

In [1]:

```

# Importing Data Manipulation Libraries
import pandas as pd
import numpy as np

# Import Data Visualization Libraries

import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Import Data Filter Libraries
import warnings
warnings.filterwarnings('ignore')

# Import Data Logging Libraries
import logging
logging.basicConfig(level = logging.INFO,
                    filename = 'model.log',
                    filemode = 'w',
                    format = '%(asctime)s - %(levelname)s - %(message)s')

# Multicollinearity test and treatment Libraries
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA

```

In [2]:

```

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 100)

```

Loading Dataset

In [3]:

```

# Loading the dataset

url = 'https://raw.githubusercontent.com/mukeshmagar543/CODEB_Internship/ref

df = pd.read_csv(url)

df.sample(frac = 1) # Data Shuffle

```

Out[3]:

	url	length_url	length_hostname	i
4329	https://user9765656787.et.r.appspot.com/jim@th...	65	31	

9236	http://encarnacionhurtado.com/img/	34	22
8552	https://www.finchainforum.it	28	20
11103	https://firebasestorage.googleapis.com/v0/b/ap...	132	30
6315	https://www.google.com/url?sa=t&rct=j&q=&esrc=...	203	14
...
7501	http://justlookapp.com/blog/wp-content/themes/...	103	15
2590	http://www.bionity.com/en/encyclopedia/Virus_I...	57	15
10106	http://work.chron.com/yearly-salary-cnc-machin...	59	14
6098	http://www.tech-faq.com/forward-error-correcti...	57	16
10471	http://bibliateologiaefilosofia.blogspot.com/	45	37

11430 rows × 89 columns



Getting Information about Dataset Like which column is object and which column is numerical

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11430 entries, 0 to 11429
Data columns (total 89 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   url                                    11430 non-null  object
1   length_url                            11430 non-null  int64
2   length_hostname                       11430 non-null  int64
3   ip                                    11430 non-null  int64
4   nb_dots                               11430 non-null  int64
5   nb_hyphens                            11430 non-null  int64
6   nb_at                                 11430 non-null  int64
7   nb_qm                                 11430 non-null  int64
8   nb_and                                11430 non-null  int64
9   nb_or                                 11430 non-null  int64
10  nb_eq                                 11430 non-null  int64
11  nb_underscore                         11430 non-null  int64
12  nb_tilde                              11430 non-null  int64
13  nb_percent                            11430 non-null  int64
14  nb_slash                              11430 non-null  int64
15  nb_star                               11430 non-null  int64
16  nb_colon                              11430 non-null  int64
17  nb_comma                              11430 non-null  int64
...
```

18	nb_semicolumn	11430	non-null	int64
19	nb_dollar	11430	non-null	int64
20	nb_space	11430	non-null	int64
21	nb_www	11430	non-null	int64
22	nb_com	11430	non-null	int64
23	nb_dslash	11430	non-null	int64
24	http_in_path	11430	non-null	int64
25	https_token	11430	non-null	int64
26	ratio_digits_url	11430	non-null	float64
27	ratio_digits_host	11430	non-null	float64
28	punycode	11430	non-null	int64
29	port	11430	non-null	int64
30	tld_in_path	11430	non-null	int64
31	tld_in_subdomain	11430	non-null	int64
32	abnormal_subdomain	11430	non-null	int64
33	nb_subdomains	11430	non-null	int64
34	prefix_suffix	11430	non-null	int64
35	random_domain	11430	non-null	int64
36	shortening_service	11430	non-null	int64
37	path_extension	11430	non-null	int64
38	nb_redirection	11430	non-null	int64
39	nb_external_redirection	11430	non-null	int64
40	length_words_raw	11430	non-null	int64
41	char_repeat	11430	non-null	int64
42	shortest_words_raw	11430	non-null	int64
43	shortest_word_host	11430	non-null	int64
44	shortest_word_path	11430	non-null	int64
45	longest_words_raw	11430	non-null	int64
46	longest_word_host	11430	non-null	int64
47	longest_word_path	11430	non-null	int64
48	avg_words_raw	11430	non-null	float64
49	avg_word_host	11430	non-null	float64
50	avg_word_path	11430	non-null	float64
51	phish_hints	11430	non-null	int64
52	domain_in_brand	11430	non-null	int64
53	brand_in_subdomain	11430	non-null	int64
54	brand_in_path	11430	non-null	int64
55	suspicious_tld	11430	non-null	int64
56	statistical_report	11430	non-null	int64
57	nb_hyperlinks	11430	non-null	int64
58	ratio_intHyperlinks	11430	non-null	float64
59	ratio_extHyperlinks	11430	non-null	float64
60	ratio_nullHyperlinks	11430	non-null	int64
61	nb_extCSS	11430	non-null	int64
62	ratio_intRedirection	11430	non-null	int64
63	ratio_extRedirection	11430	non-null	float64
64	ratio_intErrors	11430	non-null	int64
65	ratio_extErrors	11430	non-null	float64
66	login_form	11430	non-null	int64
67	external_favicon	11430	non-null	int64
68	links_in_tags	11430	non-null	float64
69	submit_email	11430	non-null	int64
70	ratio_intMedia	11430	non-null	float64
71	ratio_extMedia	11430	non-null	float64
72	sfh	11430	non-null	int64
73	iframe	11430	non-null	int64
74	popup_window	11430	non-null	int64
75	safe_anchor	11430	non-null	float64
76	onmouseover	11430	non-null	int64
77	right_clic	11430	non-null	int64
78	empty_title	11430	non-null	int64
79	domain_in_title	11430	non-null	int64
80	domain_with_copyright	11430	non-null	int64
81	whois_registered_domain	11430	non-null	int64

```

82  domain_registration_length  11430 non-null  int64
83  domain_age                  11430 non-null  int64
84  web_traffic                 11430 non-null  int64
85  dns_record                  11430 non-null  int64
86  google_index                11430 non-null  int64
87  page_rank                   11430 non-null  int64
88  status                      11430 non-null  object

```

dtypes: float64(13), int64(74), object(2)

memory usage: 7.8+ MB

Checking Null Values

- There is No Null Values are present in the given dataset.

In [5]:

```
df.isnull().sum()
```

```

Out[5]:  url                  0
        length_url          0
        length_hostname     0
        ip                  0
        nb_dots              0
        nb_hyphens           0
        nb_at                0
        nb_qm                0
        nb_and               0
        nb_or                0
        nb_eq                0
        nb_underscore        0
        nb_tilde             0
        nb_percent           0
        nb_slash             0
        nb_star              0
        nb_colon             0
        nb_comma             0
        nb_semicolumn        0
        nb_dollar            0
        nb_space             0
        nb_www               0
        nb_com               0
        nb_dslash            0
        http_in_path         0
        https_token          0
        ratio_digits_url     0
        ratio_digits_host    0
        punycode             0
        port                 0
        tld_in_path          0
        tld_in_subdomain     0
        abnormal_subdomain   0
        nb_subdomains        0
        prefix_suffix        0
        random_domain        0
        shortening_service   0
        path_extension       0
        nb_redirection       0
        nb_external_redirection 0

```

```

length_words_raw      0
char_repeat           0
shortest_words_raw    0
shortest_word_host    0
shortest_word_path    0
longest_words_raw     0
longest_word_host     0
longest_word_path     0
avg_words_raw         0
avg_word_host         0
avg_word_path         0
phish_hints           0
domain_in_brand       0
brand_in_subdomain    0
brand_in_path         0
suspicious_tld        0
statistical_report    0
nb_hyperlinks         0
ratio_intHyperlinks   0
ratio_extHyperlinks   0
ratio_nullHyperlinks  0
nb_extCSS             0
ratio_intRedirection  0
ratio_extRedirection  0
ratio_intErrors       0
ratio_extErrors       0
login_form            0
external_favicon      0
links_in_tags         0
submit_email          0
ratio_intMedia        0
ratio_extMedia        0
sfh                   0
iframe                0
popup_window          0
safe_anchor           0
onmouseover           0
right_click           0
empty_title           0
domain_in_title       0
domain_with_copyright 0
whois_registered_domain 0
domain_registration_length 0
domain_age            0
web_traffic           0
dns_record            0
google_index          0
page_rank             0
status                0
dtype: int64

```

Descriptive Analysis

In [6]:

```
df.describe()
```

Out[6]:

length_url **length_hostname**

ip

nb_dots

nb_hyphens

count	11430.000000	11430.000000	11430.000000	11430.000000	11430.000000	11
mean	61.126684	21.090289	0.150569	2.480752	0.997550	
std	55.297318	10.777171	0.357644	1.369686	2.087087	
min	12.000000	4.000000	0.000000	1.000000	0.000000	
25%	33.000000	15.000000	0.000000	2.000000	0.000000	
50%	47.000000	19.000000	0.000000	2.000000	0.000000	
75%	71.000000	24.000000	0.000000	3.000000	1.000000	
max	1641.000000	214.000000	1.000000	24.000000	43.000000	

Separating numerical and categorical columns. Then, for each numeric feature, you analyze spread, skewness, and outliers — very helpful for choosing scaling techniques or detecting which features might need transformation.

In [7]:

```
numerical_columns = df.select_dtypes(exclude= 'object')
numerical_columns
```

Out[7]:

	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_ar
0	37	19	0	3	0	0	0	
1	77	23	1	1	0	0	0	
2	126	50	1	4	1	0	1	
3	18	11	0	2	0	0	0	
4	55	15	0	2	2	0	0	
...	
11425	45	17	0	2	0	0	0	
11426	84	18	0	5	0	1	1	
11427	105	16	1	2	6	0	1	
11428	38	30	0	2	0	0	0	
11429	477	14	1	24	0	1	1	

11430 rows × 87 columns

In [8]:

```
# Descriptive statistics
from collections import OrderedDict

stats = []
```

```

for col in df.columns:
    if df[col].dtype != 'object':
        numerical_stats = OrderedDict({
            'Feature': col,
            'Minimum': df[col].min(),
            'Maximum': df[col].max(),
            'Mean': df[col].mean(),
            'Mode': df[col].mode()[0] if not df[col].mode().empty else None,
            '25%': df[col].quantile(0.25),
            '75%': df[col].quantile(0.75),
            'IQR': df[col].quantile(0.75) - df[col].quantile(0.25),
            'Standard Deviation': df[col].std(),
            'Skewness': df[col].skew(),
            'Kurtosis': df[col].kurt()
        })
        stats.append(numerical_stats)

