Nithin S
221IT085

Verifying existing URLs with VirusTotal and Extracting Features from URL Dataset to build a new dataset

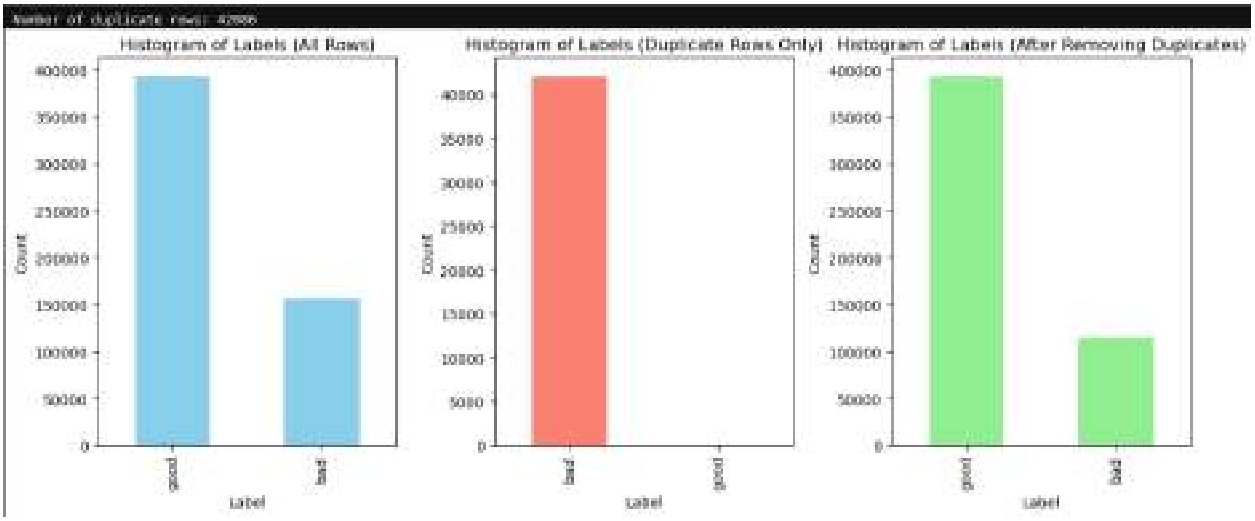
Phishing Site URLs: Dataset which contains Phishing urls and non phishing urls.

Total URLs: 549362
Total Unique URLs: 507195

URL unique URLs	Label good and bad URLs classes
507195 **unique values**	good 72% bad 28%
nobell.it/70ffb52d079109dca5664cce6f317373782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb52d...	bad
www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrcmd=_home-customer&nav=1/loading.php	bad
serviciosbys.com/paypal.cgi.bin.get-into.herf.secure.dispatch35463256rzr321654641dsf654321874/href/h...	bad

Histogram about the classes in the URLs Dataset

Total Rows present in given dataset: 549346
No of duplicates : 42086
Percentage of duplicates: 7.66 %
Total Unique URLs: 507195



No Null Values were found in the given URLs Dataset

```
phish_data.isnull().sum() # there is no missing values
```

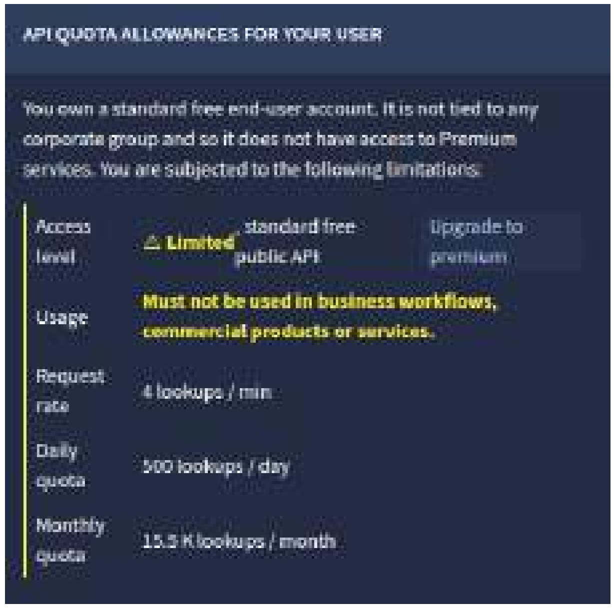
```
URL      0
Label    0
dtype: int64
```

Python Script to verify the labelling of the dataset

Please make sure to add required API Keys before running the code.

Virus Total API : https://docs.virustotal.com/reference/scan-url

Virus Total API Endpoint has strict rate limits



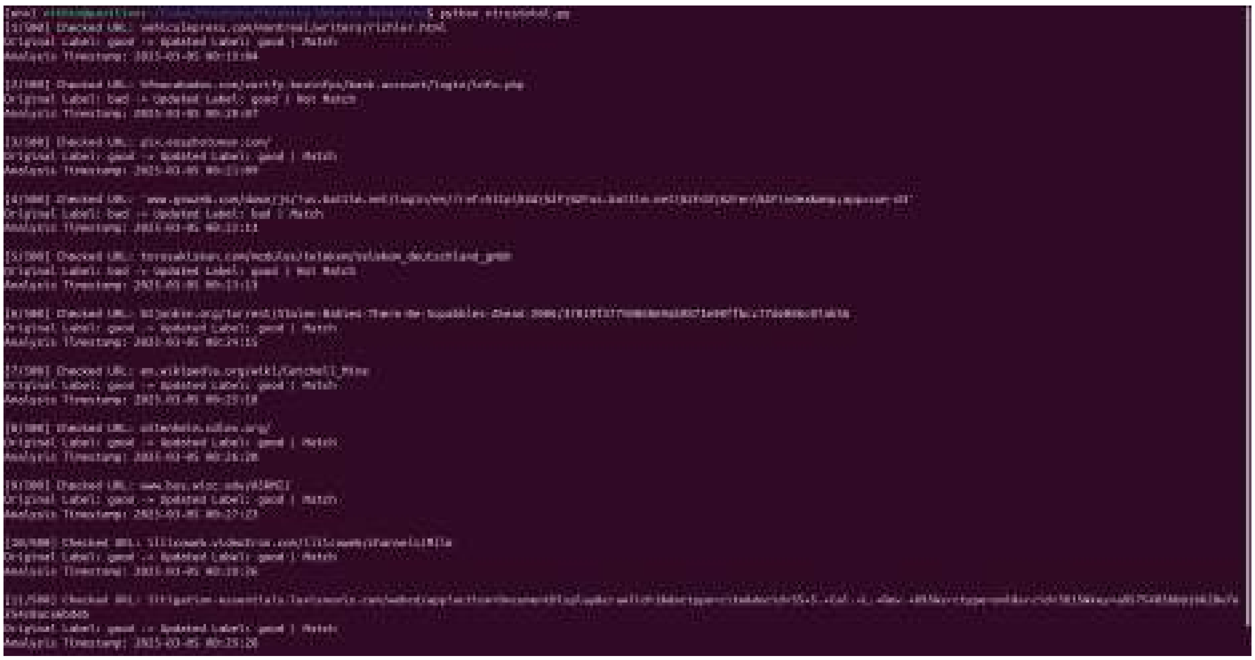
So, we will verify random 500 urls from the dataset without repetition and update them. We will have a **time delay of 60s** between each URL Verification request.

CODE

```
def check_url_virustotal(url, default_label):
    endpoint = "https://www.virustotal.com/api/v3/urls"
    headers = {"x-apikey": os.getenv("VIRUS_TOTAL_API_KEY")}
    try:
        response = requests.post(endpoint, headers=headers, data={"url": url})
        if response.status_code != 200:
            return default_label
        analysis_id = response.json()["data"]["id"]
        result_endpoint = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"
        result_response = requests.get(result_endpoint, headers=headers)
        if result_response.status_code != 200:
            return default_label
        result_data = result_response.json()["data"]["attributes"]["results"]
```

```
        malicious_count = sum(1 for scan in result_data.values() if scan["category"] == "malicious")
        return "bad" if malicious_count > 0 else "good"
    except Exception:
        return default_label
```

Output



Total mismatches : 120 out of 500
Percentage of mismatches: 120/500 * 100 = 24%

Due to size limitations I have uploaded the datasets in my google drive.

Extracted Features Dataset Google Drive Link:
https://drive.google.com/file/d/119L_eJDb8Oizm4jv3NM0tfW-Um-7sSDC/view?usp=sharing

Z Score Normalized Dataset Google Drive Link:
https://drive.google.com/file/d/1P8sOZi6j7JeM_LNLFmChU9FPHnyo_u1X/view?usp=sharing

List of Features planned to be extracted: 116 Features - (Without dropping repeated columns)

Full URL Length: Total number of characters in the entire URL.
Hostname (Domain) Length: Total number of characters in the domain name part.
Directory Length: Number of characters in the folder or path part of the URL.
File Name Length: Number of characters in the file name portion (if any).
Parameters Length: Number of characters in the query string (everything after "?").
TLD Length: Length (in characters) of the top-level domain (for example, "com" or "org").

Dot ('.') Count: Number of periods used (often separating subdomains or domain parts).
Hyphen ('-') Count: Number of hyphens used.
Underscore ('_') Count: Number of underscores.
Slash ('/') Count: Number of forward slashes.
Question Mark ('?') Count: How many "?" appear.
Equal Sign ('=') Count: Number of equal signs.
At Sign ('@') Count: Number of "@" symbols.
Ampersand ('&') Count: How many "&" symbols appear.
Exclamation Mark ('!') Count: Number of "!" symbols.
Space Count: Number of space characters.
Tilde ('~') Count: Number of tilde characters.
Comma (',') Count: How many commas appear.
Plus Sign ('+') Count: Number of "+" symbols.
Asterisk ('*') Count: Number of "*" symbols.
Hashtag ('#') Count: Number of "#" symbols.
Dollar Sign ('$') Count: Number of "$" symbols.
Percent Sign ('%') Count: Number of "%" symbols.

Common Terms Occurrence: Counts for terms such as "www", ".com", "http", and "//" that usually appear only once in normal URLs.
Email in URL: A flag indicating if an email address is embedded in the URL.

HTTPS Token: Checks if the URL uses "https" (a sign of secure connections).
IP Address in URL: A binary check to see if an IP address is used instead of a domain name.
Punycode Usage: Checks whether the domain uses punycode (which can mask its true characters).
Port Number Presence: A flag indicating if the URL explicitly shows a port (like ":80" or ":443").
TLD Position: Verifies that the top-level domain is in the right place (it should not appear in the wrong section like the path or subdomain).
Abnormal Subdomains: Detects unusual subdomain patterns (for example, variations of "www" that include numbers).
Number of Subdomains: Counts how many subdomains are present.

Prefix/Suffix with Hyphen: Checks if the domain uses hyphens to separate extra words (which might be used to mimic legitimate sites).
Random Domain Indicator: Determines if the domain seems to be made up of random characters.
URL Shortening Service: A flag to see if a URL shortener (like bit.ly) is used, which can hide the true destination.
Path Extension Check: Looks for suspicious file extensions (such as ".exe" or ".js") in the URL path.
Suspicious TLD: Checks if the top-level domain is among those known to be risky.

Digit Ratio in Full URL: Proportion of digit characters compared to the total characters in the URL.
Digit Ratio in Hostname: Proportion of digits in the domain name itself.

Word Count: Number of words found in the full URL, the hostname, or the path.
Shortest & Longest Word: Identification of the shortest and longest word in the URL parts.
Average Word Length: The average length of words in the URL, hostname, or path.
Phish Hints: Counts occurrences of suspicious or phishing-related keywords (like "login", "admin", "signin", etc.).

Brand Names in URL:
In the Domain: Presence of well-known brand names can be a sign of legitimacy.
In the Subdomain or Path: Their appearance here may indicate an attempt to deceive.
Domain in Page Title/Copyright: Checks if the domain name appears in the webpage title or copyright text (a sign of legitimacy).

Redirection Count: Total number of times the URL redirects to another page.
External Redirections: How many of these redirects go to a different domain.
Internal vs. External Hyperlinks Ratio: Compares links that point within the same site to those that point to external sites.
Null Hyperlinks Ratio: Proportion of links that lead nowhere (empty links).
Media Links Ratio: Ratio of media (images, videos, etc.) hosted on the same domain versus externally.
Connection Errors Ratio: Ratio of hyperlinks that result in errors (broken links).

Number of Hyperlinks: Total links present on the webpage.
External CSS Files Count: Number of CSS files linked from outside the domain.
Login Forms Presence: Checks for login forms, especially those with empty or suspicious action attributes.
External Favicon: Whether the page uses a favicon (the small icon in the browser tab) from an external source.
Invisible iFrame: Detects hidden iframe elements that might load content from another domain.
Pop-up Windows: Looks for pop-up windows that include text fields (which can be a sign of phishing).

Unsafe Anchors: Counts anchor (<a>) tags that use unsafe links (e.g., “javascript:” or “#”).
Right-Click Blocking: Checks for scripts that disable the right-click function (which can hide page source).
Empty Title: Flags if the webpage has no title tag.

WHOIS Registration: Whether the domain is found in the WHOIS database (a missing record is a red flag).
Domain Registration Length: The number of years for which the domain is registered (short registration periods can be suspicious).
Domain Age: How long the domain has been active.
DNS Record Check: Verifies that the domain has proper DNS records.
Google Index: Checks if the URL or domain is indexed by Google (phishing sites are often not).
Page Rank: An estimate of the webpage’s popularity.
Web Traffic: An indicator (like Alexa ranking) showing the number of visitors.

Additionally, one study mentions a “statistical report” feature that checks if the domain’s IP matches known top phishing domains.

Vowel Count in Domain: Number of vowels in the domain name.
Domain in IP Format: Whether the domain is written as an IP address.
“Server” or “Client” in Domain: Checks if these words appear in the domain name, which can hint at its purpose.

Domain Lookup Response Time: How long it takes to get a response when looking up the domain.
SPF Record: Checks if the domain has an SPF record (helps validate email sources).
ASN (Autonomous System Number): A number that identifies the network the domain’s IP belongs to.
Domain Activation Time: How many days have passed since the domain was first activated.
Domain Expiration Time: How many days remain until the domain expires.
Number of Resolved IPs: How many IP addresses are returned when the domain is looked up.
Nameservers Count: Number of DNS nameservers linked to the domain.
MX Servers Count: Number of mail servers associated with the domain.
TTL (Time-To-Live) of Hostname: The DNS record’s lifetime.
Valid TLS/SSL Certificate: Whether the site has a proper secure certificate.
URL Shortened Flag: Whether the URL has been shortened (also noted earlier under security).

TLD Present in Parameters: Checks if a top-level domain appears within the URL parameters (which is unusual).
Number of Parameters: Count of key–value pairs or parameters present in the URL query string.

Check for duplicate Columns

```
Column Index: 14, Column Name: tilde_count
Column Index: 28, Column Name: https_token
Column Index: 60, Column Name: brand_in_subdomain
Column Index: 86, Column Name: whois_registration
Column Index: 87, Column Name: domain_registration_length
Column Index: 88, Column Name: domain_age
Column Index: 96, Column Name: server_or_client_in_domain
Column Index: 98, Column Name: asn
Column Index: 99, Column Name: domain_activation_time
Column Index: 100, Column Name: domain_expiration_time
```

Check for duplicate rows

```
5339
9567
16871
18587
18965
18967
19005
19398
19673
19897
22329
22413
22605
22755
22762
22824
22892
```

Normalization

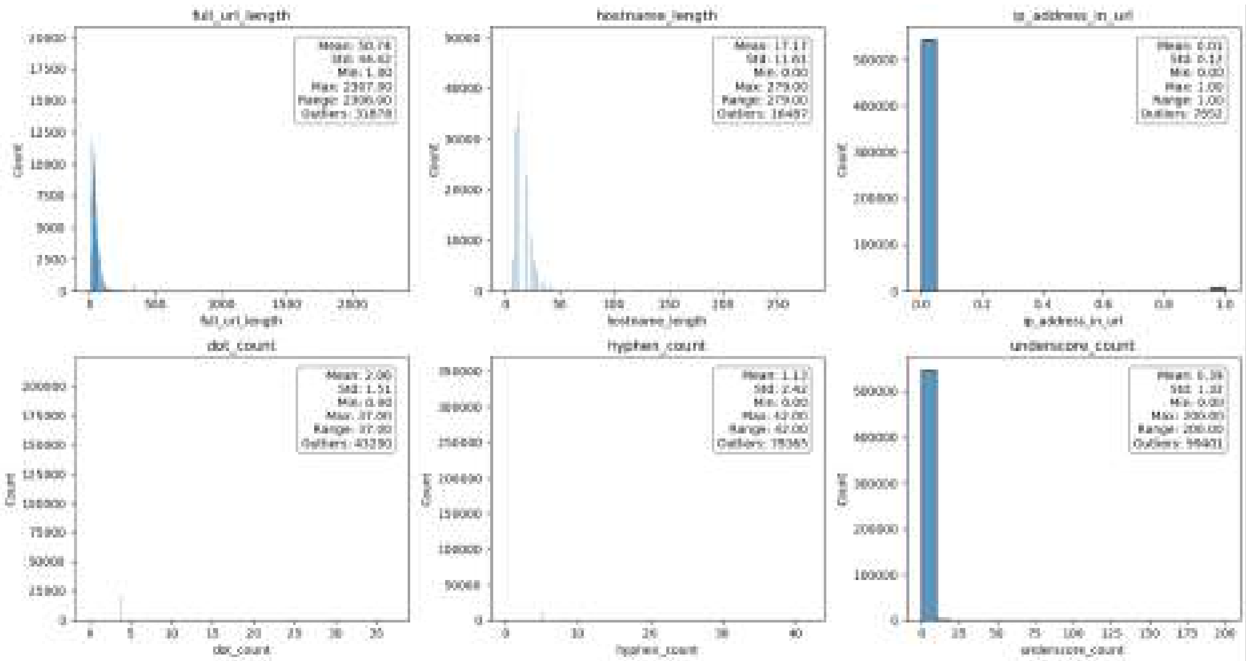
I have normalized all columns , since all were numerical.

Handle Missing Values

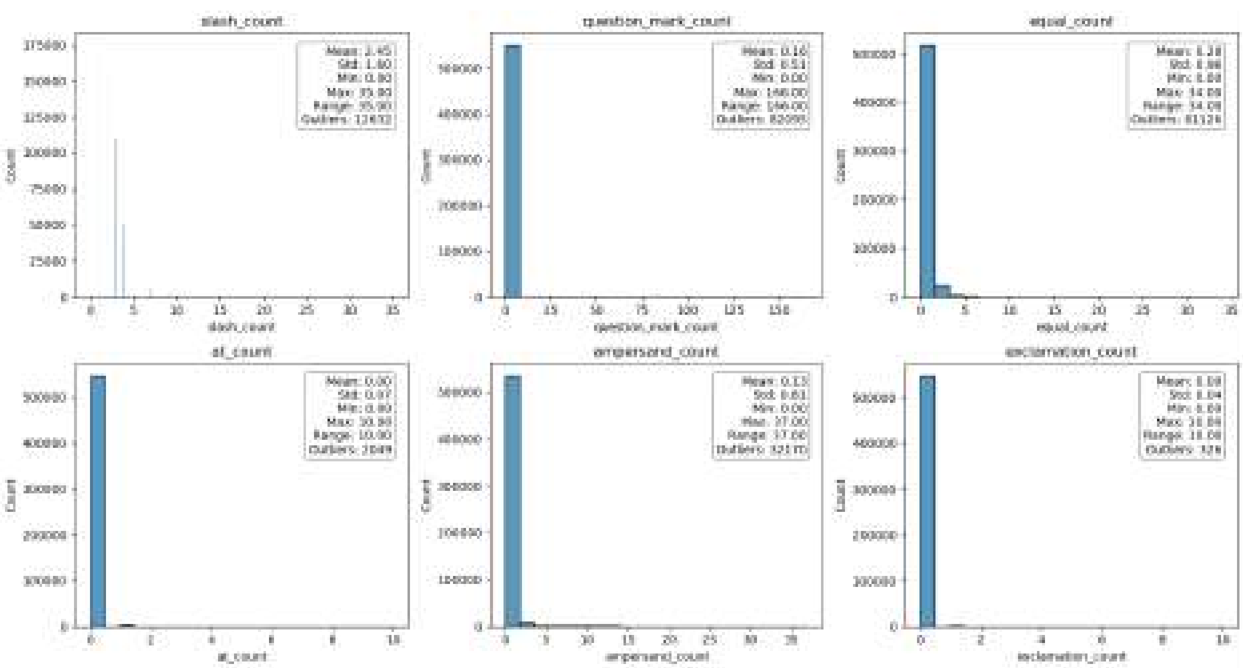
```
df.isnull().sum()

URL                  0
full_url_length      0
hostname_length      0
ip_address_in_url    0
dot_count            0
                    ..
domain_in_title      0
web_traffic          0
google_index         0
page_rank            0
Label                0
Length: 117, dtype: int64
```

NO missing values were found

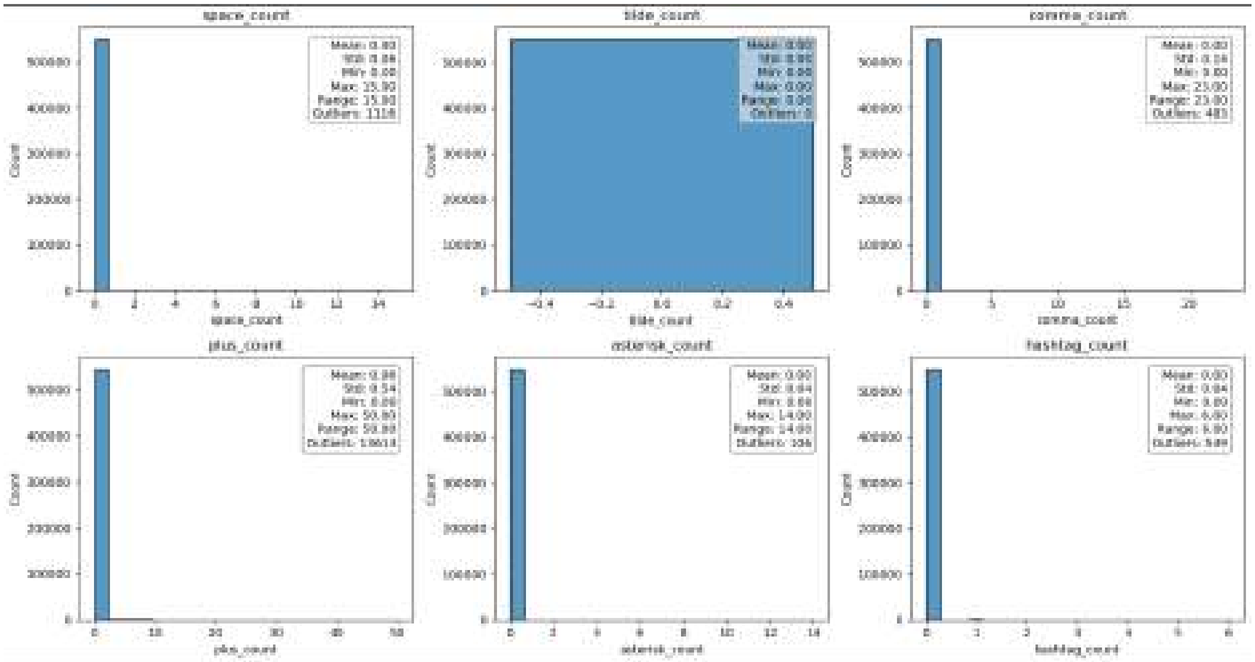
Distribution of Each Feature with its mean, standard deviation, min, max, range and outlier count

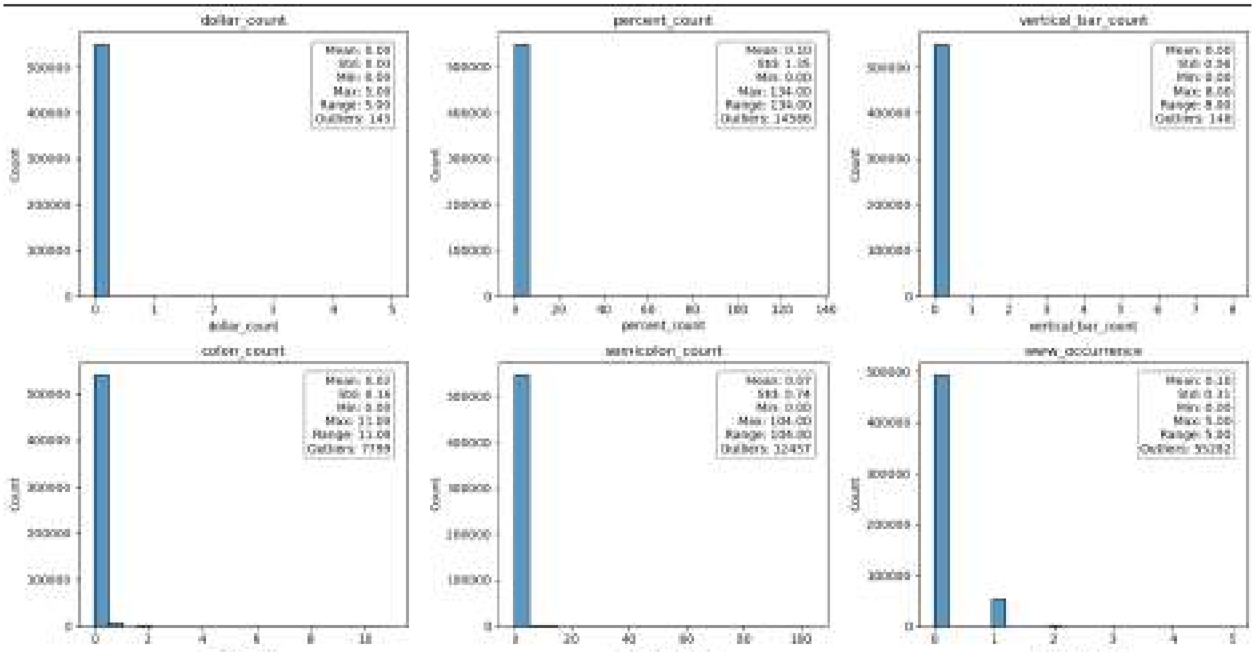


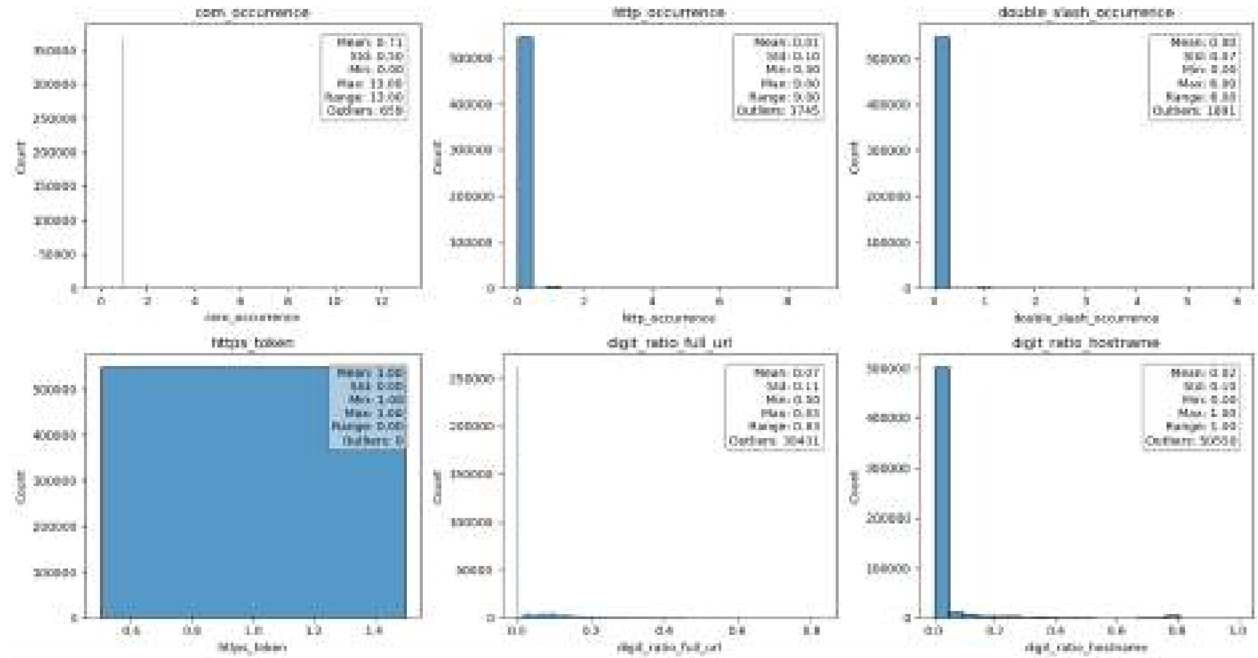
full_url_length
hostname_length
ip_address_in_url
dot_count
hyphen_count
underscore_count
Count

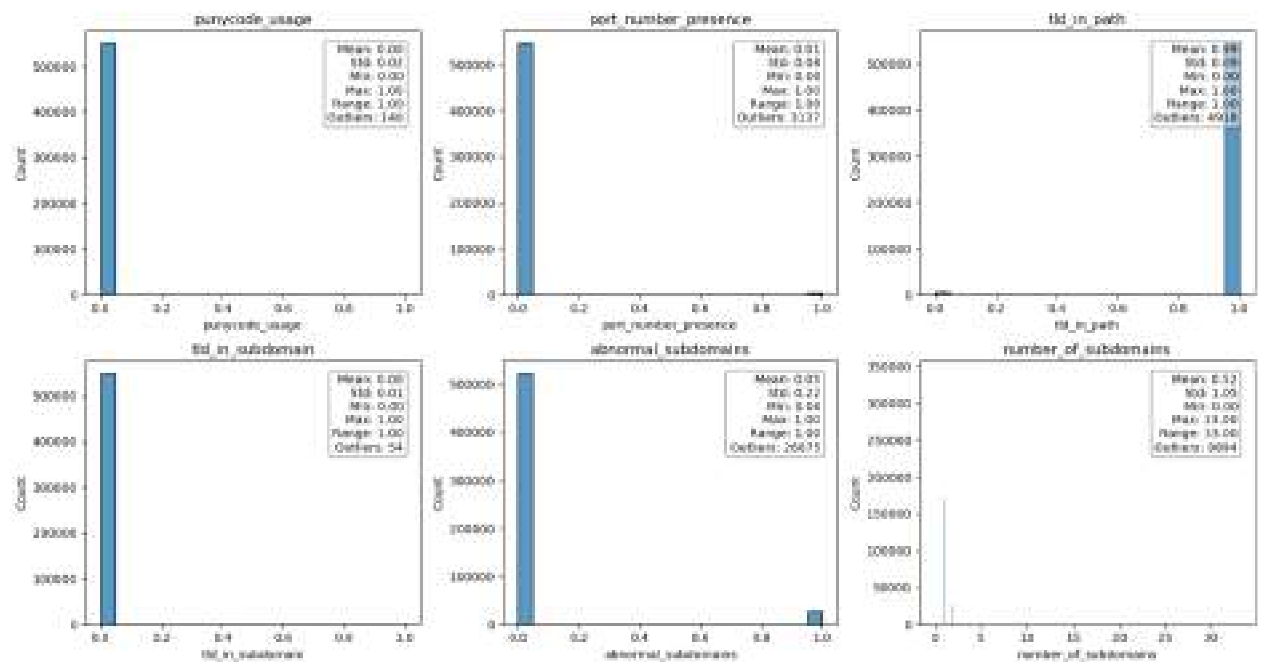


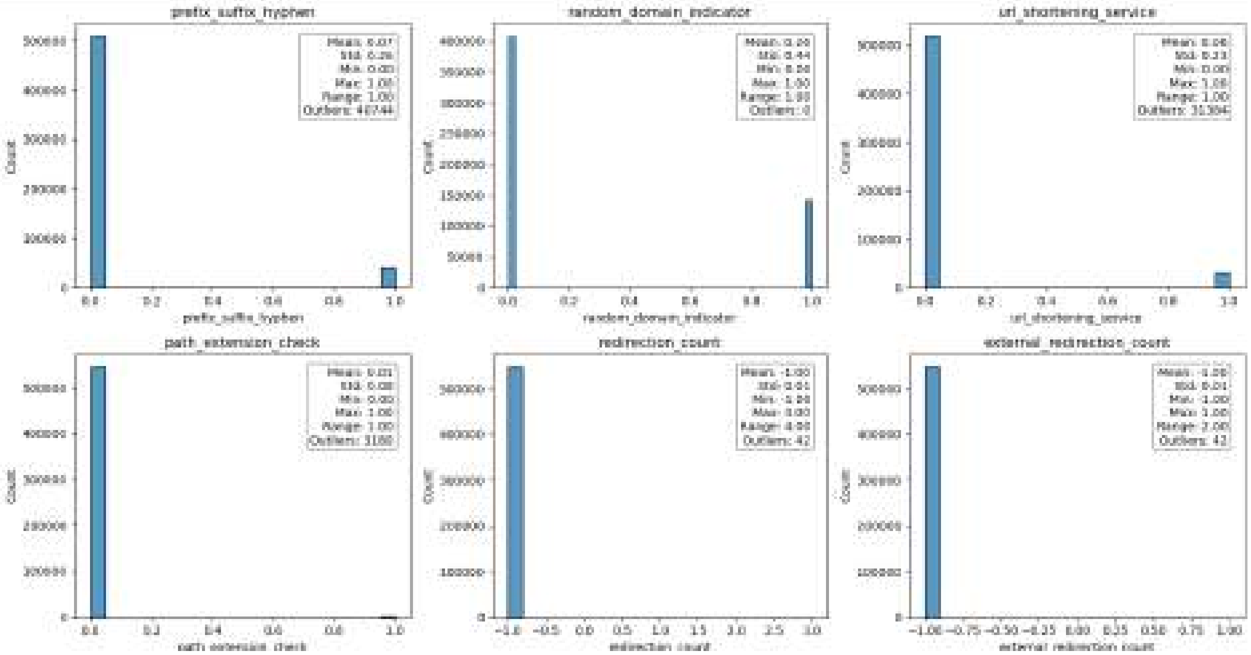
slash_count
question_mark_count
equal_count
at_count
ampersand_count
exclamation_count
Count

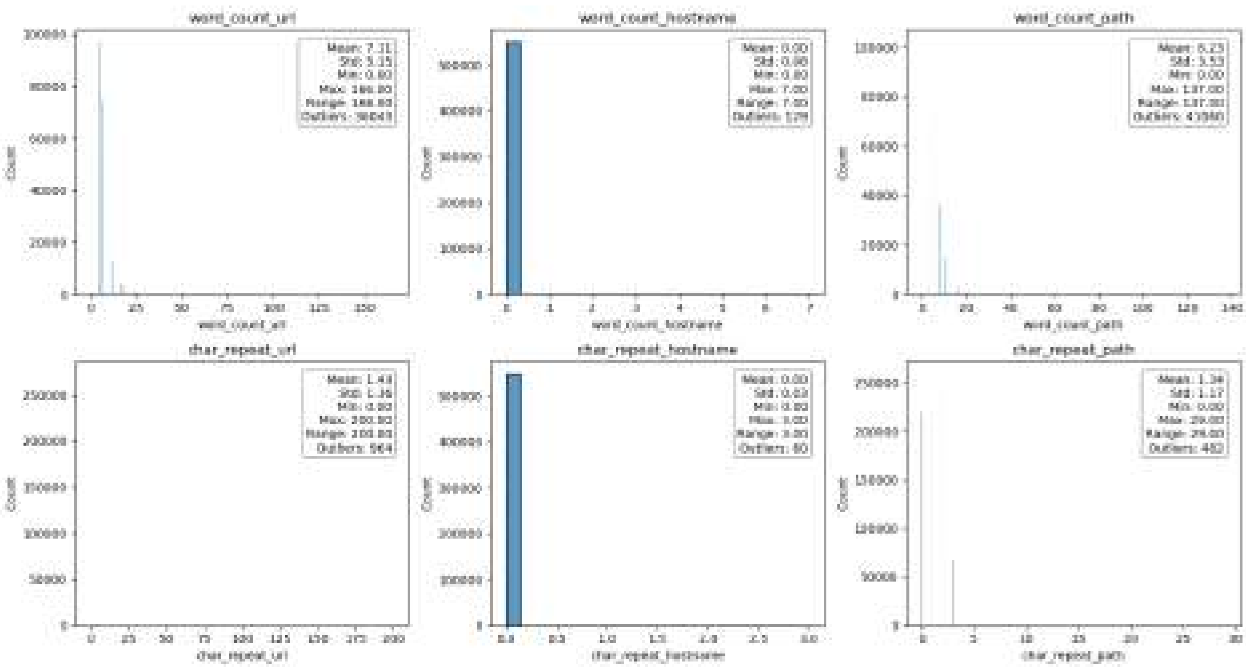

space_count
tilde_count
comma_count
plus_count
asterisk_count
hashtag_count
Count

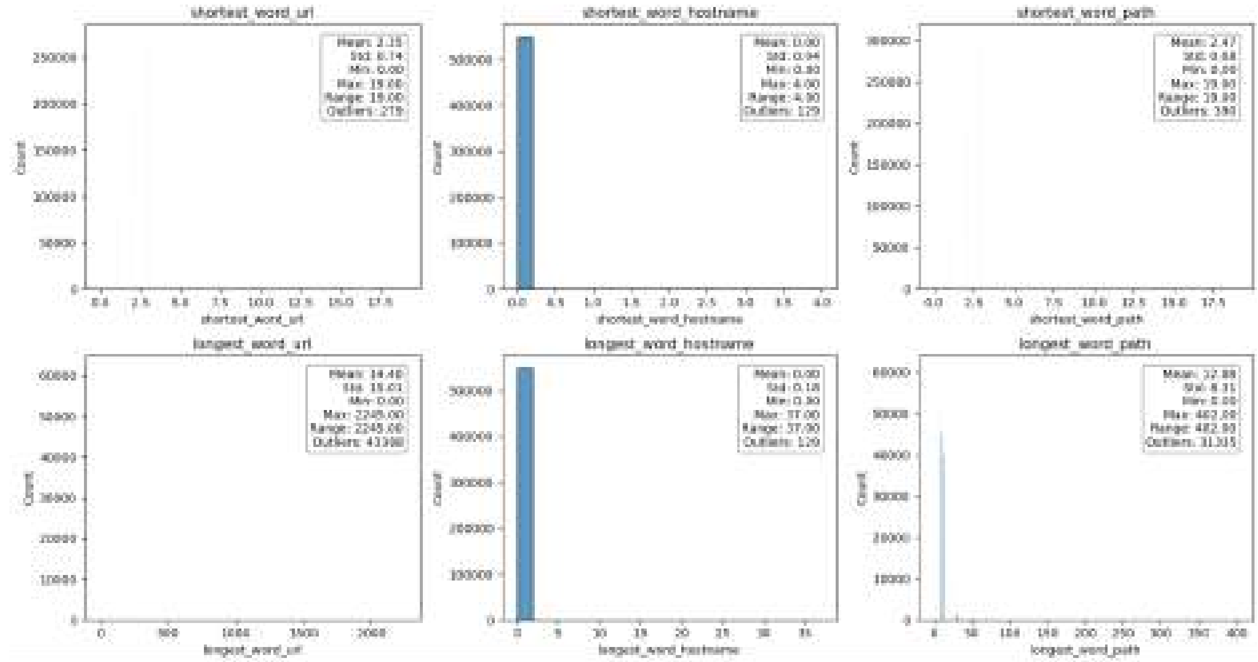

dollar_count
percent_count
vertical_bar_count
colon_count
semicolon_count
www_occurrence
Count

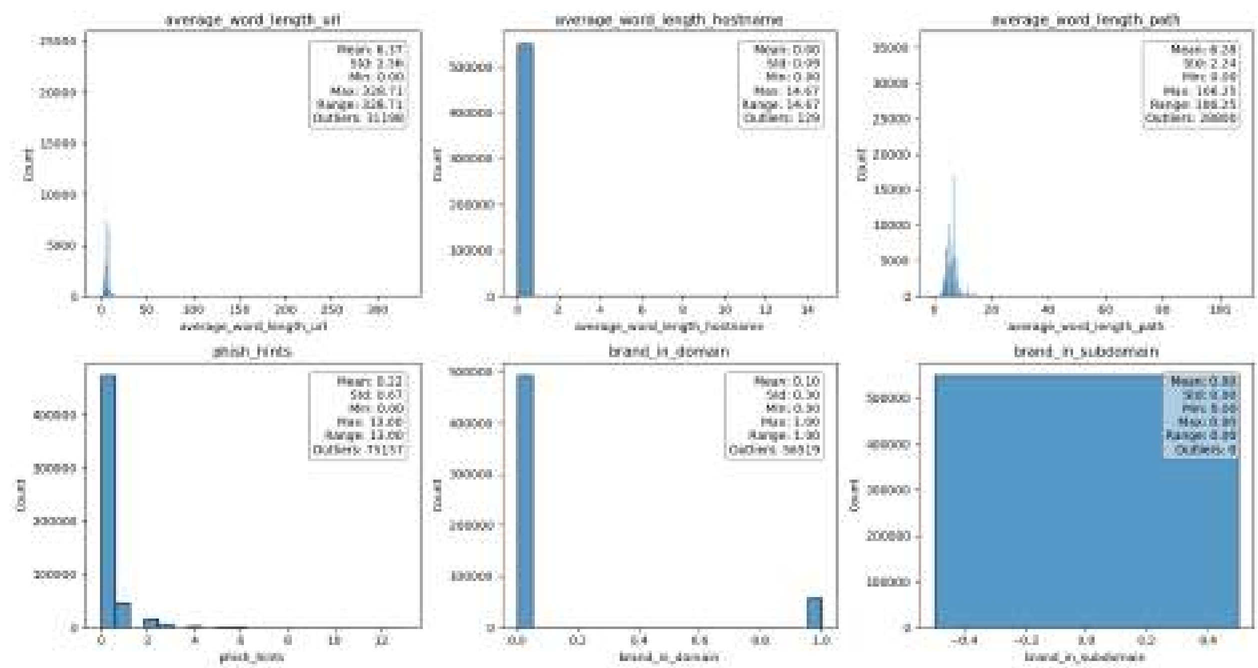

com_occurrence
http_occurrence
double_slash_occurrence
https_token
digit_ratio_full_url
digit_ratio_hostname

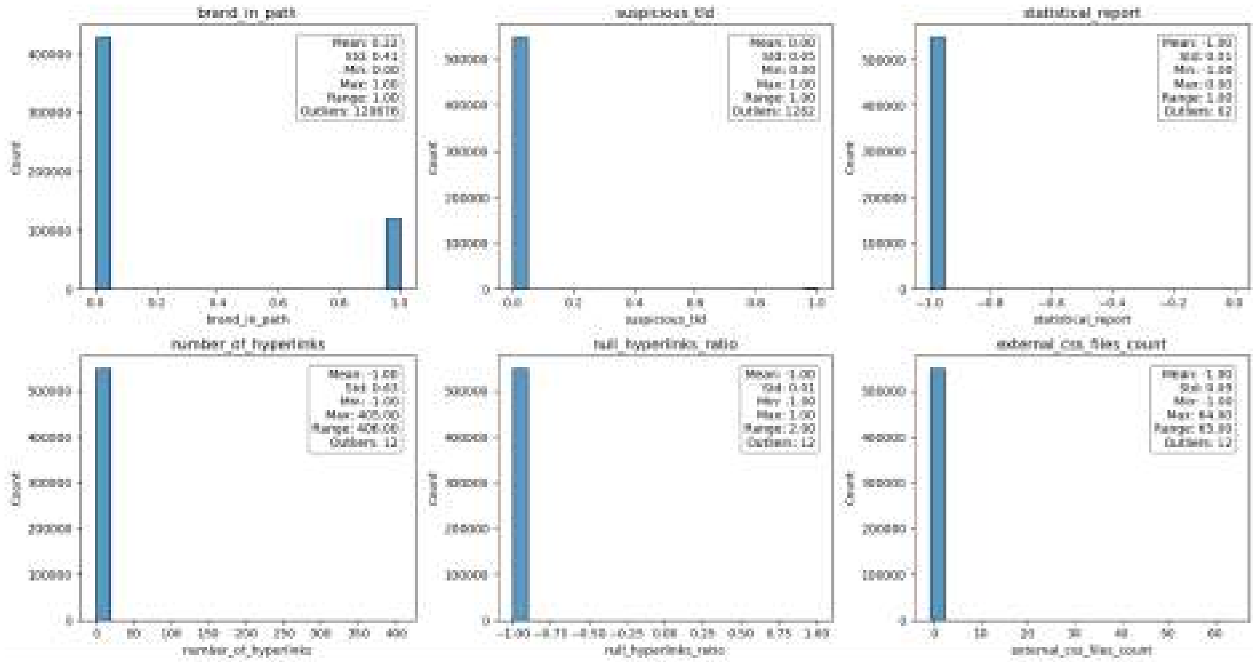

punycode_usage
port_number_presence
tld_in_path
tld_in_subdomain
abnormal_subdomains
number_of_subdomains

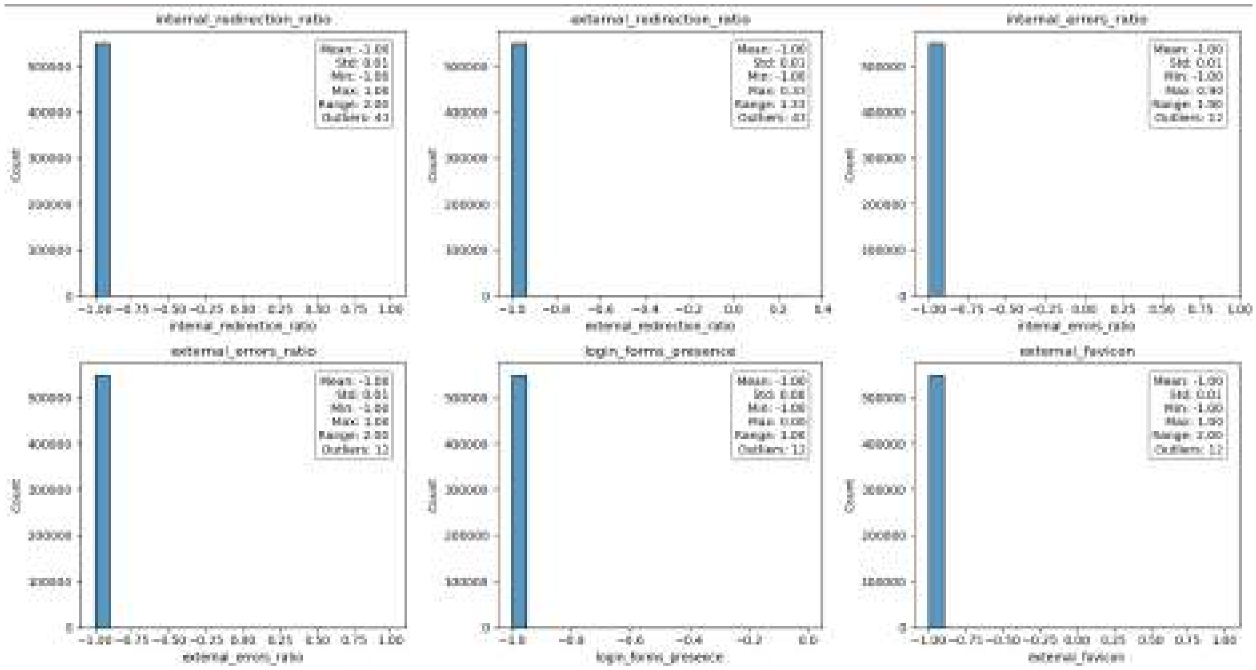

prefix_suffix_hyphen
random_domain_indicator
url_shortening_service
path_extension_check
redirection_count
external_redirection_count

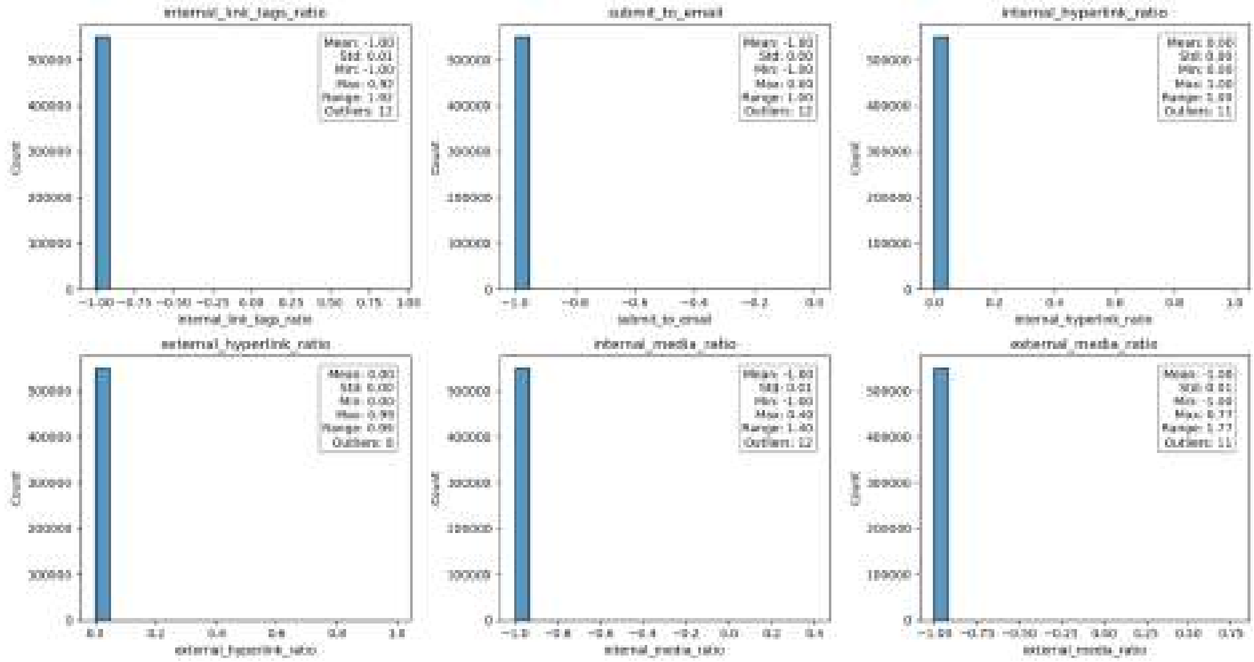

word_count_url
word_count_hostname
word_count_path
char_repeat_url
char_repeat_hostname
char_repeat_path

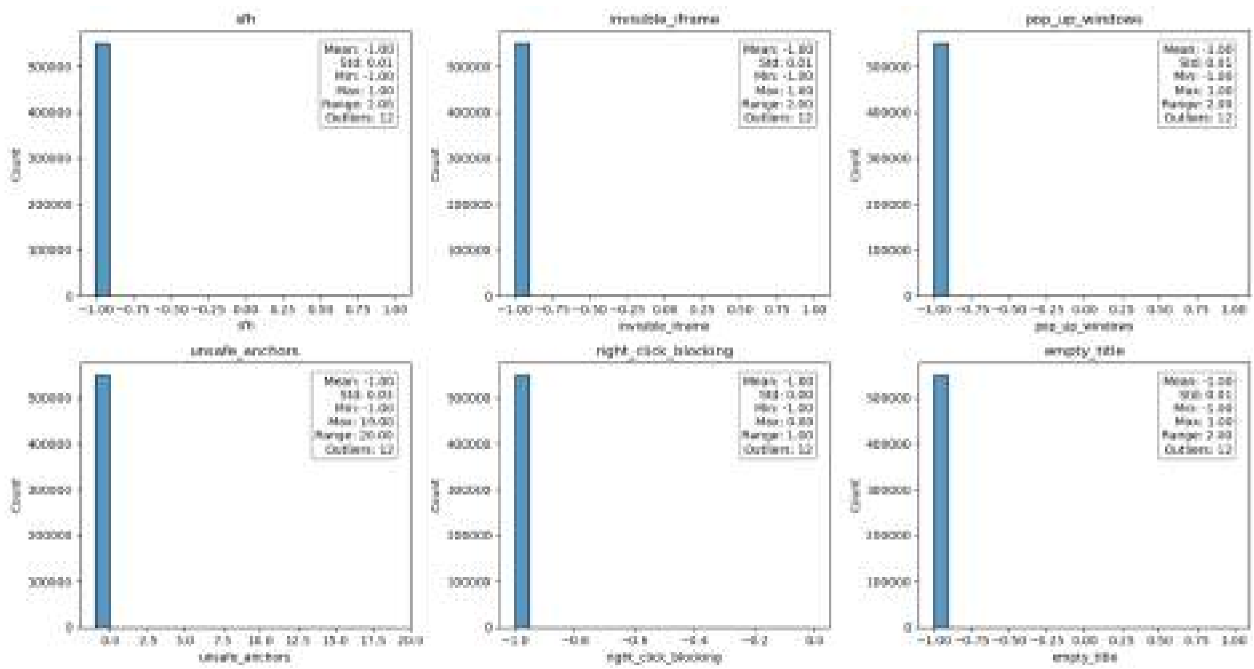

shortest_word_url
shortest_word_hostname
shortest_word_path
longest_word_url
longest_word_hostname
longest_word_path

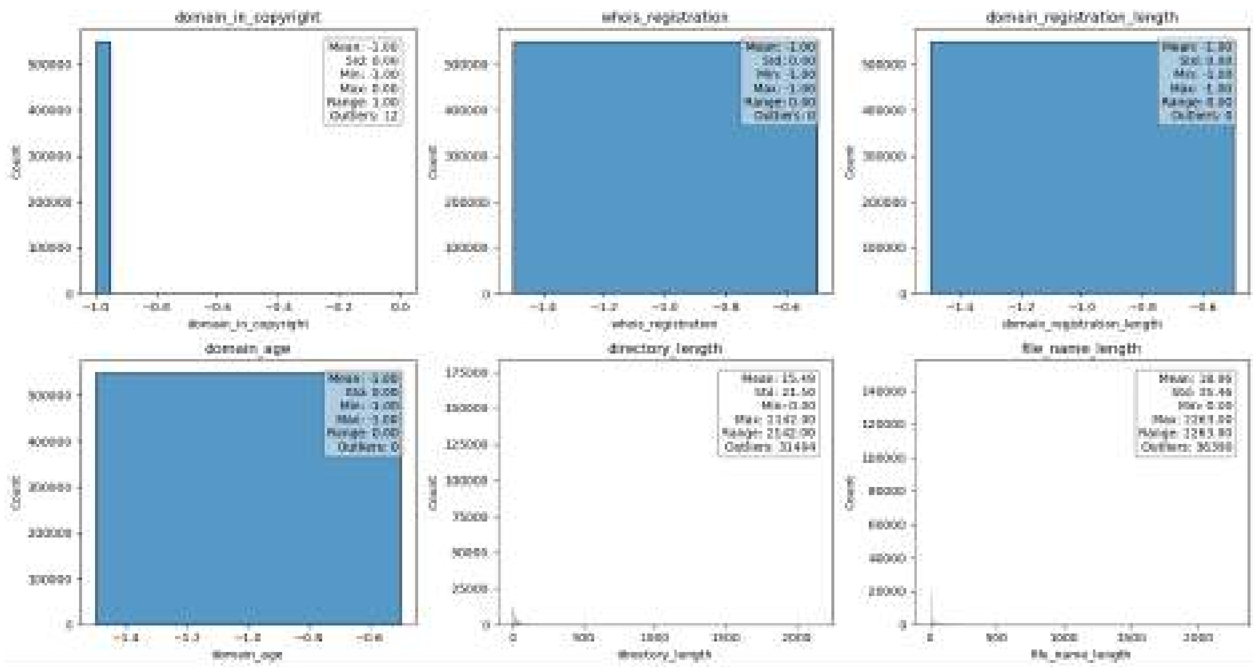

average_word_length_url
average_word_length_hostname
average_word_length_path
phish_hints
brand_in_domain
brand_in_subdomain

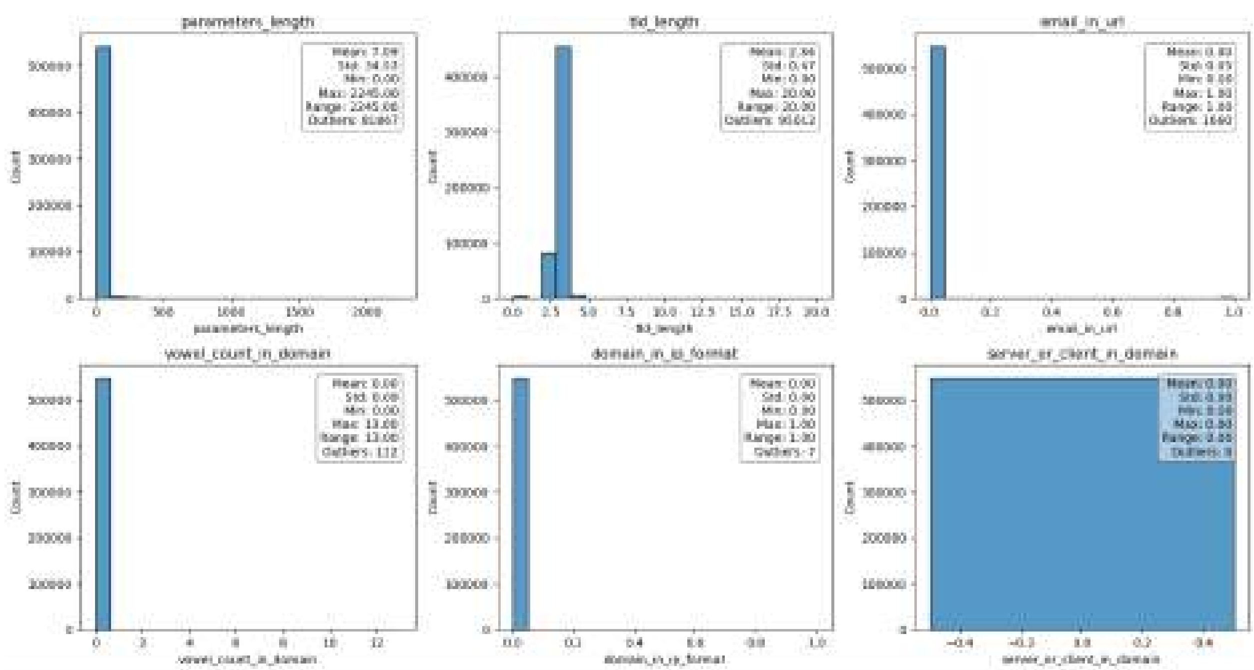

brand_in_path
suspicious_tld
statistical_report
number_of_hyperlinks
null_hyperlinks_ratio
external_css_files_count
Count

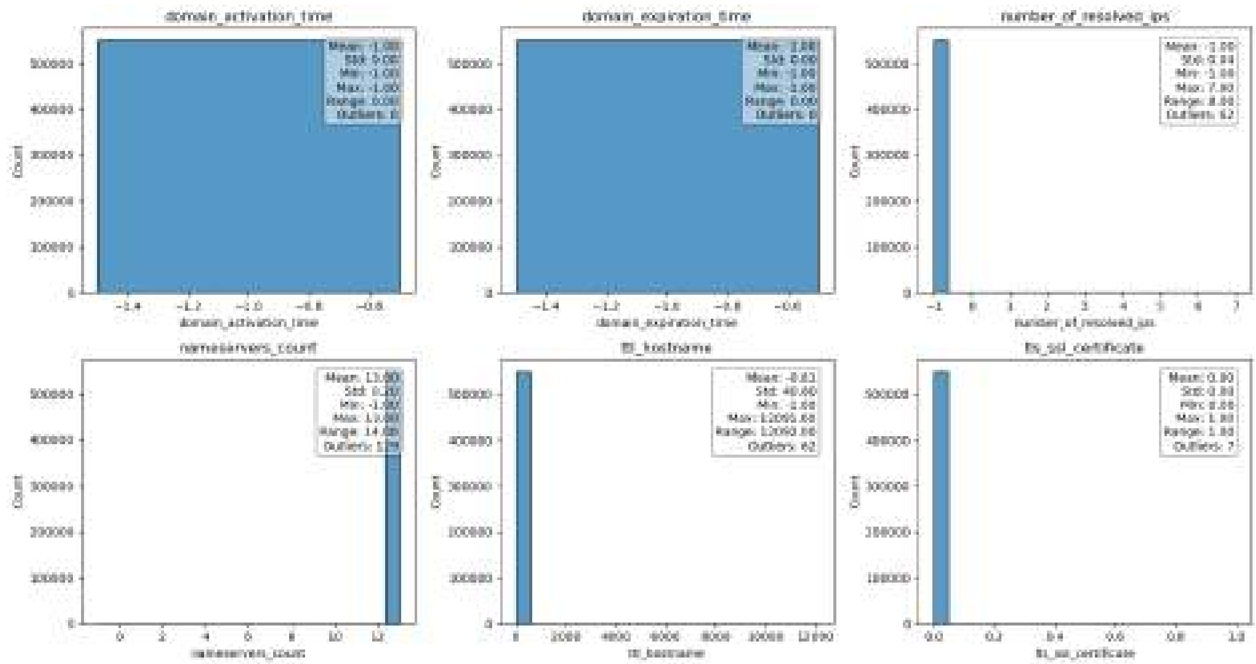

internal_redirection_ratio
external_redirection_ratio
internal_errors_ratio
external_errors_ratio
login_forms_presence
external_favicon
Count

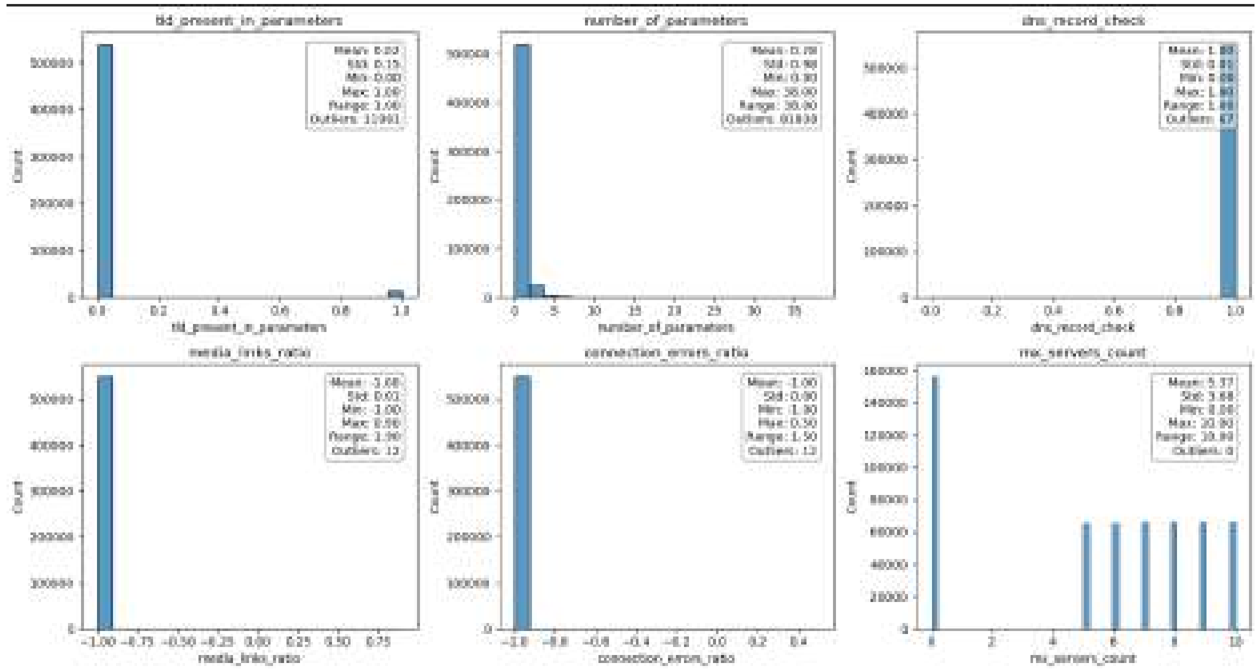

internal_link_tags_ratio
submit_to_email
internal_hyperlink_ratio
external_hyperlink_ratio
internal_media_ratio
external_media_ratio


sfh
invisible_iframe
pop_up_windows
unsafe_anchors
right_click_blocking
empty_title


domain_in_copyright
whois_registration
domain_registration_length
domain_age
directory_length
file_name_length

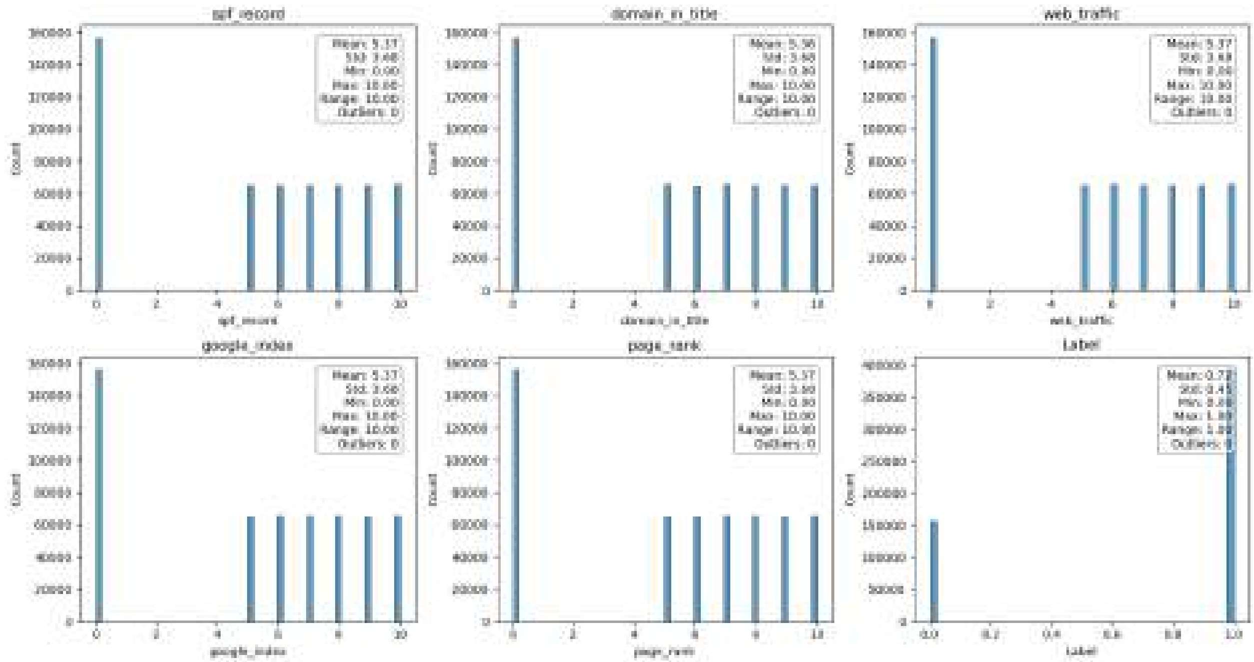

parameters_length
tld_length
email_in_url
vowel_count_in_domain
domain_in_ip_format
server_or_client_in_domain





We will z score normalize only those features which dont have gaussian distribution and have a huge range.

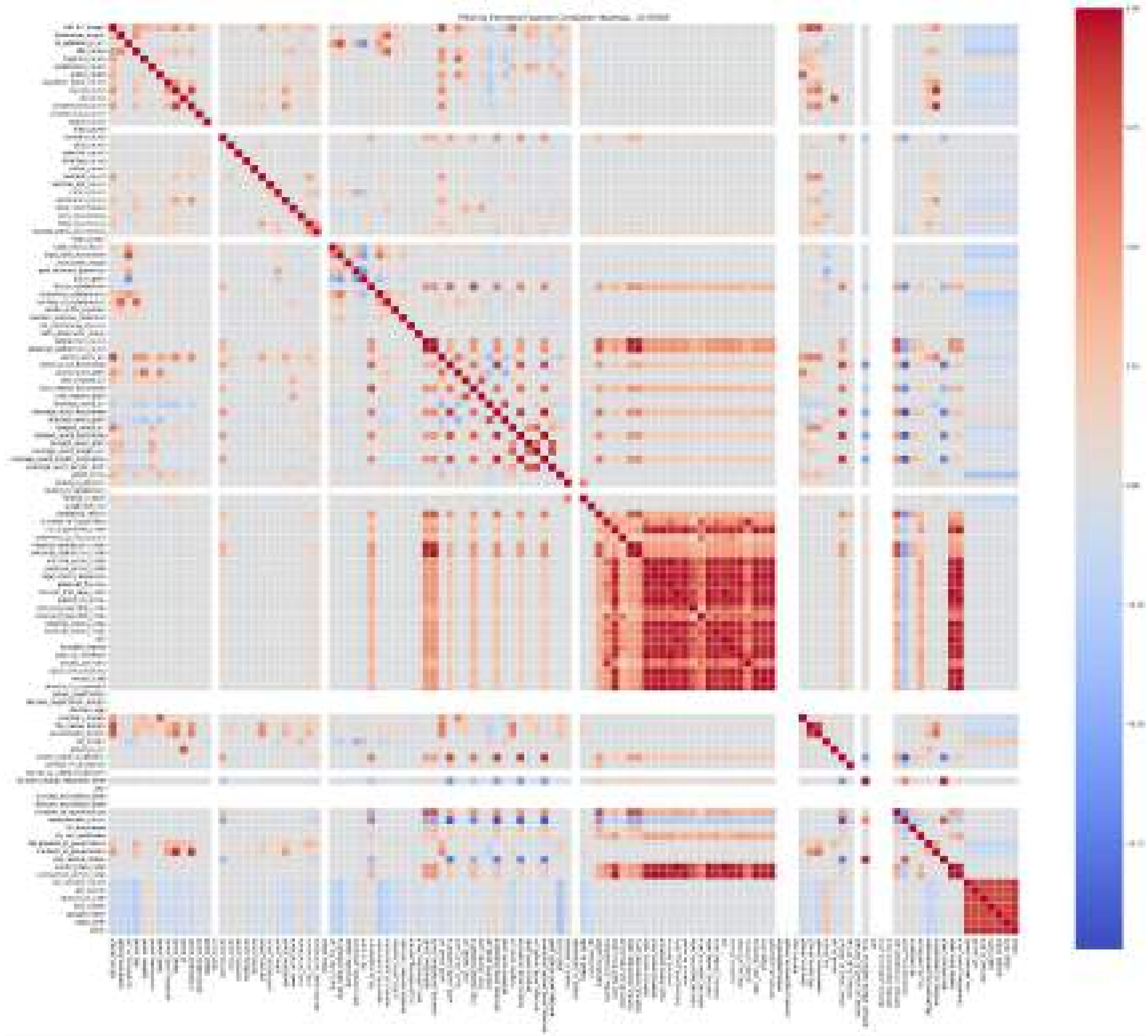
We will drop duplicate columns and highly correlated features.



Drop Constant Columns

```
Column Index: 14, Column Name: tilde_count
Column Index: 28, Column Name: https_token
Column Index: 60, Column Name: brand_in_subdomain
Column Index: 86, Column Name: whois_registration
Column Index: 87, Column Name: domain_registration_length
Column Index: 88, Column Name: domain_age
Column Index: 96, Column Name: server_or_client_in_domain
Column Index: 98, Column Name: asn
Column Index: 99, Column Name: domain_activation_time
Column Index: 100, Column Name: domain_expiration_time
```

Correlation Map



Please download the jpeg image from moodle and zoom into it for better view.

Drop highly correlated Columns with corr > 0.9

```
Removed highly correlated features: ['external_redirection_count', 'average_word_length_hostname',
'internal_redirection_ratio', 'external_redirection_ratio', 'external_errors_ratio', 'login_forms_
presence', 'external_favicon', 'internal_link_tags_ratio', 'submit_to_email', 'internal_media_rati
o', 'external_media_ratio', 'sfh', 'invisible_iframe', 'pop_up_windows', 'right_click_blocking',
'empty_title', 'domain_in_copyright', 'nameservers_count', 'dns_record_check', 'media_links_rati
o', 'connection_errors_ratio']
```

After all this we remain with 89 features out of 98 features

PCA

```
Top 10 Principal Components (Explained Variance Ratio):
PC1: 0.1089
PC2: 0.0959
PC3: 0.0668
PC4: 0.0464
PC5: 0.0459
PC6: 0.0435
PC7: 0.0386
PC8: 0.0268
PC9: 0.0241
PC10: 0.0228
```

```
PCA Component Loadings:
      full_url_length  hostname_length  ip_address_in_url  dot_count  \
PC1          0.247692         0.103775           0.049602   0.198385
PC2         -0.109453        -0.052860          -0.023370  -0.085126
PC3          0.185972        -0.028031          -0.216902  -0.008209
PC4          0.027810         0.073819          -0.095557  -0.032069
PC5         -0.056979        -0.129082           0.143479   0.022913
PC6          0.111352         0.198757           0.077052   0.099592
PC7          0.051955         0.069416           0.295438  -0.004625
PC8         -0.077486         0.345482          -0.059131   0.325239
PC9         -0.030161        -0.142746           0.041401   0.034367
PC10        -0.030644         0.034682           0.065906   0.012703

      hyphen_count  underscore_count  slash_count  question_mark_count  \
PC1       0.053982          0.055367     0.109112             0.125116
PC2      -0.027147         -0.026107    -0.038897            -0.052932
PC3       0.132240          0.104907     0.061049             0.085944
PC4      -0.104780          0.093604    -0.068713            -0.017152
PC5       0.080346         -0.139764     0.014004             0.059709
PC6       0.245754          0.058420     0.204253            -0.165554
PC7      -0.095005          0.140114    -0.247072             0.009327
PC8      -0.183324         -0.128272    -0.187942             0.085264
PC9      -0.108122          0.071843     0.140974            -0.006425
PC10      0.020416         -0.010116     0.084607             0.038613
```

	equal_count	at_count	...	ttl_hostname	tls_ssl_certificate
PC1	0.197604	0.048309	...	0.036451	0.056890
PC2	-0.081853	-0.021897	...	0.088096	0.139395
PC3	0.157023	0.012434	...	-0.003095	0.004659
PC4	-0.010582	-0.006330	...	-0.046775	0.142909
PC5	0.086981	0.020723	...	-0.040536	0.106918
PC6	-0.228910	-0.081474	...	-0.009754	0.022034
PC7	0.076882	-0.052768	...	0.002548	-0.000696
PC8	0.055638	0.056128	...	-0.003653	-0.008655
PC9	-0.062346	0.119979	...	-0.013399	-0.001424
PC10	0.002448	0.553468	...	-0.067540	0.013606

	tld_present_in_parameters	number_of_parameters	mx_servers_count
PC1	0.143674	0.197315	-0.170188
PC2	-0.049548	-0.082995	0.075657
PC3	0.054700	0.159386	0.262541
PC4	-0.035238	-0.013145	-0.086078
PC5	0.065904	0.095420	0.082835
PC6	-0.158137	-0.244641	0.085688
PC7	-0.085508	0.077037	0.143493
PC8	0.045294	0.066588	0.047232
PC9	0.238241	-0.057619	0.026281
PC10	0.164861	0.025750	0.035007

	spf_record	domain_in_title	web_traffic	google_index	page_rank
PC1	-0.170078	-0.170007	-0.170065	-0.169997	-0.170139
PC2	0.075605	0.075563	0.075601	0.075565	0.075626
PC3	0.262623	0.262590	0.262548	0.262651	0.262499
PC4	-0.086044	-0.086056	-0.085922	-0.085852	-0.086127
PC5	0.082812	0.082885	0.082649	0.082582	0.082916
PC6	0.085564	0.085562	0.085626	0.085528	0.085766
PC7	0.143229	0.143530	0.143649	0.143449	0.143319
PC8	0.046537	0.046687	0.046354	0.047282	0.046762
PC9	0.025746	0.025968	0.025977	0.026226	0.025762
PC10	0.035467	0.035071	0.035497	0.035249	0.035208

```
--- Training models using features from PCA ---

SVC using PCA features with linear kernel:
Accuracy: 93.74232817478988%
MCC: 0.9981
ROC AUC: 0.9999
Confusion Matrix:
[[10396     7]
 [   12  9585]]
Model saved as svm_PCA_linear.pkl

SVC using PCA features with poly kernel:
Accuracy: 95.56419745290249%
MCC: 0.9975
ROC AUC: 0.9998
Confusion Matrix:
[[10400     3]
 [   22  9575]]
Model saved as svm_PCA_poly.pkl

SVC using PCA features with rbf kernel:
Accuracy: 94.70038757795905%
MCC: 0.9992
ROC AUC: 1.0
Confusion Matrix:
[[10400     3]
 [    5  9592]]
Model saved as svm_PCA_rbf.pkl
```

```
SVC using PCA features with sigmoid kernel:
Accuracy: 96.78%
MCC: 0.9355
ROC AUC: 0.9867
Confusion Matrix:
[[10048   355]
 [  289  9308]]
Model saved as svm_PCA_sigmoid.pkl
```

Top 30 Features by Mutual Information FS Method

```
Top 30 features by Mutual Information:
web_traffic                  0.618173
mx_servers_count             0.618035
domain_in_title              0.617939
google_index                 0.617926
page_rank                    0.617847
spf_record                   0.617701
phish_hints                  0.098016
tld_length                   0.094013
digit_ratio_full_url         0.085915
com_occurrence               0.071726
digit_ratio_hostname         0.064054
file_name_length             0.061604
dot_count                    0.060891
brand_in_path                0.053499
shortest_word_path           0.053122
average_word_length_url      0.049774
abnormal_subdomains          0.048719
random_domain_indicator      0.046015
parameters_length            0.044766
shortest_word_url            0.042221
longest_word_url             0.041068
full_url_length              0.038074
average_word_length_path     0.036488
directory_length             0.036386
question_mark_count          0.035869
number_of_subdomains         0.035799
slash_count                  0.035339
char_repeat_path             0.034268
prefix_suffix_hyphen         0.033567
equal_count                  0.031775
dtype: float64
```

Normal SVM

```
--- Training models using features from MutualInformation ---

SVC using MutualInformation features with linear kernel:
Accuracy: 96.39244992304056%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_MutualInformation_linear.pkl

SVC using MutualInformation features with poly kernel:
Accuracy: 95.04711605381122%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_MutualInformation_poly.pkl

SVC using MutualInformation features with rbf kernel:
Accuracy: 95.1457240688026%
MCC: 0.9996
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    4  9593]]
Model saved as svm_MutualInformation_rbf.pkl

SVC using MutualInformation features with sigmoid kernel:
Accuracy: 94.69%
MCC: 0.8937
ROC AUC: 0.9829
Confusion Matrix:
[[9842  561]
 [ 501 9096]]
Model saved as svm_MutualInformation_sigmoid.pkl
```

One Class SVM

```
One-Class SVM using MutualInformation features with linear kernel:
Accuracy: 76.17%
MCC: 0.5871
Confusion Matrix (One-Class SVM):
[[10397     6]
 [ 4760  4837]]
One-Class SVM model saved as oneclasssvm_MutualInformation_linear.pkl

One-Class SVM using MutualInformation features with poly kernel:
Accuracy: 75.32%
MCC: 0.5637
Confusion Matrix (One-Class SVM):
[[10259   144]
 [ 4791  4806]]
One-Class SVM model saved as oneclasssvm_MutualInformation_poly.pkl

One-Class SVM using MutualInformation features with rbf kernel:
Accuracy: 75.99%
MCC: 0.5847
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4802  4795]]
One-Class SVM model saved as oneclasssvm_MutualInformation_rbf.pkl

One-Class SVM using MutualInformation features with sigmoid kernel:
Accuracy: 76.14%
MCC: 0.5871
Confusion Matrix (One-Class SVM):
[[10402     1]
 [ 4770  4827]]
One-Class SVM model saved as oneclasssvm_MutualInformation_sigmoid.pkl
```

Top 30 Features By RFE FS Method

```
Top features by RFE:
['hostname_length', 'ip_address_in_url', 'dot_count', 'hyphen_count', 'semicolon_count', 'com_occu
rrence', 'http_occurrence', 'digit_ratio_full_url', 'digit_ratio_hostname', 'abnormal_subdomains',
'prefix_suffix_hyphen', 'path_extension_check', 'word_count_url', 'word_count_path', 'longest_word
_url', 'phish_hints', 'brand_in_domain', 'brand_in_path', 'suspicious_tld', 'directory_length', 'f
ile_name_length', 'parameters_length', 'tld_length', 'tld_present_in_parameters', 'mx_servers_coun
t', 'spf_record', 'domain_in_title', 'web_traffic', 'google_index', 'page_rank']
```

Normal SVM

```
--- Training models using features from RFE ---

SVC using RFE features with linear kernel:
Accuracy: 95.37249145312066%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_RFE_linear.pkl

SVC using RFE features with poly kernel:
Accuracy: 96.98559990717254%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_RFE_poly.pkl

SVC using RFE features with rbf kernel:
Accuracy: 95.55890943974204%
MCC: 0.9997
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    3  9594]]
Model saved as svm_RFE_rbf.pkl

SVC using RFE features with sigmoid kernel:
Accuracy: 97.06%
MCC: 0.9411
ROC AUC: 0.9944
Confusion Matrix:
[[10088   315]
 [  273  9324]]
Model saved as svm_RFE_sigmoid.pkl
```

One Class SVM

```
One-Class SVM using RFE features with linear kernel:
Accuracy: 76.01%
MCC: 0.5845
Confusion Matrix (One-Class SVM):
[[10396     7]
 [ 4791  4806]]
One-Class SVM model saved as oneclasssvm_RFE_linear.pkl

One-Class SVM using RFE features with poly kernel:
Accuracy: 75.08%
MCC: 0.5573
Confusion Matrix (One-Class SVM):
[[10223   180]
 [ 4804  4793]]
One-Class SVM model saved as oneclasssvm_RFE_poly.pkl

One-Class SVM using RFE features with rbf kernel:
Accuracy: 76.03%
MCC: 0.5853
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4794  4803]]
One-Class SVM model saved as oneclasssvm_RFE_rbf.pkl

One-Class SVM using RFE features with sigmoid kernel:
Accuracy: 76.03%
MCC: 0.585
Confusion Matrix (One-Class SVM):
[[10399     4]
 [ 4790  4807]]
One-Class SVM model saved as oneclasssvm_RFE_sigmoid.pkl
```

Top 30 Features By Anova

Normal SVM

```
--- Training models using features from ANOVAFtest ---

SVC using ANOVAFtest features with linear kernel:
Accuracy: 93.60219480880879%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_ANOVAFtest_linear.pkl

SVC using ANOVAFtest features with poly kernel:
Accuracy: 96.09237261178717%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_ANOVAFtest_poly.pkl

SVC using ANOVAFtest features with rbf kernel:
Accuracy: 93.78426166986037%
MCC: 0.9996
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    4  9593]]
Model saved as svm_ANOVAFtest_rbf.pkl

SVC using ANOVAFtest features with sigmoid kernel:
Accuracy: 98.48%
MCC: 0.9695
ROC AUC: 0.9971
Confusion Matrix:
[[10241   162]
 [  143  9454]]
Model saved as svm_ANOVAFtest_sigmoid.pkl
```

One Class SVM

```
One-Class SVM using ANOVAFtest features with linear kernel:
Accuracy: 76.23%
MCC: 0.5885
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4754  4843]]
One-Class SVM model saved as oneclasssvm_ANOVAFtest_linear.pkl

One-Class SVM using ANOVAFtest features with poly kernel:
Accuracy: 75.7%
MCC: 0.5737
Confusion Matrix (One-Class SVM):
[[10316    87]
 [ 4773  4824]]
One-Class SVM model saved as oneclasssvm_ANOVAFtest_poly.pkl

One-Class SVM using ANOVAFtest features with rbf kernel:
Accuracy: 75.89%
MCC: 0.583
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4823  4774]]
One-Class SVM model saved as oneclasssvm_ANOVAFtest_rbf.pkl

One-Class SVM using ANOVAFtest features with sigmoid kernel:
Accuracy: 76.1%
MCC: 0.5864
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4780  4817]]
One-Class SVM model saved as oneclasssvm_ANOVAFtest_sigmoid.pkl
```

Top 30 Features by ExtraTreesClassifier FS Method

```
Top 30 features by ExtraTreesClassifier importance:
google_index                0.200716
domain_in_title             0.189509
spf_record                  0.184622
mx_servers_count            0.163016
web_traffic                 0.116818
page_rank                   0.098673
abnormal_subdomains         0.012135
digit_ratio_hostname        0.004866
ip_address_in_url           0.004252
brand_in_path               0.003856
tld_present_in_parameters   0.003215
phish_hints                 0.002958
tld_length                  0.001557
random_domain_indicator     0.001502
brand_in_domain             0.001496
path_extension_check        0.001200
digit_ratio_full_url        0.000917
prefix_suffix_hyphen        0.000907
tld_in_path                 0.000838
shortest_word_path          0.000743
hyphen_count                0.000681
port_number_presence        0.000626
com_occurrence              0.000583
equal_count                 0.000551
number_of_subdomains        0.000420
slash_count                 0.000366
www_occurrence              0.000333
email_in_url                0.000306
longest_word_path           0.000202
dot_count                   0.000200
dtype: float64
```

Normal SVM

```
--- Training models using features from ExtraTrees ---

SVC using ExtraTrees features with linear kernel:
Accuracy: 93.659460649639%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_ExtraTrees_linear.pkl

SVC using ExtraTrees features with poly kernel:
Accuracy: 96.89735029476759%
MCC: 1.0
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    0  9597]]
Model saved as svm_ExtraTrees_poly.pkl

SVC using ExtraTrees features with rbf kernel:
Accuracy: 94.24821921129515%
MCC: 0.9997
ROC AUC: 1.0
Confusion Matrix:
[[10403     0]
 [    3  9594]]
Model saved as svm_ExtraTrees_rbf.pkl

SVC using ExtraTrees features with sigmoid kernel:
Accuracy: 94.51%
MCC: 0.8901
ROC AUC: 0.9832
Confusion Matrix:
[[9819  584]
 [ 514 9083]]
Model saved as svm_ExtraTrees_sigmoid.pkl
```

One Class SVM

```
One-Class SVM using ExtraTrees features with linear kernel:
Accuracy: 76.3%
MCC: 0.5892
Confusion Matrix (One-Class SVM):
[[10398     5]
 [ 4736  4861]]
One-Class SVM model saved as oneclasssvm_ExtraTrees_linear.pkl

One-Class SVM using ExtraTrees features with poly kernel:
Accuracy: 75.84%
MCC: 0.575
Confusion Matrix (One-Class SVM):
[[10300   103]
 [ 4728  4869]]
One-Class SVM model saved as oneclasssvm_ExtraTrees_poly.pkl

One-Class SVM using ExtraTrees features with rbf kernel:
Accuracy: 76.21%
MCC: 0.5882
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4758  4839]]
One-Class SVM model saved as oneclasssvm_ExtraTrees_rbf.pkl

One-Class SVM using ExtraTrees features with sigmoid kernel:
Accuracy: 76.25%
MCC: 0.5888
Confusion Matrix (One-Class SVM):
[[10403     0]
 [ 4750  4847]]
One-Class SVM model saved as oneclasssvm_ExtraTrees_sigmoid.pkl
```

Remove Redundant Features and Reduce Dimensionality using Autoencoders and Train a SVM Model on different kernels.

Drop Repeated Columns

```
Column Index: 14, Column Name: tilde_count
Column Index: 28, Column Name: https_token
Column Index: 60, Column Name: brand_in_subdomain
Column Index: 86, Column Name: whois_registration
Column Index: 87, Column Name: domain_registration_length
Column Index: 88, Column Name: domain_age
Column Index: 96, Column Name: server_or_client_in_domain
Column Index: 98, Column Name: asn
Column Index: 99, Column Name: domain_activation_time
Column Index: 100, Column Name: domain_expiration_time
```

Drop highly correlated Columns with corr > 0.9

connection_errors_ratio,
'internal_link_tags_ratio',
'sfh',
'Nameservers_count',
'Pop_up_windows',
'internal_redirection_ratio',
'External_favicon',
'Internal_media_ratio',
'External_errors_ratio',
'External_redirection_count',
'dns_record_check',
'right_click_blocking',
'External_redirection_ratio',
'internal_errors_ratio',
'Domain_in_copyright',
'Average_word_length_hostname',
'number_of_parameters',
'Vowel_count_in_domain',
'unsafe_anchors',
'Media_links_ratio',
'login_forms_presence',
'Empty_title',
'Invisible_iframe',
'Submit_to_email',
'longest_word_hostname',
'external_media_ratio'

After this 89 features remain out of 98

Autoencoders to reduce Dimensionality to 15

Simple 2 layered architecture

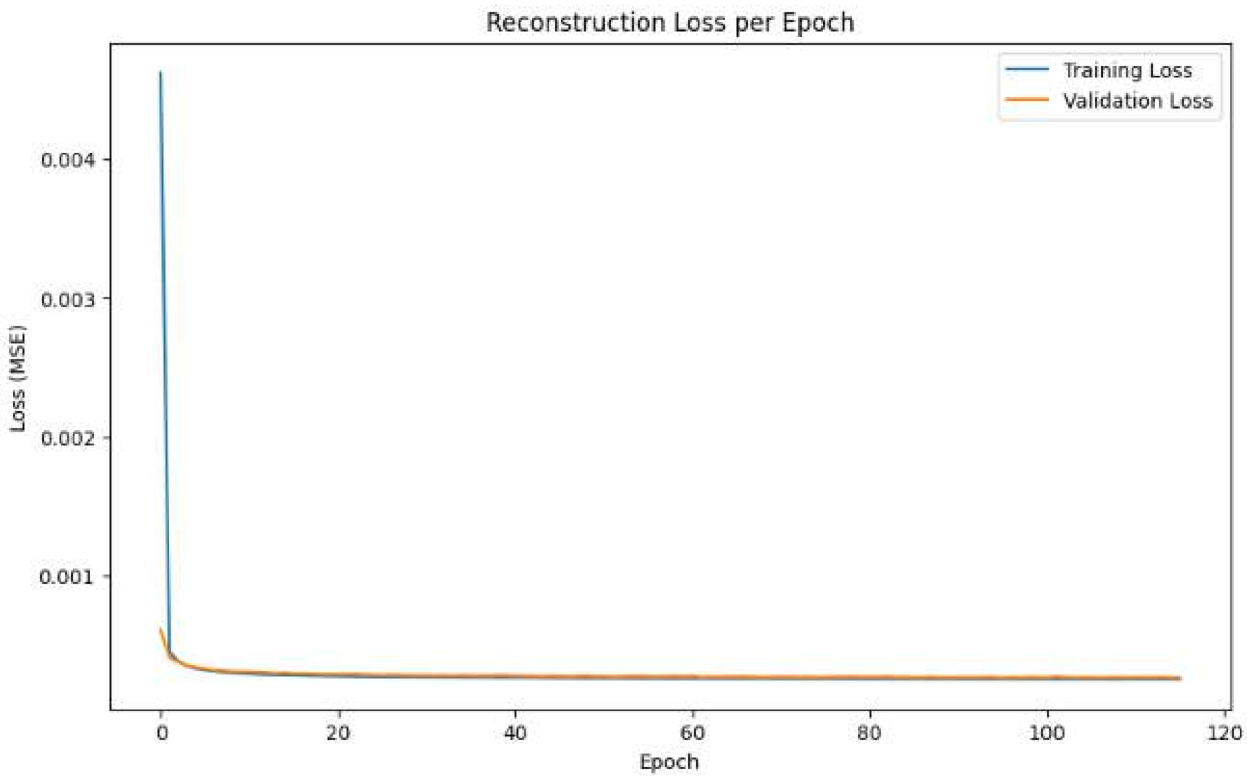
```
Model: "model_28"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_15 (InputLayer)       [(None, 97)]              0

 dense_28 (Dense)            (None, 15)                1470

 dense_29 (Dense)            (None, 97)                1552

=================================================================
Total params: 3,022
Trainable params: 3,022
Non-trainable params: 0
_________________________________________________________________
```

We have used a simple architecture for now due to the time it takes to train. Depending on how it performs we will increase the depth or keep it as it is.

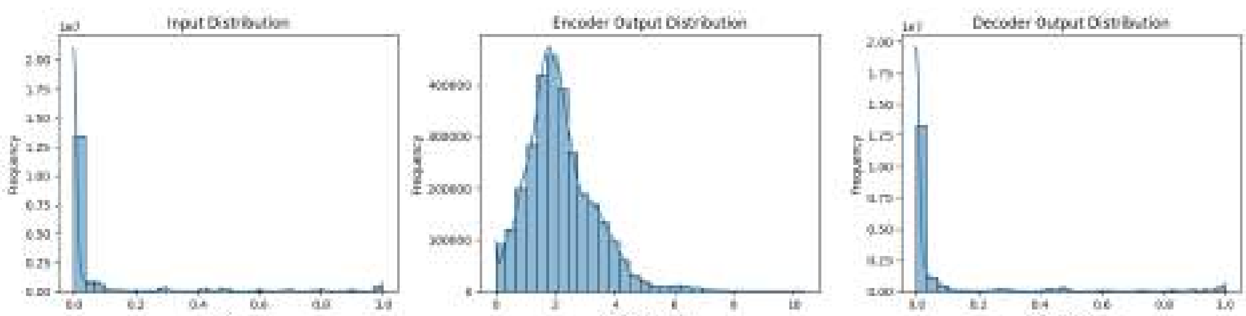
Trained till 128 epochs with validation loss of 0.0002

```
Epoch 122: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 123: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 124: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 125: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 126: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 127: Training Loss = 0.0001, Validation Loss = 0.0002
Epoch 128: Training Loss = 0.0001, Validation Loss = 0.0002
```

Final Data has 15 features extracted

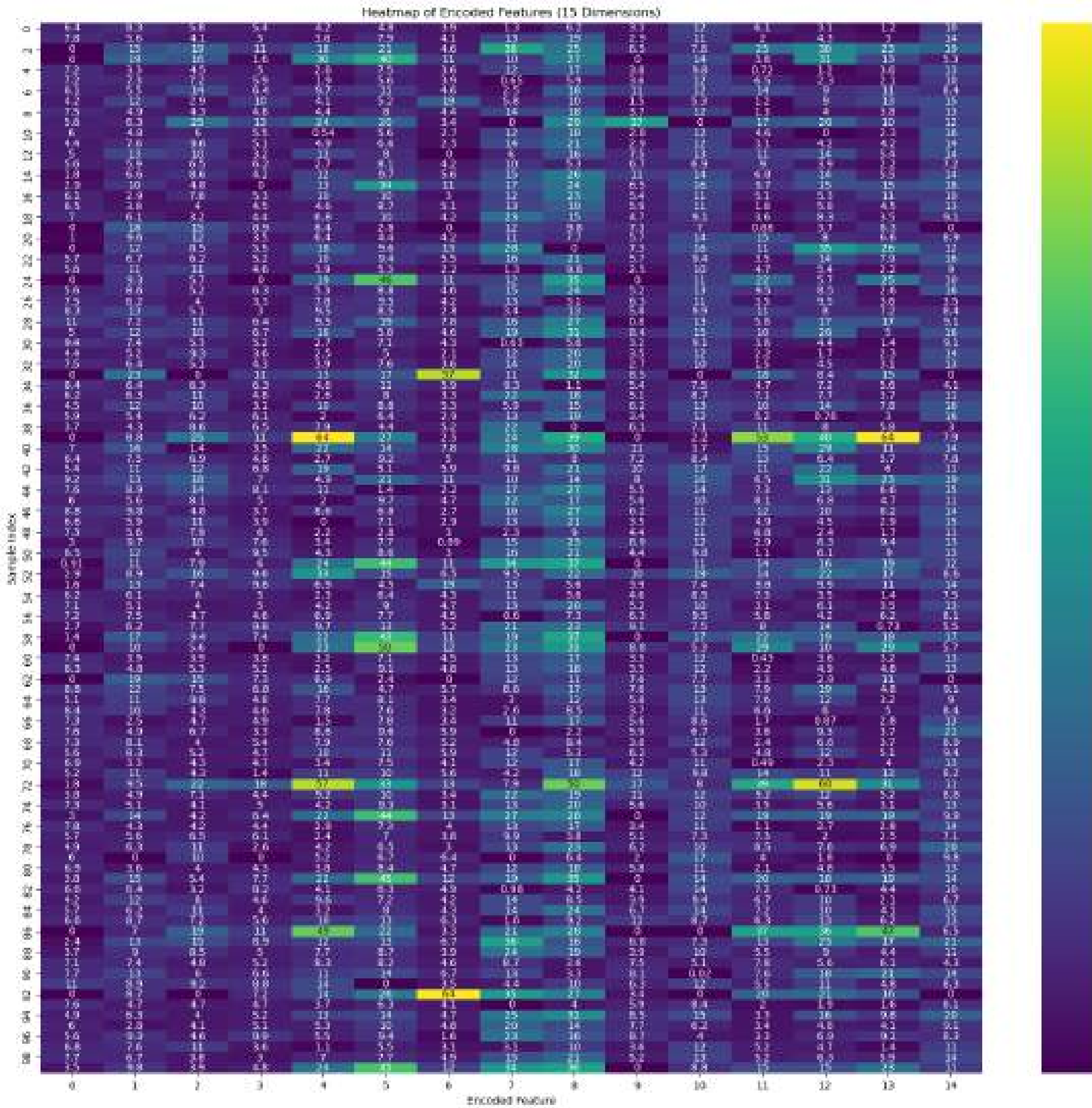


Plots of input data to each layer



Weights of Extracted Features for the first 100 samples

Instead of directly mentioning numerical weights , I have color coded them for better understanding.


Heatmap of Encoded Features (15 Dimensions)
Sample Index
Encoded Feature

SVM Model trained on those 15 extracted features using 4 kernels

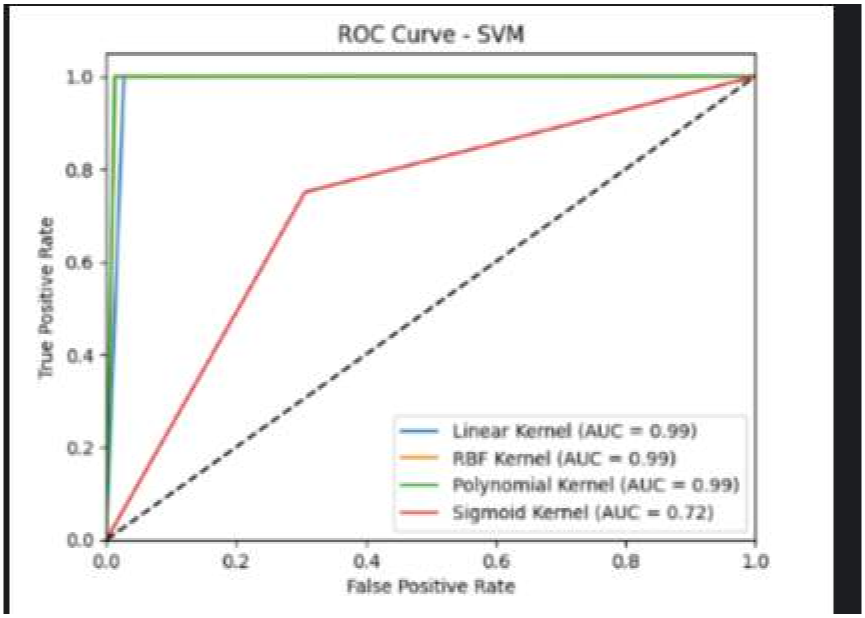
Data Splitting First: The train-test split is performed using the original features, ensuring the autoencoder is not influenced by the test data.
Training Autoencoder on Training Data Only: This prevents any leakage from the test set into the autoencoder, leading to a more realistic evaluation.
Consistent Transformation: Both the training and test sets are transformed using the same trained encoder, ensuring consistent feature representation for the SVM.

```
Training SVM with kernel: linear
SVM training time with linear kernel: 3.4347 seconds
For kernel 'linear': Misclassified 456 out of 100000 samples
Metrics for kernel 'linear':
  Accuracy: 97.3
  Precision: 98.3
  Recall: 98.67
  F1-Score: 98.9
  MCC: 0.95
  Confusion Matrix:
[[93607    60]
 [   96 72739]]
  Average Prediction Time per sample: 0.1084 ms
```

```
Training SVM with kernel: poly
SVM training time with poly kernel: 1.7084 seconds
For kernel 'poly': Misclassified 789 out of 100000 samples
Metrics for kernel 'poly':
  Accuracy: 96.5
  Precision: 97.5
  Recall: 99.34
  F1-Score: 97.8
  MCC: 0.93
  Confusion Matrix:
[[79871   277]
 [  268 72750]]
  Average Prediction Time per sample: 0.1098 ms
```

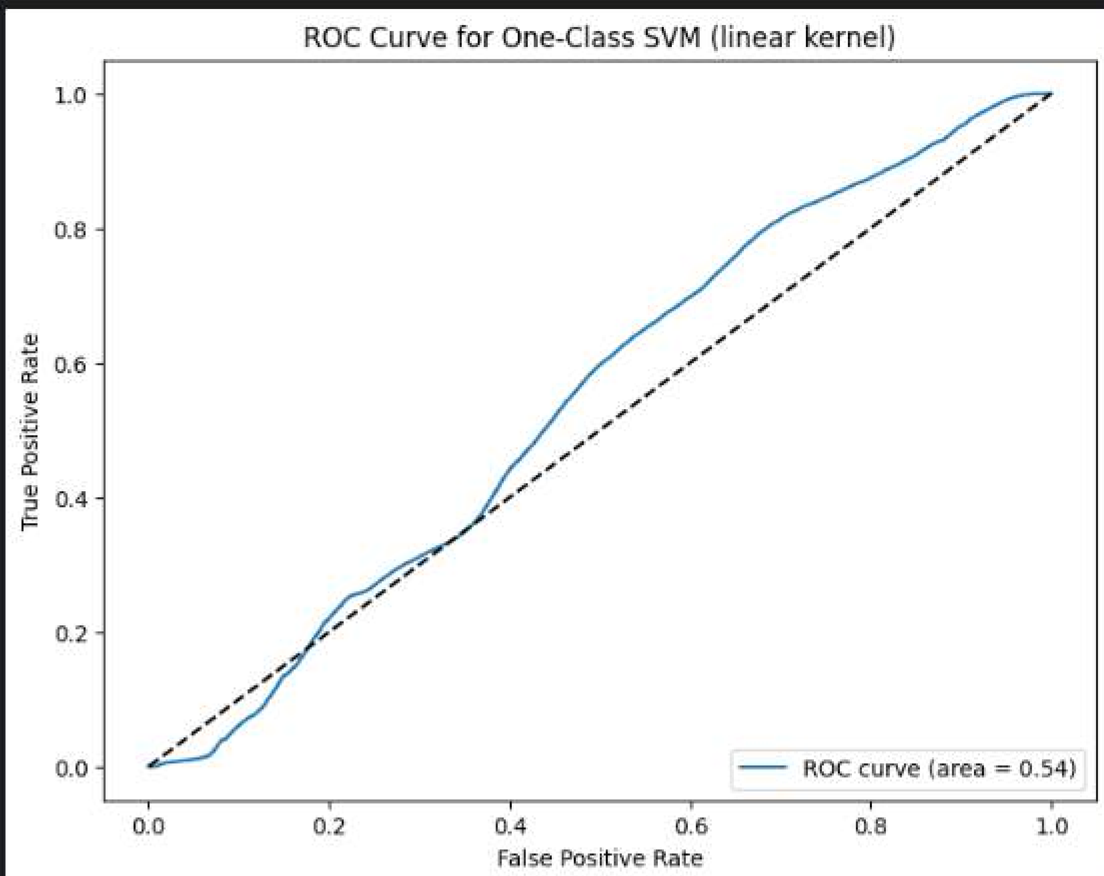
```
Training SVM with kernel: rbf
SVM training time with rbf kernel: 15.8549 seconds
For kernel 'rbf': Misclassified 123 out of 100000 samples
Metrics for kernel 'rbf':
  Accuracy: 99.53
  Precision: 99.22
  Recall: 98.97
  F1-Score: 98.78
  MCC: 1.0
  Confusion Matrix:
[[20889   122]
 [  182 77644]]
  Average Prediction Time per sample: 0.1279 ms
```

```
Training SVM with kernel: sigmoid
SVM training time with sigmoid kernel: 2346.7509 seconds
For kernel 'sigmoid': Misclassified 980 out of 100000 samples
Metrics for kernel 'sigmoid':
  Accuracy: 79.54599999999999
  Precision: 86.92520705248718
  Recall: 87.07120317589576
  F1-Score: 86.99814386330696
  MCC: 0.39079008805918647
  Confusion Matrix:
[[69293   322]
 [  399 71852]]
  Average Prediction Time per sample: 1.8532 ms
```

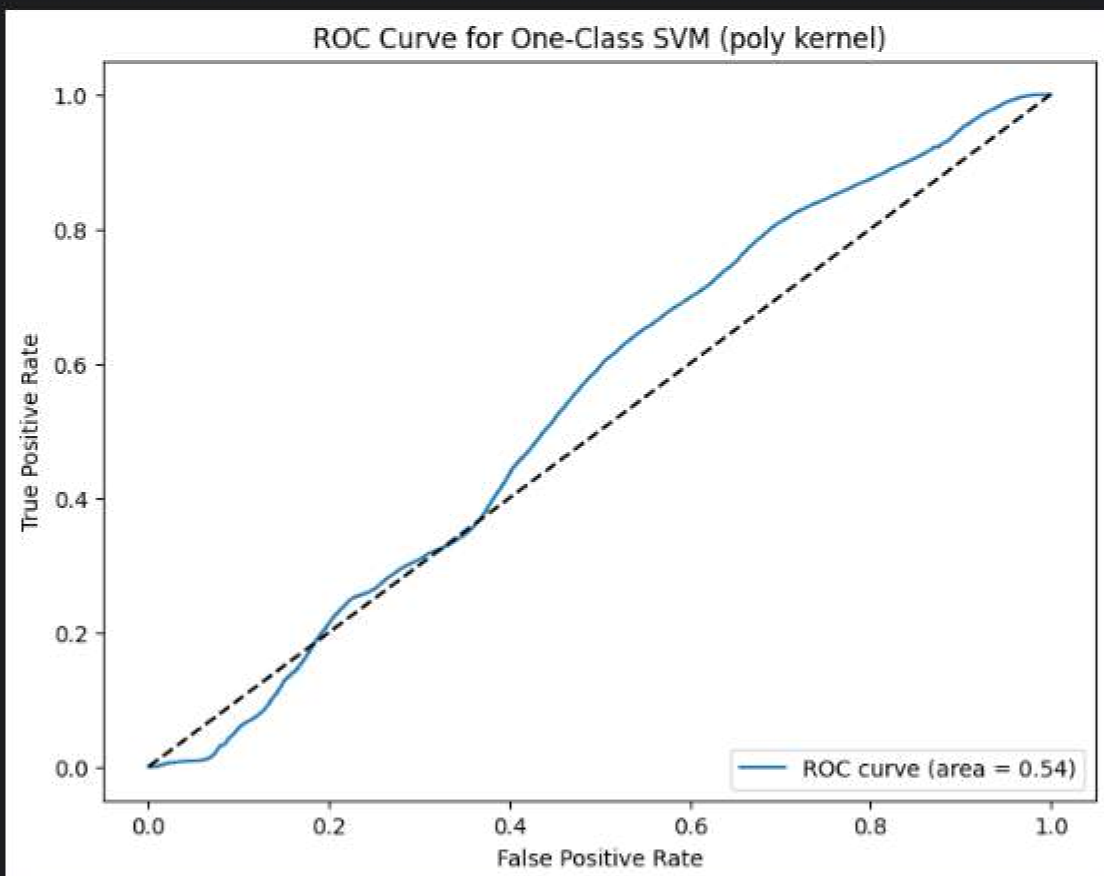


One Class SVM

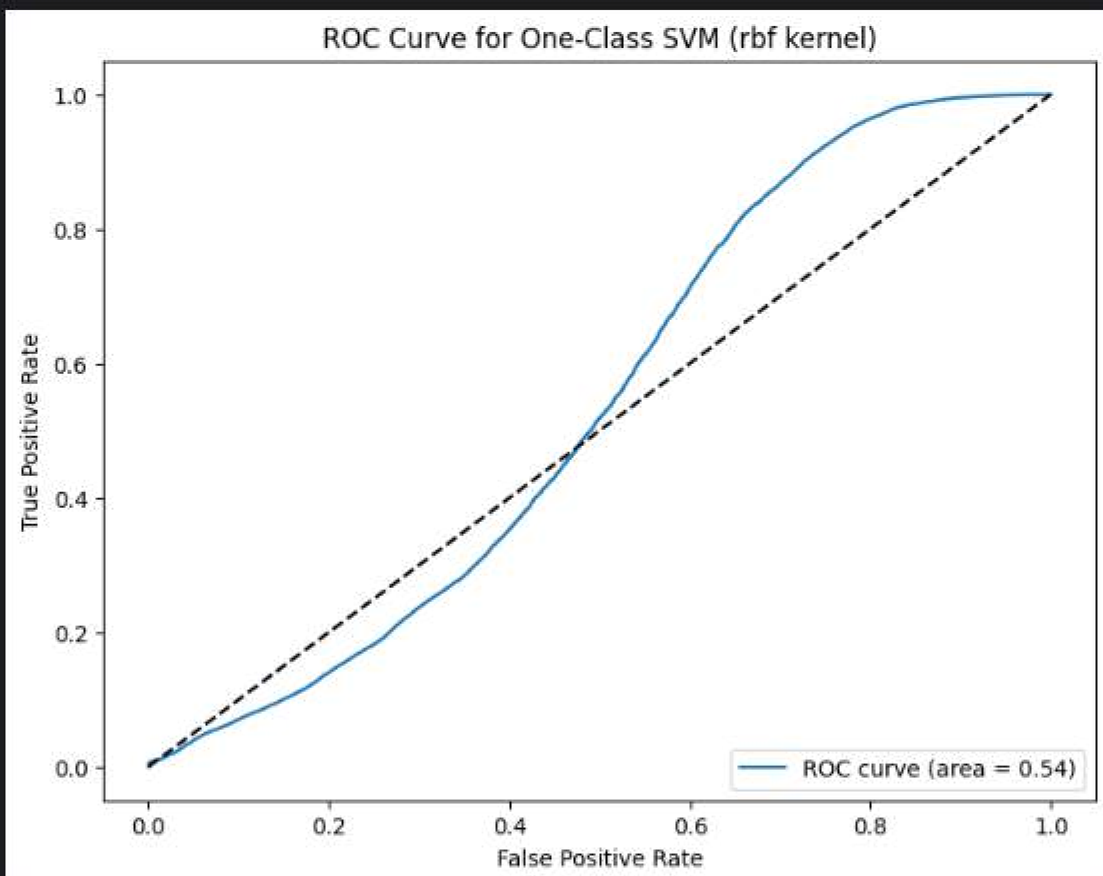
```
Training One-Class SVM with kernel: linear
One-Class SVM Metrics for kernel 'linear':
  Accuracy: 0.7484
  Precision: 0.7969
  Recall: 0.9125
  F1-Score: 0.8508
  MCC: 0.0801
  Confusion Matrix:
[[ 3126 18282]
 [ 6874 71718]]
```



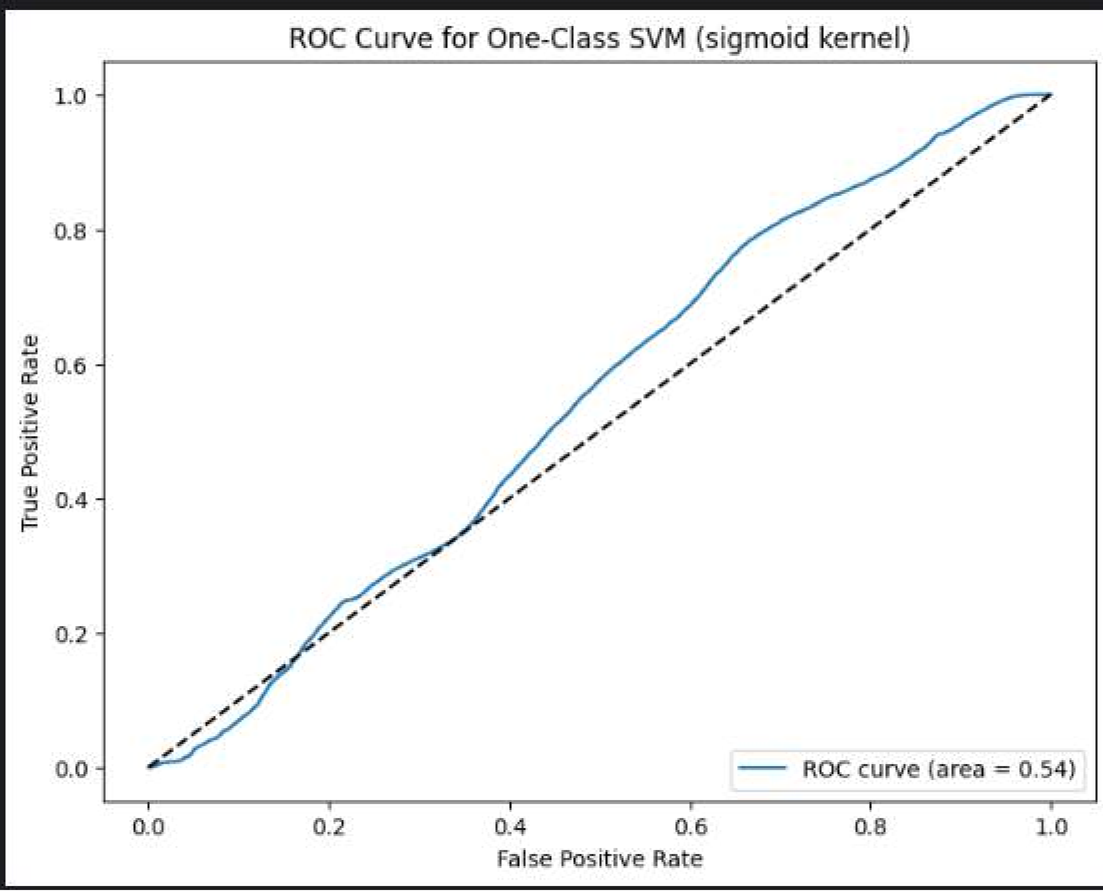
```
Training One-Class SVM with kernel: poly
One-Class SVM Metrics for kernel 'poly':
  Accuracy: 0.7467
  Precision: 0.7959
  Recall: 0.9114
  F1-Score: 0.8498
  MCC: 0.0731
  Confusion Matrix:
[[ 3040 18368]
 [ 6960 71632]]
```

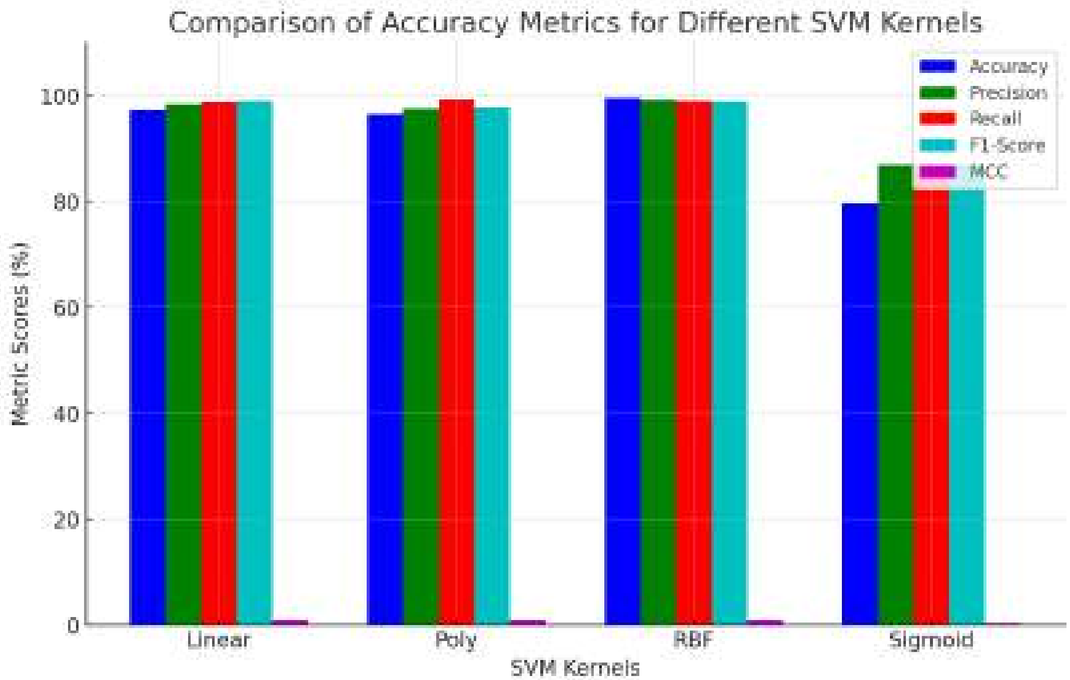


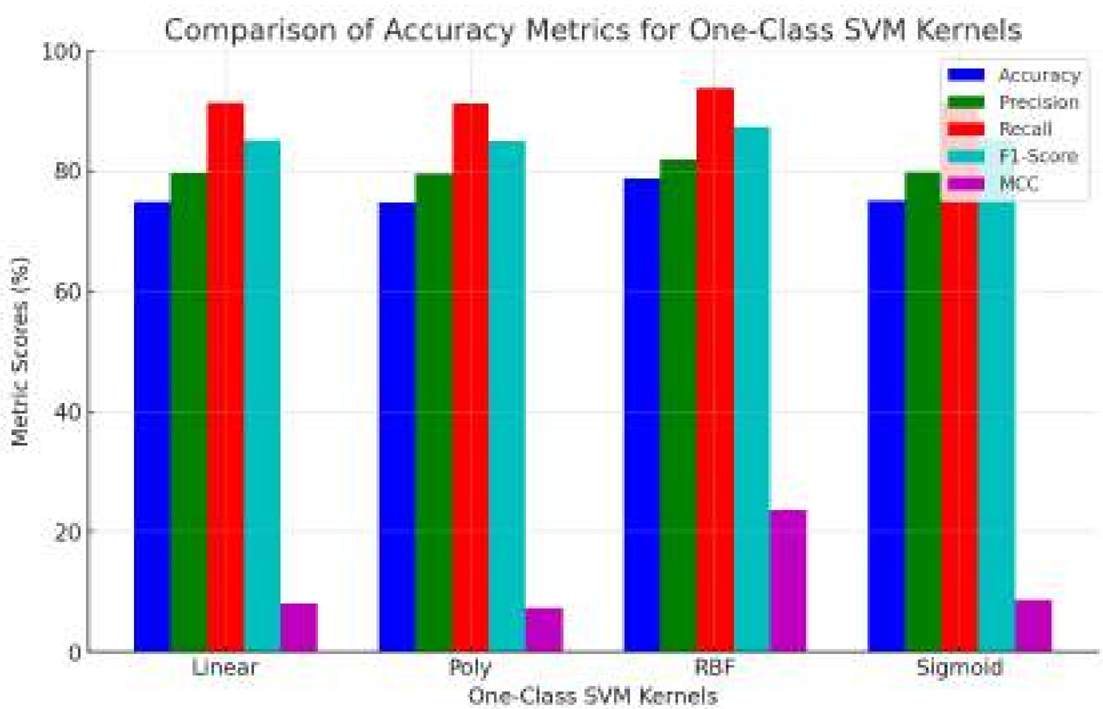
```
Training One-Class SVM with kernel: rbf
One-Class SVM Metrics for kernel 'rbf':
  Accuracy: 0.7868
  Precision: 0.8182
  Recall: 0.9370
  F1-Score: 0.8735
  MCC: 0.2358
  Confusion Matrix:
[[ 5042 16366]
 [ 4955 73637]]
```

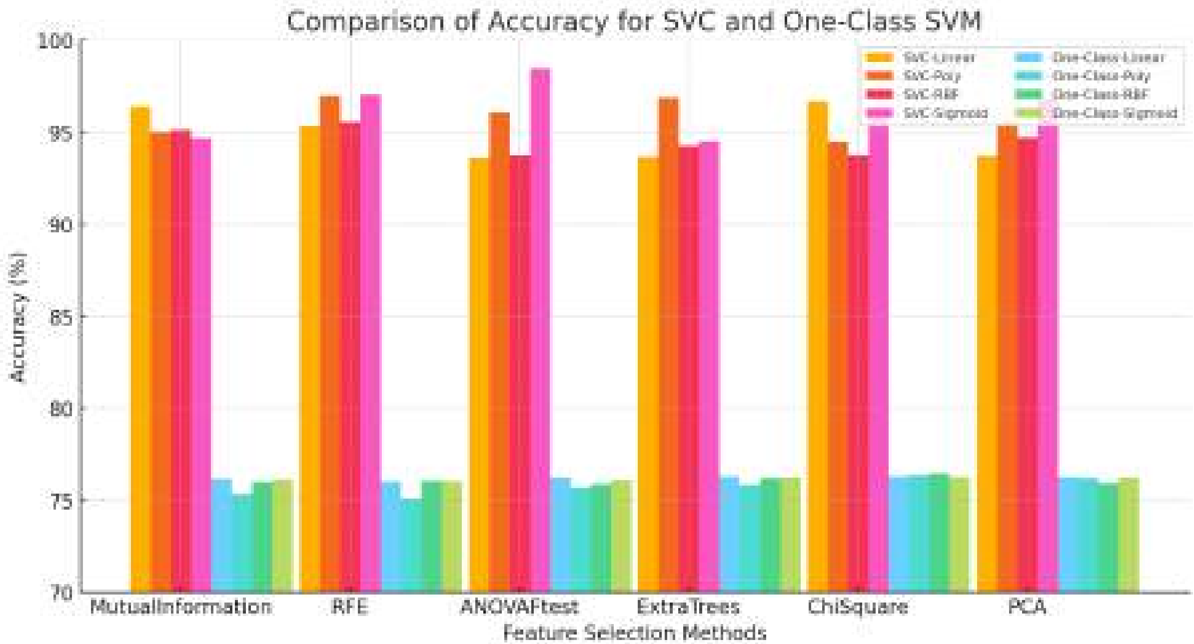


```
Training One-Class SVM with kernel: sigmoid
One-Class SVM Metrics for kernel 'sigmoid':
  Accuracy: 0.7497
  Precision: 0.7976
  Recall: 0.9133
  F1-Score: 0.8515
  MCC: 0.0852
  Confusion Matrix:
[[ 3190 18218]
 [ 6811 71781]]
```




Comparison of Accuracy Metrics for Different SVM Kernels
Accuracy
Precision
Recall
F1-Score
MCC
Metric Scores (%)
0
20
40
60
80
100
Linear
Poly
RBF
Sigmoid
SVM Kernels


Comparison of Accuracy Metrics for One-Class SVM Kernels
Accuracy
Precision
Recall
F1-Score
MCC
Metric Scores (%)
0
20
40
60
80
100
Linear
Poly
RBF
Sigmoid
One-Class SVM Kernels

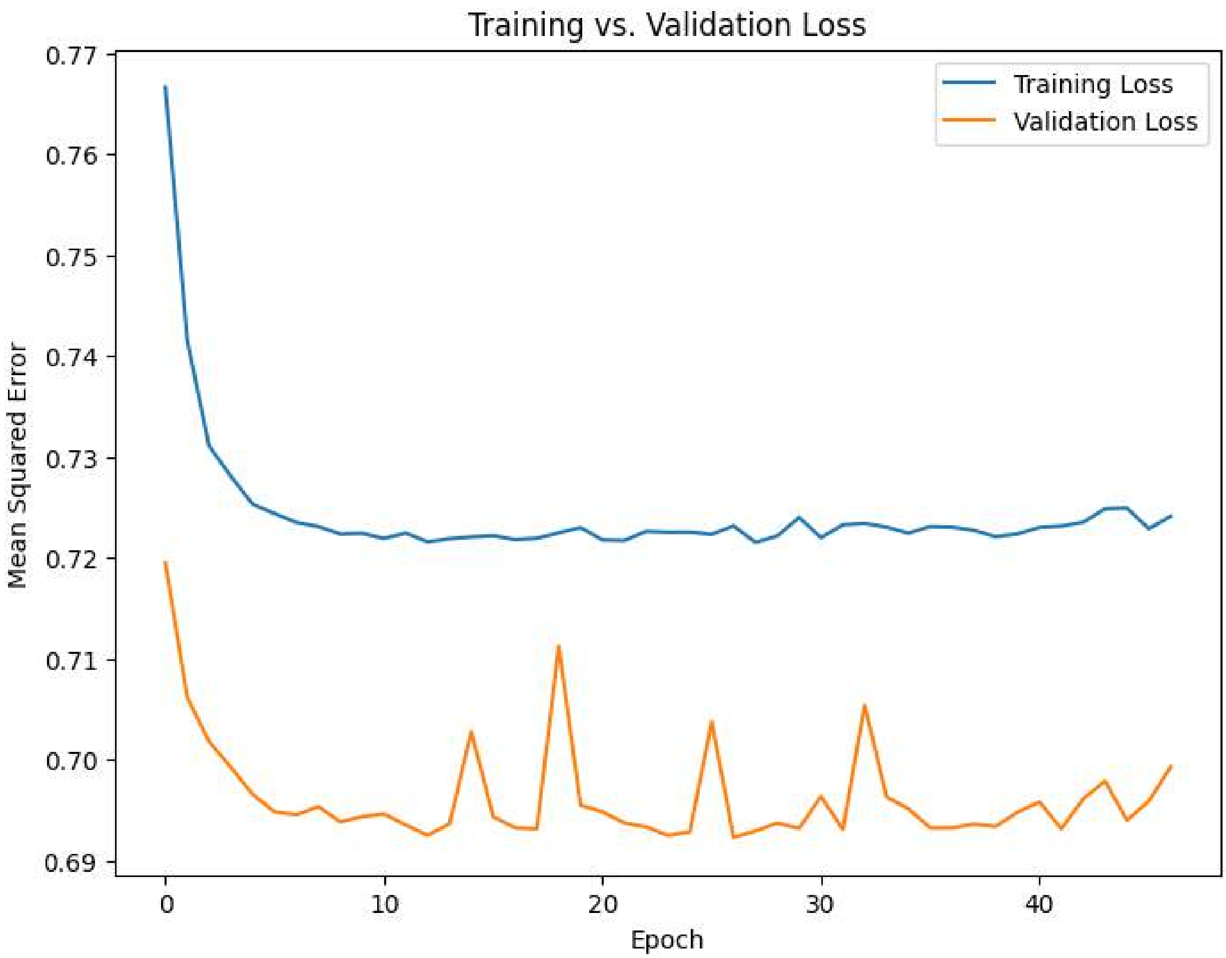


Complex Autoencoder architecture

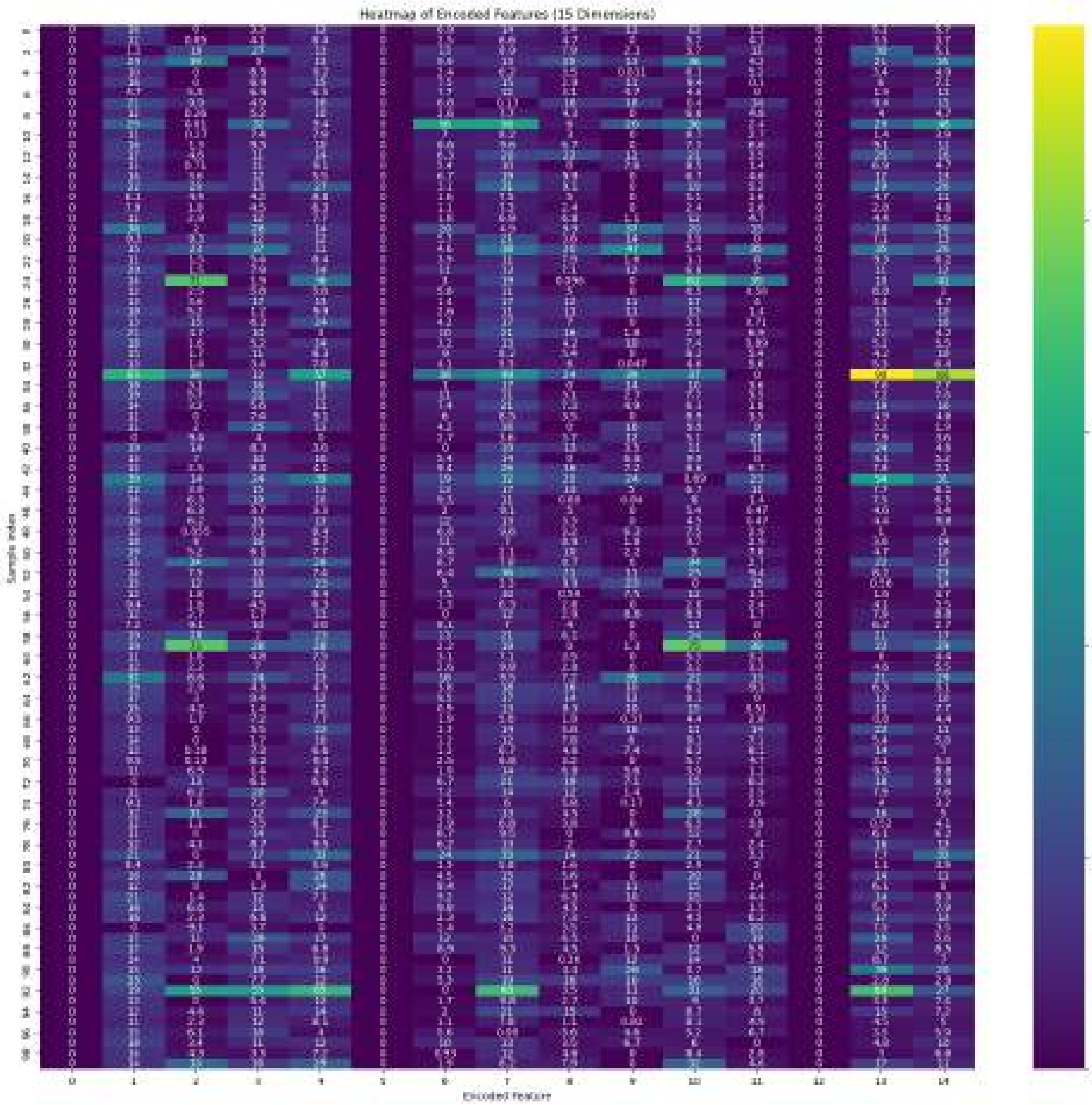
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 97)	0
encoder_dense_128 (Dense)	(None, 128)	12,544
encoder_dense_64 (Dense)	(None, 64)	8,256
encoder_dense_32 (Dense)	(None, 32)	2,080
encoder_output (Dense)	(None, 15)	495
decoder_dense_32 (Dense)	(None, 32)	512
decoder_dense_64 (Dense)	(None, 64)	2,112
decoder_dense_128 (Dense)	(None, 128)	8,320
decoder_output (Dense)	(None, 97)	12,513

```
3995/4000 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.7570Epoch 046 - loss: 0.7229 - val_lo
ss: 0.6959
4000/4000 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - loss: 0.7570 - val_loss: 0.6959
Epoch 47/500
3985/4000 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.7523Epoch 047 - loss: 0.7241 - val_lo
ss: 0.6993
4000/4000 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - loss: 0.7522 - val_loss: 0.6993
```

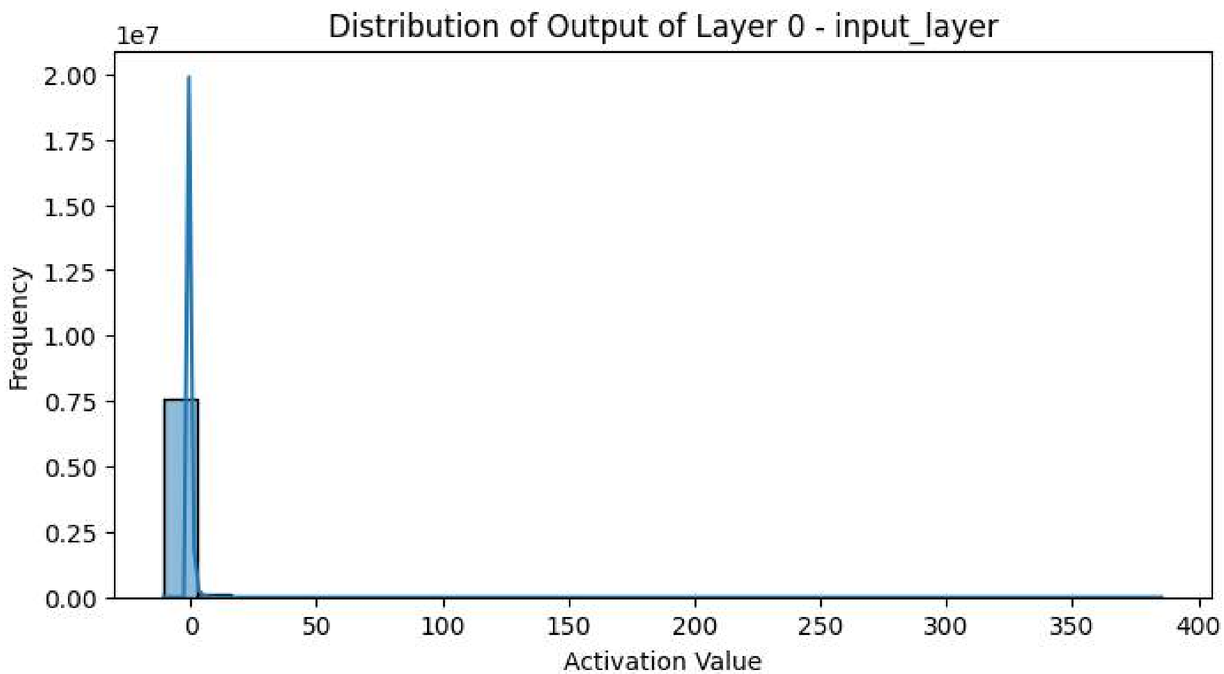
Training and Validation loss

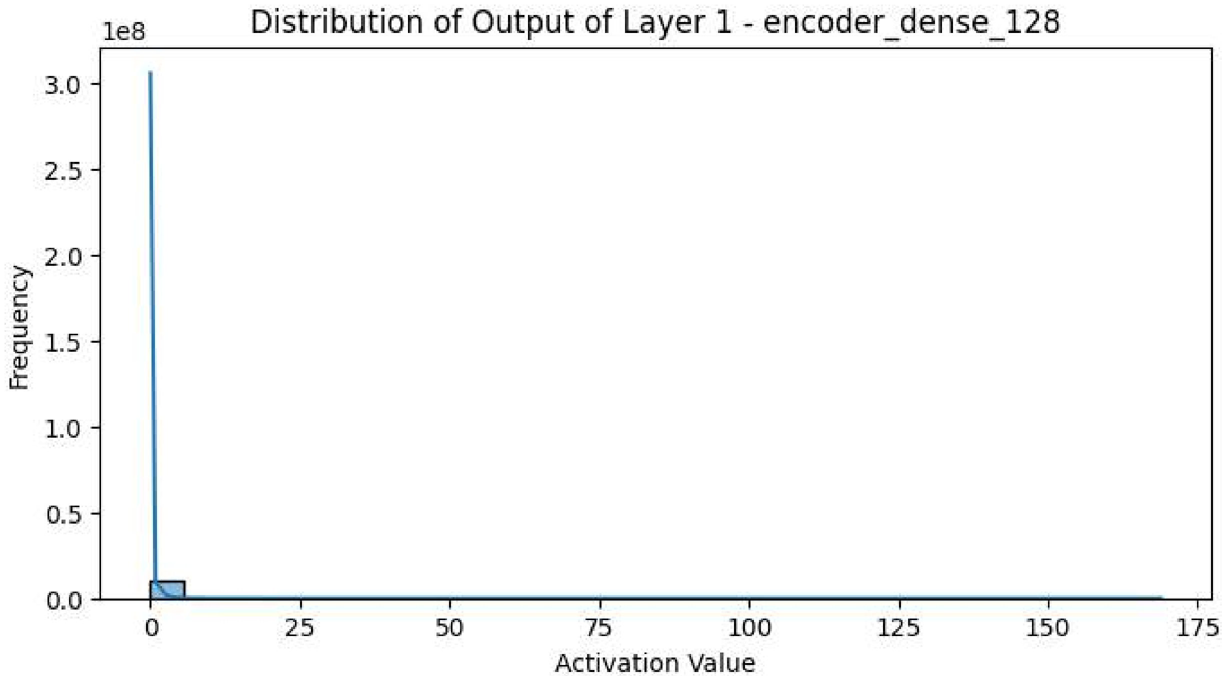


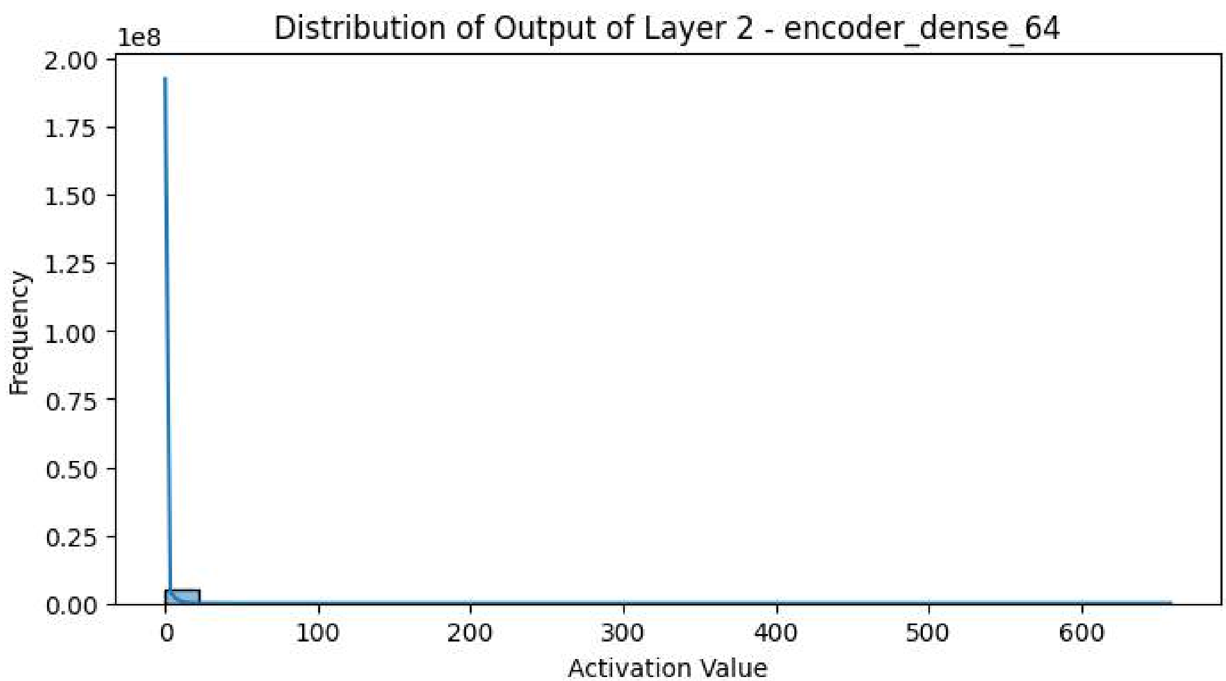
Weights of 15 extracted features for the first 100 samples

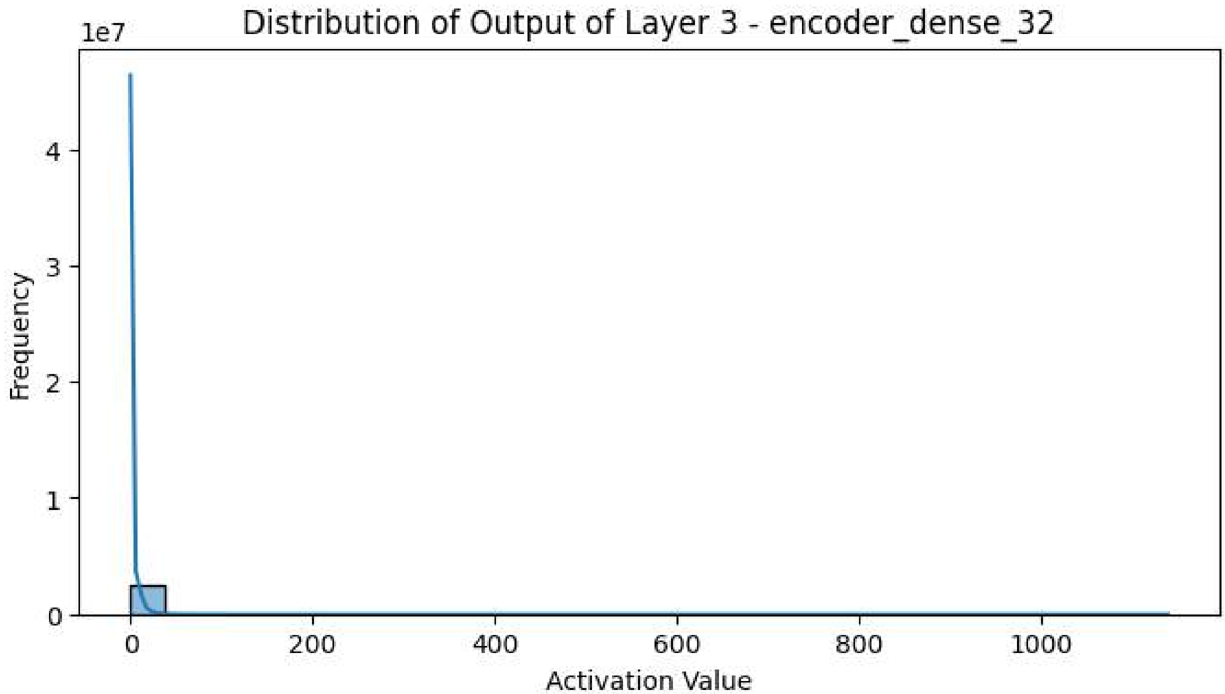


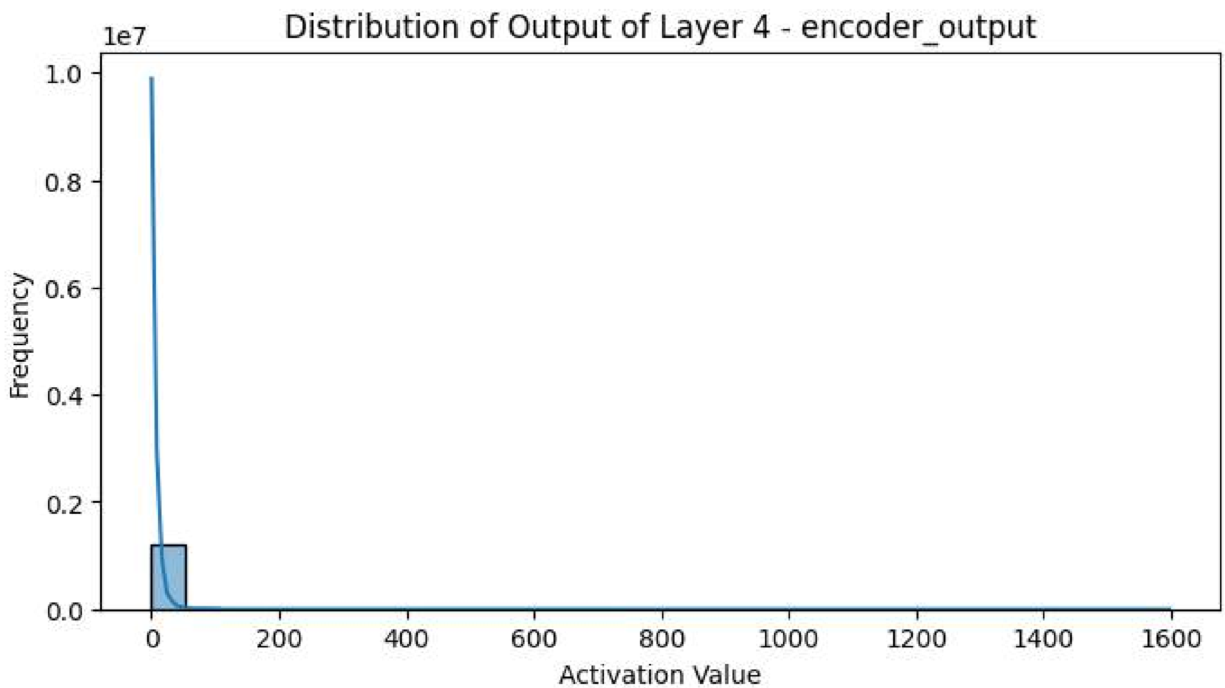
Layer Output Plots in autoencoder

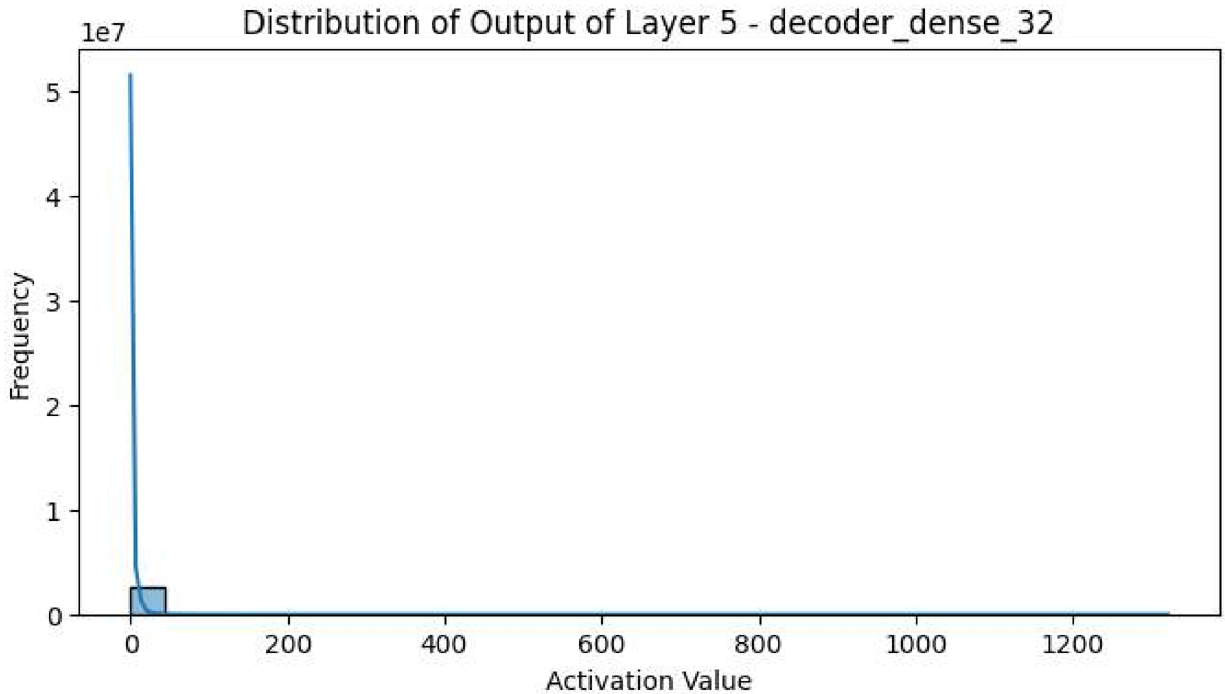


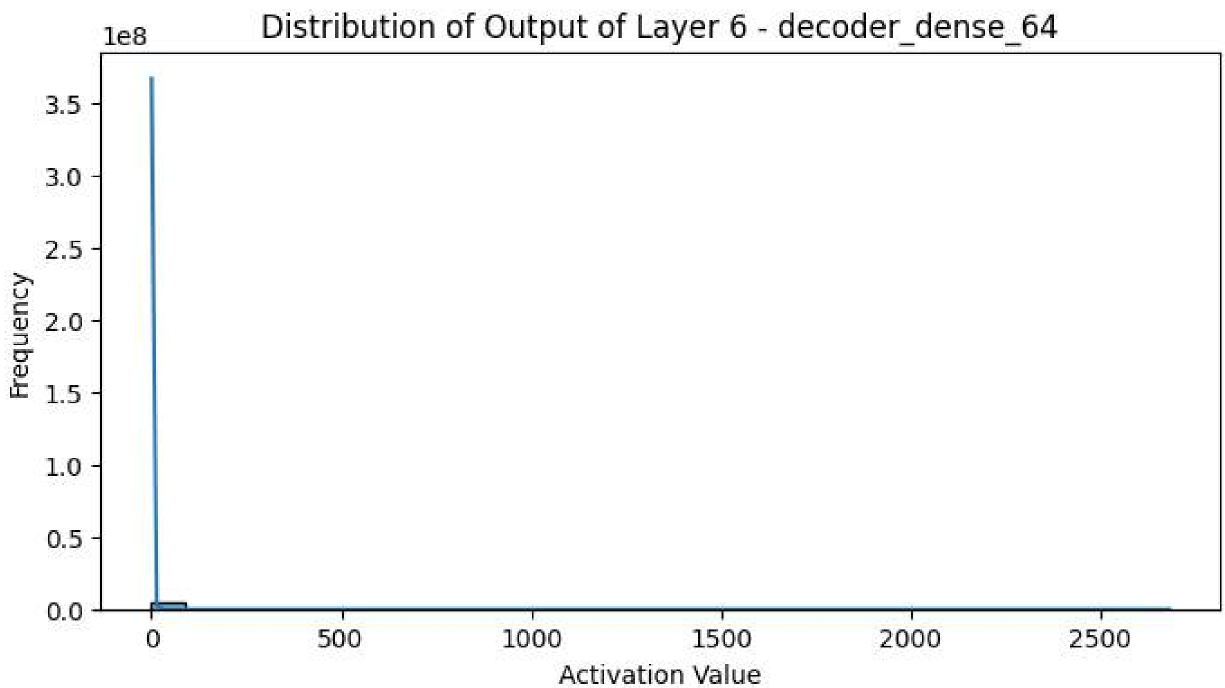

Distribution of Output of Layer 1 - encoder_dense_128
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Frequency
0
25
50
75
100
125
150
175
Activation Value

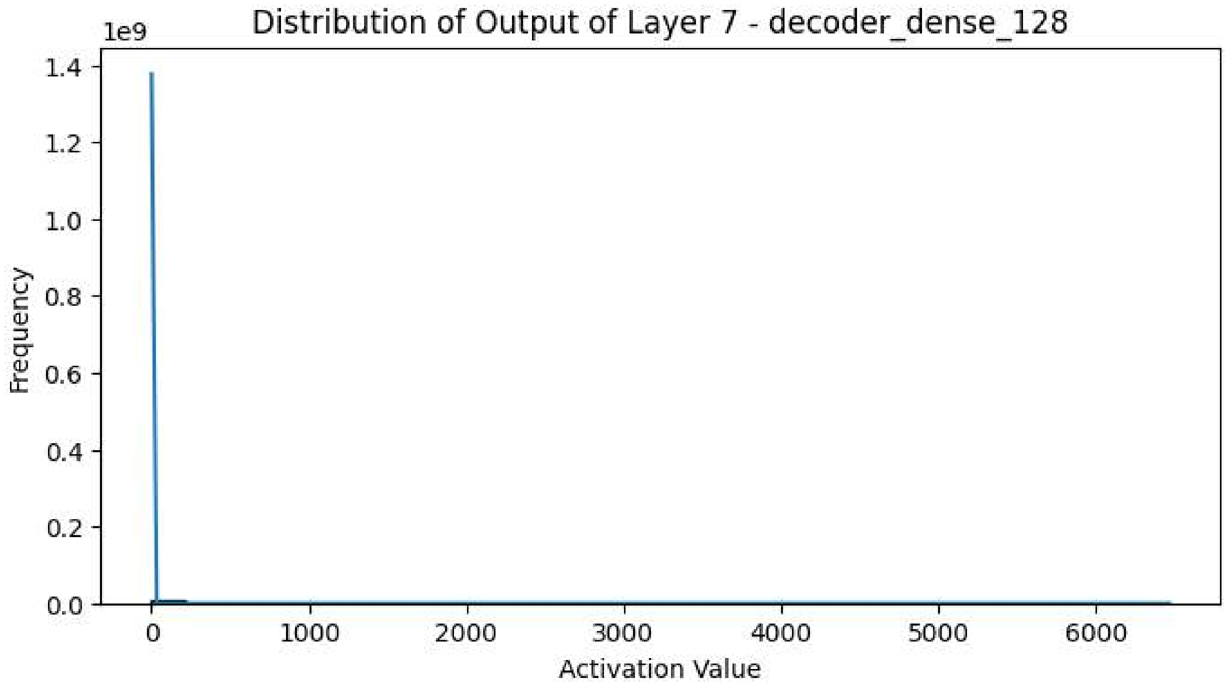

Distribution of Output of Layer 2 - encoder_dense_64
1e8
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Frequency
0
100
200
300
400
500
600
Activation Value

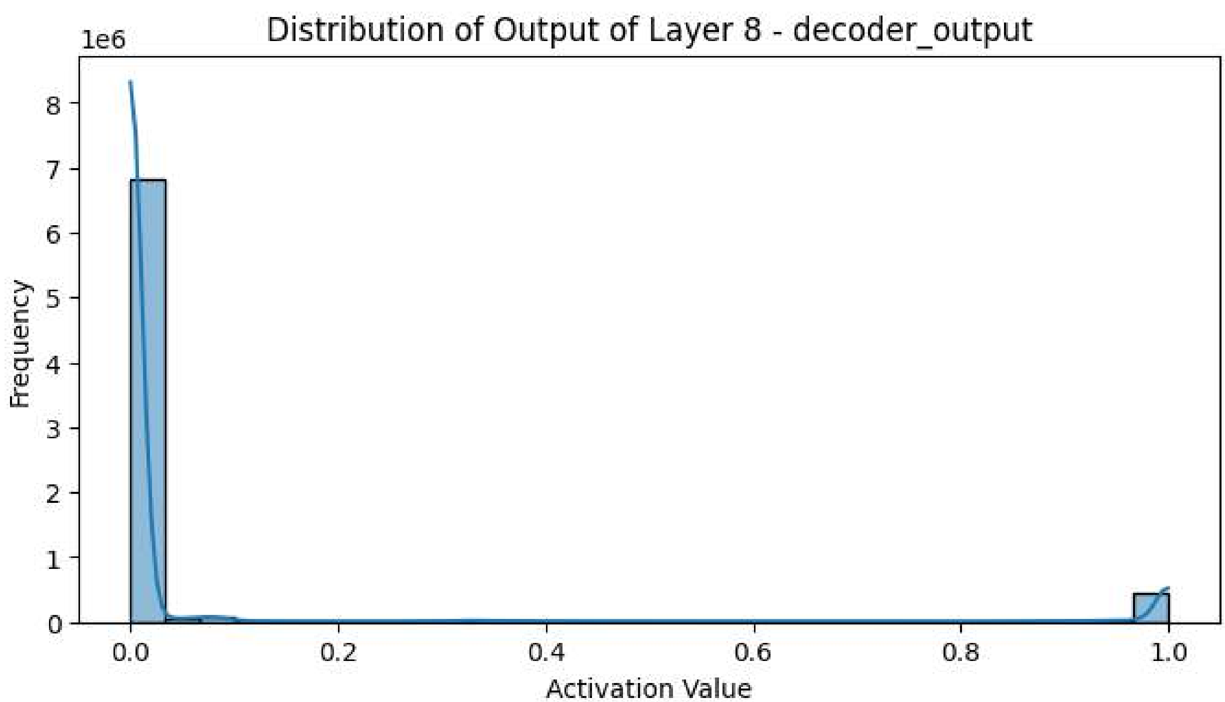

Distribution of Output of Layer 3 - encoder_dense_32
1e7
4
3
2
1
0
Frequency
0
200
400
600
800
1000
Activation Value


Distribution of Output of Layer 4 - encoder_output
1e7
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
0
200
400
600
800
1000
1200
1400
1600
Activation Value


Distribution of Output of Layer 5 - decoder_dense_32
1e7
5
4
3
2
1
0
Frequency
0
200
400
600
800
1000
1200
Activation Value


Distribution of Output of Layer 6 - decoder_dense_64
1e8
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Frequency
0
500
1000
1500
2000
2500
Activation Value





SVM Model trained on those 15 extracted features using 4 kernels

```
--- Training SVM with kernel: linear ---
Pickled SVM model saved as: svm_model_linear.pkl
Accuracy: 0.8674
Confusion Matrix:
 [[8893 1510]
 [1142 8455]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.85      0.87     10403
           1       0.85      0.88      0.86      9597

    accuracy                           0.87     20000
   macro avg       0.87      0.87      0.87     20000
weighted avg       0.87      0.87      0.87     20000
```

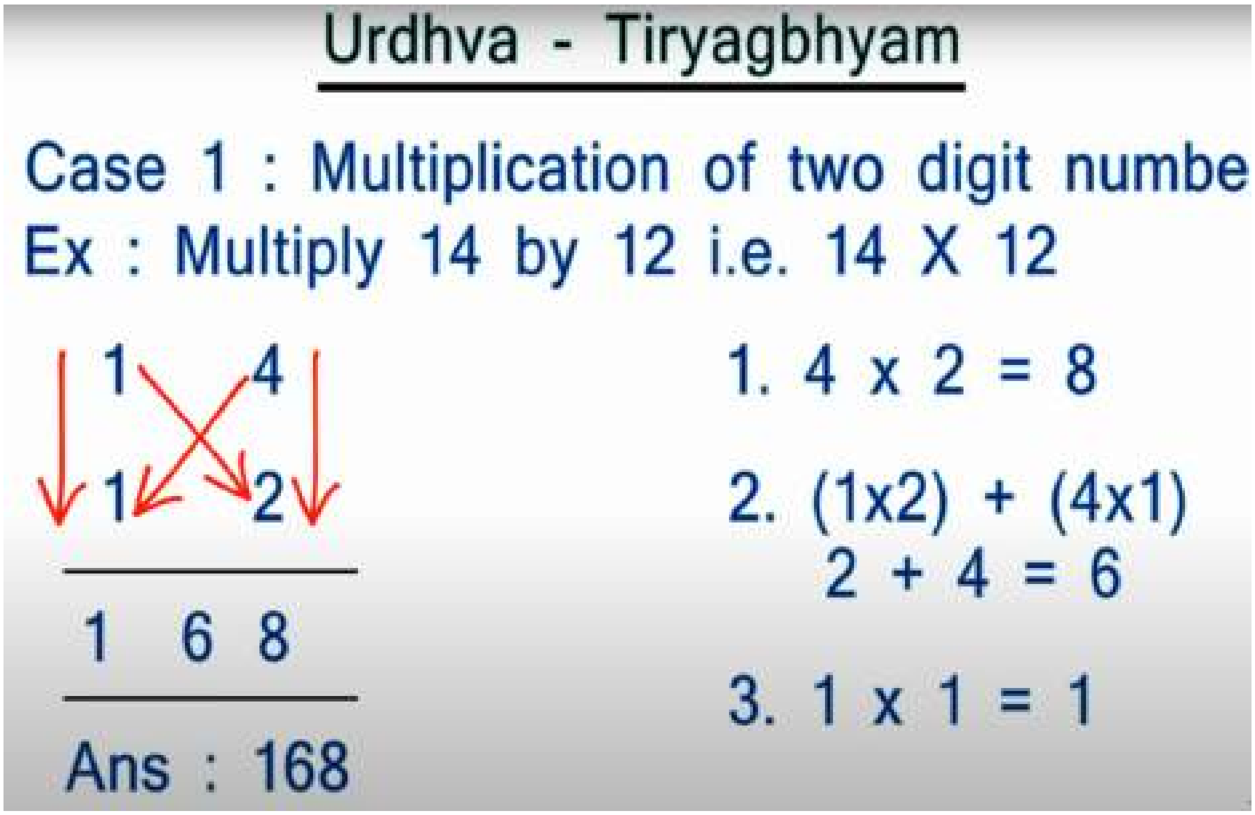
```
--- Training SVM with kernel: poly ---
Pickled SVM model saved as: svm_model_poly.pkl
Accuracy: 0.9056
Confusion Matrix:
 [[8847 1556]
 [ 332 9265]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.85      0.90     10403
           1       0.86      0.97      0.91      9597

    accuracy                           0.91     20000
   macro avg       0.91      0.91      0.91     20000
weighted avg       0.91      0.91      0.91     20000
```

```
--- Training SVM with kernel: rbf ---
Pickled SVM model saved as: svm_model_rbf.pkl
Accuracy: 0.9469
Confusion Matrix:
 [[9876  527]
 [ 535 9062]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     10403
           1       0.95      0.94      0.94      9597

    accuracy                           0.95     20000
   macro avg       0.95      0.95      0.95     20000
weighted avg       0.95      0.95      0.95     20000
```

```
--- Training SVM with kernel: sigmoid ---
Pickled SVM model saved as: svm_model_sigmoid.pkl
Accuracy: 0.7641
Confusion Matrix:
 [[7988 2415]
 [2303 7294]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.77      0.77     10403
           1       0.75      0.76      0.76      9597

    accuracy                           0.76     20000
   macro avg       0.76      0.76      0.76     20000
weighted avg       0.76      0.76      0.76     20000
```

IKS - Vedic Maths

Vedic Multiplication (Urdhva-Tiryagbhyam)



Example: Multiplying 23 × 45

Step 1 – Write the Numbers as Digits:

23 → digits: 2 and 3
45 → digits: 4 and 5

Step 2 – Multiply the Right-most Digits:

Multiply 3 (from 23) by 5 (from 45): $3 \times 5 = 15$

Write down 5 and carry over 1.

Step 3 – Cross-Multiply and Add:

Multiply cross-wise:

$(2 \times 5) + (3 \times 4) = 10 + 12 = 22$

Add the carried over 1:
$22 + 1 = 23$

Write down the unit digit 3 and carry over 2.

Step 4 – Multiply the Left-most Digits:

Multiply 2 (from 23) by 4 (from 45):

$2 \times 4 = 8$

Add the carry 2: $8 + 2 = 10$

Write down 10 (which gives the remaining digits).

Step 5 – Combine the Results:

The digits (from left to right) become 10, 3, 5

When you combine them (taking care of any place-value adjustments), the final product is 1035.

Matrix Dot Product Using Vedic Multiplication

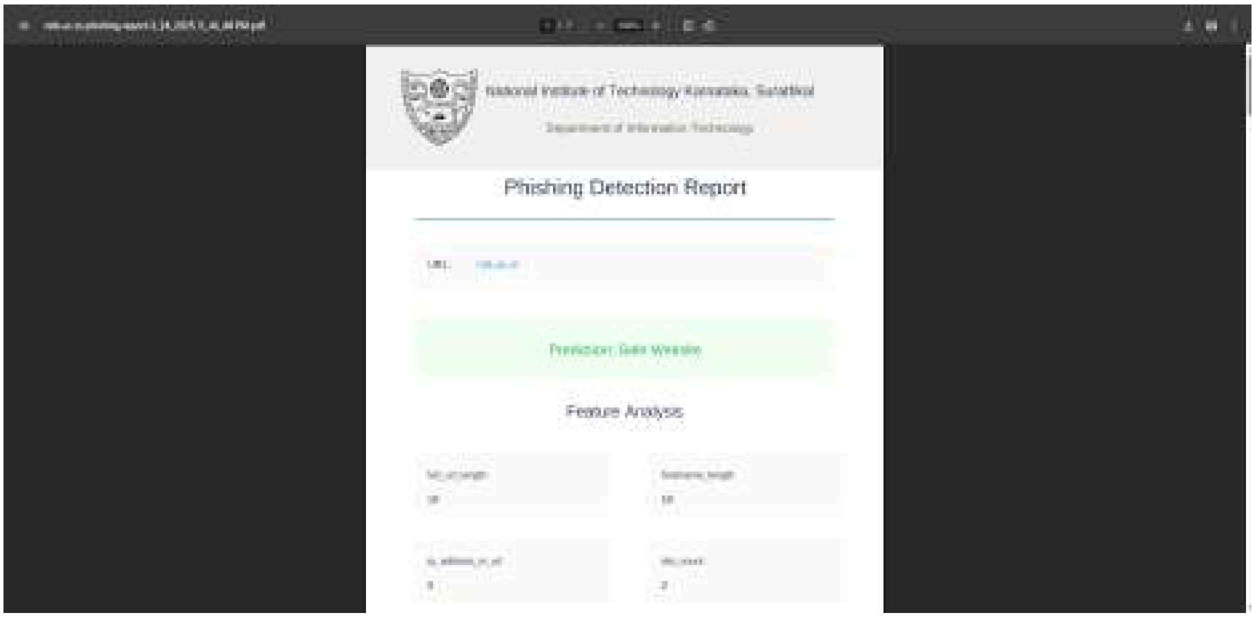
$$\text{result}[i,j] = \sum \text{vedic\_multiply}(A[i,k], B[k,j])$$

Web Tool

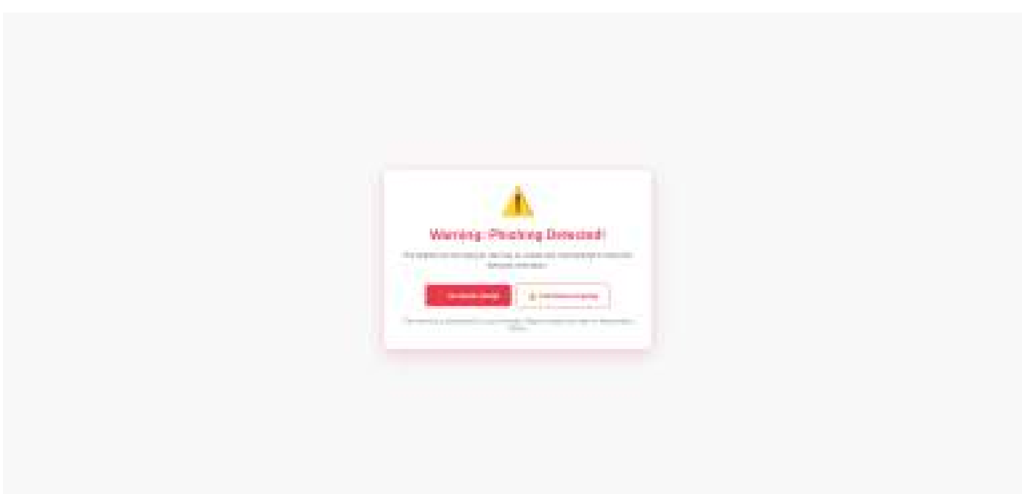

National Institute of Technology Karnataka, Surathkal
Phishing Website Detection System [IT352 Course Project Jan - May 2025]
Single URL Analysis
Analyze Single URL


National Institute of Technology Karnataka, Surathkal
Phishing Website Detection System [IT352 Course Project Jan - May 2025]
Analyze Single URL

Report PDF



Extension



Phishing Website Examples

smilesvoegol.servebbs.org/voegol.php

premierpaymentprocessing.com/includes/boleto-2via-07-2012.php

super1000.info/docs

www.coincoele.com.br/Scripts/smiles/?pt-br/Paginas/default.aspx

www.avedeoiro.com/site/plugins/chase