In [23]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [24]:
```
for each in os.listdir():
    print(each)
```

```
.ipynb_checkpoints
AI-ML-Project-2-Cyber Security
AI-ML-Project-2-Cyber Security.zip
cyber_security-Copy1.ipynb
cyber_security.ipynb
dataset.csv
Helper files
Helper files.zip
```

In [25]:
```
df= pd.read_csv("dataset.csv")
df
```

Out[25]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Dor
0	1	-1	1	1	1	-1	-1	
1	2	1	1	1	1	1	-1	
2	3	1	0	1	1	1	-1	
3	4	1	0	1	1	1	-1	
4	5	1	0	-1	1	1	-1	
...	...	...	...	...	...	...	...	
11050	11051	1	-1	1	-1	1	1	
11051	11052	-1	1	1	-1	-1	-1	
11052	11053	1	-1	1	1	1	-1	
11053	11054	-1	-1	1	1	1	-1	
11054	11055	-1	-1	1	1	1	-1	

11055 rows × 32 columns

In [26]:
```
df.head()
```

Out[26]:

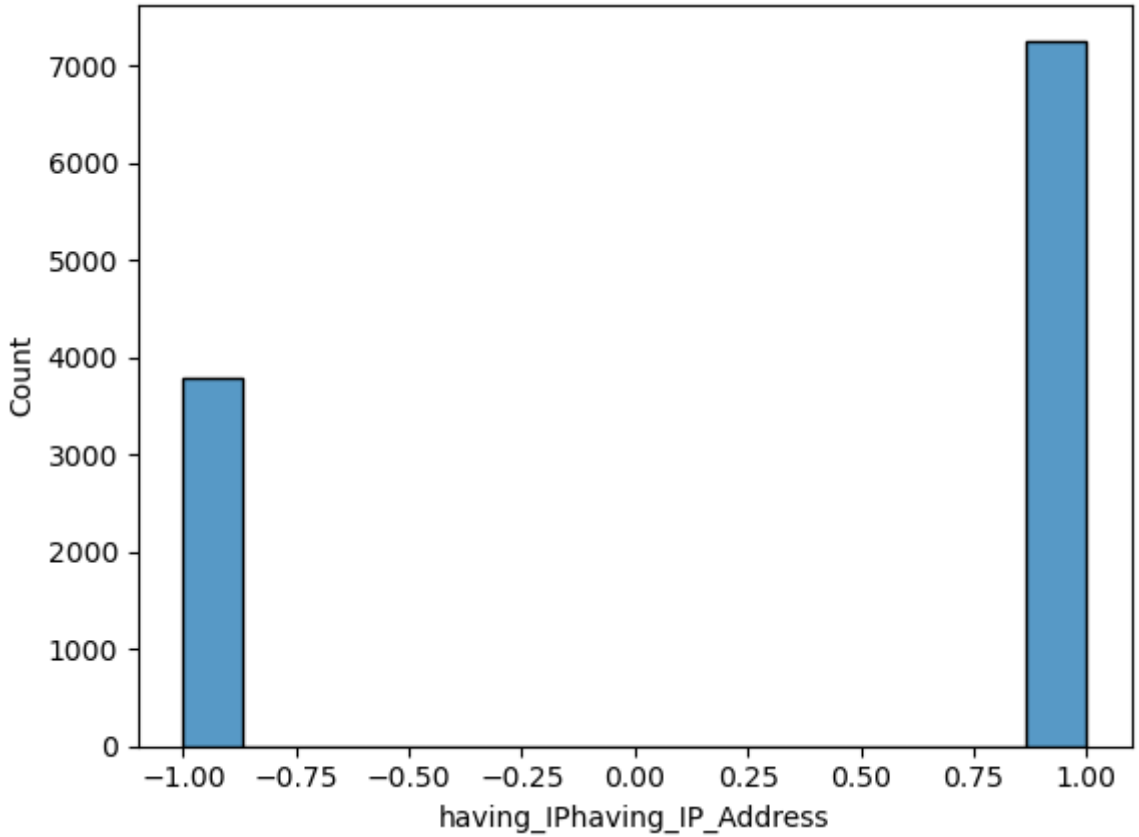
	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	S
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

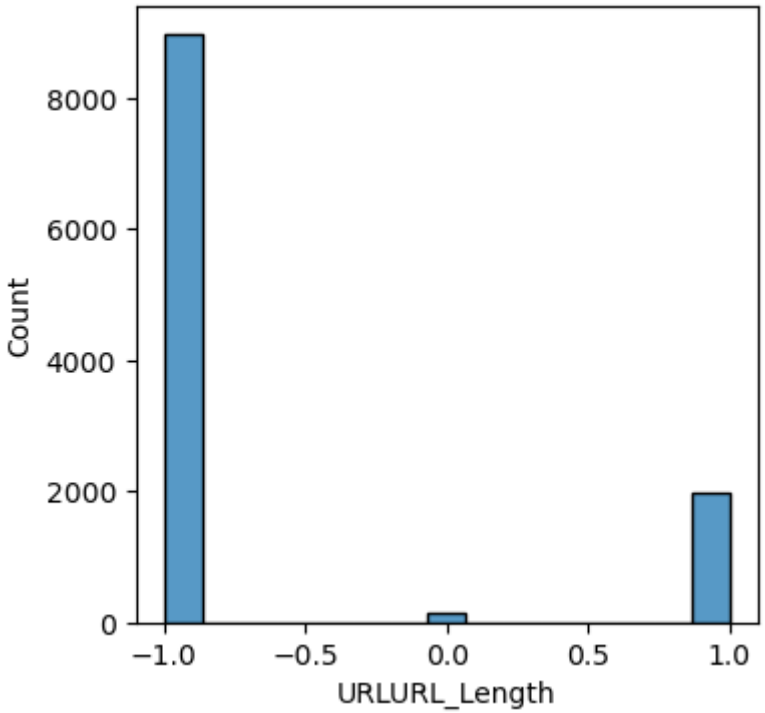
5 rows × 32 columns

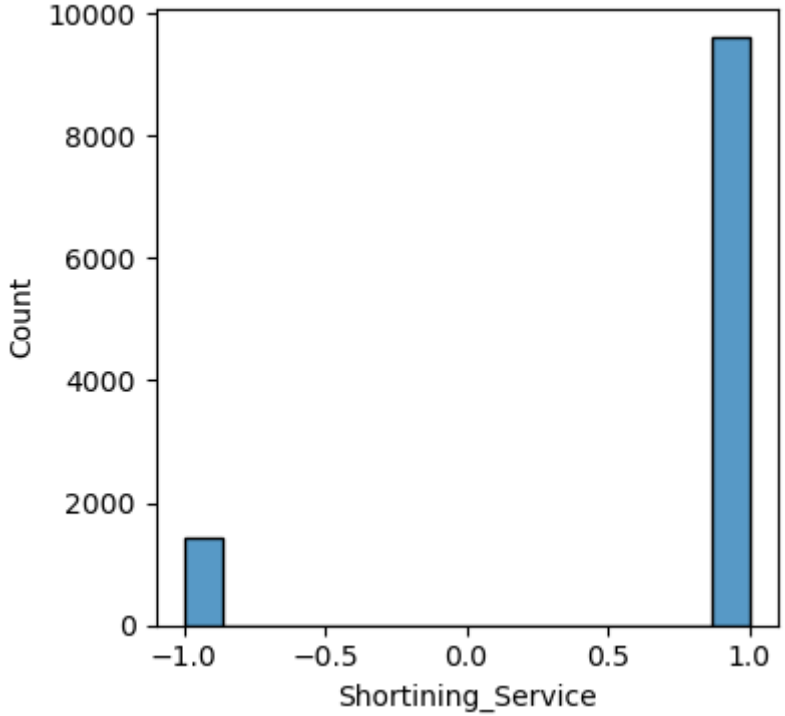
In [27]:
```
# Exploratory Data Analysis:
```

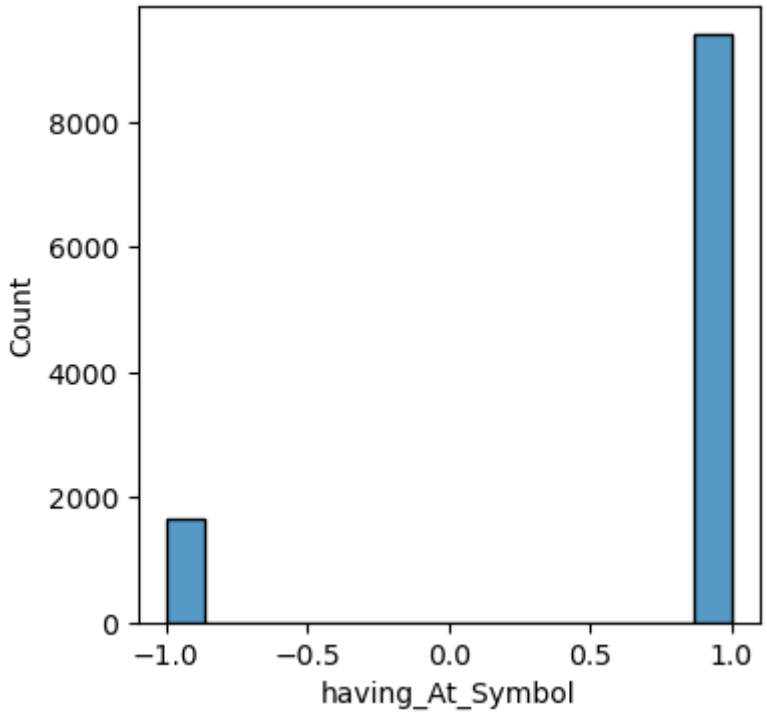
In [28]:
```
# 1.Each sample has 32 features ranging from -1,0,1. Explore the data using histogram, heatmaps.
```

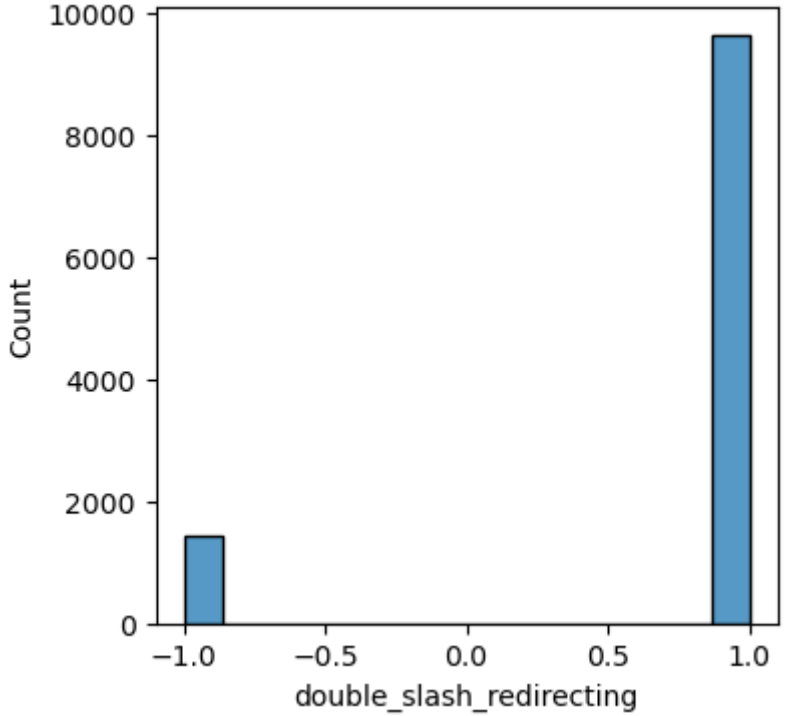
In [29]:
```
df1 = df.drop('index', axis=1)
for i, col in enumerate(df1.columns):
    plt.figure(figsize=(4,4))
    plt.figure(i)
    sns.histplot(df1[col])
    sns.histplot()
```