# Convert to DataFrame
report = pd.DataFrame(stats)

report

```

Out[8]:

	Feature	Minimum	Maximum	Mean	Mode	
0	length_url	12.0	1.641000e+03	61.126684	26.0	33.0C
1	length_hostname	4.0	2.140000e+02	21.090289	16.0	15.0C
2	ip	0.0	1.000000e+00	0.150569	0.0	0.0C
3	nb_dots	1.0	2.400000e+01	2.480752	2.0	2.0C
4	nb_hyphens	0.0	4.300000e+01	0.997550	0.0	0.0C
5	nb_at	0.0	4.000000e+00	0.022222	0.0	0.0C
6	nb_qm	0.0	3.000000e+00	0.141207	0.0	0.0C
7	nb_and	0.0	1.900000e+01	0.162292	0.0	0.0C
8	nb_or	0.0	0.000000e+00	0.000000	0.0	0.0C
9	nb_eq	0.0	1.900000e+01	0.293176	0.0	0.0C
10	nb_underscore	0.0	1.800000e+01	0.322660	0.0	0.0C
11	nb_tilde	0.0	1.000000e+00	0.006649	0.0	0.0C
12	nb_percent	0.0	9.600000e+01	0.123097	0.0	0.0C
13	nb_slash	2.0	3.300000e+01	4.289589	3.0	3.0C
14	nb_star	0.0	1.000000e+00	0.000700	0.0	0.0C
15	nb_colon	1.0	7.000000e+00	1.027909	1.0	1.0C
16	nb_comma	0.0	4.000000e+00	0.004024	0.0	0.0C
17	nb_semicolumn	0.0	2.000000e+01	0.062292	0.0	0.0C

18	nb_dollar	0.0	6.000000e+00	0.001925	0.0	0.00
19	nb_space	0.0	1.800000e+01	0.034821	0.0	0.00
20	nb_www	0.0	2.000000e+00	0.448469	0.0	0.00
21	nb_com	0.0	6.000000e+00	0.127997	0.0	0.00
22	nb_dslash	0.0	1.000000e+00	0.006562	0.0	0.00
23	http_in_path	0.0	4.000000e+00	0.016710	0.0	0.00
24	https_token	0.0	1.000000e+00	0.610936	1.0	0.00
25	ratio_digits_url	0.0	7.238806e-01	0.053137	0.0	0.00
26	ratio_digits_host	0.0	8.000000e-01	0.025024	0.0	0.00
27	punycode	0.0	1.000000e+00	0.000350	0.0	0.00
28	port	0.0	1.000000e+00	0.002362	0.0	0.00
29	tld_in_path	0.0	1.000000e+00	0.065617	0.0	0.00
30	tld_in_subdomain	0.0	1.000000e+00	0.050131	0.0	0.00
31	abnormal_subdomain	0.0	1.000000e+00	0.021610	0.0	0.00
32	nb_subdomains	1.0	3.000000e+00	2.231671	2.0	2.00
33	prefix_suffix	0.0	1.000000e+00	0.202450	0.0	0.00
34	random_domain	0.0	1.000000e+00	0.083290	0.0	0.00
35	shortening_service	0.0	1.000000e+00	0.123447	0.0	0.00
36	path_extension	0.0	1.000000e+00	0.000175	0.0	0.00
37	nb_redirection	0.0	6.000000e+00	0.498250	0.0	0.00
38	nb_external_redirection	0.0	1.000000e+00	0.003150	0.0	0.00
39	length_words_raw	1.0	1.060000e+02	6.232808	2.0	2.00
40	char_repeat	0.0	1.460000e+02	2.927472	3.0	1.00
41	shortest_words_raw	1.0	3.100000e+01	3.127297	3.0	2.00
42	shortest_word_host	1.0	3.900000e+01	5.019773	3.0	3.00
43	shortest_word_path	0.0	4.000000e+01	2.398950	0.0	0.00
44	longest_words_raw	2.0	8.290000e+02	15.393876	9.0	9.00
45	longest_word_host	1.0	6.200000e+01	10.467979	9.0	7.00
46	longest_word_path	0.0	8.290000e+02	10.561505	0.0	0.00
47	avg_words_raw	2.0	1.282500e+02	7.258882	6.0	5.25
48	avg_word_host	1.0	3.900000e+01	7.678075	5.0	5.25
49	avg_word_path	0.0	2.500000e+02	5.092425	0.0	0.00
50	phish_hints	0.0	1.000000e+01	0.327734	0.0	0.00
51	domain_in_brand	0.0	1.000000e+00	0.104199	0.0	0.00
52	brand in subdomain	0.0	1.000000e+00	0.004112	0.0	0.00

53	brand_in_path	0.0	1.000000e+00	0.004899	0.0	0.00
54	suspecious_tld	0.0	1.000000e+00	0.017935	0.0	0.00
55	statistical_report	0.0	2.000000e+00	0.059755	0.0	0.00
56	nb_hyperlinks	0.0	4.659000e+03	87.189764	0.0	9.00
57	ratio_intHyperlinks	0.0	1.000000e+00	0.602457	0.0	0.22
58	ratio_extHyperlinks	0.0	1.000000e+00	0.276720	0.0	0.00
59	ratio_nullHyperlinks	0.0	0.000000e+00	0.000000	0.0	0.00
60	nb_extCSS	0.0	1.240000e+02	0.784864	0.0	0.00
61	ratio_intRedirection	0.0	0.000000e+00	0.000000	0.0	0.00
62	ratio_extRedirection	0.0	2.000000e+00	0.158926	0.0	0.00
63	ratio_intErrors	0.0	0.000000e+00	0.000000	0.0	0.00
64	ratio_extErrors	0.0	1.000000e+00	0.062469	0.0	0.00
65	login_form	0.0	1.000000e+00	0.063605	0.0	0.00
66	external_favicon	0.0	1.000000e+00	0.442170	0.0	0.00
67	links_in_tags	0.0	1.000000e+02	51.978211	0.0	0.00
68	submit_email	0.0	0.000000e+00	0.000000	0.0	0.00
69	ratio_intMedia	0.0	1.000000e+02	42.870444	0.0	0.00
70	ratio_extMedia	0.0	1.000000e+02	23.236293	0.0	0.00
71	sfh	0.0	0.000000e+00	0.000000	0.0	0.00
72	iframe	0.0	1.000000e+00	0.001312	0.0	0.00
73	popup_window	0.0	1.000000e+00	0.006037	0.0	0.00
74	safe_anchor	0.0	1.000000e+02	37.063922	0.0	0.00
75	onmouseover	0.0	1.000000e+00	0.001137	0.0	0.00
76	right_clic	0.0	1.000000e+00	0.001400	0.0	0.00
77	empty_title	0.0	1.000000e+00	0.124759	0.0	0.00
78	domain_in_title	0.0	1.000000e+00	0.775853	1.0	1.00
79	domain_with_copyright	0.0	1.000000e+00	0.439545	0.0	0.00
80	whois_registered_domain	0.0	1.000000e+00	0.072878	0.0	0.00
81	domain_registration_length	-1.0	2.982900e+04	492.532196	0.0	84.00
82	domain_age	-12.0	1.287400e+04	4062.543745	-1.0	972.25
83	web_traffic	0.0	1.076799e+07	856756.643307	0.0	0.00
84	dns_record	0.0	1.000000e+00	0.020122	0.0	0.00
85	google_index	0.0	1.000000e+00	0.533946	1.0	0.00
86	page_rank	0.0	1.000000e+01	3.185739	0.0	1.00

Frequency distribution for categorical features

Several features showed significant skewness, suggesting non-normal distributions.

Wide ranges and high standard deviations in some columns (e.g., web_traffic, length_url) indicate the presence of outliers.

Features with high kurtosis are likely to have heavy tails or sharp peaks.

Checking frequency counts for categorical columns — this helps you see whether categories are balanced or dominated by one class (like the target label status).

In [9]:

```
# Frequency distribution for categorical features (if any)
for col in df.columns:
    if df[col].dtype == 'object':
        print(f"\nFrequency distribution for {col}:\n")
        print(df[col].value_counts())
```

Frequency distribution for url:

```
url
http://e710z0ear.du.r.appspot.com/c:/users/user/downlo
2
https://lt.mydplr.com/16672ac75448ecdb528e1c663c0df3a7-f10ed321df1a4fbc893c86f
bb12f0913
1
http://appleid.apple.com-app.es/
1
http://174.139.46.123/ap/signin?openid.pape.max_auth_age=0&openid.return_t
o=https%3A%2F%2Fwww.amazon.co.jp%2F%3Fref_%3Dnav_em_hd_re_signin&openid.id
entity=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&open
id.assoc_handle=jpflex&openid.mode=checkid_setup&key=a@b.c&openid.
claimed_id=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&
openid.ns=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0&&&ref_=nav_em_hd_c
lc_signin 1
http://www.crestonwood.com/router.php
1
..
https://www.dissernet.org/
1
https://workprotocoles-com.webs.com/
1
http://www.vg247.com/2017/04/24/best-nintendo-switch-games/
1
https://www.facebook.com/Publictransporthub/
1
http://www.game.co.uk/en/games/nintendo-switch/nintendo-switch/
1
Name: count, Length: 11429, dtype: int64
```

Frequency distribution for status:

```
status
legitimate    5715
phishing      5715
Name: count, dtype: int64
```

The target label is balanced — There is no need to use SMOTE techniques to Balance the Target column.

In [10]:

```
df['status'].mode()
```

```
Out[10]: 0    legitimate
         1     phishing
         Name: status, dtype: object
```

In [11]:

```
df['url'].mode()
```

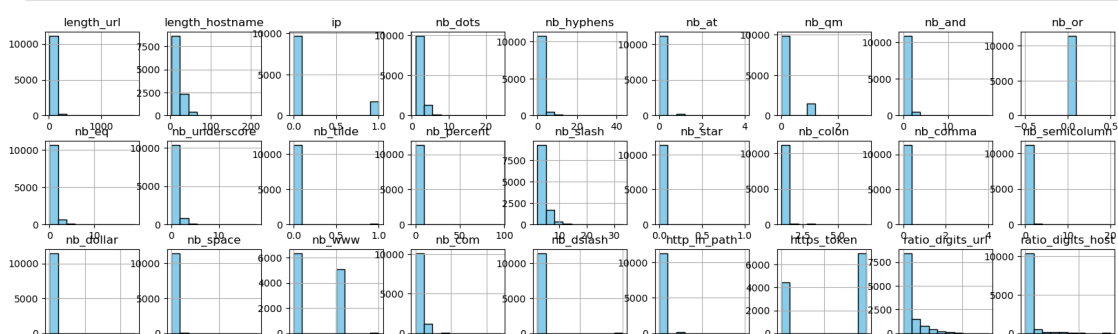
```
Out[11]: 0    http://e710z0ear.du.r.appspot.com/c:/users/use...
         Name: url, dtype: object
```

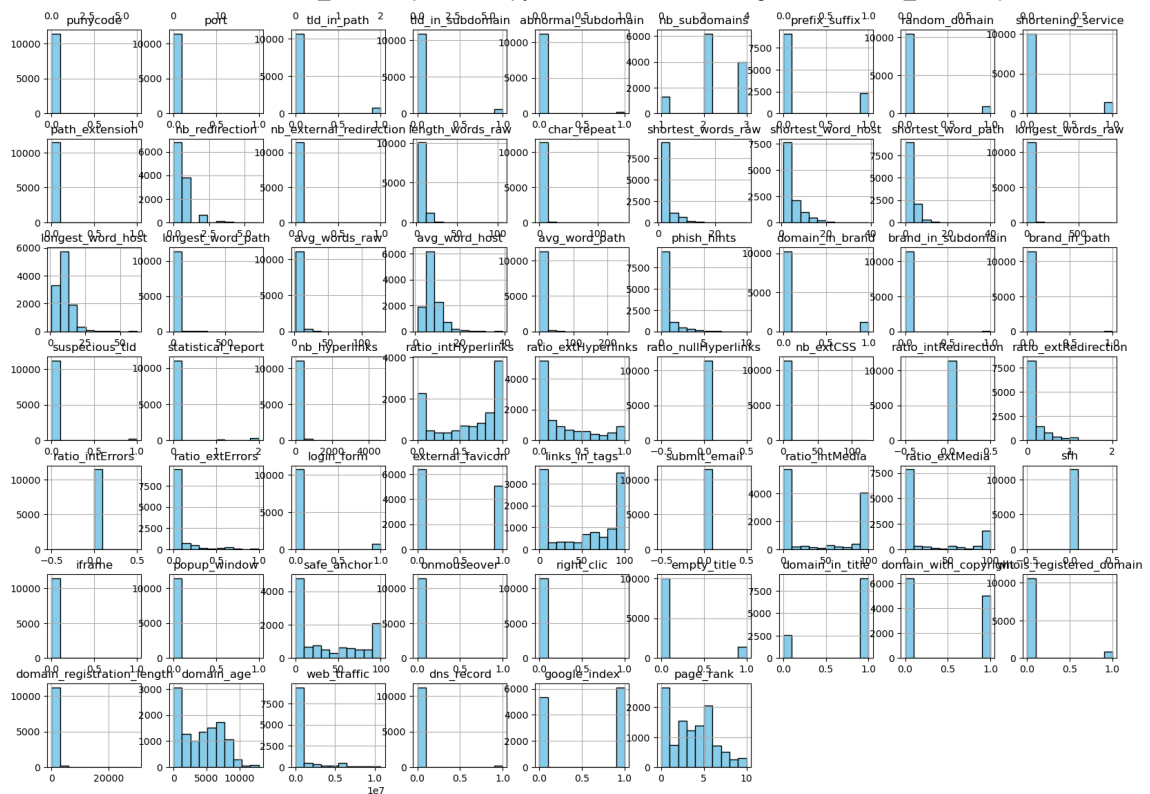
Histogram

Histograms Reveal skewed features and possible outliers. Some features like web_traffic or length_url may need scaling or normalization.

In [12]:

```
# Histograms for numerical features
numerical_columns.hist(figsize=(20, 20), bins= 10, color= 'skyblue', edgecolor= 'black')
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



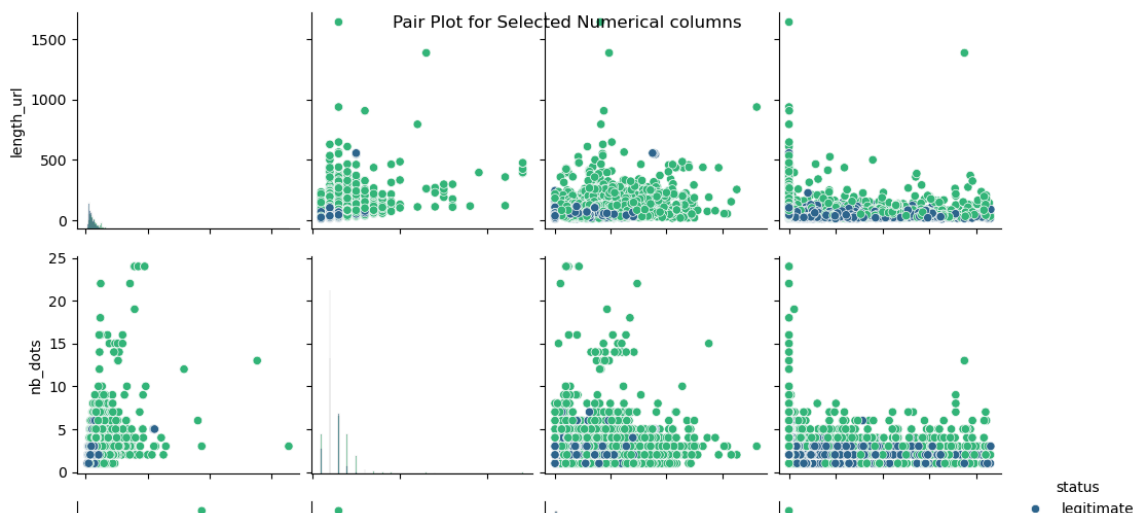


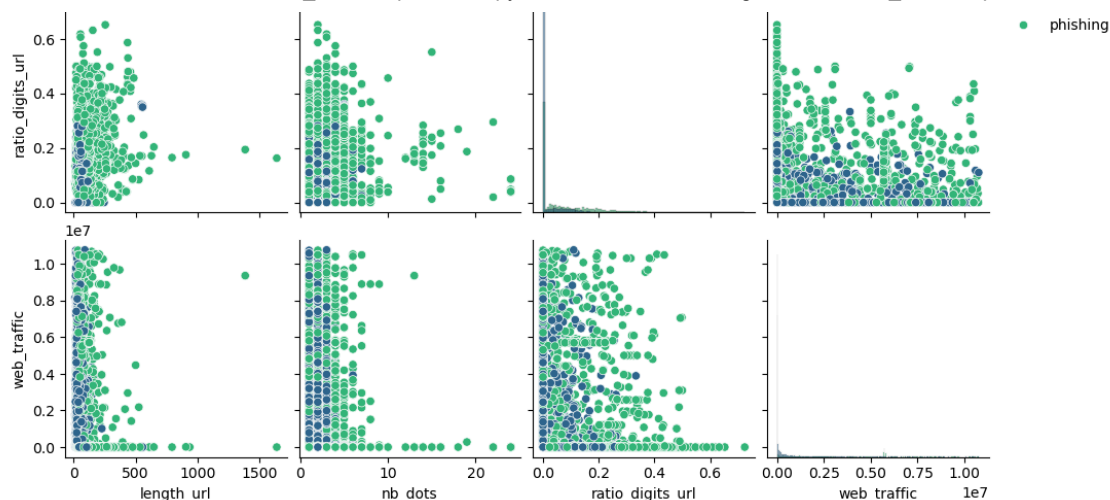
Pair Plot

- We have use only selected important features to create the Pair Plot
- The pairplot shows some visual separation between phishing and legitimate classes in selected features — especially in ratio_digits_url and web_traffic. That means these features might be strong indicators for classification.

In [13]:

```
selected_features = ['length_url', 'nb_dots', 'ratio_digits_url', 'web_traffic']
# plot pair plot
sns.pairplot(df[selected_features], hue='status', diag_kind='hist', palette=
plt.suptitle('Pair Plot for Selected Numerical columns')
plt.show()
```





Using Replace function to 'legitimate' and 'phishing' into 0 and 1 — readying the target for machine learning models.

In [14]:

```
df['status'] = df['status'].replace({'legitimate' : 0, 'phishing' : 1})
```

Label encoding to url column — to convert the categorical data into numerical

In [15]:

```
# Using Label Encoding in Url column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df['url1'] = le.fit_transform(df['url'])
df['url1'].value_counts()
```

```
Out[15]: url1
1065    2
8258    1
363      1
62       1
4501     1
..
9799     1
9324     1
6684     1
9920     1
4919     1
Name: count, Length: 11429, dtype: int64
```

Insights and Recommendations

- Features like `web_traffic`, `SSLfinal_State`, and `page_rank` are crucial indicators.
- The Dataset has huge amount of Outliers

The Dataset has huge amount of Outliers.

- Outliers can be capped using the IQR method.
- Use `RobustScaler` to normalize numerical features.
- Remove redundant features with high multicollinearity.
- The target is balance hence, there is no need for SMOTE.
- We can use Feature Engineering.
- The Dataset have doesn't have any null values.

Checking Duplicates

Label Encoding was applied to the url column to convert categorical values into numeric form. One-Hot Encoding was avoided because it would have significantly increased the number of columns due to the high number of unique URLs. Label Encoding keeps the dataset compact and efficient without adding unnecessary dimensions.

In [16]:

```
# Checking Duplicates
duplicates = df.duplicated()
duplicates.value_counts()
```

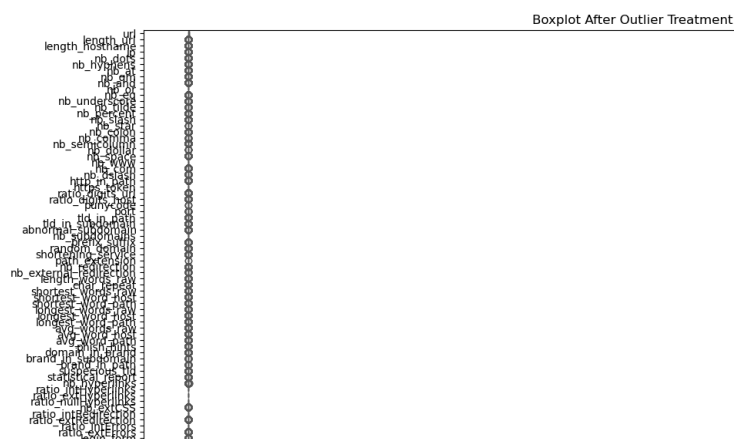
Out[16]: False 11430
Name: count, dtype: int64

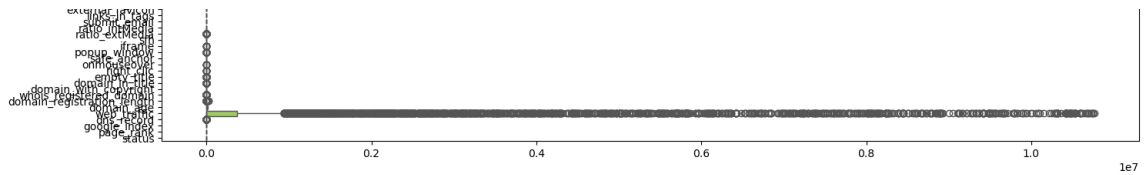
In [17]:

```
# Set figure size
plt.figure(figsize=(15, 8))

# Create boxplot for all numerical columns
sns.boxplot(data=df, orient='h', palette='Set2')

# Set title
plt.title('Boxplot After Outlier Treatment')
plt.tight_layout()
plt.show()
```





In [18]:

```
# Replace Outliers with Median Statergy

for col in df.select_dtypes(include='number').columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = (df[col] < lower_bound) | (df[col] > upper_bound)
    outlier_count = outliers.sum()

    if outlier_count > 0:
        replacement = df[col].median()
        df.loc[outliers, col] = replacement
        print(f"Replaced {outlier_count} outliers in '{col}' with median.")
    else:
        print(f"No outliers found in '{col}'.")
```

```
No outliers found in 'url'.
Replaced 620 outliers in 'length_url' with median.
Replaced 775 outliers in 'length_hostname' with median.
Replaced 1721 outliers in 'ip' with median.
Replaced 567 outliers in 'nb_dots' with median.
Replaced 1371 outliers in 'nb_hyphens' with median.
Replaced 245 outliers in 'nb_at' with median.
Replaced 1555 outliers in 'nb_qm' with median.
Replaced 761 outliers in 'nb_and' with median.
No outliers found in 'nb_or'.
Replaced 1564 outliers in 'nb_eq' with median.
Replaced 1695 outliers in 'nb_underscore' with median.
Replaced 76 outliers in 'nb_tilde' with median.
Replaced 355 outliers in 'nb_percent' with median.
Replaced 401 outliers in 'nb_slash' with median.
Replaced 8 outliers in 'nb_star' with median.
Replaced 197 outliers in 'nb_colon' with median.
Replaced 24 outliers in 'nb_comma' with median.
Replaced 248 outliers in 'nb_semicolumn' with median.
Replaced 11 outliers in 'nb_dollar' with median.
Replaced 210 outliers in 'nb_space' with median.
No outliers found in 'nb_www'.
Replaced 1327 outliers in 'nb_com' with median.
Replaced 75 outliers in 'nb_dslash' with median.
Replaced 150 outliers in 'http_in_path' with median.
No outliers found in 'https_token'.
Replaced 933 outliers in 'ratio_digits_url' with median.
Replaced 1503 outliers in 'ratio_digits_host' with median.
Replaced 4 outliers in 'punycode' with median.
Replaced 27 outliers in 'port' with median.
Replaced 750 outliers in 'tld_in_path' with median.
Replaced 573 outliers in 'tld_in_subdomain' with median.
```

Replaced 373 outliers in 'tld_in_subdomain' with median.
Replaced 247 outliers in 'abnormal_subdomain' with median.
No outliers found in 'nb_subdomains'.
Replaced 2314 outliers in 'prefix_suffix' with median.
Replaced 952 outliers in 'random_domain' with median.
Replaced 1411 outliers in 'shortening_service' with median.
Replaced 2 outliers in 'path_extension' with median.
Replaced 166 outliers in 'nb_redirection' with median.
Replaced 36 outliers in 'nb_external_redirection' with median.
Replaced 264 outliers in 'length_words_raw' with median.
Replaced 310 outliers in 'char_repeat' with median.
Replaced 1435 outliers in 'shortest_words_raw' with median.
Replaced 1093 outliers in 'shortest_word_host' with median.
Replaced 428 outliers in 'shortest_word_path' with median.
Replaced 1035 outliers in 'longest_words_raw' with median.
Replaced 220 outliers in 'longest_word_host' with median.
Replaced 929 outliers in 'longest_word_path' with median.
Replaced 725 outliers in 'avg_words_raw' with median.
Replaced 568 outliers in 'avg_word_host' with median.
Replaced 282 outliers in 'avg_word_path' with median.
Replaced 2041 outliers in 'phish_hints' with median.
Replaced 1191 outliers in 'domain_in_brand' with median.
Replaced 47 outliers in 'brand_in_subdomain' with median.
Replaced 56 outliers in 'brand_in_path' with median.
Replaced 205 outliers in 'suspicious_tld' with median.
Replaced 377 outliers in 'statistical_report' with median.
Replaced 953 outliers in 'nb_hyperlinks' with median.
No outliers found in 'ratio_intHyperlinks'.
No outliers found in 'ratio_extHyperlinks'.
No outliers found in 'ratio_nullHyperlinks'.
Replaced 1019 outliers in 'nb_extCSS' with median.
No outliers found in 'ratio_intRedirection'.
Replaced 999 outliers in 'ratio_extRedirection' with median.
No outliers found in 'ratio_intErrors'.
Replaced 2149 outliers in 'ratio_extErrors' with median.
Replaced 727 outliers in 'login_form' with median.
No outliers found in 'external_favicon'.
No outliers found in 'links_in_tags'.
No outliers found in 'submit_email'.
No outliers found in 'ratio_intMedia'.
Replaced 2012 outliers in 'ratio_extMedia' with median.
No outliers found in 'sfh'.
Replaced 15 outliers in 'iframe' with median.
Replaced 69 outliers in 'popup_window' with median.
No outliers found in 'safe_anchor'.
Replaced 13 outliers in 'onmouseover' with median.
Replaced 16 outliers in 'right_click' with median.
Replaced 1426 outliers in 'empty_title' with median.
Replaced 2562 outliers in 'domain_in_title' with median.
No outliers found in 'domain_with_copyright'.
Replaced 833 outliers in 'whois_registered_domain' with median.
Replaced 1529 outliers in 'domain_registration_length' with median.
No outliers found in 'domain_age'.
Replaced 2138 outliers in 'web_traffic' with median.
Replaced 230 outliers in 'dns_record' with median.
No outliers found in 'google_index'.
No outliers found in 'page_rank'.
No outliers found in 'status'.

A ranked list of features based on Variance Variance Inflation Factor (VIF)

In [19]:

```
# Checking VIF:
def calculate_vif(dataset):
    vif = pd.DataFrame()
    vif['features'] = dataset.columns
    vif['VIF_Values'] = [variance_inflation_factor(dataset.values,i) for i in range(dataset.shape[0])]
    vif['VIF_Values'] = round(vif['VIF_Values'], 2)
    vif = vif.sort_values(by = 'VIF_Values', ascending=False)
    return (vif)

calculate_vif(df.drop('status',axis = 1))
```

Out[19]:

	features	VIF_Values
0	url	9.42
25	https_token	7.70
47	longest_word_path	5.86
33	nb_subdomains	5.36
46	longest_word_host	5.18
4	nb_dots	4.88
58	ratio_intHyperlinks	4.49
45	longest_words_raw	4.38
21	nb_www	4.21
50	avg_word_path	4.16
40	length_words_raw	4.08
1	length_url	3.81
49	avg_word_host	3.45
68	links_in_tags	3.04
48	avg_words_raw	2.49
2	length_hostname	2.36
44	shortest_word_path	2.35
67	external_favicon	2.33
59	ratio_extHyperlinks	2.32
41	char_repeat	2.25
87	page_rank	2.20
86	google_index	1.88
42	shortest_words_raw	1.88
14	nb_slash	1.80

43	shortest_word_host	1.76
83	domain_age	1.70
70	ratio_intMedia	1.65
57	nb_hyperlinks	1.54
61	nb_extCSS	1.49
75	safe_anchor	1.47
26	ratio_digits_url	1.28
38	nb_redirection	1.24
63	ratio_extRedirection	1.20
80	domain_with_copyright	1.18
82	domain_registration_length	1.18
5	nb_hyphens	1.15
71	ratio_extMedia	1.11
65	ratio_extErrors	1.08
84	web_traffic	1.07
16	nb_colon	0.00
79	domain_in_title	0.00
3	ip	NaN
6	nb_at	NaN
7	nb_qm	NaN
8	nb_and	NaN
9	nb_or	NaN
10	nb_eq	NaN
11	nb_underscore	NaN
12	nb_tilde	NaN
13	nb_percent	NaN
15	nb_star	NaN
17	nb_comma	NaN
18	nb_semicolumn	NaN
19	nb_dollar	NaN
20	nb_space	NaN
22	nb_com	NaN
23	nb_dslash	NaN
24	http_in_path	NaN
27	ratio_digits_host	NaN

27	ratio_digits_host	NaN
28	punycode	NaN
29	port	NaN
30	tld_in_path	NaN
31	tld_in_subdomain	NaN
32	abnormal_subdomain	NaN
34	prefix_suffix	NaN
35	random_domain	NaN
36	shortening_service	NaN
37	path_extension	NaN
39	nb_external_redirection	NaN
51	phish_hints	NaN
52	domain_in_brand	NaN
53	brand_in_subdomain	NaN
54	brand_in_path	NaN
55	suspicious_tld	NaN
56	statistical_report	NaN
60	ratio_nullHyperlinks	NaN
62	ratio_intRedirection	NaN
64	ratio_intErrors	NaN
66	login_form	NaN
69	submit_email	NaN
72	sfh	NaN
73	iframe	NaN
74	popup_window	NaN
76	onmouseover	NaN
77	right_clic	NaN
78	empty_title	NaN
81	whois_registered_domain	NaN
85	dns_record	NaN

In [20]:

```
# Splitting Data into Independent And target Column
X=df.drop(columns='status')
y=df['status']
```

In [21]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.70,random_st
```

In [22]:

```
X_train_original = X_train.copy()
```

Scaling Technique:- Robust Scaler

Robust Scaler was used to handle outliers effectively, as boxplots showed many extreme values in the numerical features. It scales data based on the median and IQR, making it less sensitive to outliers compared to StandardScaler or MinMaxScaler.

In [23]:

```
from sklearn.preprocessing import MinMaxScaler,StandardScaler,RobustScaler
scaler=RobustScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

In [24]:

```
X_train_scaled=X_train.copy()
# If X_train is a NumPy array, convert it to a DataFrame
X_train_df = pd.DataFrame(X_train_original)
X_train_scaled_df = pd.DataFrame(X_train_scaled)

# Plot before and after scaling side by side
plt.figure(figsize=(14, 6))

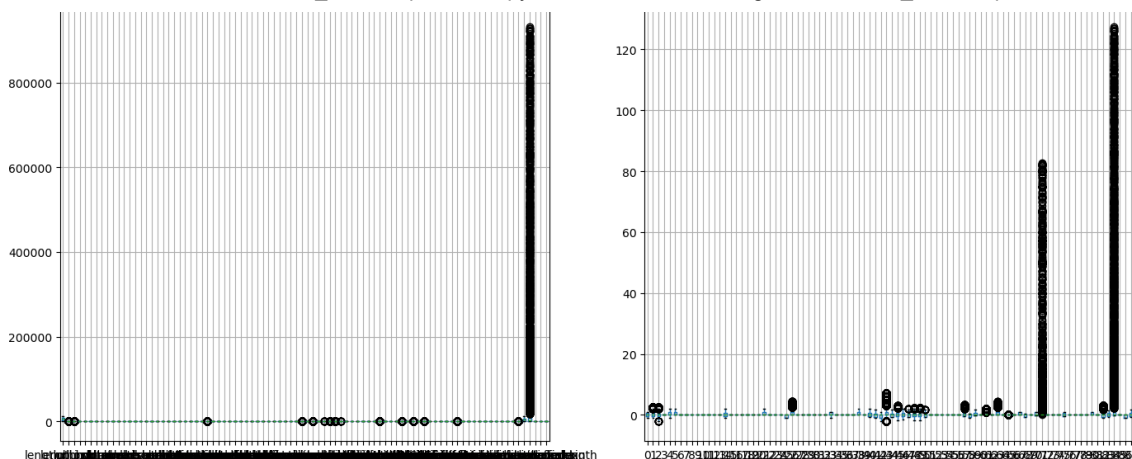
plt.subplot(1, 2, 1)
X_train_df.boxplot()
plt.title("Before Scaling")

plt.subplot(1, 2, 2)
X_train_scaled_df.boxplot()
plt.title("After Robust Scaling")

plt.tight_layout()
plt.show()
```

Before Scaling

After Robust Scaling



In [25]:

```
# Table summarizing feature correlations
df.corr()['status']
```

```
Out[25]: url -0.290971
length_url 0.217898
length_hostname 0.105306
ip NaN
nb_dots 0.109748
nb_hyphens 0.158158
nb_at NaN
nb_qm NaN
nb_and NaN
nb_or NaN
nb_eq NaN
nb_underscore NaN
nb_tilde NaN
nb_percent NaN
nb_slash 0.201618
nb_star NaN
nb_colon NaN
nb_comma NaN
nb_semicolumn NaN
nb_dollar NaN
nb_space NaN
nb_www -0.443468
nb_com NaN
nb_dslash NaN
http_in_path NaN
https_token 0.114669
ratio_digits_url 0.222690
ratio_digits_host NaN
punycode NaN
port NaN
tld_in_path NaN
tld_in_subdomain NaN
abnormal_subdomain NaN
nb_subdomains 0.112891
prefix_suffix NaN
random_domain NaN
shortening_service NaN
path_extension NaN
nb_redirection -0.043685
nb_external_redirection NaN
length_url 0.217898
```

```

length_words_raw      0.135110
char_repeat            -0.122545
shortest_words_raw     -0.167907
shortest_word_host      0.102649
shortest_word_path      0.094549
longest_words_raw      0.164277
longest_word_host      0.094016
longest_word_path      0.187312
avg_words_raw          0.122929
avg_word_host          0.140864
avg_word_path          0.229877
phish_hints            NaN
domain_in_brand        NaN
brand_in_subdomain     NaN
brand_in_path          NaN
suspicious_tld         NaN
statistical_report     NaN
nb_hyperlinks          -0.442032
ratio_intHyperlinks    -0.243982
ratio_extHyperlinks    0.083357
ratio_nullHyperlinks   NaN
nb_extCSS              -0.080464
ratio_intRedirection   NaN
ratio_extRedirection   -0.267394
ratio_intErrors        NaN
ratio_extErrors        -0.179422
login_form             NaN
external_favicon       -0.146565
links_in_tags          -0.184401
submit_email           NaN
ratio_intMedia         -0.193333
ratio_extMedia         -0.098609
sfh                    NaN
iframe                 NaN
popup_window           NaN
safe_anchor            -0.173397
onmouseover            NaN
right_click            NaN
empty_title            NaN
domain_in_title        NaN
domain_with_copyright  -0.173098
whois_registered_domain NaN
domain_registration_length -0.146138
domain_age             -0.331889
web_traffic            -0.203653
dns_record             NaN
google_index           0.731171
page_rank              -0.511137
status                 1.000000
Name: status, dtype: float64

```

Applying PCA for Dimension Reduction

Displaying Variance Ratio

In [26]:

```

# Using PCA Concept:

# Step 1: Standardize the data

```

```

from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()

X_scaled = scaler.fit_transform(df)

# Step 2: Determine number of components to retain 90% variance

for i in range(1, df.shape[1] + 1):
    pca = PCA(n_components=i)
    pca.fit(X_scaled)
    evr = np.cumsum(pca.explained_variance_ratio_)
    if evr[i - 1] >= 0.90:
        pcs = i
        break

print("Explained Variance Ratio:", evr)
print("Number of components selected:", pcs)

# Step 3: Apply PCA

pca = PCA(n_components=pcs)
pca_data = pca.fit_transform(X_scaled)

# Step 4: Create DataFrame

pca_columns = [f'PC{j+1}' for j in range(pcs)]
pca_df = pd.DataFrame(pca_data, columns=pca_columns)

# Step 5: Join Target Column with PCA:

pca_df = pca_df.join(df['status'], how = 'left')

pca_df

```

Explained Variance Ratio: [0.57046592 0.970943]

Number of components selected: 2

Out[26]:

	PC1	PC2	status
0	-7.873171	-5.366775	0
1	-6.311023	14.559213	1
2	-7.736537	-5.395951	1
3	5.910682	-2.885161	0
4	-6.772865	-5.403439	0
...
11425	-1.094859	73.295599	0
11426	-7.901367	-5.380939	1
11427	-7.912587	-5.342337	0
11428	-7.664649	-5.369196	0
11429	-7.893053	-5.346426	1

11430 rows × 3 columns

Training Machine Learning Model

*1 *Logistic Regression**

In []:

```
from sklearn.model_selection import train_test_split
X = pca_df.drop(columns= 'status', axis=1)
y = pca_df['status']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size= 0.2, ran

from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()

LR.fit(X_train, y_train)

y_pred_LR = LR.predict(X_test)

from sklearn.metrics import accuracy_score, confusion_matrix
accuracy_score_LR = accuracy_score(y_pred_LR, y_test)
accuracy_score_LR
print(f'Accuracy : {round(accuracy_score_LR * 100,2)}%')
```

Accuracy : 61.42%

2 Decision Tree Classifier

In [42]:

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(X_train, y_train)

y_pred_DT = DT.predict(X_test)

accuracy_score_DT = accuracy_score(y_pred_DT, y_test)
accuracy_score_DT
print(f'Accuracy : {round(accuracy_score_DT * 100,2)}%')
```

Accuracy : 80.66%

3 Random Forest Classifier

In [47]:

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
```



```
RF = RandomForestClassifier()  
  
RF.fit(X_train,y_train)  
  
y_pred_RF = RF.predict(X_test)  
  
from sklearn.metrics import accuracy_score  
accuracy_score_RF = accuracy_score(y_pred_RF,y_test)  
accuracy_score_RF  
print(f'Accuracy : {round(accuracy_score_RF * 100,2)}%')
```

Accuracy : 85.21%