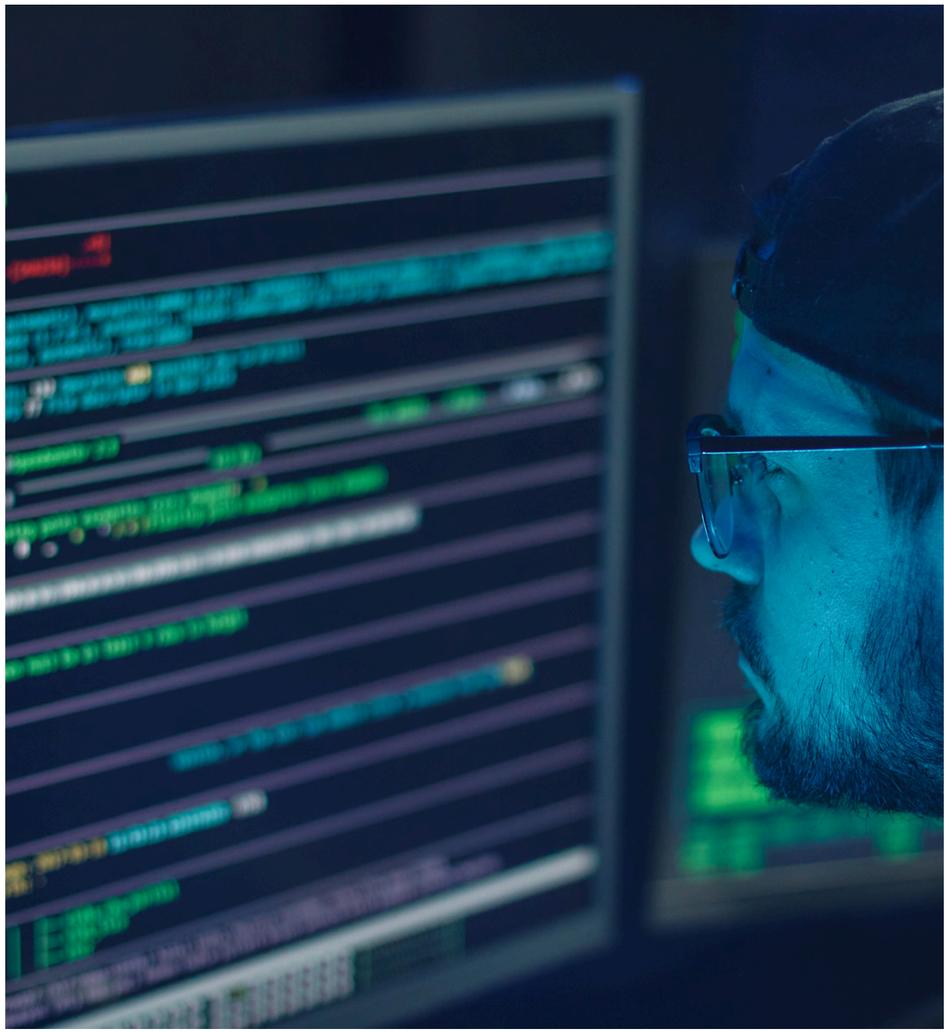
Phishing Website Detection

Introduction

Cybercriminals employ deceptive tactics, often exploiting human vulnerability, to gain unauthorized access to sensitive information.

Phishing attacks have evolved from simple email scams to highly targeted and convincing campaigns that can compromise personal and organizational security. The consequences of falling victim to such attacks can be severe, ranging from financial losses to data breaches with far-reaching implications.



What is Phishing?

Phishing, at its core, is a form of cybercrime that employs deceptive tactics to trick individuals into revealing sensitive information such as usernames, passwords, and financial details. It's a term that encapsulates a variety of techniques, often exploiting human psychology and trust in order to gain unauthorized access.

Common Tactics Used by Phishers:

- Email Spoofing
- Social Engineering
- Phishing Websites

Related Work

Research Paper- I

A new hybrid ensemble feature selection framework for machine learning-based phishing detection system
By Kang Leng Chiewa, Choon Lin Tan, KokSheik Wong, Kelvin S.C. Yong, Wei King Tiong

- Proposed a new feature selection framework for machine learning-based phishing detection systems called the **Hybrid Ensemble Feature Selection (HEFS)**.

- The framework consists of two phases: the first phase utilizes the Cumulative Distribution Function gradient **(CDF-g)** algorithm to generate primary feature subsets. The second phase derives baseline features from the secondary feature subsets.

- Results show that HEFS, when integrated with the Random Forest classifier, performs best, identifying baseline features that correctly distinguish 94.6% of phishing and legitimate websites using only 20.8% of the original features.

Research Paper- I

- The researchers utilized filter measures, such as **Information Gain, Relief-F, and Symmetrical Uncertainty**, to rank features and generate primary feature subsets. The CDF-g algorithm was then applied to identify the optimal cut-off rank of features. The hybrid ensemble strategy combined **data perturbation and function perturbation** techniques to produce secondary feature subsets and the final baseline feature set.

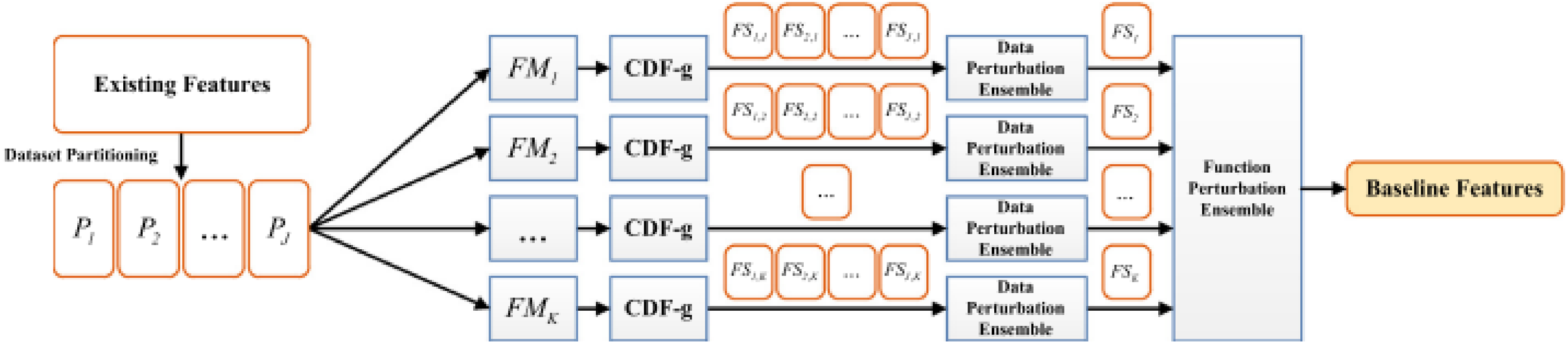


Fig. 1. Overview of the proposed feature selection framework.

ADVANTAGES

01

Enhances the detection accuracy and computational efficiency.

02

Novel approach to determine the cut-off ranks(ie., cut-off point) for the features ranked by filter measures.

03

Not affected by the order of the instances present in the dataset as it randomly partitions the datasets into parts and uses data perbutation and function perbutation for feature extraction.

DISADVANTAGES

01

CDF-g uses gradient for calculating the cut-off rank, therefore it can have limitations like, low convergence rate, unbalanced selection effect, and biased estimation

02

NumDots, UrlLength, AtSymbol, NoHttps, and IpAddress which are considered essential for phishing detection, actually contribute only little to the accuracy of a phishing detection technique

Related Work

Research Paper- II

An effective detection approach for phishing websites using URL and HTML features By Ali Aljofey, Qingshan Jiang, Abdur Rasool, Hui Chen, Wenyin Liu, Qiang Qu & YangWang

- This paper provides an efficient solution for phishing detection that extracts the features from a **website's URL and HTML source code**.

- Specifically, the authors proposed a hybrid feature set including URL character sequence features, various hyperlink information, plaintext, and noisy HTML data-based features within the HTML source code. These features are then used to create a feature vector required for training the proposed approach by the **XGBoost classifier.**

- The authors have introduced eleven hyperlink features (F3–F13), two login form features (F14 and F15), character level TF-IDF features (F2), and URL character sequence features (F1).The proposed features determine the relationship between the URL of the webpage and the webpage content.

Research Paper- II

- From the experimental results, it is noticed that **TF-IDF character level features** outperformed other features with significant accuracy(96.76%), precision, F-Score, Recall, and AUC using the **XGBoost classifier**. Hence, we implemented the TF-IDF character level technique to generate text features (F2) of the webpage.

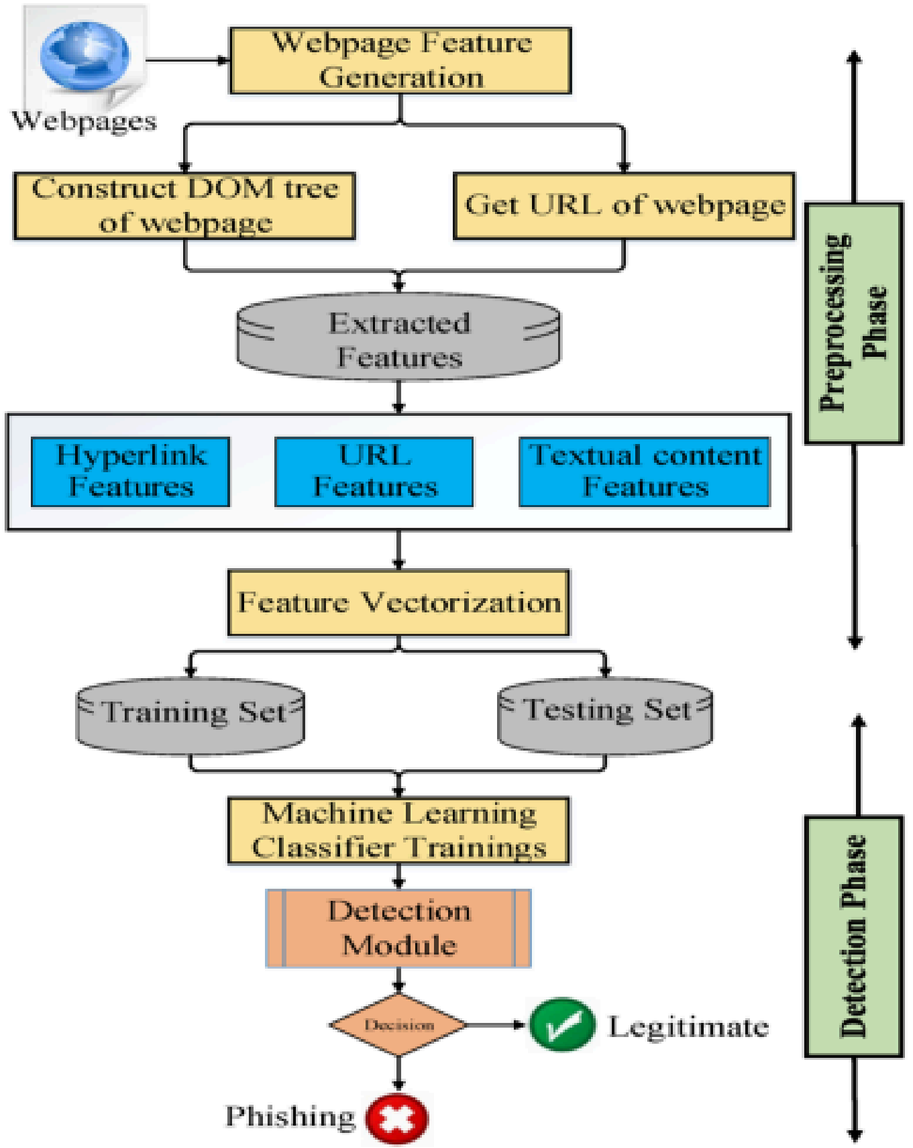


Fig. 2 General architecture of proposed approach

ADVANTAGES

01

The approach is dynamic and more suitable.

02

Classifies the websites based on both url and web page content, hence enhances its accuracy.

DISADVANTAGES

01 Web-page language dependent

02 Does not identify the textual content hidden in embedded objects

03 Large training time due to high dimensional vector

04 Not sufficiently capable of detecting attached malware

Implementation

Dataset Used

Dataset consists of 11,055 instances, each associated with 32 features. These features are intricately crafted to encapsulate various aspects of a website's URL and its associated characteristics.

```
print(df.shape)
print(df.columns)
```

```
(11055, 32)
Index(['id', 'having_IP_Address', 'URL_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'],
      dtype='object')
```

Implementation

01 Data Preprocessing

02 Feature Selection

03 Classification Models

04 Comparison of Accuracy of various Classification Moodels

Data Preprocessing

Checked for null values in the dataset

Analysed the dataset for missing values

Drop the column with unique values, having non predictive nature.

Checked the class distribution of the target class in the dataset

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

```
id                             0
having_IP_Address              0
URL_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
```

Fig. 3 Null Values

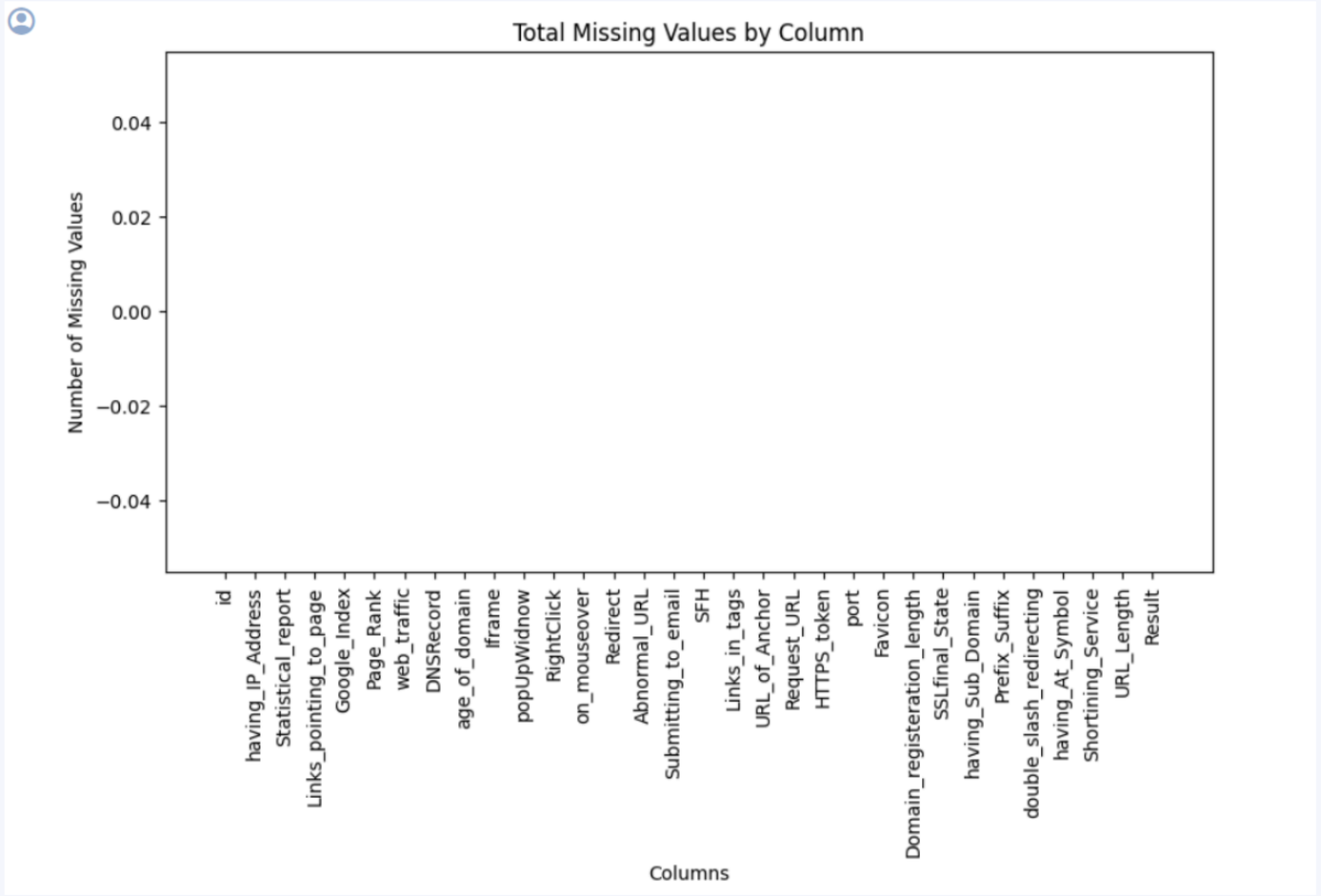


Fig. 4 Missing Values

```
df.drop(['id'], axis=1, inplace=True)
print(df)
```

```
       having_IP_Address  URL_Length  Shortining_Service  h
0                     -1           1                   1
1                      1           1                   1
2                      1           0                   1
3                      1           0                   1
4                      1           0                  -1
...                  ...         ...                 ...
11050                  1          -1                   1
11051                 -1           1                   1
11052                  1          -1                   1
11053                 -1          -1                   1
11054                 -1          -1                   1

       double_slash_redirecting  Prefix_Suffix  having_Sub_
0                            -1             -1
1                             1             -1
2                             1             -1
3                             1             -1
4                             1             -1
...                         ...            ...
```

Fig. 5 Drop id column

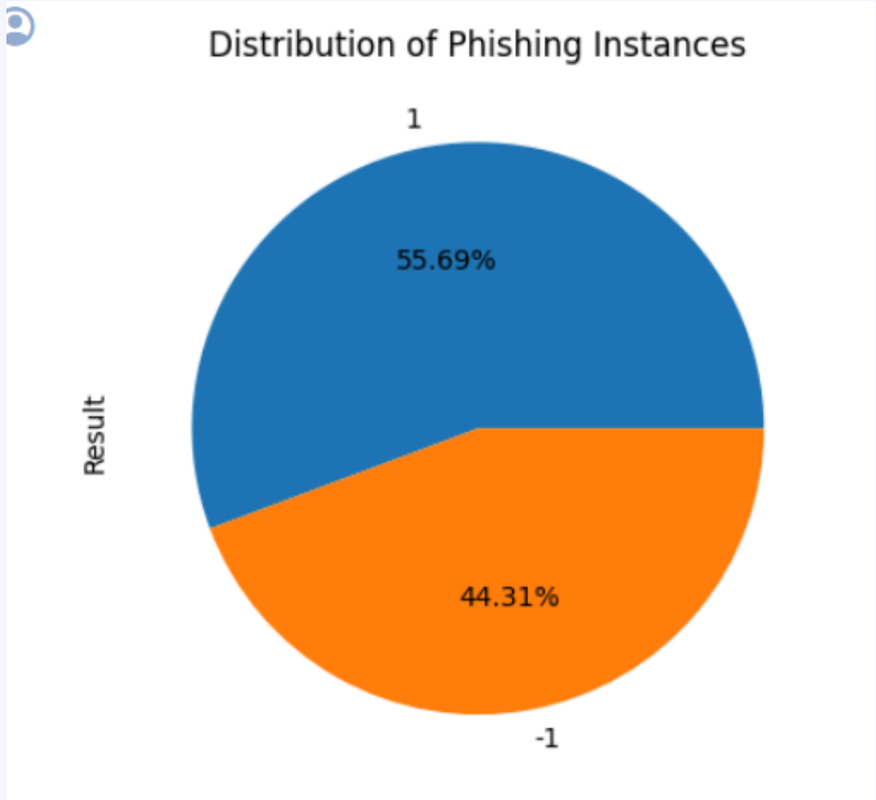


Fig. 6 Target class distribution

Feature Selection Measures Used

Pearson Correlation between features

Pearson Correlation between feature and target class

Mutual Information Gain between features and target class

Variance, to identify existence of any constant feature

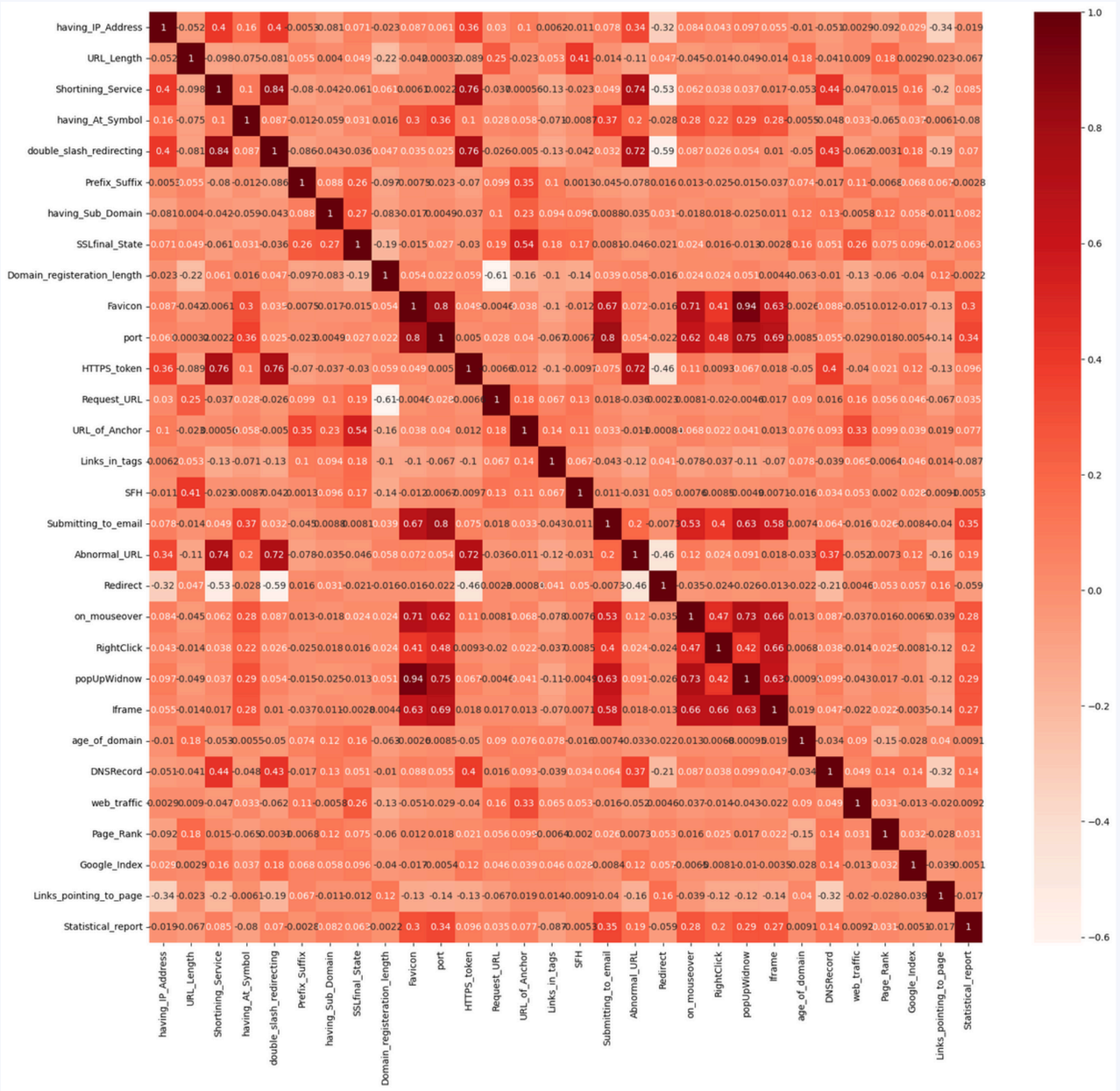


Fig. 7

Heatmap for Pearson's correlation having all 30 attributes

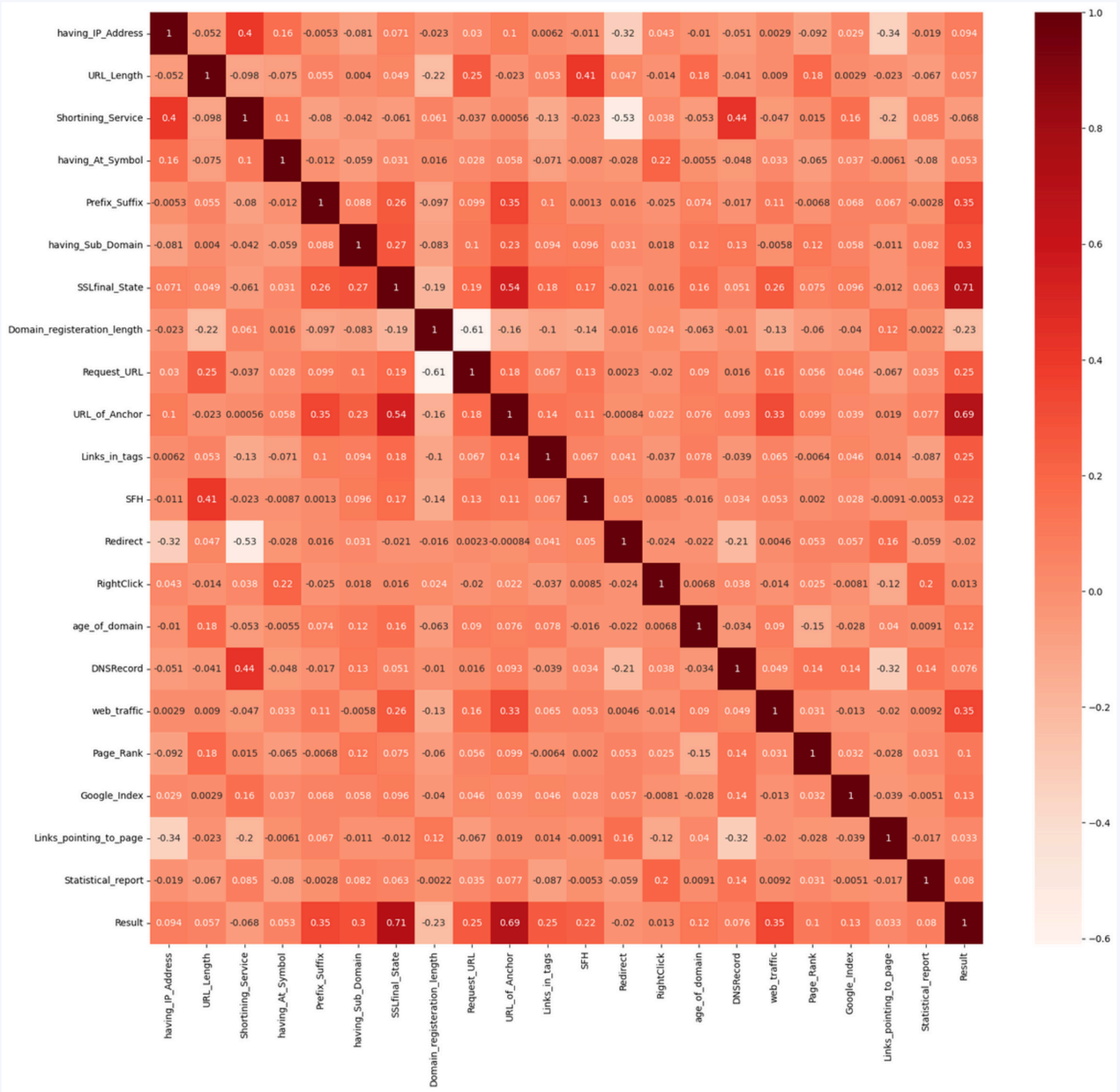


Fig. 8

Heatmap for Pearson's correlation after dropping, 'double_slash_redirecting', 'on_mouseover', 'HTTPS_token', 'popUpWidnow', 'port', 'Submitting_to_email', 'Abnormal_URL' (7 attributes)

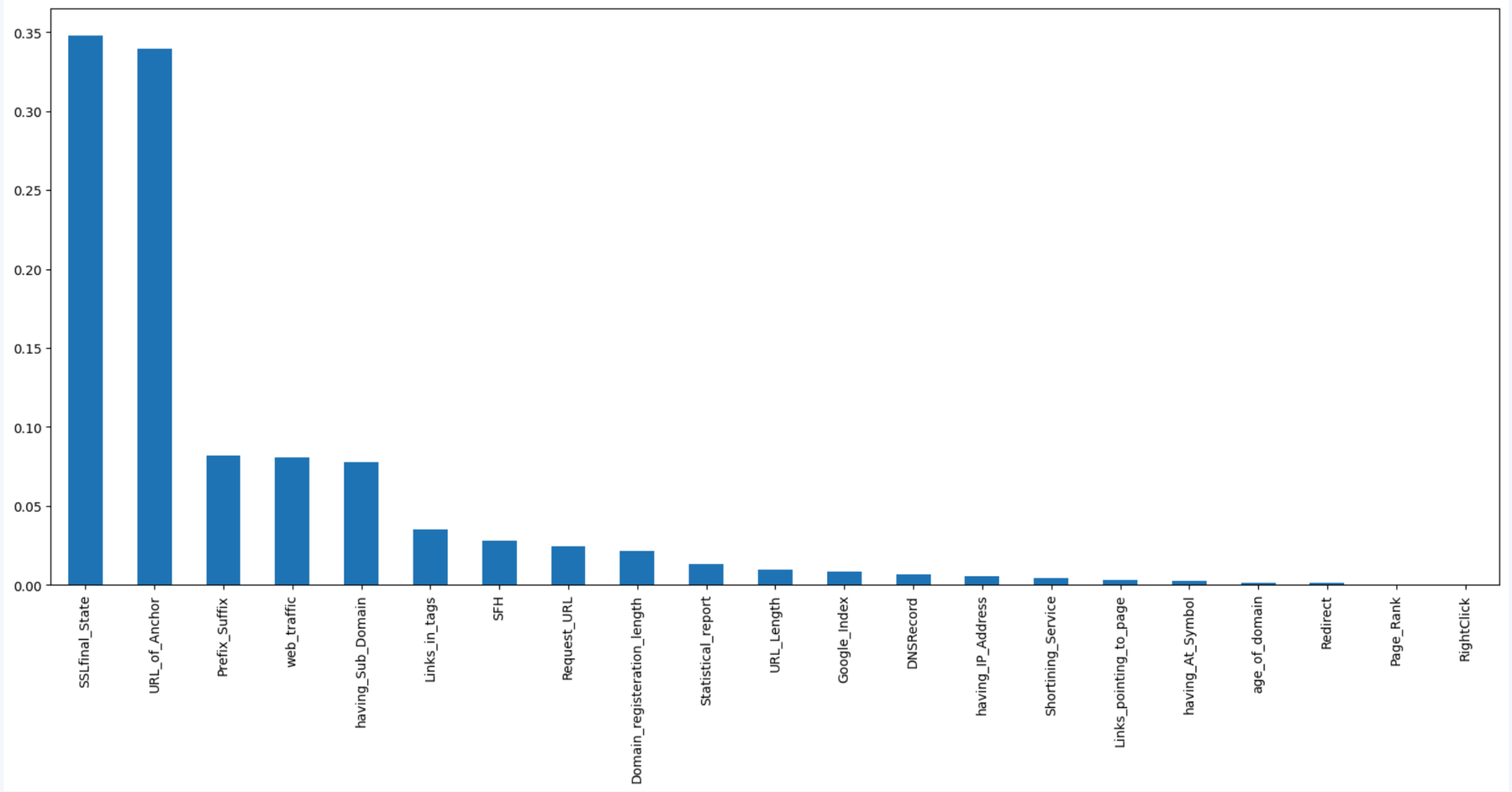


Fig. 9

Information gain for the remaining attributes and dropping 3 attributes

```
[ ]  #removes all low(zero)-variance features
     from sklearn.feature_selection import VarianceThreshold
     var_thres = VarianceThreshold(threshold=0)
     #drop features with variance 0 , remove the features that have the same value in all samples
     var_thres.fit(df)
```

```
VarianceThreshold(threshold=0)
```

```
[ ]  #finding non-constant features
     # get_support() returns integer index, of the features selected, features that have value more than threshold are retented
     var_thres.get_support(indices=False)
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

Fig. 10

Applying variance on remaining attributes

Classfication Models Used

Decison Tree

Without dropping features - **0.9435**

After dropping features - **0.9412**

Measure for classification : **Entropy**
Maximum depth : **10**

Random Forest

Without dropping features - **0.953**

After dropping features - **0.9525**

Maximum Depth : **10 levels**

Naive Bayes

Without dropping features - **0.6099**

After dropping features - **0.6027**

XGBoost Classifier

Without dropping features - **0.9263**

After dropping features - **0.9263**

Learning Rate : **0.1**
Maximum Depth : **10 levels**

Comparison of Accuracy of various Classification Model

(All 30 features)

	ML Model	Train Accuracy	Test Accuracy	Precision	Recall	f1 score	ROC_AUC	Time
0	Decision Tree	0.9582	0.9435	0.9501	0.9469	0.9485	0.9431	old
1	Random Forest	0.9617	0.9530	0.9416	0.9749	0.9580	0.9505	old
2	Naive Bayes	0.6005	0.6099	0.9958	0.2916	0.4512	0.6451	old
3	XGBoost	0.9308	0.9263	0.9068	0.9650	0.9350	0.9220	old

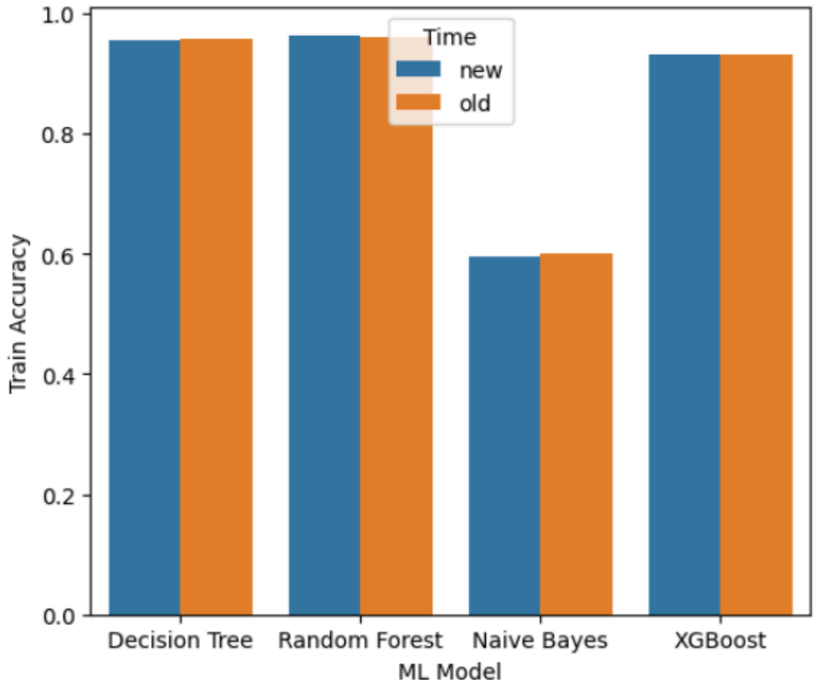
Comparison of Accuracy of various Classification Model

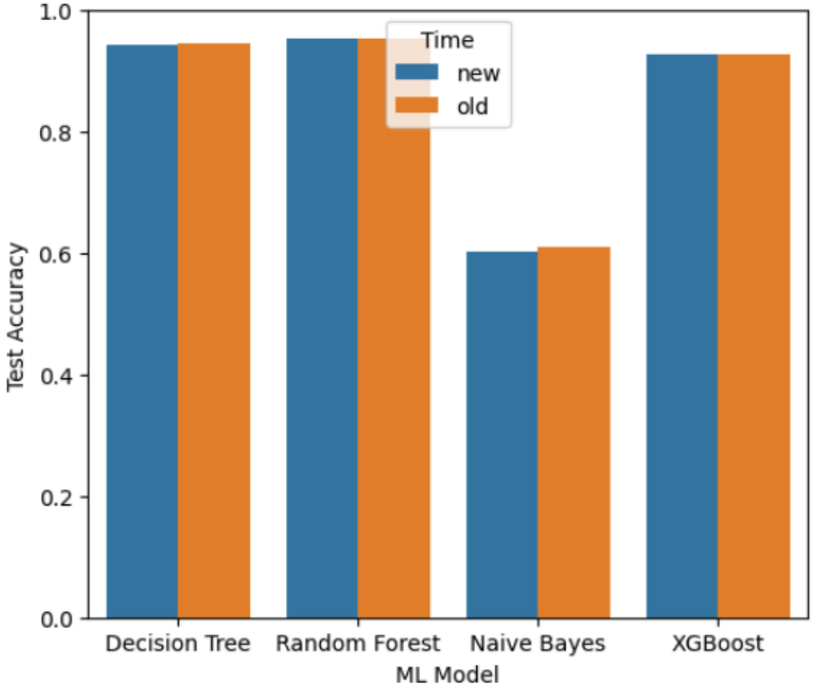
(19 features)

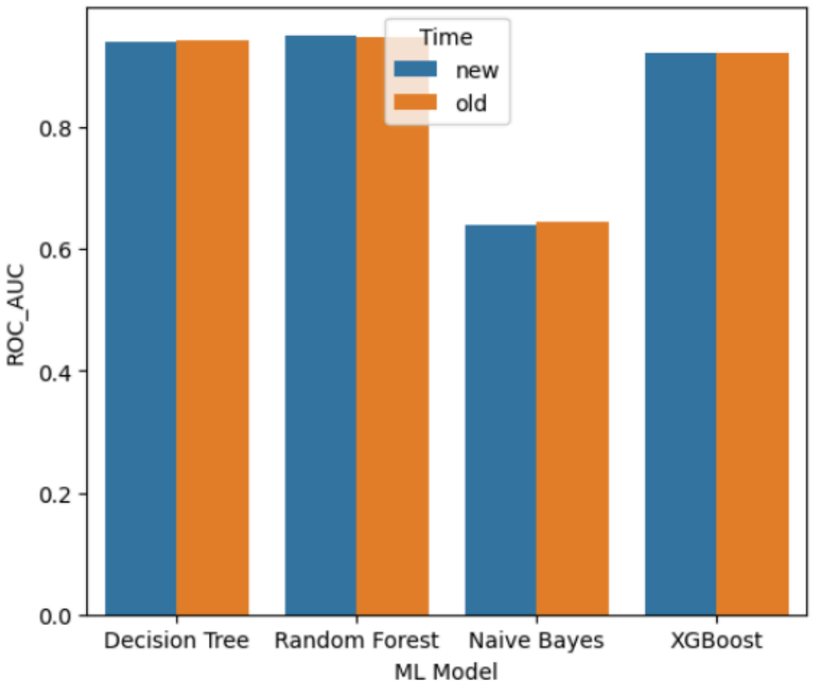
	ML Model	Train Accuracy	Test Accuracy	Precision	Recall	f1 score	ROC_AUC	Time
0	Decision Tree	0.9566	0.9412	0.9436	0.9498	0.9467	0.9403	new
1	Random Forest	0.9631	0.9525	0.9408	0.9749	0.9576	0.9500	new
2	Naive Bayes	0.5945	0.6027	1.0000	0.2773	0.4341	0.6386	new
3	XGBoost	0.9308	0.9263	0.9068	0.9650	0.9350	0.9220	new

CONCLUSION

Based on the results of the experiments performed we can conclude that **Random Forest Classifier** gave the best results with the **train accuracy of 0.96 and test accuracy of 0.95**. In addition to it, we can also conclude that the accuracy of the classification was increased after the feature selection, even if it is increased in a very small proportion, but it has increased.







The following bar chart depicts the test accuracy, train accuracy, and roc_accuracy of classification models before feature selection and after feature selection.

Limitations of our implementaion

01

Considering parameters of URL as hard-coded and not performing feature extraction directly from the URL

02

This methodology works on URL features and does not consider any textual content of the websites

Future Work

01 Including textual features along with url features for classification

02 Including feature extraction code, so that features can be extracted from a given URL.

03 Other classification algorithms can be used to compare the results and conclude the best classification algorithm.

REFERENCES

1. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system By Kang Leng Chiewa, Choon Lin Tan, KokSheik Wong, Kelvin S.C. Yong, Wei King Tiong
2. An effective detection approach for phishing websites using URL and HTML features By Ali Aljofey, Qingshan Jiang, Abdur Rasool, Hui Chen, Wenyin Liu, Qiang Qu & YangWang
3. A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms By M. Amaad Ul Haq Tahir , Sohail Asghar , Ayesha Zafar, Saira Gillani


THANK
YOU