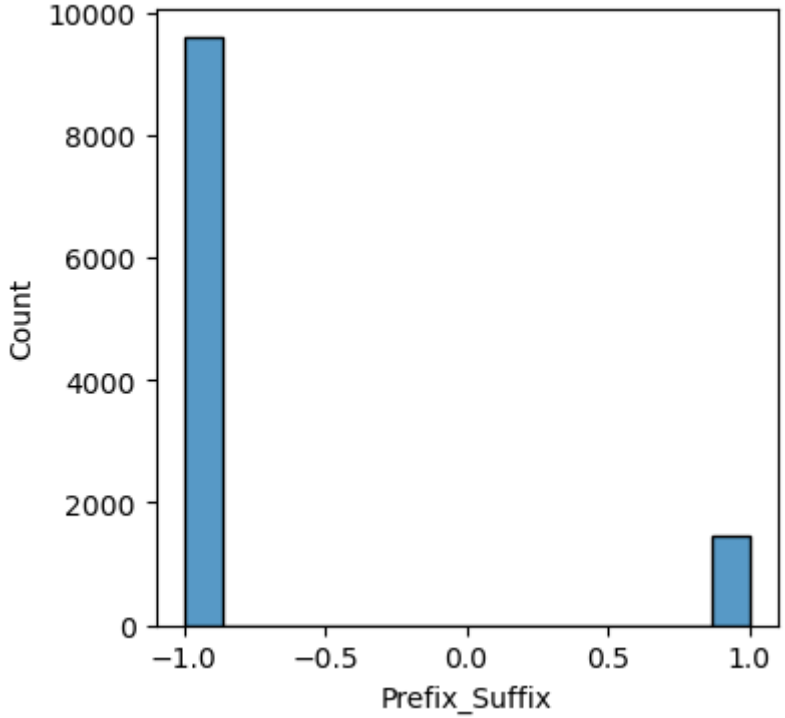


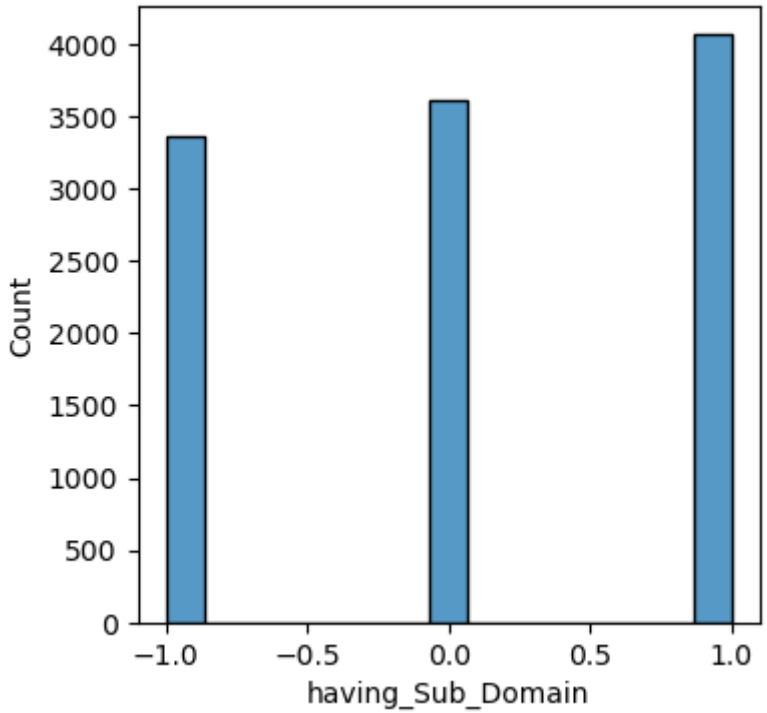

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
URLURL_Length

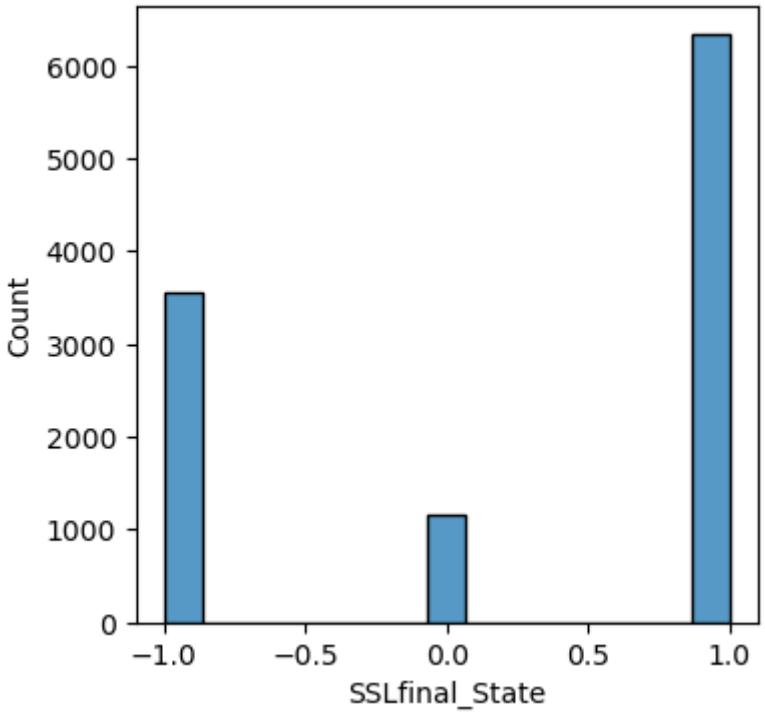

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Shortining_Service

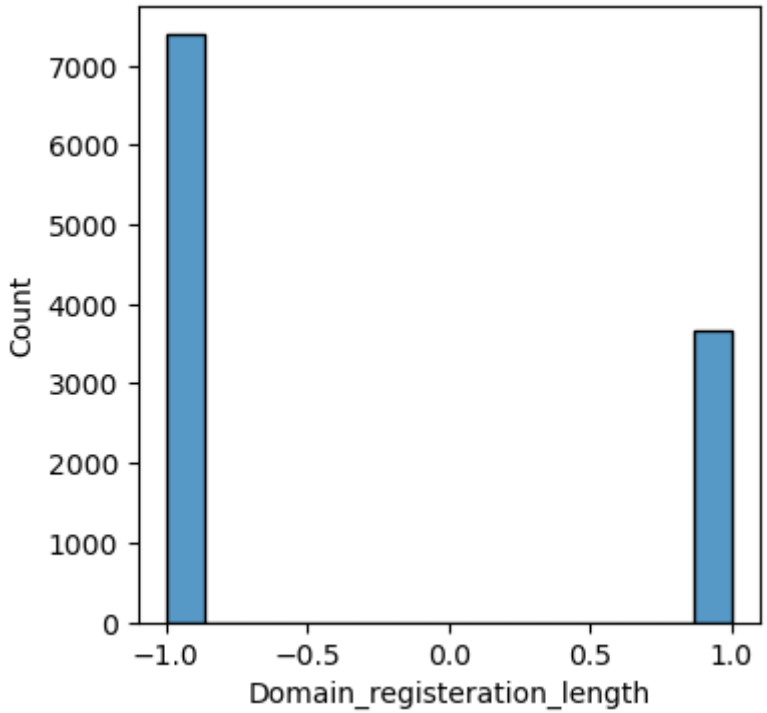

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
having_At_Symbol

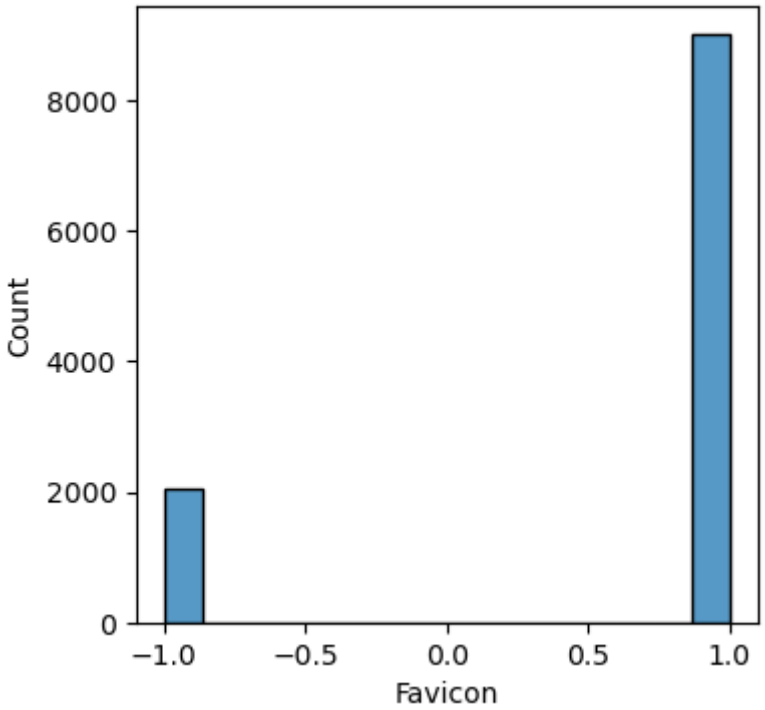

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
double_slash_redirecting

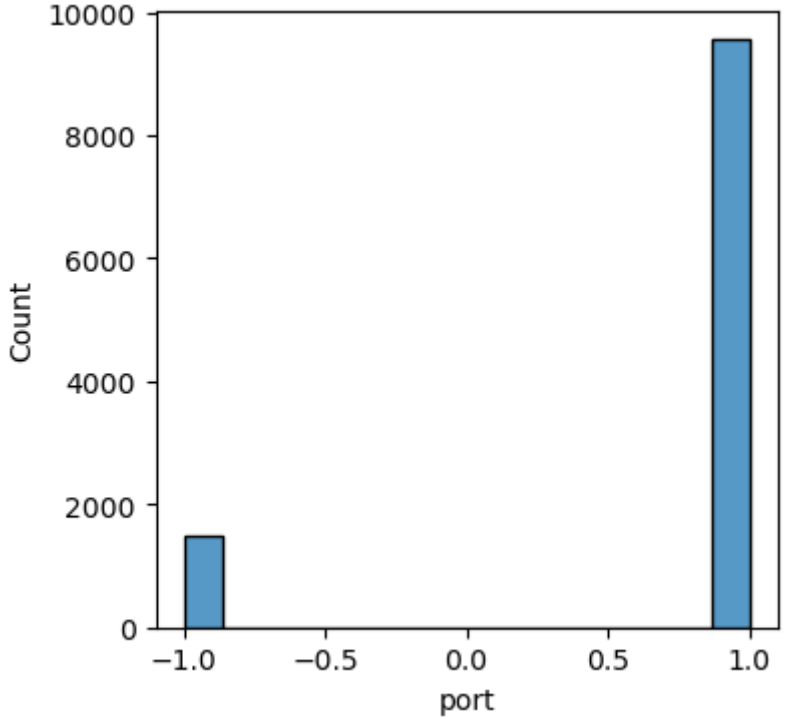

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Prefix_Suffix

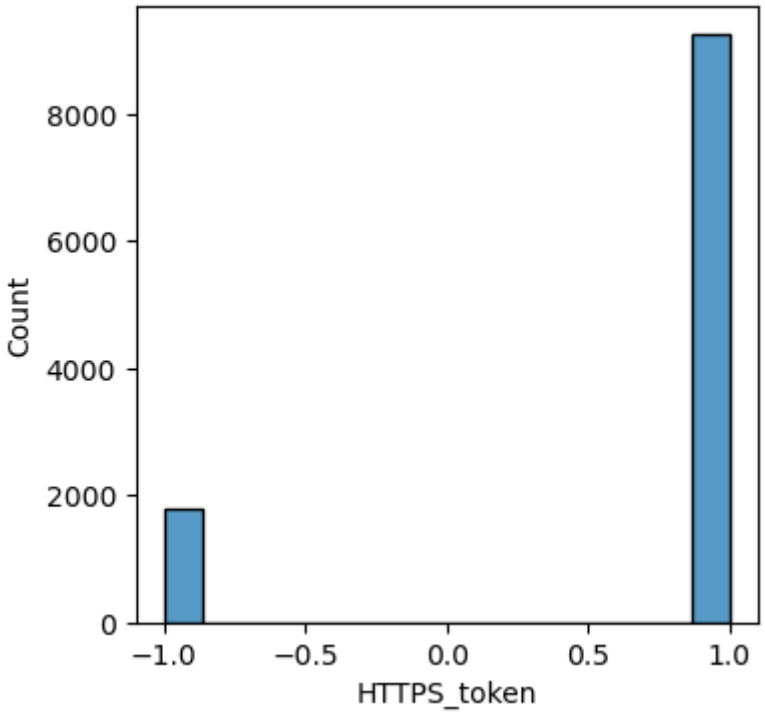

4000
3500
3000
2500
2000
1500
1000
500
0
Count
−1.0
−0.5
0.0
0.5
1.0
having_Sub_Domain

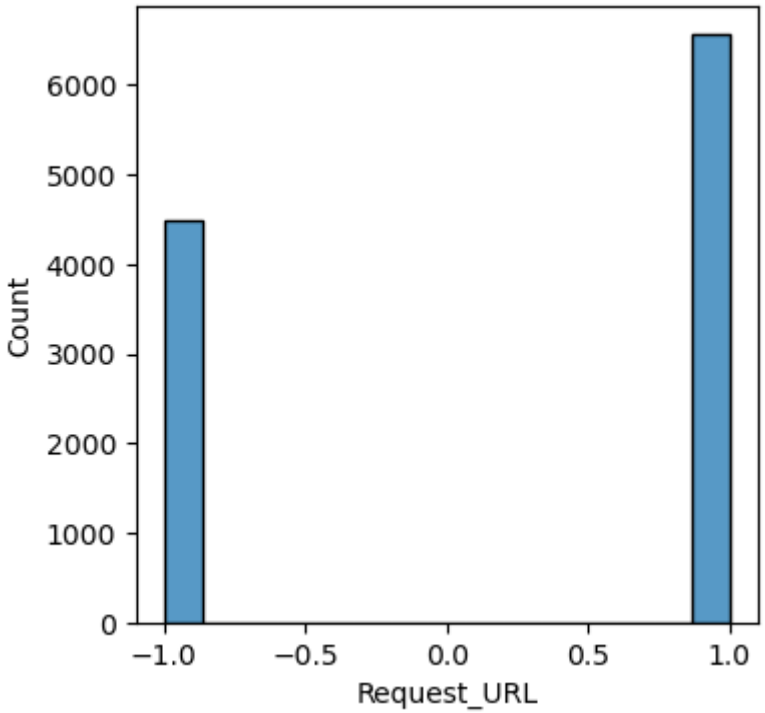

6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
SSLfinal_State

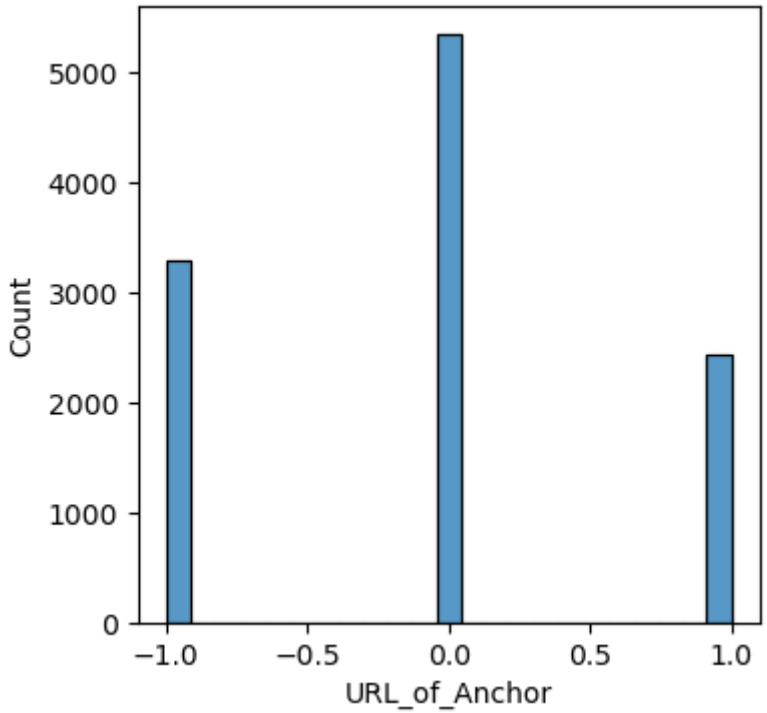

7000
6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Domain_registeration_length

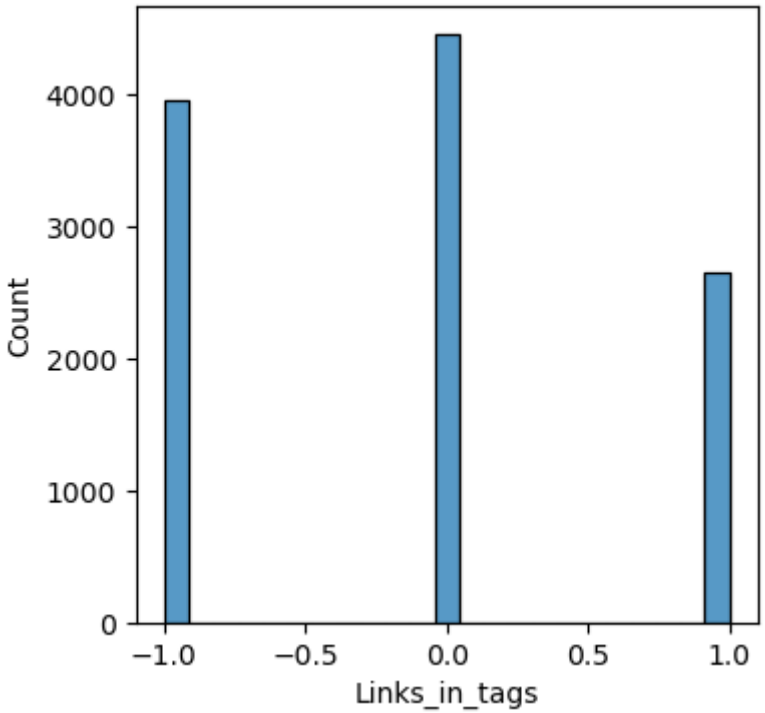

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Favicon

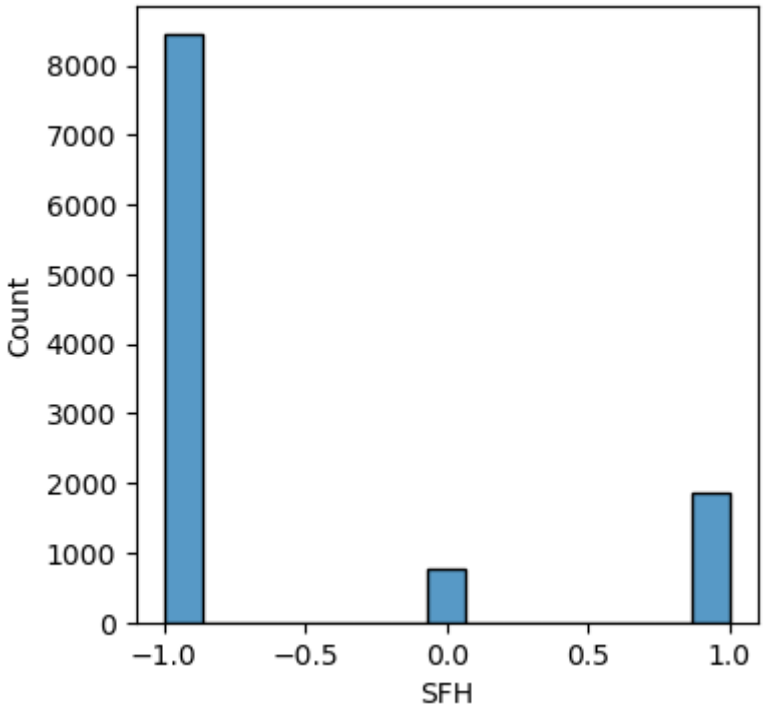

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
port

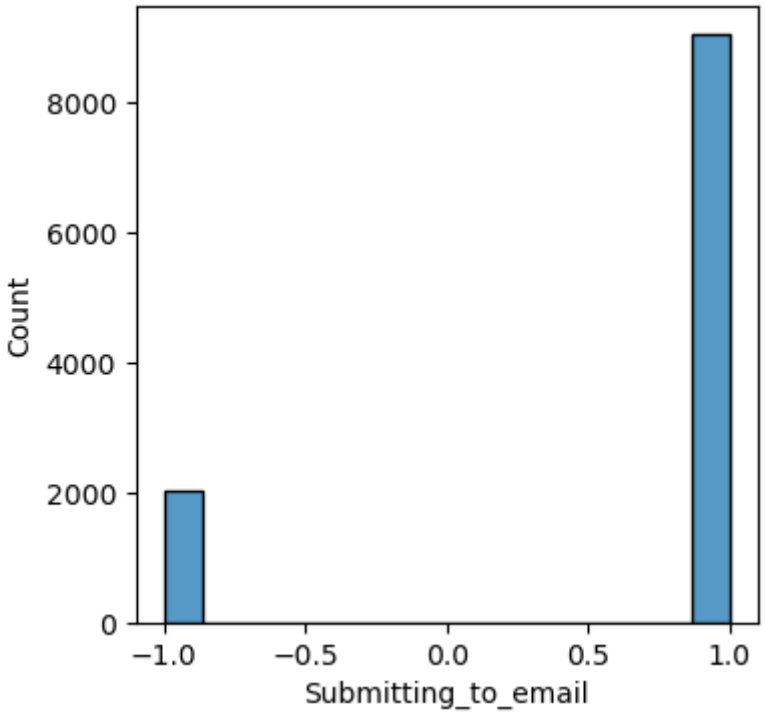

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
HTTPS_token

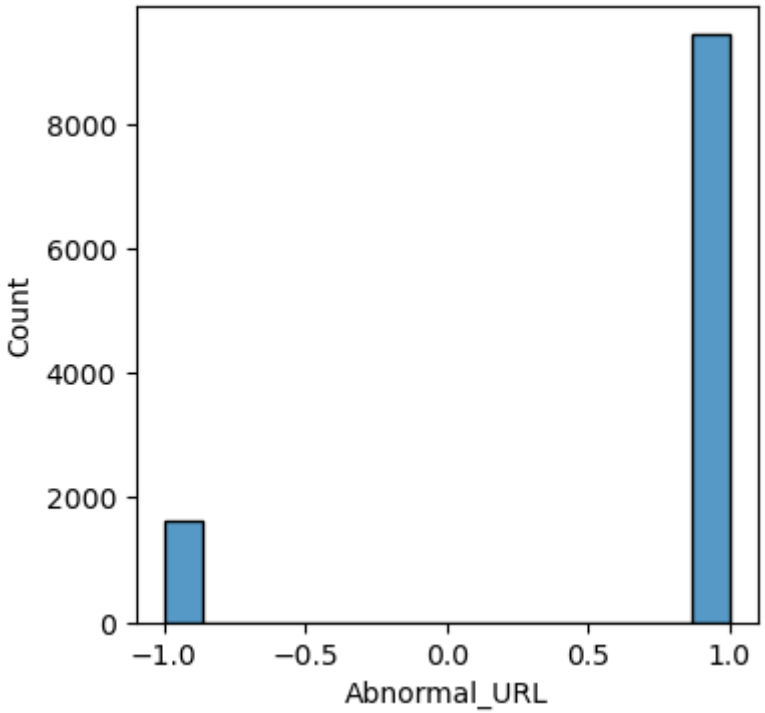

6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Request_URL

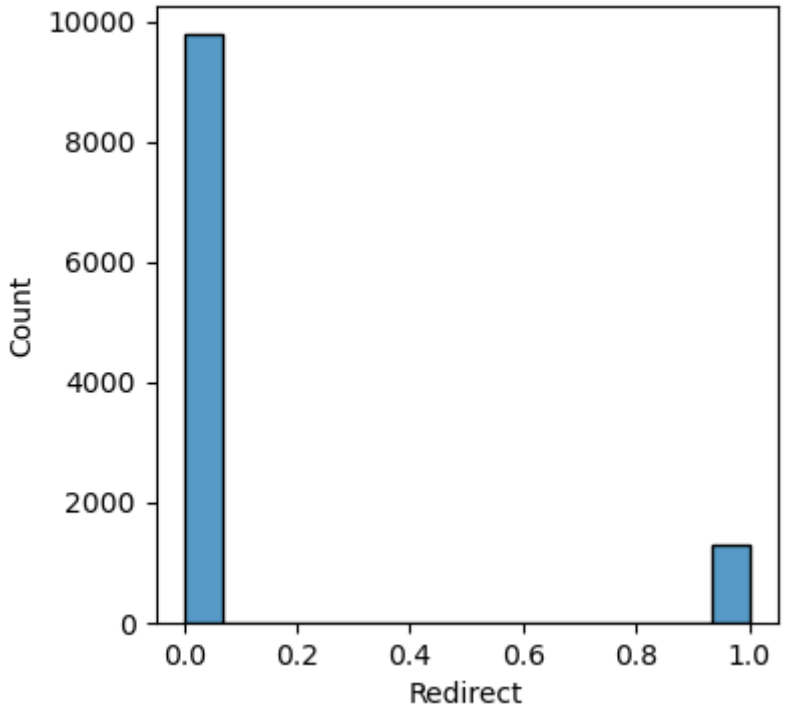

5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
URL_of_Anchor

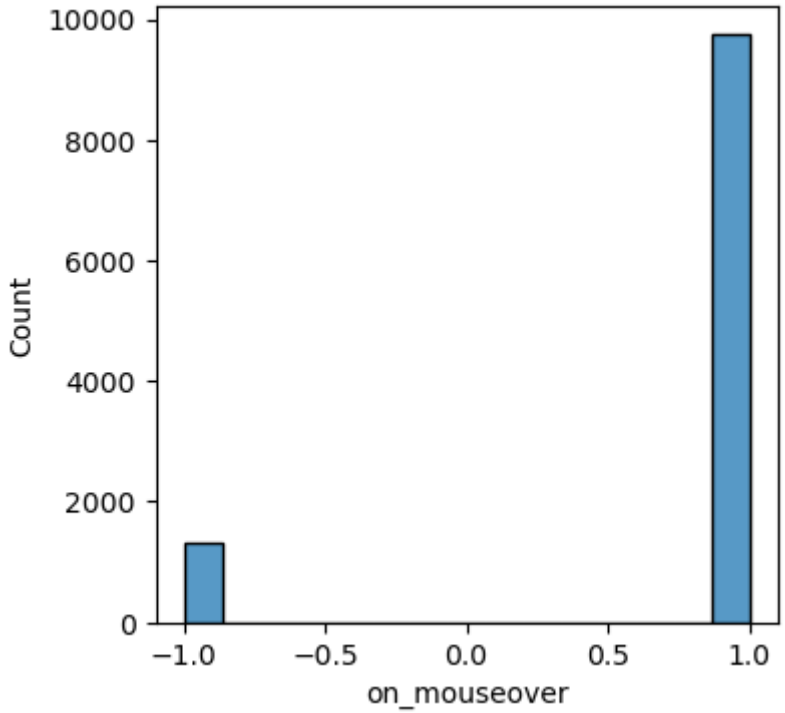

4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Links_in_tags

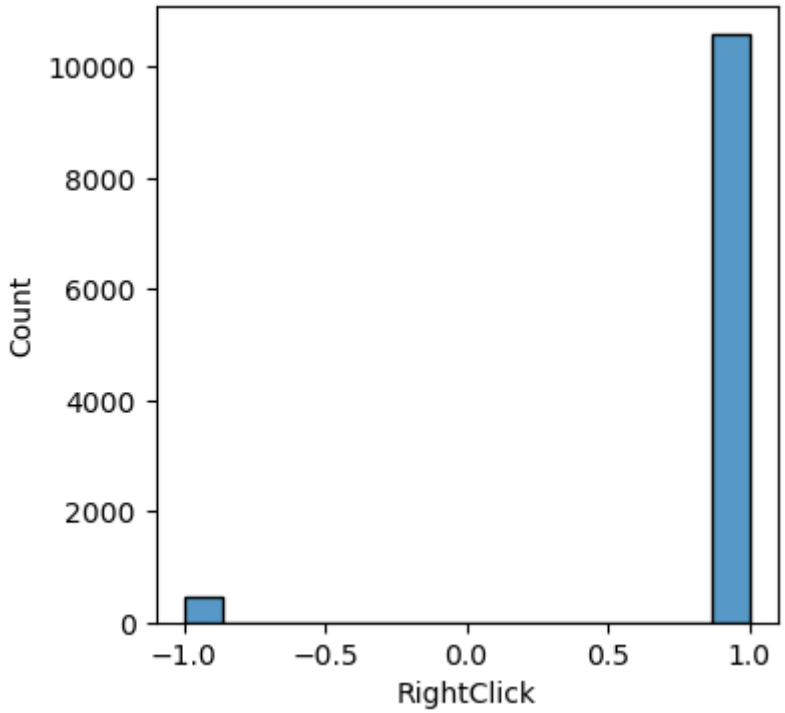

8000
7000
6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
SFH

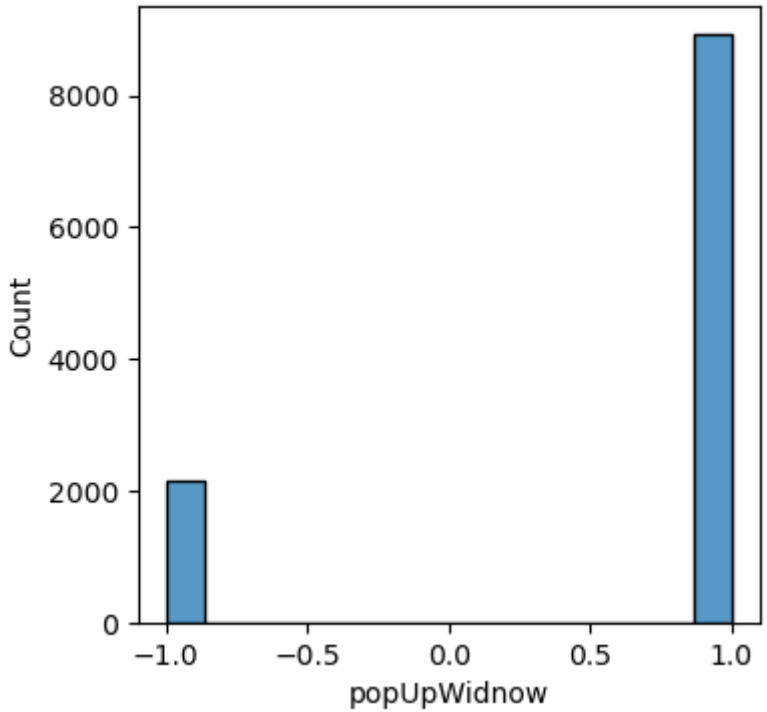

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Submitting_to_email

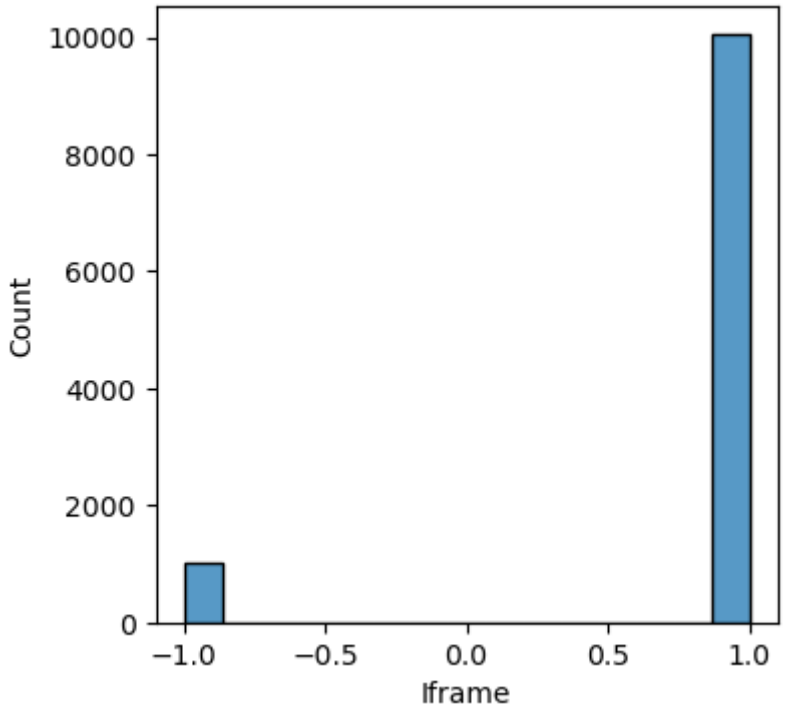

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Abnormal_URL

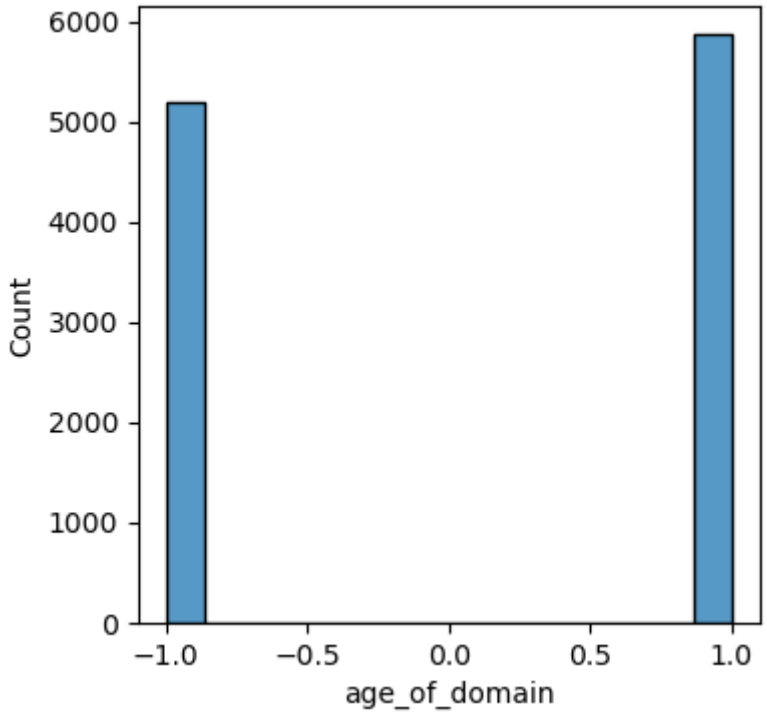

10000
8000
6000
4000
2000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Redirect

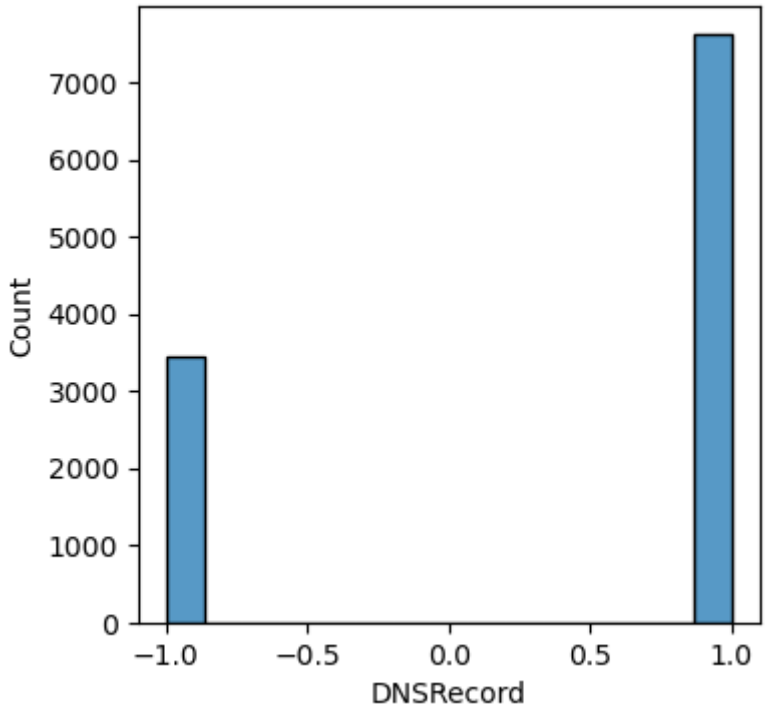

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
on_mouseover

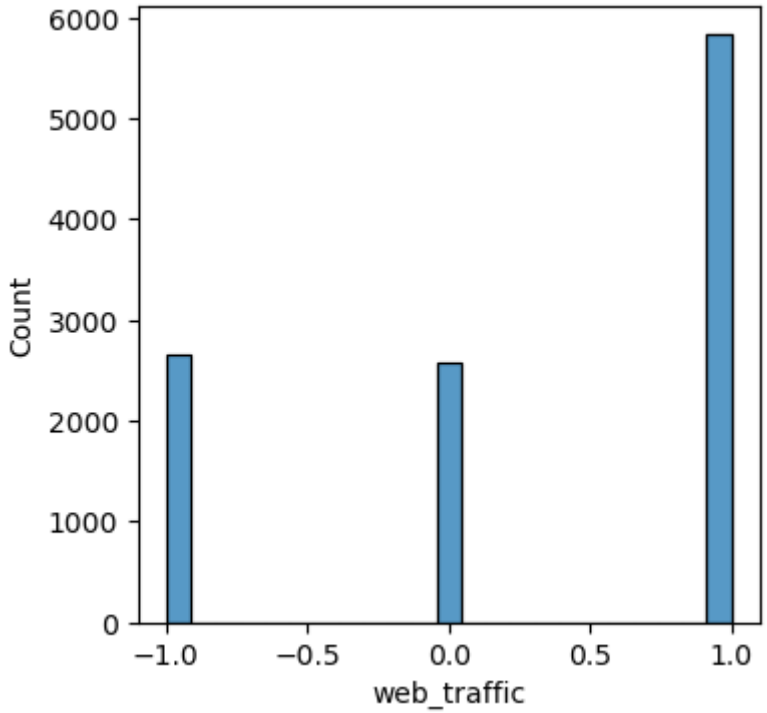

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
RightClick

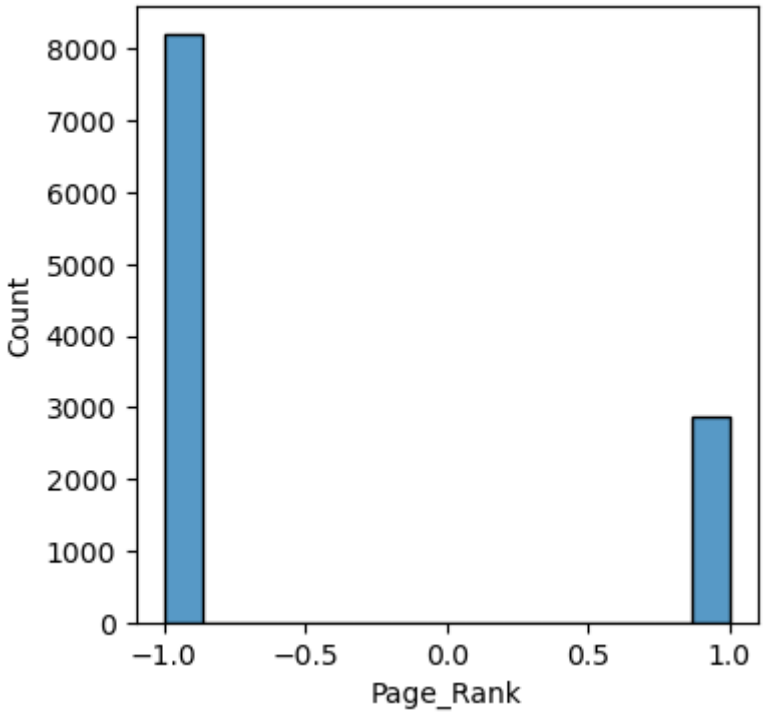

8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
popUpWidnow

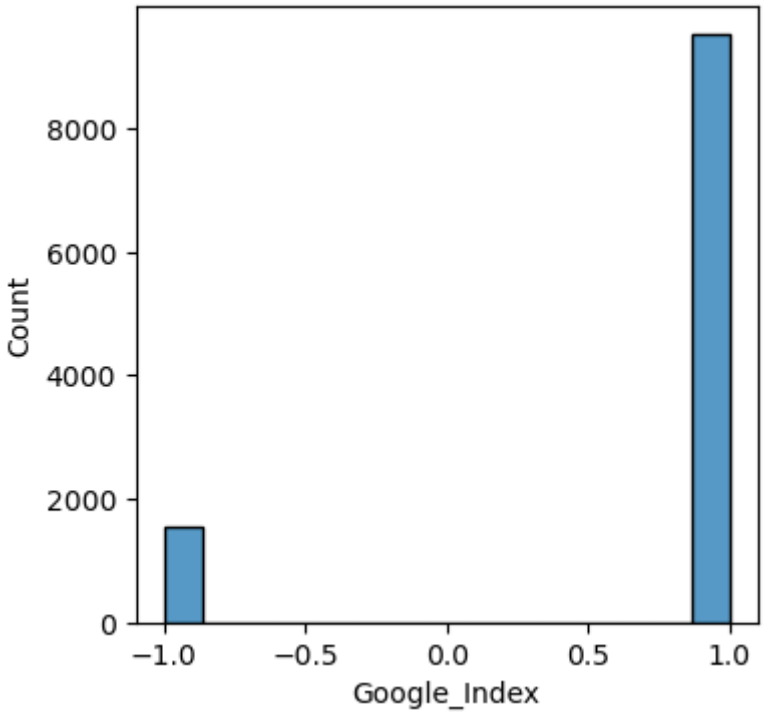

10000
8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Iframe

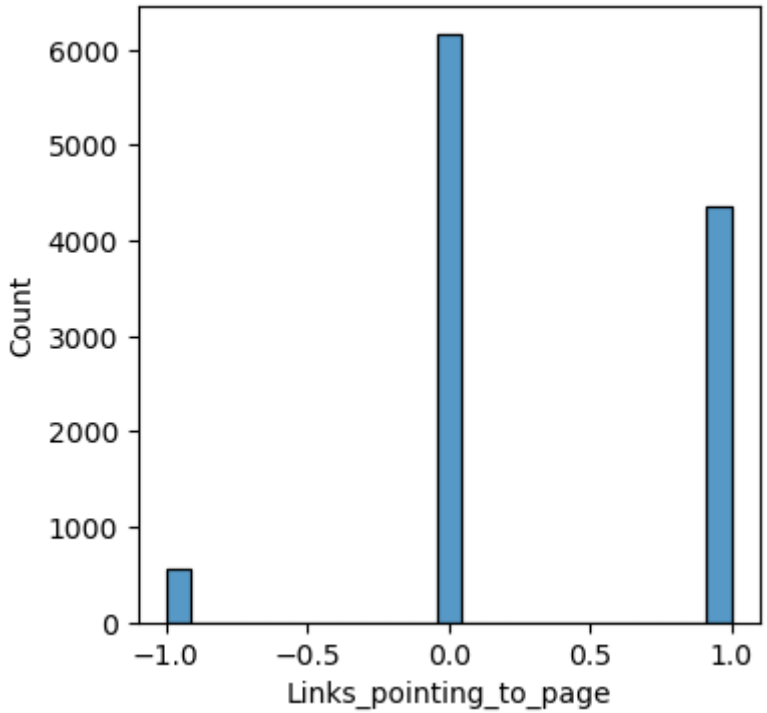

6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
age_of_domain

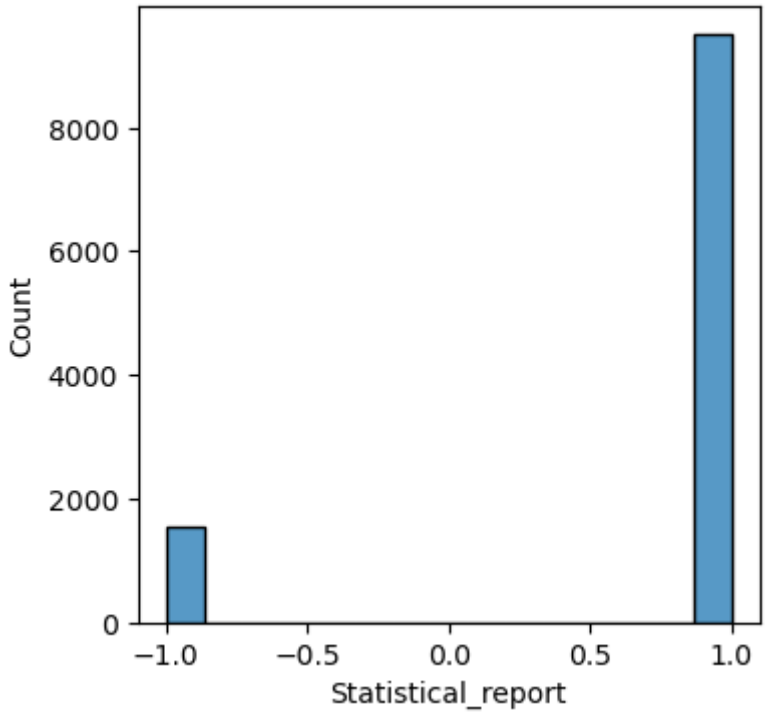

7000
6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
DNSRecord


6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
web_traffic

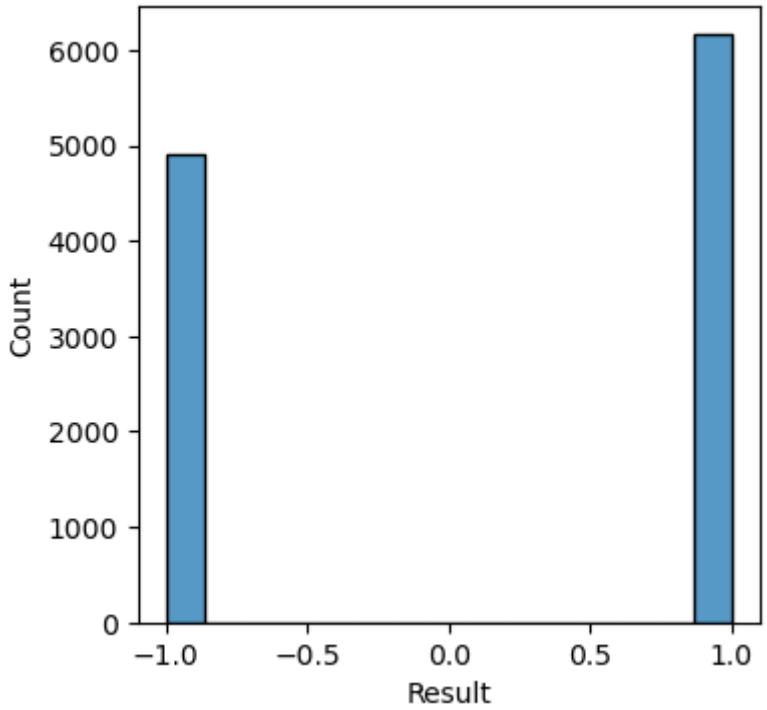

8000
7000
6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Page_Rank


8000
6000
4000
2000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Google_Index


6000
5000
4000
3000
2000
1000
0
Count
−1.0
−0.5
0.0
0.5
1.0
Links_pointing_to_page



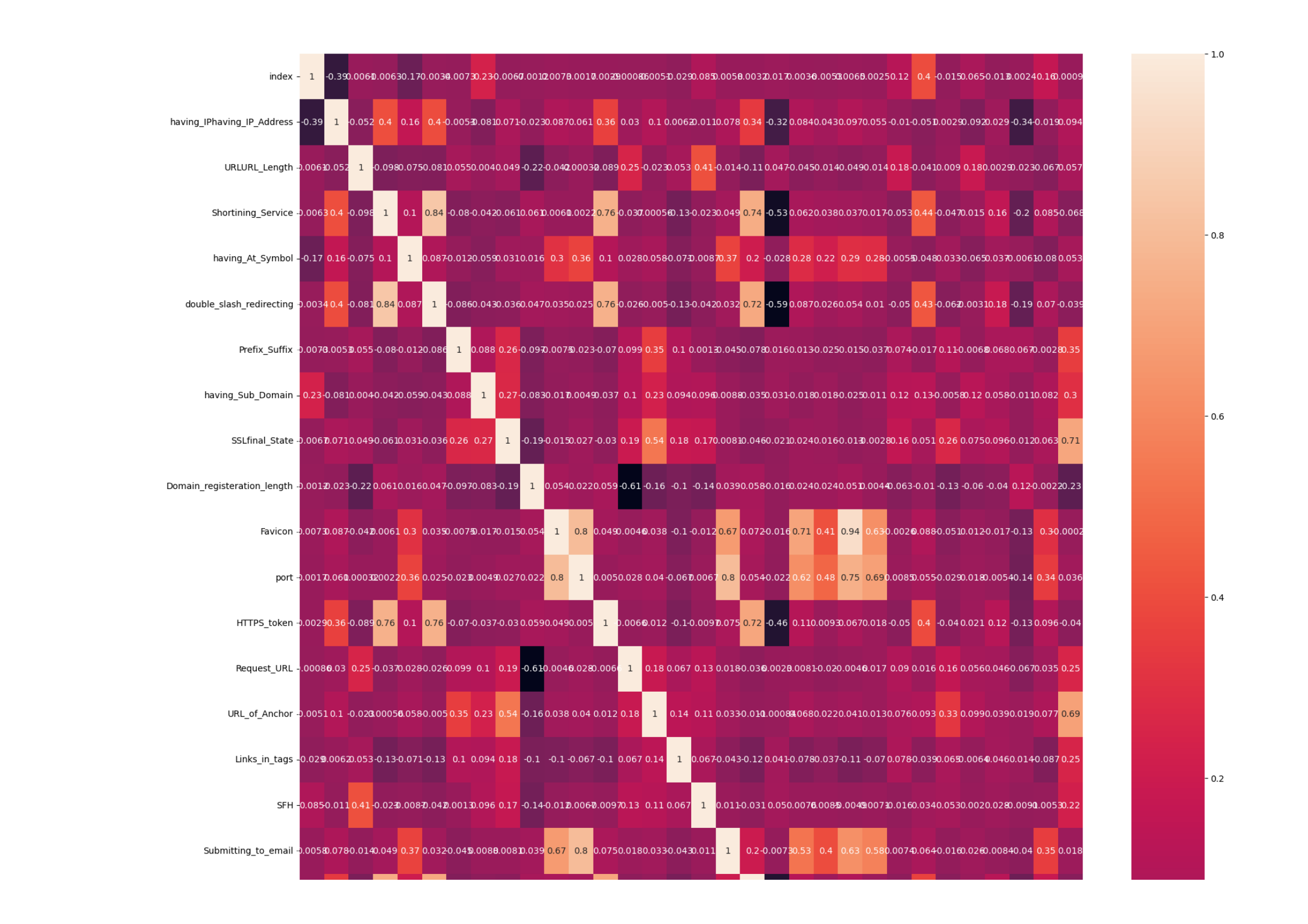
<Figure size 400x400 with 0 Axes>

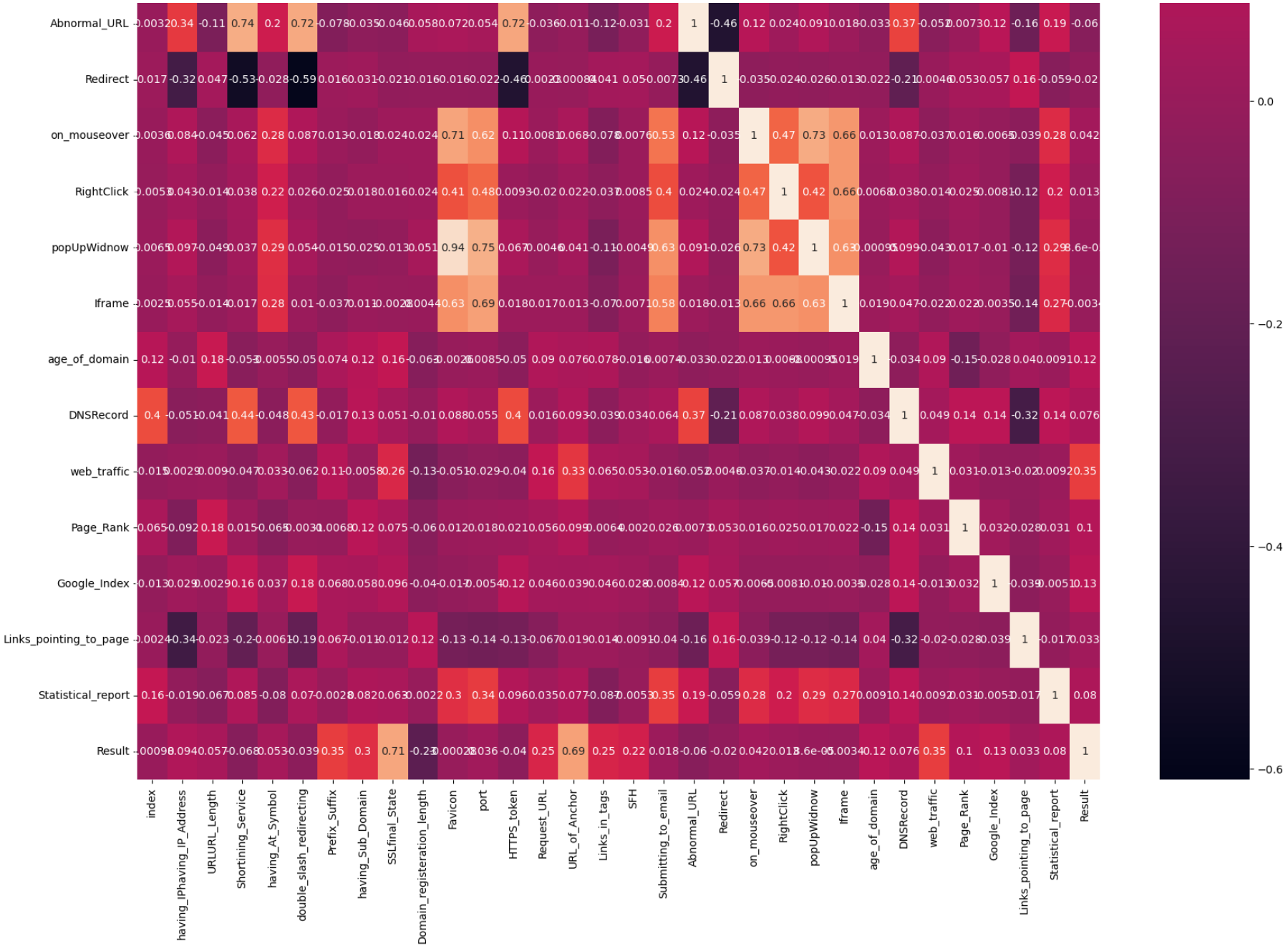


In [30]:
```
# features 'URLURL_Length','Prefix_Suffix','Domain_registeration_length','SFH','Page_Rank' have more impact on value -1
# and the rest features have high impact on value 1.

# features 'URL_of_Anchor', 'Links_in_tags' has more impact on value 0 and then -1 and has less impact on 1.
```

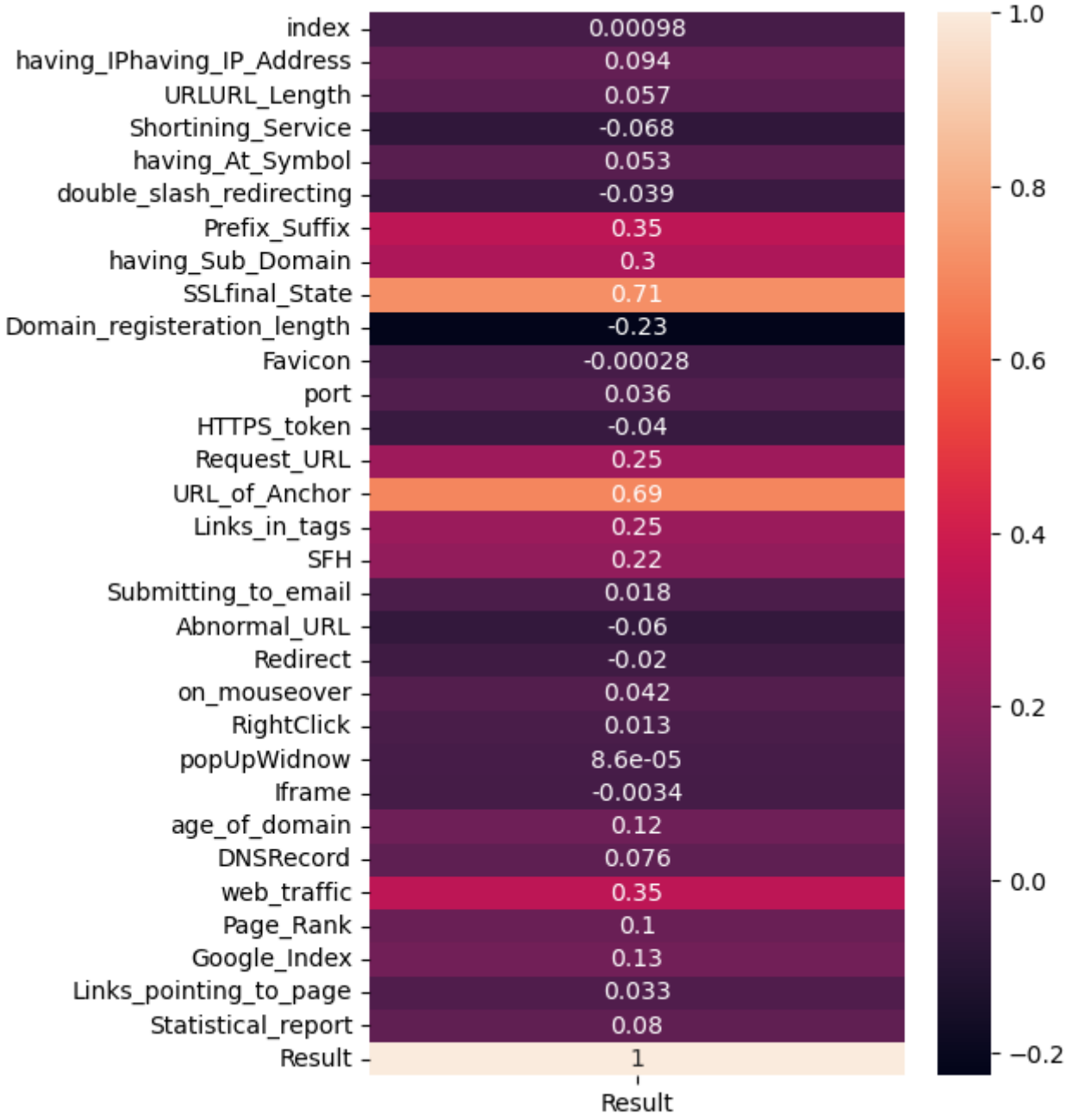
In [31]:
```
plt.figure(figsize = (20,30))
sns.heatmap(data = df.corr(), annot = True)
plt.show()
```


index
having_IPhaving_IP_Address
URLURL_Length
Shortining_Service
having_At_Symbol
double_slash_redirecting
Prefix_Suffix
having_Sub_Domain
SSLfinal_State
Domain_registeration_length
Favicon
port
HTTPS_token
Request_URL
URL_of_Anchor
Links_in_tags
SFH
Submitting_to_email
1.0
0.8
0.6
0.4
0.2


Abnormal_URL
Redirect
on_mouseover
RightClick
popUpWidnow
Iframe
age_of_domain
DNSRecord
web_traffic
Page_Rank
Google_Index
Links_pointing_to_page
Statistical_report
Result
index
having_IPhaving_IP_Address
URLURL_Length
Shortining_Service
having_At_Symbol
double_slash_redirecting
Prefix_Suffix
having_Sub_Domain
SSLfinal_State
Domain_registeration_length
Favicon
port
HTTPS_token
Request_URL
URL_of_Anchor
Links_in_tags
SFH
Submitting_to_email
Abnormal_URL
Redirect
on_mouseover
RightClick
popUpWidnow
Iframe
age_of_domain
DNSRecord
web_traffic
Page_Rank
Google_Index
Links_pointing_to_page
Statistical_report
Result
0.0
−0.2
−0.4
−0.6

In [32]:
```
plt.figure(figsize = (5,8))
sns.heatmap(df.corr()[['Result']], annot = True)
```

Out[32]:
```
<Axes: >
```



In [33]:
```
#feature 'SSLfinal_State' has high correlation 0.71 with 'Result
```

In [34]:
```
#2.  Determine the number of samples present in the data, unique elements in all the features.
```

In [35]:
```
df.shape
```

Out[35]:
```
(11055, 32)
```

In [36]:
```
df.describe().T
```

Out[36]:

	count	mean	std	min	25%	50%	75%	max
index	11055.0	5528.000000	3191.447947	1.0	2764.5	5528.0	8291.5	11055.0
having_IPhaving_IP_Address	11055.0	0.313795	0.949534	-1.0	-1.0	1.0	1.0	1.0
URLURL_Length	11055.0	-0.633198	0.766095	-1.0	-1.0	-1.0	-1.0	1.0
Shortining_Service	11055.0	0.738761	0.673998	-1.0	1.0	1.0	1.0	1.0
having_At_Symbol	11055.0	0.700588	0.713598	-1.0	1.0	1.0	1.0	1.0
double_slash_redirecting	11055.0	0.741474	0.671011	-1.0	1.0	1.0	1.0	1.0
Prefix_Suffix	11055.0	-0.734962	0.678139	-1.0	-1.0	-1.0	-1.0	1.0
having_Sub_Domain	11055.0	0.063953	0.817518	-1.0	-1.0	0.0	1.0	1.0
SSLfinal_State	11055.0	0.250927	0.911892	-1.0	-1.0	1.0	1.0	1.0
Domain_registeration_length	11055.0	-0.336771	0.941629	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11055.0	0.628584	0.777777	-1.0	1.0	1.0	1.0	1.0
port	11055.0	0.728268	0.685324	-1.0	1.0	1.0	1.0	1.0
HTTPS_token	11055.0	0.675079	0.737779	-1.0	1.0	1.0	1.0	1.0
Request_URL	11055.0	0.186793	0.982444	-1.0	-1.0	1.0	1.0	1.0
URL_of_Anchor	11055.0	-0.076526	0.715138	-1.0	-1.0	0.0	0.0	1.0
Links_in_tags	11055.0	-0.118137	0.763973	-1.0	-1.0	0.0	0.0	1.0
SFH	11055.0	-0.595749	0.759143	-1.0	-1.0	-1.0	-1.0	1.0
Submitting_to_email	11055.0	0.635640	0.772021	-1.0	1.0	1.0	1.0	1.0
Abnormal_URL	11055.0	0.705292	0.708949	-1.0	1.0	1.0	1.0	1.0
Redirect	11055.0	0.115694	0.319872	0.0	0.0	0.0	0.0	1.0
on_mouseover	11055.0	0.762099	0.647490	-1.0	1.0	1.0	1.0	1.0
RightClick	11055.0	0.913885	0.405991	-1.0	1.0	1.0	1.0	1.0
popUpWidnow	11055.0	0.613388	0.789818	-1.0	1.0	1.0	1.0	1.0
Iframe	11055.0	0.816915	0.576784	-1.0	1.0	1.0	1.0	1.0

	count	mean	std	min	25%	50%	75%	max
age_of_domain	11055.0	0.061239	0.998168	-1.0	-1.0	1.0	1.0	1.0
DNSRecord	11055.0	0.377114	0.926209	-1.0	-1.0	1.0	1.0	1.0
web_traffic	11055.0	0.287291	0.827733	-1.0	0.0	1.0	1.0	1.0
Page_Rank	11055.0	-0.483673	0.875289	-1.0	-1.0	-1.0	1.0	1.0
Google_Index	11055.0	0.721574	0.692369	-1.0	1.0	1.0	1.0	1.0
Links_pointing_to_page	11055.0	0.344007	0.569944	-1.0	0.0	0.0	1.0	1.0
Statistical_report	11055.0	0.719584	0.694437	-1.0	1.0	1.0	1.0	1.0
Result	11055.0	0.113885	0.993539	-1.0	-1.0	1.0	1.0	1.0

In [37]:
```
df['Result'].unique()
```

Out[37]:
```
array([-1,  1], dtype=int64)
```

In [38]:
```
df['Result'].nunique()
```

Out[38]:
```
2
```

In [39]:
```
for i in df.columns:
 print (i , ":", df[i].unique(),"\n len :",df[i].nunique())
 print ("\n")
```

```
index : [    1     2     3 ... 11053 11054 11055]
 len : 11055


having_IPhaving_IP_Address : [-1  1]
 len : 2


URLURL_Length : [ 1  0 -1]
 len : 3


Shortining_Service : [ 1 -1]
 len : 2


having_At_Symbol : [ 1 -1]
 len : 2


double_slash_redirecting : [-1  1]
 len : 2


Prefix_Suffix : [-1  1]
 len : 2


having_Sub_Domain : [-1  0  1]
 len : 3


SSLfinal_State : [-1  1  0]
 len : 3


Domain_registeration_length : [-1  1]
 len : 2


Favicon : [ 1 -1]
 len : 2
```

```
port : [ 1 -1]
 len : 2


HTTPS_token : [-1  1]
 len : 2


Request_URL : [ 1 -1]
 len : 2


URL_of_Anchor : [-1  0  1]
 len : 3


Links_in_tags : [ 1 -1  0]
 len : 3


SFH : [-1  1  0]
 len : 3


Submitting_to_email : [-1  1]
 len : 2


Abnormal_URL : [-1  1]
 len : 2


Redirect : [0 1]
 len : 2


on_mouseover : [ 1 -1]
 len : 2


RightClick : [ 1 -1]
 len : 2
```

```
popUpWidnow : [ 1 -1]
 len : 2


Iframe : [ 1 -1]
 len : 2


age_of_domain : [-1  1]
 len : 2


DNSRecord : [-1  1]
 len : 2


web_traffic : [-1  0  1]
 len : 3


Page_Rank : [-1  1]
 len : 2


Google_Index : [ 1 -1]
 len : 2


Links_pointing_to_page : [ 1  0 -1]
 len : 3


Statistical_report : [-1  1]
 len : 2


Result : [-1  1]
 len : 2
```

In [40]:
```
# 3.Check if there is any null value in any features.
```

In [41]:
```
df.isnull().sum()
```

Out[41]:
```
index                           0
having_IPhaving_IP_Address      0
URLURL_Length                   0
Shortining_Service              0
having_At_Symbol                0
double_slash_redirecting        0
Prefix_Suffix                   0
having_Sub_Domain               0
SSLfinal_State                  0
Domain_registeration_length     0
Favicon                         0
port                            0
HTTPS_token                     0
Request_URL                     0
URL_of_Anchor                   0
Links_in_tags                   0
SFH                             0
Submitting_to_email             0
Abnormal_URL                    0
Redirect                        0
on_mouseover                    0
RightClick                      0
popUpWidnow                     0
Iframe                          0
age_of_domain                   0
DNSRecord                       0
web_traffic                     0
Page_Rank                       0
Google_Index                    0
Links_pointing_to_page          0
Statistical_report              0
Result                          0
dtype: int64
```

In [42]:
```
# Correlation of features and feature selection:
```

In [43]:
```
# 4. Next, we have to find if there are any correlated features present in the data.
    # Remove the feature which might be correlated with some threshold.
```

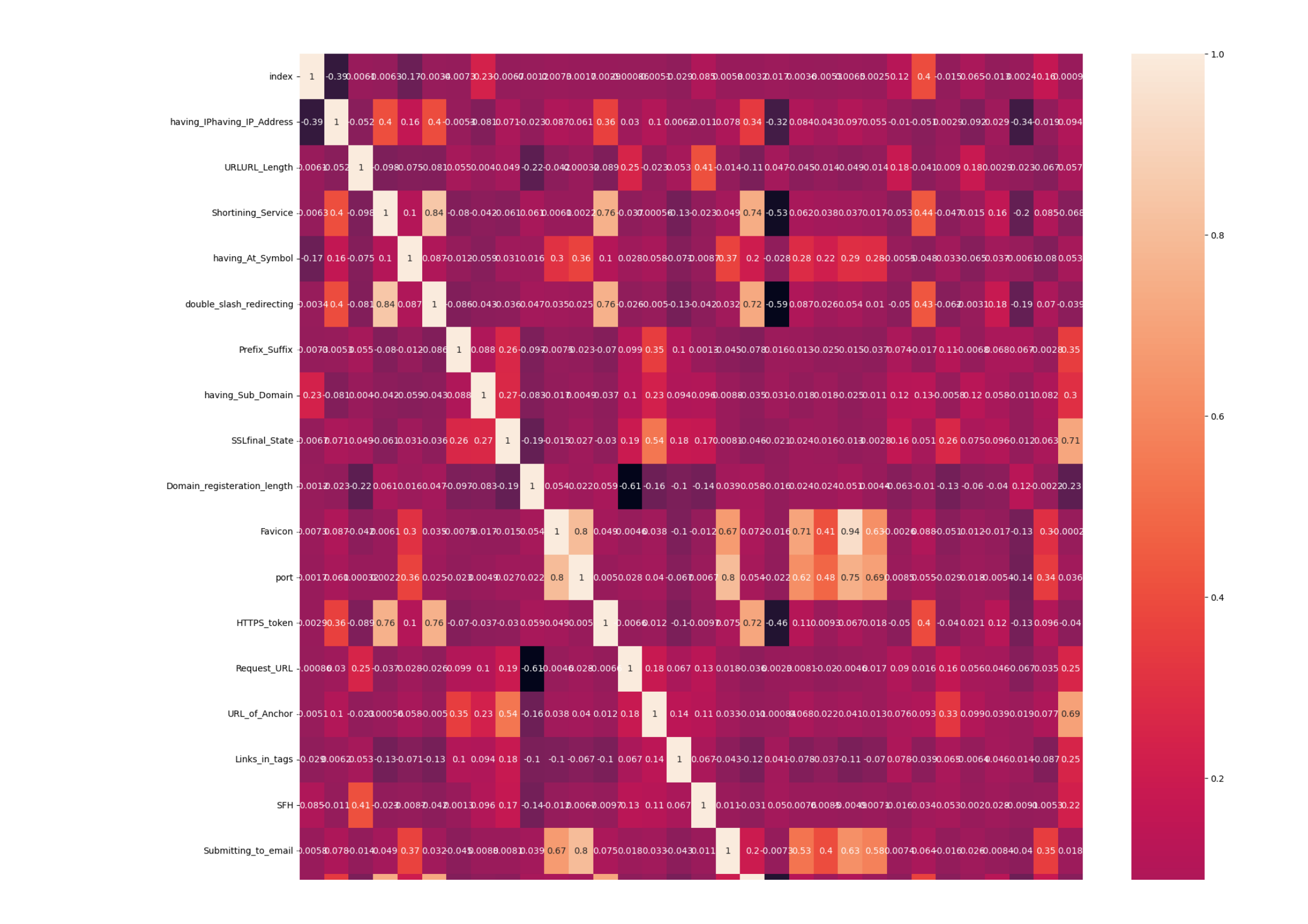
In [44]:
```
df.corr()
```

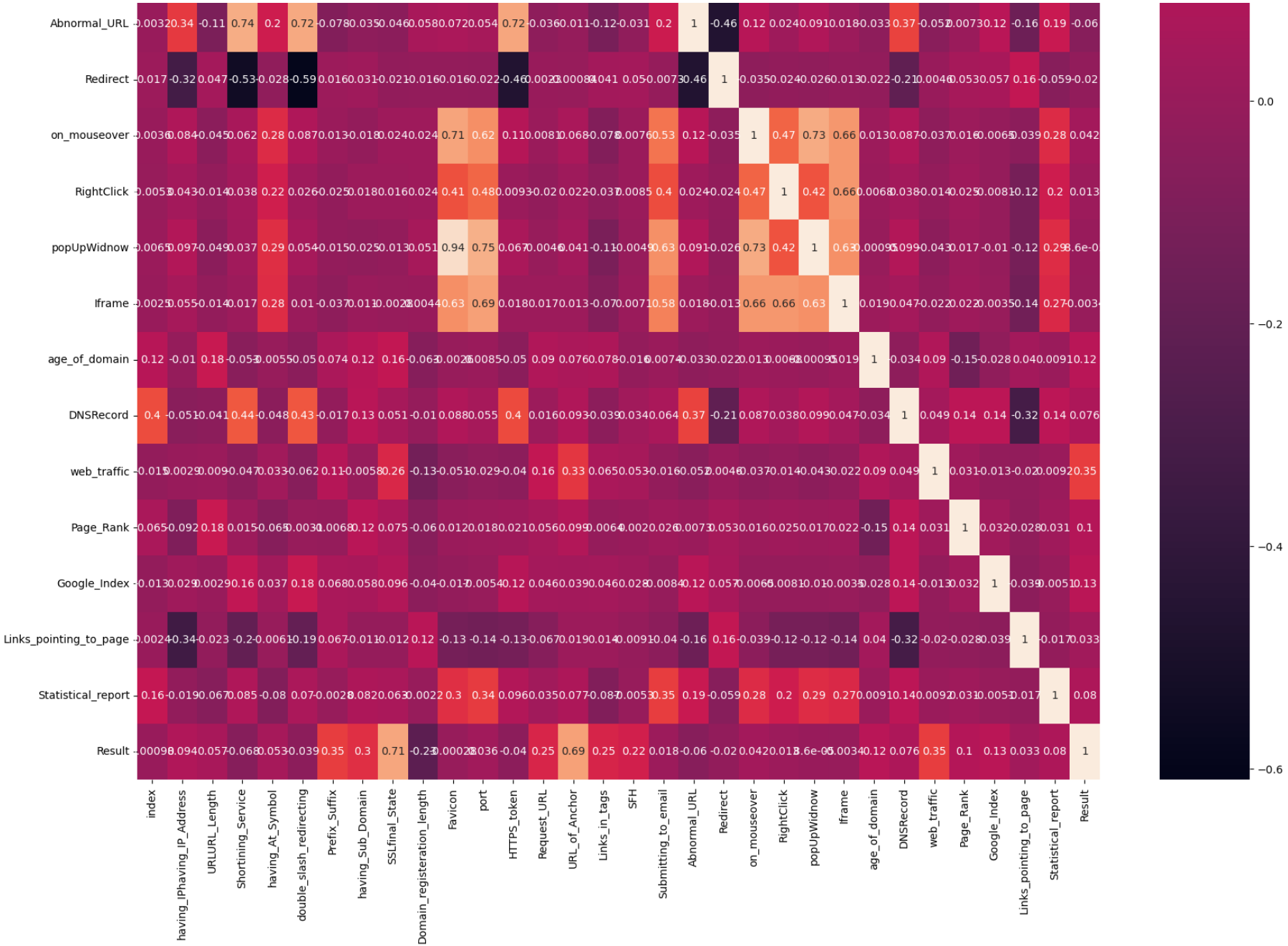
Out[44]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Pref
index	1.000000	-0.388317	0.006105	-0.006281	-0.169478	-0.003363	-
having_IPhaving_IP_Address	-0.388317	1.000000	-0.052411	0.403461	0.158699	0.397389	-
URLURL_Length	0.006105	-0.052411	1.000000	-0.097881	-0.075108	-0.081247	
Shortining_Service	-0.006281	0.403461	-0.097881	1.000000	0.104447	0.842796	-
having_At_Symbol	-0.169478	0.158699	-0.075108	0.104447	1.000000	0.086960	-
double_slash_redirecting	-0.003363	0.397389	-0.081247	0.842796	0.086960	1.000000	-
Prefix_Suffix	-0.007340	-0.005257	0.055247	-0.080471	-0.011726	-0.085590	
having_Sub_Domain	0.234091	-0.080745	0.003997	-0.041916	-0.058976	-0.043079	
SSLfinal_State	-0.006682	0.071414	0.048754	-0.061426	0.031220	-0.036200	
Domain_registeration_length	-0.001180	-0.022739	-0.221892	0.060923	0.015522	0.047464	-
Favicon	0.007293	0.087025	-0.042497	0.006101	0.304899	0.035100	-
port	0.001656	0.060979	0.000323	0.002201	0.364891	0.025060	-
HTTPS_token	0.002916	0.363534	-0.089383	0.757838	0.104561	0.760799	-
Request_URL	-0.000862	0.029773	0.246348	-0.037235	0.027909	-0.026368	
URL_of_Anchor	-0.005071	0.099847	-0.023396	0.000561	0.057914	-0.005036	
Links_in_tags	-0.028865	0.006212	0.052869	-0.133379	-0.070861	-0.125583	
SFH	0.085354	-0.010962	0.414196	-0.022723	-0.008672	-0.041672	
Submitting_to_email	0.005828	0.077989	-0.014457	0.049328	0.370123	0.031898	-
Abnormal_URL	0.003228	0.336549	-0.106761	0.739290	0.203945	0.723724	-
Redirect	0.016804	-0.321181	0.046832	-0.534530	-0.028160	-0.591478	
on_mouseover	0.003649	0.084059	-0.045103	0.062383	0.279697	0.086635	
RightClick	-0.005265	0.042881	-0.013613	0.038118	0.219503	0.025863	-
popUpWidnow	0.006515	0.096882	-0.049381	0.036616	0.290893	0.054463	-
Iframe	0.002533	0.054694	-0.013838	0.016581	0.284410	0.010459	-

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Pref
age_of_domain	0.115320	-0.010446	0.179426	-0.052596	-0.005499	-0.050107	
DNSRecord	0.400890	-0.050733	-0.040823	0.436064	-0.047872	0.431409	-
web_traffic	-0.014900	0.002922	0.008993	-0.047074	0.032918	-0.062369	
Page_Rank	0.065117	-0.091774	0.183518	0.014591	-0.064735	-0.003132	-
Google_Index	-0.012527	0.029153	0.002902	0.155844	0.037061	0.178415	
Links_pointing_to_page	0.002442	-0.339065	-0.022987	-0.198410	-0.006080	-0.194165	
Statistical_report	0.163799	-0.019103	-0.067153	0.085461	-0.080357	0.070390	-
Result	0.000978	0.094160	0.057430	-0.067966	0.052948	-0.038608	

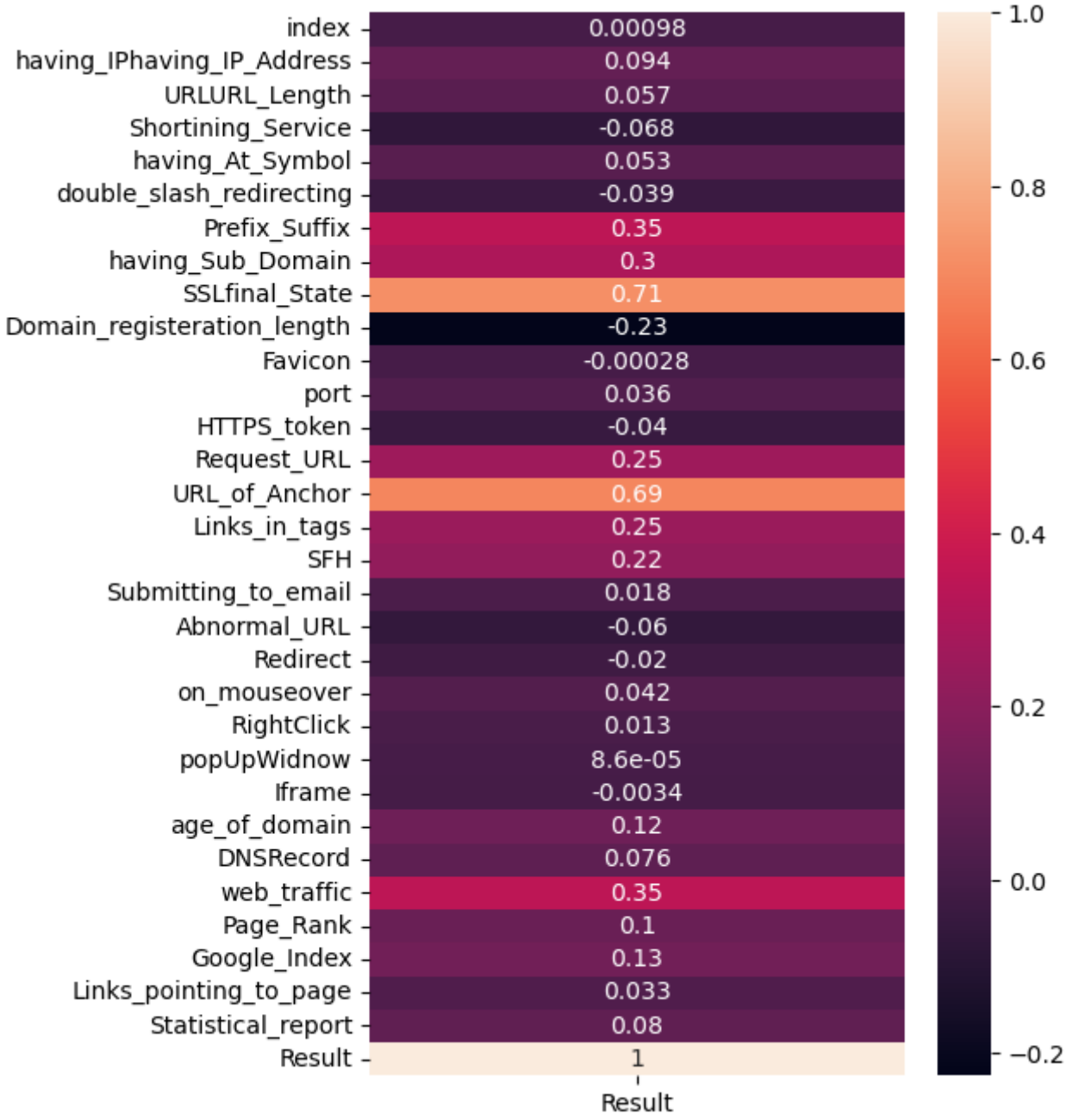
In [45]:
```
plt.figure(figsize = (20,30))
sns.heatmap(data = df.corr(), annot = True)
plt.show()
```


index
having_IPhaving_IP_Address
URLURL_Length
Shortining_Service
having_At_Symbol
double_slash_redirecting
Prefix_Suffix
having_Sub_Domain
SSLfinal_State
Domain_registeration_length
Favicon
port
HTTPS_token
Request_URL
URL_of_Anchor
Links_in_tags
SFH
Submitting_to_email
1.0
0.8
0.6
0.4
0.2


Abnormal_URL
Redirect
on_mouseover
RightClick
popUpWidnow
Iframe
age_of_domain
DNSRecord
web_traffic
Page_Rank
Google_Index
Links_pointing_to_page
Statistical_report
Result
index
having_IPhaving_IP_Address
URLURL_Length
Shortining_Service
having_At_Symbol
double_slash_redirecting
Prefix_Suffix
having_Sub_Domain
SSLfinal_State
Domain_registeration_length
Favicon
port
HTTPS_token
Request_URL
URL_of_Anchor
Links_in_tags
SFH
Submitting_to_email
Abnormal_URL
Redirect
on_mouseover
RightClick
popUpWidnow
Iframe
age_of_domain
DNSRecord
web_traffic
Page_Rank
Google_Index
Links_pointing_to_page
Statistical_report
Result
0.0
−0.2
−0.4
−0.6

In [46]:
```
plt.figure(figsize = (5,8))
sns.heatmap(df.corr()[['Result']], annot = True)
```

Out[46]:
```
<Axes: >
```


index 0.00098
having_IPhaving_IP_Address 0.094
URLURL_Length 0.057
Shortining_Service -0.068
having_At_Symbol 0.053
double_slash_redirecting -0.039
Prefix_Suffix 0.35
having_Sub_Domain 0.3
SSLfinal_State 0.71
Domain_registeration_length -0.23
Favicon -0.00028
port 0.036
HTTPS_token -0.04
Request_URL 0.25
URL_of_Anchor 0.69
Links_in_tags 0.25
SFH 0.22
Submitting_to_email 0.018
Abnormal_URL -0.06
Redirect -0.02
on_mouseover 0.042
RightClick 0.013
popUpWidnow 8.6e-05
Iframe -0.0034
age_of_domain 0.12
DNSRecord 0.076
web_traffic 0.35
Page_Rank 0.1
Google_Index 0.13
Links_pointing_to_page 0.033
Statistical_report 0.08
Result 1
Result
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [47]:
```
# with 0.84 'Shortining_Service' and 'double_slash_redirecting' are correlated with each other
# with 0.8  'Favicon', 'port' are correlated with each other
# not going to drop any columns has no high correlation
```

In [48]:
```
# Building Classification Model
```

In [49]:
```
#1.Finally, build a robust classification system that classifies whether the URL sample is a phishing site or not.
```

In [50]:
```
# Build classification models using a binary classifier to detect malicious or phishing URLs.
```

In [51]:
```
#Identify x and y
```

In [52]:
```
features = ['index', 'having_IPhaving_IP_Address', 'URLURL_Length',
       'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting',
       'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State',
       'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token',
       'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
       'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover',
       'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord',
       'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page',
       'Statistical_report']
target = ['Result']
```

In [53]:
```
x = df[features]
y = df[target]
```

In [54]:
```
#splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [57]:
```
from sklearn.linear_model import LogisticRegression
```

In [58]:
```
logreg=LogisticRegression()
```

In [59]:
```
logreg.fit(X_train,y_train)
```

Out[59]:
```
▾ LogisticRegression

LogisticRegression()
```

In [60]:
```
pred=logreg.predict(X_test)
```

In [61]:
```
logreg.score(X_train,y_train)
```

Out[61]:
```
0.9261409667836888
```

In [62]:
```
logreg.score(X_test,y_test)
```

Out[62]:
```
0.9199780761852563
```

In [63]:
```
#testing
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(y_test, pred)
```

Out[63]:
```
array([[1421,  144],
       [ 148, 1936]], dtype=int64)
```

In [64]:
```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

          -1       0.91      0.91      0.91      1565
           1       0.93      0.93      0.93      2084

    accuracy                           0.92      3649
   macro avg       0.92      0.92      0.92      3649
weighted avg       0.92      0.92      0.92      3649
```

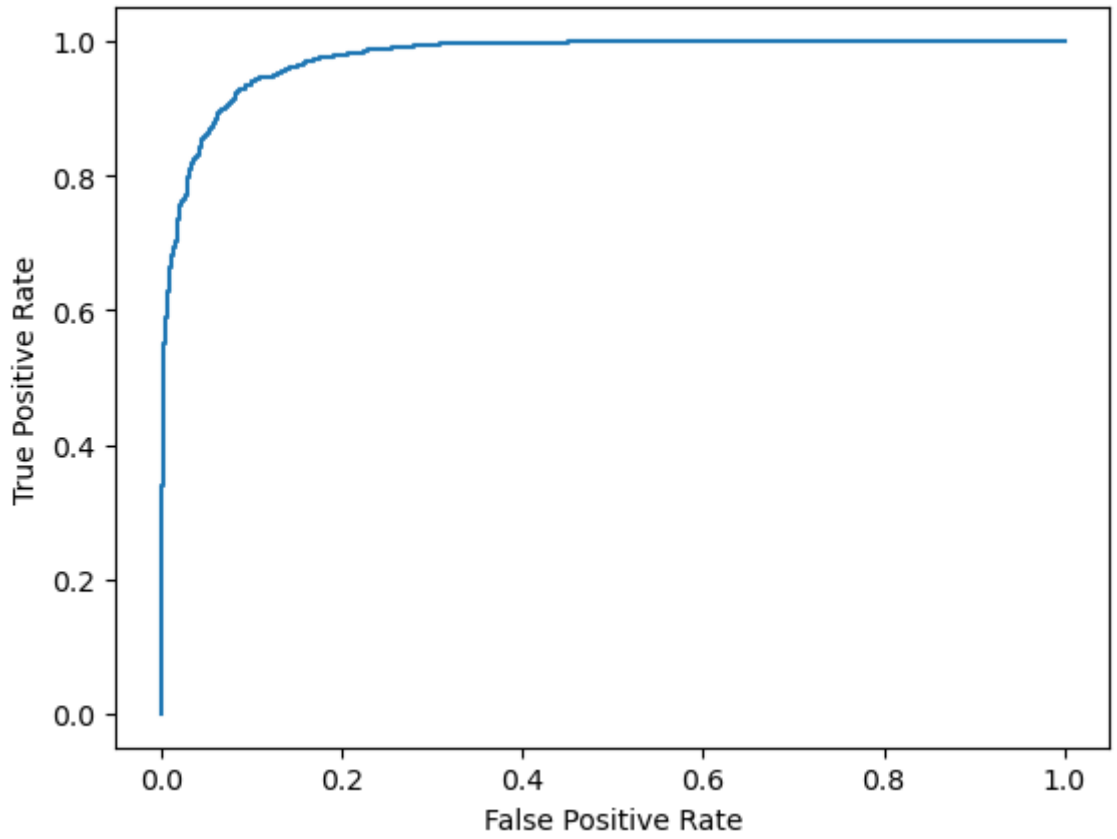
In [65]:
```
# Illustrate the diagnostic ability of this binary classifier by plotting the ROC curve.
```

In [66]:
```
from sklearn import metrics

y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)

#create ROC curve
```

```
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [67]:
```
# Validate the accuracy of data by the K-Fold cross-validation technique.
```

In [68]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
from numpy import absolute
from numpy import sqrt
```

In [69]:
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
```

In [70]:
```
#use k-fold CV to evaluate model
scores = cross_val_score(logreg, x, y, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
```

In [71]:
```
#view mean absolute error
mean(absolute(scores))
```

Out[71]:
```
0.15413630301195455
```

In [72]:
```
# average absolute error between the model prediction and the actual observed data is 0.154
```

In [73]:
```
#view RMSE
sqrt(mean(absolute(scores)))
```

Out[73]:
```
0.39260196511473877
```

In [74]:
```
# root mean squared error (RMSE) was 0.392
```

In [75]:
```
# The final output consists of the model, which will give maximum accuracy on the validation dataset with selected attributes.
```

In [76]:
```
#testing
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(y_test, pred)
```

Out[76]:
```
array([[1421,  144],
       [ 148, 1936]], dtype=int64)
```

In [77]:
```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

          -1       0.91      0.91      0.91      1565
           1       0.93      0.93      0.93      2084

    accuracy                           0.92      3649
   macro avg       0.92      0.92      0.92      3649
weighted avg       0.92      0.92      0.92      3649
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: