805270e-03, 8.14960124e-03, 1.84766226e-03, 1.47693011e-02,
```

```
       4.56182546e-03, 1.45965566e-03, 0.00000000e+00, 0.00000000e+00,
       2.24557470e-03, 0.00000000e+00, 2.94429119e-03, 1.20517631e-04,
       1.15568160e-03, 0.00000000e+00, 0.00000000e+00, 9.97185766e-04,
       2.97906086e-03, 1.17408055e-04, 1.08356479e-02, 0.00000000e+00,
       4.48980262e-03, 1.27953970e-03, 1.04022962e-02, 8.10677086e-03,
       0.00000000e+00, 0.00000000e+00, 4.38838549e-03, 0.00000000e+00,
       2.04234058e-04, 3.11280373e-03, 0.00000000e+00, 6.60880716e-03,
       0.00000000e+00, 0.00000000e+00, 9.54015202e-03, 0.00000000e+00,
       0.00000000e+00, 9.03714171e-03, 4.87356827e-04, 0.00000000e+00,
       0.00000000e+00, 4.12995103e-03, 5.67670635e-03, 1.77747021e-03,
       6.49559466e-03, 0.00000000e+00, 0.00000000e+00, 5.08611800e-03,
       1.16769607e-02, 1.13694301e-03, 0.00000000e+00, 0.00000000e+00,
       1.21474874e-03, 0.00000000e+00, 0.00000000e+00, 1.45934502e-04,
       0.00000000e+00, 8.02668963e-03, 6.18980103e-03, 0.00000000e+00,
       6.77381713e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 8.76804734e-03, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 4.18000228e-03, 7.98015984e-04, 8.14256706e-04,
       1.80113708e-03, 2.06366710e-03, 0.00000000e+00, 4.33449117e-03,
       1.30935660e-03, 9.59636969e-04, 1.09163817e-02, 1.27604686e-04,
       1.05372105e-02, 5.33525205e-03, 0.00000000e+00, 0.00000000e+00,
       5.10115986e-03, 5.83960775e-03, 5.63409801e-04, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.70279963e-03, 0.00000000e+00, 4.16154123e-03,
       2.75483564e-03, 0.00000000e+00, 4.50818412e-03, 6.05263894e-03,
       6.54358856e-02, 5.40014567e-02, 6.21839878e-02, 3.81255839e-02,
       1.30815051e-02, 4.39974168e-02, 3.23049896e-02, 6.17479587e-02,
       7.91417433e-03, 0.00000000e+00, 6.37550708e-03, 3.15730507e-02,
       4.26863533e-03, 0.00000000e+00, 0.00000000e+00, 2.33481490e-02,
       0.00000000e+00, 0.00000000e+00, 1.03455437e-01, 2.00539264e-02,
       0.00000000e+00, 3.98614252e-03, 0.00000000e+00, 9.32610830e-02,
       4.11918465e-02, 0.00000000e+00, 7.91251032e-03, 8.36091889e-03,
       2.73681868e-02, 7.87953614e-03, 2.34962843e-02, 3.11903861e-02,
       0.00000000e+00, 9.11895529e-03, 0.00000000e+00, 0.00000000e+00,
       3.34702166e-03, 4.72488947e-02, 8.23973059e-02, 5.77405740e-02,
       7.55073284e-02, 4.80486730e-02, 6.68063050e-02, 1.67742327e-02,
       1.12502521e-01, 6.76388560e-02, 7.84961342e-02, 7.94527993e-02,
       8.43456926e-02, 1.28443828e-02, 9.34779136e-03, 0.00000000e+00,
       1.08909246e-02, 1.17713232e-02, 2.32459767e-01, 2.34291061e-01,
       2.32305669e-01, 1.22047440e-02, 1.16134580e-01, 7.57822601e-02,
       0.00000000e+00, 5.22248219e-03, 1.19532969e-01, 6.86505505e-02,
       6.45713362e-02, 0.00000000e+00, 1.27498234e-03, 1.79479573e-01,
       8.55811347e-03, 1.57083470e-03, 2.37805275e-02, 6.02418476e-02,
       5.90060561e-03, 1.63716024e-02, 1.51856284e-01, 2.72006275e-01,
       2.89398906e-01, 4.45014326e-03, 2.99082357e-01, 2.15360317e-01])
```

In [105…

```
x_train
```

Out[105…

```
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 [106…

```
mutual_info_classif(x_train,y_train)    # mutual information always gives either pos
```

Out[106…

```
array([1.76090497e-03, 0.00000000e+00, 0.00000000e+00, 5.69984964e-03,
       1.28339530e-02, 0.00000000e+00, 6.74799799e-03, 0.00000000e+00,
       1.10942248e-02, 1.29071552e-03, 0.00000000e+00, 0.00000000e+00,
       6.20279080e-03, 0.00000000e+00, 3.30152747e-03, 5.16219282e-03,
       4.75943946e-03, 1.60572361e-02, 1.54782225e-03, 3.83900819e-03,
       0.00000000e+00, 0.00000000e+00, 2.58315696e-04, 5.99173948e-04,
       4.39855243e-03, 0.00000000e+00, 8.00894775e-03, 1.16597384e-03,
       0.00000000e+00, 6.32635894e-03, 0.00000000e+00, 6.97708293e-03,
       1.50868542e-03, 1.77257016e-03, 0.00000000e+00, 8.40146623e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       1.20791458e-03, 3.77773475e-03, 1.54926129e-02, 0.00000000e+00,
       9.88921829e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
```

```
0.00000000e+00, 0.00000000e+00, 4.13521247e-03, 3.00845717e-04,
3.08201146e-03, 0.00000000e+00, 4.87421373e-03, 5.57472457e-03,
2.02299604e-03, 0.00000000e+00, 8.87866233e-03, 2.86586372e-05,
2.39124424e-03, 0.00000000e+00, 6.19998838e-03, 0.00000000e+00,
2.00525221e-03, 0.00000000e+00, 2.79009184e-04, 0.00000000e+00,
6.06096392e-04, 4.81162367e-03, 8.50449799e-04, 3.71950824e-04,
4.49259800e-03, 0.00000000e+00, 1.78071361e-03, 0.00000000e+00,
9.82532956e-04, 3.25147737e-03, 0.00000000e+00, 0.00000000e+00,
1.46059773e-03, 0.00000000e+00, 3.51187191e-03, 1.61797920e-03,
1.00484713e-02, 1.27065133e-02, 1.12858184e-02, 8.17406197e-03,
1.20595980e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
2.11702480e-03, 7.44168030e-04, 0.00000000e+00, 0.00000000e+00,
4.81542980e-03, 5.59273155e-03, 4.19929357e-03, 0.00000000e+00,
0.00000000e+00, 2.16506370e-03, 1.17315348e-02, 1.03066626e-02,
0.00000000e+00, 0.00000000e+00, 2.45436505e-03, 1.43203083e-03,
7.34072169e-04, 0.00000000e+00, 3.43671380e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.68288001e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.23296838e-03,
0.00000000e+00, 3.25818561e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
6.72179696e-03, 0.00000000e+00, 0.00000000e+00, 7.02001594e-03,
5.14169113e-03, 0.00000000e+00, 6.95496824e-03, 0.00000000e+00,
8.82017290e-03, 9.15120851e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 9.21484714e-03, 0.00000000e+00, 0.00000000e+00,
7.58726932e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 3.47960076e-03, 0.00000000e+00, 1.57338000e-02,
0.00000000e+00, 8.12766109e-03, 7.87875378e-03, 2.43220948e-04,
0.00000000e+00, 0.00000000e+00, 2.65884606e-03, 0.00000000e+00,
4.85806546e-03, 7.70781618e-03, 5.06289837e-03, 3.64618452e-04,
0.00000000e+00, 7.04500604e-03, 0.00000000e+00, 0.00000000e+00,
7.09976202e-03, 0.00000000e+00, 9.09864234e-05, 1.73431813e-03,
4.60651112e-03, 0.00000000e+00, 0.00000000e+00, 9.52544026e-03,
1.01893623e-02, 0.00000000e+00, 0.00000000e+00, 1.31394057e-03,
0.00000000e+00, 0.00000000e+00, 8.64706655e-04, 5.72191994e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 5.27167638e-03,
2.33570582e-03, 0.00000000e+00, 9.74223751e-03, 0.00000000e+00,
4.69338096e-03, 0.00000000e+00, 1.60176944e-03, 3.71310703e-04,
0.00000000e+00, 0.00000000e+00, 5.94956302e-03, 3.58167767e-04,
0.00000000e+00, 0.00000000e+00, 6.40575441e-03, 4.49140350e-03,
1.19877444e-03, 5.60935044e-03, 0.00000000e+00, 1.62666988e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 5.30161260e-03, 2.14717438e-03, 9.65062577e-03,
0.00000000e+00, 0.00000000e+00, 6.35535441e-03, 3.43650678e-03,
4.53739930e-03, 8.25646041e-04, 3.38154138e-03, 0.00000000e+00,
0.00000000e+00, 1.58267956e-03, 1.12488085e-02, 7.39772718e-03,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 7.81062274e-03, 0.00000000e+00, 9.24858248e-03,
5.29838590e-03, 0.00000000e+00, 5.51788992e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 2.33271251e-03, 9.45682967e-04,
0.00000000e+00, 2.59201240e-03, 5.00669781e-04, 6.00082837e-03,
2.42586586e-03, 3.01626927e-03, 0.00000000e+00, 0.00000000e+00,
7.63079957e-04, 7.69856206e-03, 0.00000000e+00, 4.65472164e-04,
3.05444584e-03, 0.00000000e+00, 1.18600806e-03, 2.87715896e-03,
1.86783212e-03, 0.00000000e+00, 0.00000000e+00, 4.70360257e-03,
2.04454413e-03, 2.62915046e-03, 3.04414170e-03, 0.00000000e+00,
0.00000000e+00, 5.49700289e-03, 0.00000000e+00, 0.00000000e+00,
6.82448823e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
2.02453793e-03, 0.00000000e+00, 0.00000000e+00, 3.49481831e-03,
1.07792630e-02, 1.19765997e-03, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 5.41195566e-03, 0.00000000e+00, 3.66276789e-03,
3.85392319e-04, 0.00000000e+00, 0.00000000e+00, 8.90625928e-03,
9.00424096e-03, 8.00579700e-03, 0.00000000e+00, 4.00525994e-03,
0.00000000e+00, 3.48799138e-03, 8.03750381e-03, 0.00000000e+00,
1.02153164e-02, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.89243472e-03, 3.13917831e-03, 1.38496961e-03, 0.00000000e+00,
0.00000000e+00, 1.04598658e-02, 1.03929878e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 3.81474160e-03, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 5.34655668e-03, 5.62639571e-03,
3.80407796e-03, 9.72170560e-03, 2.31017974e-03, 0.00000000e+00,
```

```
       6.48025156e-02, 4.96747878e-02, 8.15705212e-02, 3.50136425e-02,
       2.11709904e-02, 5.66073784e-02, 3.30246380e-02, 4.94110017e-02,
       1.07246962e-02, 8.64734335e-03, 2.96723638e-03, 2.84802562e-02,
       1.64069982e-02, 1.63259803e-03, 7.29765356e-03, 1.34322100e-02,
       2.81209359e-03, 1.94272668e-02, 1.06754823e-01, 1.63801572e-02,
       1.81791952e-02, 0.00000000e+00, 8.31526142e-03, 1.02959066e-01,
       5.07511337e-02, 0.00000000e+00, 6.30581522e-03, 1.00079794e-02,
       2.41936664e-02, 1.09148604e-02, 2.63485295e-02, 2.79403004e-02,
       3.11793604e-03, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       9.75346666e-04, 3.09582158e-02, 8.35521816e-02, 4.90197385e-02,
       6.69417679e-02, 3.64981884e-02, 7.98666909e-02, 2.25810945e-02,
       1.16242870e-01, 6.47078104e-02, 8.39454254e-02, 8.49774065e-02,
       7.43451591e-02, 0.00000000e+00, 0.00000000e+00, 8.14298129e-03,
       9.59842322e-03, 1.58062629e-03, 2.31233040e-01, 2.31136905e-01,
       2.32102466e-01, 1.92631045e-02, 1.19682220e-01, 7.64027458e-02,
       7.86817534e-03, 7.13504863e-03, 1.12565706e-01, 7.31681324e-02,
       6.73904993e-02, 0.00000000e+00, 0.00000000e+00, 1.78960045e-01,
       1.22273845e-03, 0.00000000e+00, 2.78760646e-02, 5.89007626e-02,
       1.36938852e-02, 1.07537262e-02, 1.44602393e-01, 2.74590659e-01,
       2.87366075e-01, 8.76898946e-03, 2.98813476e-01, 2.18223981e-01])
```

In [107…

```
y_train
```

Out[107…

```
array([0, 0, 0, ..., 1, 1, 1])
```

In [108…

```
import pandas as pd
mutual_data=pd.Series(mutual_info)
mutual_data
```

Out[108…

```
0      0.001531
1      0.000000
2      0.000000
3      0.008089
4      0.003207
         ...
399    0.272006
400    0.289399
401    0.004450
402    0.299082
403    0.215360
Length: 404, dtype: float64
```

In [109…

```
pd.Series(mutual_info).sort_values(ascending=False)
```

Out[109…

```
402    0.299082
400    0.289399
399    0.272006
379    0.234291
378    0.232460
         ...
127    0.000000
126    0.000000
125    0.000000
124    0.000000
201    0.000000
Length: 404, dtype: float64
```

In [110…

```
mutual_data.nlargest(10)     # Features with top 10 Largest Score
```

Out[110…

```
402    0.299082
400    0.289399
399    0.272006
379    0.234291
378    0.232460
380    0.232306
403    0.215360
391    0.179480
```

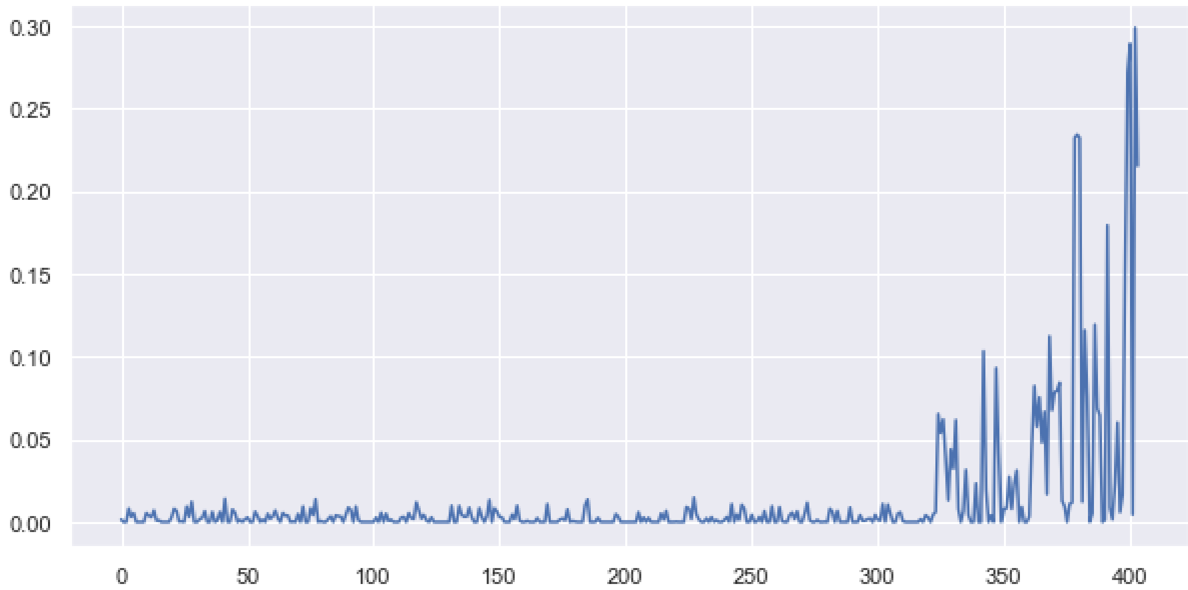
```
398    0.151856
386    0.119533
dtype: float64
```

In [111…

```
import matplotlib.pyplot as plt
mutual_data.plot()
```

Out[111…

```
<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 [112…

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

Out[112…

```
ExtraTreesClassifier()
```

In [113…

```
print(model.feature_importances_)
```

```
[0.00000000e+00 5.65576451e-05 0.00000000e+00 1.54932579e-04
 2.23342519e-04 3.31281514e-04 4.27115783e-04 5.17656184e-04
 6.36721069e-04 8.15509061e-04 5.59268107e-04 1.08016687e-03
 1.10654248e-03 8.60703307e-04 1.05226241e-03 1.21595904e-03
 1.21596065e-03 9.66381304e-04 7.59753033e-04 7.74570646e-04
 1.01199210e-03 1.27920364e-03 9.90383360e-04 9.65017709e-04
 9.69964859e-04 1.12576078e-03 8.26537450e-04 1.10151436e-03
 7.54547918e-04 1.02397290e-03 9.50514516e-04 1.23601570e-03
 9.03674453e-04 1.04160361e-03 7.10982830e-04 1.08502987e-03
 9.76677995e-04 1.10831407e-03 6.20382613e-04 8.11825169e-04
 6.08291507e-04 6.38841923e-04 5.63750517e-04 6.00441708e-04
 7.26481119e-04 5.84875006e-04 6.13326627e-04 5.54768602e-04
```

7.18278353e-04	4.88970479e-04	3.68852846e-04	6.24486271e-04
5.70939702e-04	2.74377738e-04	3.77427648e-04	5.14881504e-04
5.68803207e-04	1.70233556e-04	5.42268938e-04	3.20949954e-04
3.77311947e-04	2.35437657e-04	5.24373331e-04	5.31886513e-04
3.37586934e-04	3.68331259e-04	3.40746374e-04	3.49459887e-04
5.18843506e-04	7.60695647e-04	2.41373695e-04	3.30682177e-04
6.70692798e-04	3.11891575e-04	1.81477206e-04	2.18366754e-04
2.29778615e-04	3.26938356e-04	1.34113503e-04	2.37502628e-04
1.32548026e-04	1.28193158e-04	1.16361547e-04	9.01831437e-05
2.25765011e-04	1.18831588e-04	1.64984672e-04	3.81794497e-04
8.90715344e-05	2.18539840e-04	9.64518174e-05	1.62465693e-04
9.99446462e-05	1.38929287e-04	1.06730658e-04	9.10212376e-05
9.30351188e-05	3.23636184e-05	1.21439907e-04	7.99116941e-05
1.07920614e-04	1.83932640e-04	2.16833153e-05	2.17199626e-04
6.37618595e-05	1.04997782e-04	1.19345706e-04	7.81943717e-05
7.96026776e-05	8.05362553e-05	3.53489806e-05	1.31580735e-04
1.64609121e-04	3.84451361e-05	3.31124784e-04	2.51324801e-04
5.19398308e-04	3.05745554e-05	5.89316117e-05	3.45847420e-05
4.70227109e-04	1.85553287e-05	2.62832502e-07	8.77022982e-05
8.54738724e-06	9.70124742e-06	8.40719433e-05	1.04744589e-05
1.61993619e-05	2.54044020e-05	6.76019225e-05	8.06023313e-05
8.28066526e-06	1.55060136e-05	3.51189399e-05	9.82530278e-06
2.06694206e-05	1.21662008e-05	2.88370309e-06	4.00878791e-06
4.20216978e-06	2.13244712e-05	1.65616980e-08	2.84045395e-05
0.00000000e+00	3.25916006e-06	3.69525494e-06	5.02710824e-05
1.09504664e-06	0.00000000e+00	6.15183845e-06	5.66039344e-06
3.27373908e-06	1.73556948e-05	2.34585784e-06	1.74629793e-05
2.16566248e-06	3.88117185e-05	0.00000000e+00	1.96061278e-09
2.82026608e-07	1.77492347e-05	0.00000000e+00	8.33260432e-07
0.00000000e+00	2.63907329e-06	1.07133484e-06	8.42976834e-06
0.00000000e+00	5.64081252e-05	0.00000000e+00	2.74701241e-08
0.00000000e+00	1.09269719e-06	1.23066156e-07	1.59121346e-05
3.26058430e-07	0.00000000e+00	4.85259267e-05	8.80973446e-06
6.94383693e-08	0.00000000e+00	0.00000000e+00	0.00000000e+00
7.28711679e-06	3.16891252e-05	2.80271928e-05	4.13822103e-06
0.00000000e+00	3.51621778e-06	3.69015334e-07	1.80274341e-05
0.00000000e+00	1.58716273e-08	6.25598862e-06	0.00000000e+00
0.00000000e+00	1.12211033e-09	1.92656777e-06	3.65013625e-06
0.00000000e+00	0.00000000e+00	1.84429433e-05	0.00000000e+00
0.00000000e+00	0.00000000e+00	3.15683980e-05	9.76005139e-06
1.75294319e-06	9.78175290e-07	0.00000000e+00	0.00000000e+00
2.93274995e-06	0.00000000e+00	2.68804387e-06	6.54734068e-06
8.33260432e-07	0.00000000e+00	0.00000000e+00	1.58042455e-05
1.03291004e-05	0.00000000e+00	6.37866355e-06	2.62385838e-06
0.00000000e+00	8.32951931e-09	3.17839048e-06	0.00000000e+00
3.80496462e-06	9.52297637e-08	0.00000000e+00	0.00000000e+00
1.03341478e-06	0.00000000e+00	3.70846317e-06	6.16134054e-06
8.23457368e-07	2.59001561e-06	2.67518238e-05	0.00000000e+00
0.00000000e+00	0.00000000e+00	8.47672811e-06	4.69533435e-06
3.85680543e-06	3.34965157e-07	9.32928871e-08	3.33304173e-06
1.00999169e-05	0.00000000e+00	0.00000000e+00	2.77032040e-06
8.27069204e-06	0.00000000e+00	0.00000000e+00	1.23005111e-07
0.00000000e+00	0.00000000e+00	0.00000000e+00	8.88811128e-07
5.64173357e-06	5.52790465e-06	4.54505690e-08	0.00000000e+00
2.15667406e-06	1.81762987e-05	3.65776399e-06	0.00000000e+00
1.66652086e-07	1.17058002e-05	0.00000000e+00	0.00000000e+00
0.00000000e+00	6.60751046e-06	0.00000000e+00	2.49978130e-06
0.00000000e+00	0.00000000e+00	4.84453740e-07	2.66643338e-06
0.00000000e+00	0.00000000e+00	5.69023300e-06	0.00000000e+00
3.52626068e-06	8.33260432e-08	0.00000000e+00	7.22231438e-06
1.07913109e-05	2.58310734e-06	0.00000000e+00	0.00000000e+00
0.00000000e+00	0.00000000e+00	0.00000000e+00	9.61454345e-08
0.00000000e+00	0.00000000e+00	2.40219224e-07	0.00000000e+00
0.00000000e+00	0.00000000e+00	3.74967194e-06	0.00000000e+00
0.00000000e+00	0.00000000e+00	1.79830459e-06	5.35185785e-06
2.65635330e-05	3.70813556e-06	1.76804995e-05	1.81989702e-05
5.38006543e-06	9.49963927e-06	3.55765332e-05	0.00000000e+00
2.66643338e-06	0.00000000e+00	3.76890103e-07	0.00000000e+00
0.00000000e+00	0.00000000e+00	4.27906511e-06	0.00000000e+00

```
 1.21884168e-02 2.08003871e-02 9.77614264e-03 9.02540291e-03
 3.15822504e-03 1.08162173e-02 2.90703941e-03 5.34509380e-03
 3.54202235e-03 4.66226581e-04 1.68100136e-03 1.15179815e-02
 1.19964573e-05 1.70902487e-03 1.74443599e-04 9.84044339e-04
 1.50702867e-05 1.42801518e-03 5.38748441e-02 2.61489911e-03
 7.95065276e-04 1.73330552e-03 6.15139497e-03 1.70211251e-02
 9.03624502e-03 7.75588325e-06 4.58127997e-04 2.24047919e-03
 5.58142304e-03 3.31506390e-03 1.12222926e-02 1.07542224e-02
 2.38691857e-03 7.28134021e-03 1.16446098e-04 7.97173646e-03
 7.69078806e-05 9.53723233e-03 8.98565767e-03 8.36371390e-03
 1.15477088e-02 9.47801851e-03 1.00130278e-02 7.39412526e-03
 1.08880497e-02 7.40789622e-03 9.50125065e-03 8.31453605e-03
 2.09125755e-02 7.97186251e-03 2.95512636e-04 2.13573054e-04
 3.21321985e-03 3.32842150e-03 1.62771293e-02 2.42261988e-02
 1.68935014e-02 5.59407116e-03 9.50740483e-03 7.74470574e-03
 2.89440832e-03 9.62561440e-03 1.78429333e-02 1.16884106e-02
 1.05997973e-02 1.80774537e-04 6.99393513e-04 2.41259840e-02
 4.63893835e-05 2.27986070e-04 1.07964365e-02 3.20680161e-02
 1.94702914e-02 2.56871357e-03 1.12377596e-02 2.86855750e-02
 8.24321118e-03 2.96699619e-03 2.13683604e-01 7.94163167e-02]
```

In [114…

```
model.feature_importances_.shape
```

Out[114…

```
(404,)
```

In [115…

```
Feature_imp=pd.Series(model.feature_importances_)
Feature_imp
```

Out[115…

```
0      0.000000
1      0.000057
2      0.000000
3      0.000155
4      0.000223
         ...
399    0.028686
400    0.008243
401    0.002967
402    0.213684
403    0.079416
Length: 404, dtype: float64
```

In [116…

```
pd.Series(model.feature_importances_).nlargest(10)
```

Out[116…

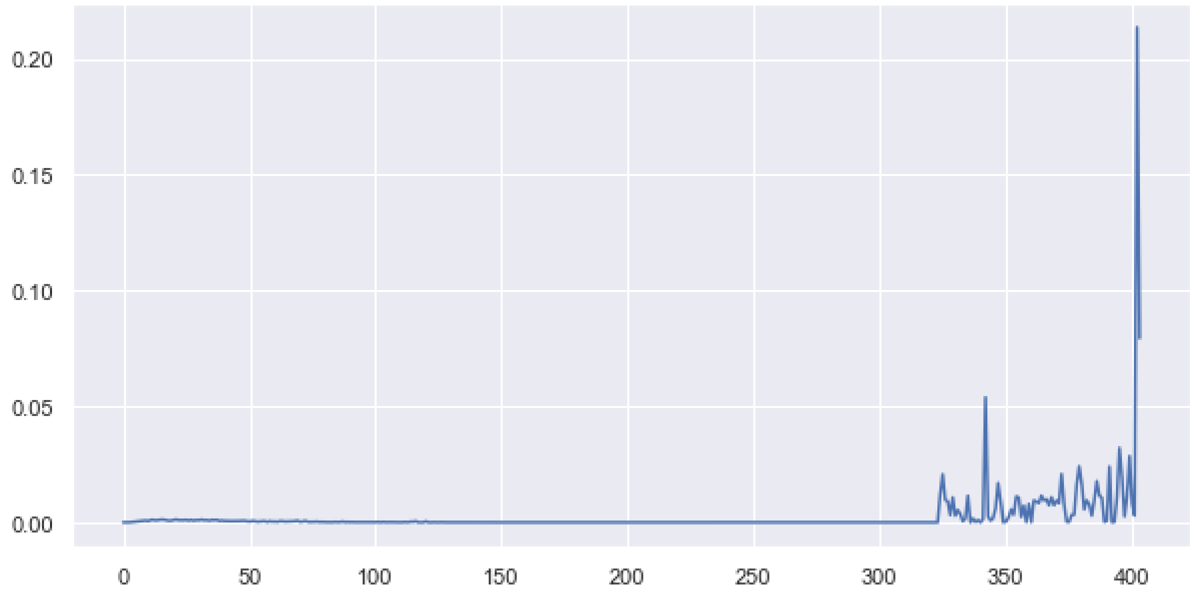
```
402    0.213684
403    0.079416
342    0.053875
395    0.032068
399    0.028686
379    0.024226
391    0.024126
372    0.020913
325    0.020800
396    0.019470
dtype: float64
```

In [117…

```
Feature_imp.plot()
```

Out[117…

```
<AxesSubplot:>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: