

**Feature Extraction Based Ensemble Stacking for Combating Cyber Threat in Phishing URLs ​**

*A project report submitted in partial fulfilment of the requirement*

*for the award of degree of*

**BACHELOR OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

*Submitted by*

|  |  |
| --- | --- |
| **T. Lavanya**  **(19341A05D6)** | **P. Satyamani Sai**  **(19341A05D9)** |
| **Renuka Kola**  **(19341A05E4)** | **Sri Samhita. P**  **(19341A05D6)** |
| **Sk Sadhik Basha**  **(19341A05F6)** | 1. **Hari Krishna**   **(19341A05G6)** |

**U. Manikanta**

**(19341A05H4)**

*Under the esteemed guidance of*

**Mr D.Siva Krishna**

Assistant Professor, Dept. of CSE

**GMR Institute of Technology**

**An Autonomous Institute Affiliated to JNTUK, Kakinada**

(Accredited by NBA, NAAC with ‘A’ Grade & ISO 9001:2008 Certified Institution)

**GMR Nagar, Rajam – 532127,**

**Andhra Pradesh, India**

**October 2022**



**Department of Computer Science and Engineering**

**CERTIFICATE**

This is to certify that the thesis entitled **Feature Extraction Based Stacking model for combating Cyber threat in Phishing URLs** submitted by **T. Lavanya(19341A05H0), P. Satyamani Sai (19341A05D9), Renuka Kola (19341A05E4), Sri Samhita. P (19341A05F9), Sk Sadhik Basha (19341A05F6), T. Hari Krishna(19341A05G6)​, U. Manikanta(19341A05H4)​**has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **GMRIT, Rajam** affiliated to **JNTUV, VIZIANAGARAM** is a record of Bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

**Signature of Supervisor Signature of HOD**

**Mr.D.Siva Krishna Dr. A. V. Ramana**

Assistant Professor Professor & Head

Department of CSE Department of CSE

GMRIT, Rajam. GMRIT, Rajam.

The report is submitted for the viva-voice examination held on ……………….

Signature of Internal Examiner Signature of External Examiner

**ACKNOWLEDGEMENT**

It gives us an immense pleasure to express deep sense of gratitude to my guide **Mr.D.Siva Krishna,** Assistant Professor, Department of Computer Science and Engineering for his whole hearted and invaluable guidance throughout the project work. Without his sustained and sincere effort, this project work would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr. C. L. V. R. S. V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Computer Science Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

**T. Lavanya(19341A05H0)​**

**P. Satyamani Sai(19341A05D9) ​**

**Renuka Kola(19341A05E4)​**

**Sri Samhita. P(19341A05F9)​**

**Sk Sadhik Basha(19341A05F6)​**

**T. Hari Krishna(19341A05G6)​**

**U. Manikanta(19341A05H4)​**

**ABSTRACT**

Cyber threat is a process of exploiting sensitive information by losing the golden triad (Confidentiality, Integrity, and Availability). The threats may include Trojans, Phishing, and DDOS attacks. Phishing works as a social engineering technique where in the malicious attacker attempts in such a way that the appearance to the user would seem legitimate thereby causing the user to fall in a trap which may lead to the compromising of the user’s sensitive information by means of illegitimate URLs or links etc. According to previous records approximately 18% of businesses lost their financial information due to phishing attack. Phishing can exploit data such as credentials, personal data, medical, etc. In this work we proposed a feature extraction-based stacking model where the URL (uniform resource locators) is identified as a legitimate or illegitimate URL. In the process of feature extraction, the features are considered based on hostname\_length, path\_length, fd\_length, tld\_length, count of -, count of @, count of? count of %, count of., count of =, count of http, count of https, count of www, count of digits, count of letters, dir., Ip (v6 or v4). We trained the model with different machine learning algorithms for chosen the best models for stacking the models. We considered the best four machine learning algorithms to be stacked and a meta layer for the final prediction. We compare the metrics such as accuracy, recall, precision, and F1-score with other state of art models.

**Keywords:** Data Pre-processing, Tokenization, Word Embedding, Stacking, Machine learning,Feature Extraction.*​*

**TABLE OF CONTENTS**

**ACKNOWLEDGEMENT**  iii

**ABSTRACT** iv

**LIST OF TABLES** v

**LIST OF FIGURES** vi

**LIST OF SYMBOLS & ABBREVIATIONS** vii

**1. INTRODUCTION** 1

**1.1** INTRODUCTION TO PHISHING

**1.2** INTRODUCTION TO MACHINE LEARNING

**1.3** INTRODUCTION TO STACKING

**1.4** INTRODUCTION TO FEATURE ENGINEERING

**2. RELATED WORK**

**3. SYSTEM DESIGN**

**4. DATASET**

**5.** **METHODOLOGY**

**5.1** PREPROCESSING

**5.2** FEATURE ENGINEERING

**5.3** VECTORIZATION AND WORD2VEC

**5.3.1** TFID

**5.3.2** WORD2VEC

**5.4 S**TACKING

**6. IMPLEMENTATION**

**7.** **EXPERIMENTAL STUDY**

**4. RESULTS AND DISCUSSIONS**

**5**. **CONCLUSIONS**

**7. LIMITATIONS AND FUTURE SCOPE**

**REFERENCES**

**BIBLIOGRAPHY** 60

**LIST OF PUBLICATIONS**

**LIST OF TABLES**

**TABLE NO**  **TITLE**  **PAGE NO**

1 Dataset Samples 12

2 Performance of different machine algorithms with WordVec 23

3 Performance of different machine algorithms with TF-IDF

4 Performance of different machine algorithms with proposed method

**LIST OF FIGURES**

**FIGURE NO** **TITLE** **PAGE NO**

1 Fraud mail from attacker 12

2 Phishing link in a mail 23

3 Flow Chart

4 URL Pattern

5 Data set

6 Splitting the data set

7 TF-IDF

8 Word2Vec

9 Ensemble Stacking

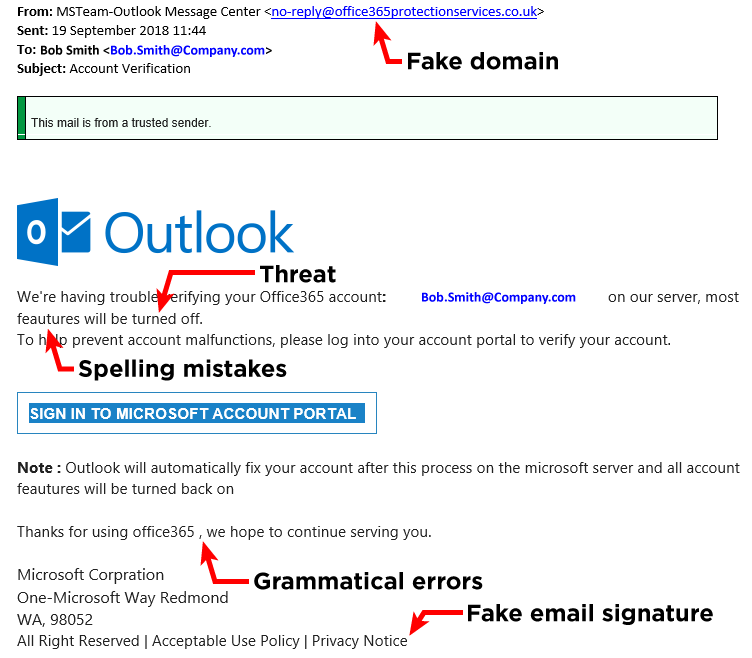
**LIST OF SYMBOLS & ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| **S NO** | **SYMBOL** | **ABBREVIATIONS** |
| 1 | MITM | Man In the Middle |
| 2 | URL | Unified Resource Locator |
| 3 | UI | User Interface |
| 4 | HTTP | Hyper Text Transfer Protocol |
| 5 | HTTPS | Hyper Text Transfer Protocol (Secure Socket Layer) |
| 6 | FTP | File Transfer Protocol |
| 7 | NLP | Natural Language Processing |
| 8 | IP | Internet Protocol |
| 9 | IPv4 | Internet Protocol Version 4 |
| 10 | IPv6 | Internet Protocol Version 6 |
| 11 | TF-IDF | Term Frequency Inverse Document Frequency |
| 12 | FD length |  |
| 13 | TLD | Top Level Domain |
| 14 | WWW | World Wide Web |
| 15 | IR | Information Retrieval |
| 16 | CBOW | Counting Bag of Words |
| 17 | W2V | Word to Vector |
| 18 | PERT | Project Evaluation Review Technique |
| 19 | Gaussian | Gaussian Naive Bayes |
| 20 | SVM | Support Vector Machine |
| 21 | KNN | K-Nearest Neighbors |
| 22 | TWSVM | Twin Support Vector Machine |

**1. INTRODUCTION**

**1.1. Introduction to phishing**

A cyber-attack is an attempt to disable computers, steal data, or use a breached computer system to launch additional attacks. Cybercriminals use different methods to launch a cyber-attack that includes malware, phishing, ransomware, MITM Attack, or other methods. Frequency of attacks: 66% have experienced a cyber-attack in the past 12 months. Background of attacks: 69% say that cyber-attacks are becoming more targeted. Phishing remains a major concern, not only due to an increase in the number of phishing attacks, but also due to the sophisticated methods used by attackers to carry out the attack. Phishing attacks became profitable and simple to carry out thanks to sophisticated techniques such as the use of Phishing toolkits and email flooding. Phishing is an attack in which an attacker attempts to trick users into disclosing sensitive and personal information such as credit card numbers, passwords, and so on. The attacker's goal in carrying out a phishing attack is to sell the victims' identities, obtain ransom, exploit system vulnerabilities, or gain financial benefits. Making a false website that is a copy of a legitimate website (like PayPal, eBay, etc.) and hosting it on a compromised, free, or paid domain is one such typical assault. Human eyes find it challenging to distinguish between authentic and fraudulent web pages since they look similar. Sensitive data will be stolen via scripts after the user accesses the false website. Each year, more people utilise the Internet, which leads to an increase in phishing assaults. Attackers can carry out phishing assaults using emails, websites, and malicious software**.** According to the report,36% of all security breaches are a direct result of phishing attacks. The number of brands being targeted in phishing attacks declined in December after months of growth. However, there were still over 500 attacks, which shows how cybercriminals are extending the scope of their attempts to lure in victims. Facebook and Google: Between 2013 and 2015, Facebook and Google were tricked out of $100 million due to an extended phishing campaign. The phisher took advantage of the fact that both companies used Quanta, a Taiwan-based company, as a vendor. This is one of the famous phishing attacks which had led to data breach. Phishing is an email-based deception where a perpetrator (phisher) camouflages emails to appear as a legitimate request for personal and sensitive information. This is the reason people get phished. Only phishing links lasts Based on the data, according to number of observations several observations. Hosted phishing pages become inactive faster than the others. A quarter of the pages survived for no more than 8 hours, and only 12.3% of all pages remained active after 30 days. According to the survey, 99percent of the phishing occurs through emails and remaining 3 percentage occur through malicious websites which can cause loss of confidential information or sensitive information. Cyber criminals use phishing attack to breach the data because it is easy, cheap and effective. Email addresses are easy to obtain, and emails are virtually free to send. With little effort and cost, attackers can quickly gain access to valuable data.

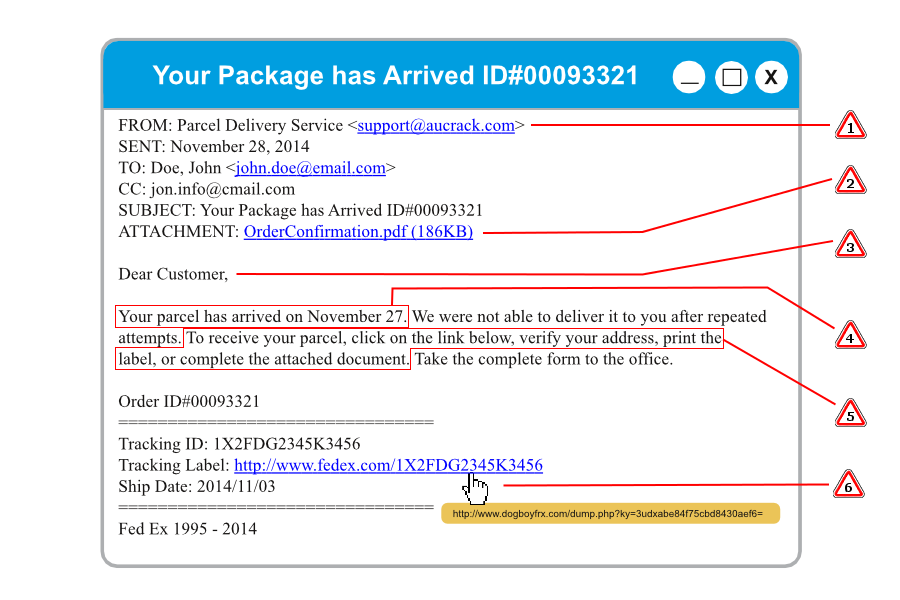
 *Figure 1. Fraud mail from attacker*

Some of the ways to detect phishing mails is: The message is sent from a public email domain. The domain name is misspelt. The email is poorly written. It includes infected attachments or suspicious links. The message creates a sense of urgency. Prevent phishing by educating your employees.

Phishing awareness is not only about knowing the definition of phishing but employees in an organization need to be trained about:

* The various kinds of phishing attacks
* What a phishing email looks like
* How to respond to emails that request personal information
* How it could affect the organization, and, in turn, the employees
* Recent instances of phishing attacks and how it cost millions for organizations around the world

Employees also need to be taught about how to respond and report potential phishing attacks immediately to the internal security teams. Immediate reporting will help security teams to alert other employees and contain the threat to a great extent.

*Figure 2. Phishing link in a mail*

**1.2. Introduction to Machine Learning**

Machine Learning algorithms will be used for the prediction of Phishing URLs. In the previous works it was observed that the machine learning algorithms are used. According to previous, we can see that machine learning algorithms like ensemble Machine Learning and combination of machine learning and deep learning algorithms, KNN, Random Forest classifier and decision tree, kernel-SVM. In few papers CNN model was used to recover word-level feature representations of URLs and attention based hierarchical recurrent neural network (RNN) module to extract character-level spatial feature representations of URLs. While, some papers produced LSTM technique to identify malicious sites and the accuracy achieved is 86%. Then there is another paper which analyzed the following algorithms SVM, KNN, Decision Tree classifier and random forest classifier where the accuracy obtained is about 87.32%. The main limitation in the previously published papers is that phishing URLs are not completely identified by the models, and feature selection and other techniques take more time. The Random Forest method performed less accurately compared to others since it took more features into consideration than were necessary to determine whether a URL was valid or not.

**1.3. Introduction to Stacking**

Stacking involves using a machine learning model to learn how to best combine the predictions from contributing ensemble members. In voting, ensemble members are typically a diverse collection of model types, such as a decision tree, naive Bayes, and support vector machine. Combining both the learnings with hybrid stacking will make it more effective to recognize authentic URLs and non-authentic URLs. In our proposed schema, a hybrid stacking model is utilized to determine whether a URL is authentic or not. The URL-related data is first pre-processed, and then words are converted to vectors, which are then fed into multiple machine learning algorithms, from which the best feature selection is made. To train and anticipate the outcomes based on the voting process, these attributes will be provided to various Deep Learning algorithms.

**1.4. Introduction to Feature Extraction**

Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones (and then discarding the original features). These new reduced set of features should then be able to summarize most of the information contained in the original set of features. Feature extraction helps to reduce the amount of redundant data from the data set. In the end, the reduction of the data helps to build the model with less machine effort and increases the speed of learning and generalization steps in the machine learning process. Consideration of more features when training the model is the key issue with several of the models in the earlier works. Therefore, the model can be optimized by excluding unimportant features from consideration. The bulk of models achieved good accuracy by using word embedding approaches rather than feature selection. Every single model that has been proposed either uses deep learning or machine learning.

**2. LITERATURE SURVEY**

1. ***S. Ariyadasa, S. Fernando and S. Fernando,* "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML," in IEEE Access, vol. 10, pp. 82355-82375, 2022, Doi: 10.1109/ACCESS.2022.3196018.**

In the current inquiry and experiment, three datasets were used. One of them is exclusive to our study, whereas the other two may be found in publicly available databases. For ease of use, the datasets were given the designations Dataset A, Dataset B, and Benchmark Dataset. The experiment explored training, testing, and validation on various scales using these three datasets. There are 45003 reliable URLs and 50000 real URLs altogether. To Extract Features examines three main feature sets for existing anti-phishing technology. These are characteristics that are external, content-based, and URL-based. URL-based features are lexical features that may be extracted directly from the URL, as opposed to content-based features, which are obtained directly from HTML text. Random forest, KNN, naive bayes, and deep learning models like LSTM and GRE are some of the machine learning algorithms that have been employed. By comparing the findings, it was found that LSTM outperformed other algorithms. Accuracy, precision, recall, and f1-score for URL detection are the performance measures. Accuracy, precision, recall, and f1 score were attained at 96.42%, 96.40%, and 96.44%, respectively.

**[2] *M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki and V. González-Castro,* "Phishing URL Detection: A Real-Case Scenario Through Login URLs," in IEEE Access, vol. 10, pp. 42949-42960, 2022, Doi: 10.1109/ACCESS.2022.3168681.**

A Real-Case Scenario Through Login URLs produced a new dataset called Phishing Index Login URL (PILU-90K), which is made up of 30K phishing URLs and 60K genuine URLs, comprising login and index websites. Raw words are extracted from the various parts of the URL by segmenting the string using several symbols, including "/," "-," ".," "@," "?"," "?", "&," "=", Eight supervised classifiers are trained and compared to the model, including Light Gradient Boosting Machine, Extreme Gradient Boosting, Adaptive Boosting, Random Forest, Support Vector Machines, k-Nearest Neighbors, Nave Bayes, and Logistic Regression. Accuracy, precision, recall, and f1-score are the performance measures for URL detection. When combined with Term Frequency - Inverse Document Frequency (TF-IDF) feature extraction, the logistic regression model achieves 96.50% accuracy on the recently released login URL dataset.

**[3] *Anurag Pandey, Jay Chadawar 2022*, “Phishing URL Detection using Hybrid Ensemble Model,” INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 04 (April 2022).**

A total of 20,000 normal and malicious URL combinations makes up the dataset under consideration. The dataset's URLs are sent to a variety of features, which, depending on the circumstances, return 0 or 1. The returned values are then tabulated saved in a csv file. Training and testing data are separated from the dataset in varying ratios. The dataset is applied to the imported classifiers, and the corresponding accuracies are determined. To create weak learners in this work, we define a set of models a variable number of times. Finally, the Max Voting Classifier approach is employed, and the ensemble model's final class prediction will be the one that has been primarily predicted by the weak learners. This hybrid model has an accuracy of 85.37%.

**[4] *Maheshwari, Shantanu & Janet, B & Kumar, R. (2021),* “Malicious URL Detection: A Comparative Study”.1147-1151. 10.1109/ICAIS50930.2021.9396014.**

A 450000 URLs public dataset from Kaggle is the data's source. The top classifier locates harmful URLs on the publicly accessible phishing website. The collection includes labels for the URLs that indicate whether they are malicious or benign. There are 104438 harmful URLs and 104438 benign URLs. Pre-processing includes management of missing data, extrapolation of new features, normalization, encoding of categorical values, and standardization of values. The number of layers and kernels affect the CNN's performance in general. Thesklearn Python Library is used to train the model on 80% of the data using several machine learning techniques, such as Nave Bayes, Nave Regression, Stochastic Gradient Descent, Random Forest, Support Vector Machine, K-Nearest Neighbors, and Decision Tree. The metrics recall, precision, support, and f1 were used to train and assess a variety of models. The Random Forest model produced the best results. The recall is 91.2%, the recall score is 90%, the accuracy is 97%, and the precision is 91%.

**[5] *M. H. Alkawaz, S. J. Steven, A. I. Hajamydeen and R. Ramli,* "A Comprehensive Survey on Identification and Analysis of Phishing Website based on Machine Learning Methods," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2021, pp. 82-87, Doi: 10.1109/ISCAIE51753.2021.9431794.**

Author offered a Comprehensive Survey on Identifying and Analysing Phishing Websites. Phish Alert fared better on an experimental dataset of 500 phishing sites and 500 real sites. While the phishing URL was received using phish tank, the real URLs were retrieved through the stuff gate server. Data tokenizing techniques like stemming and lemmatization have been utilized for pre-processing. Decision Tree, Random Forest, and Support Vector Machine are the machine learning models that were applied in this instance. The model's accuracy has altered based on the splitting ratio. Accuracy, a false positive rate, and a false negative rate are the measures used for assessment. The accuracy is 97.14 and false positive rate 2.62%, false negative rate is 3.14 for Random Forest model training and testing.

**[6] *Rao, R.S., Pais, A.R. & Anand, P*. “A heuristic technique to detect phishing websites using TWSVM classifier. Neural Computer & Application 33, 5733–5752 (2021)”, https://doi.org/10.1007/s00521-020-05354-z.**

A heuristic method is used to identify phishing websites. The dataset is derived from Alexa's ranking of the top websites, and the phishing websites are gathered from the Phish Tank website. In contrast to other research, we chose 5500 trustworthy websites at random from the Alexa database, avoiding bias toward highly ranked websites. Moreover, 5500 phishing URLs. The new characteristics are created using TF-IDF words similarity, copyright similarity, title similarity, description similarity, maximum frequency domain similarity, and filename similarity. The machine learning methods Support Vector Machine, Twin Support Vector Machine, and Proximal Support Vector Machine have been used to train and test the dataset. Using a Dell Precision T1700 CPU and 16 GB of RAM, the SVM, PSVM, and TWSVM machine learning algorithms have all been simulated in the MATLAB 12.0 environment. The criteria employed are recall, specificity, precision, f1 score, and accuracy; TWSVM received results of 98.05% accuracy, 97.77% specificity, 98.33% recall, 98.74% precision, and 98.03% f1 score.

**[7] *P. L. Indrasiri, M. N. Halgamuge and A. Mohammad*, "Robust Ensemble Machine Learning Model for Filtering Phishing URLs: Expandable Random Gradient Stacked Voting Classifier (ERG-SVC)," in IEEE Access, vol. 9, pp. 150142-150161, 2021, Doi: 10.1109/ACCESS.2021.3124628.**

A reliable and acceptable dataset for an ML-based detection technique for URL validity predictions is Expandable Random Gradient Stacked Voting Classifier (ERG-SVC). A dataset including 102400 legitimate URLs and 137375 phishing URLs is utilized to accomplish this study. A feature extraction process is used to determine whether a website is authentic if it has the following characteristics: IP, pagerank0, position (/)>7, non-standard port number, short URL with https in the domain, URL length>75, and domain age= (10 months). Machine learning classifiers including DT, RF, Boost, AdaBoost, KNN, Gradient Boost, and Logistic Regression were used in this project. The Gradient Boost method outperformed the other algorithms in terms of accuracy, precision, recall, and f1 score. Accuracy 96.71%, precision 96.34%, recall 97.2%, and f1 score 96% were attained.

**[8] *A. El Aassal, S. Baki, A. Das and R. M. Verma*, "An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs," in IEEE Access, vol. 8, pp. 22170-22192, 2020, Doi: 10.1109/ACCESS.2020.2969780.**

Authorutilized 10,000 emails that included URLs in total. It retrieved 8,433 emails from the Nazario phishing email collection in addition to 1,048 emails from its recently released 2015 to 2017 emails. The 1,019 Spam Assassin emails were included. Utilizing Information Gain, the Gini Index, the chi-square test, and Recursive Feature Elimination, the features are processed and ranked. SVM, Random Forest, Decision Tree, Gaussian & Multinomial Naive Bayes, Logistic Regression, K Nearest Neighbors, Boosting, Bagging, Online Learning, Deep Neural Networks, Imbalanced learning, and Hellinger Distance Decision Tree are the models that are employed in this model. Accuracy, Recall, F1-score, Precision, Geometric Mean, Balanced Detection Rate, and Matthew's Correlation Coefficient are the performance indicators for identifying URLs. Gaussian Naive Bayes among trained models had an accuracy of 94.7%, F1 score of 94.36%, GMean of 94.48%, and BDR of 17.35%.

**[9]  *X. Liu and J. Fu*, "SPWalk: Similar Property Oriented Feature Learning for Phishing Detection," in IEEE Access, vol. 8, pp. 87031-87045, 2020, Doi: 10.1109/ACCESS.2020.2992381.**

There are 0.5 million harmful URLs from Phish Tank and Open Phish and 1 million reliable URLs from Alexa and Dmoz. These annotated URLs were used in the calculation of the URL quality score. The TF-IDF will be used to separate the numeric data from the word data once the URLs have been separated into three training layers, L0, L1, and L2. Network embedding models like Deep Walk and Node2vec as well as conventional phishing detection models like URL Pattern Mining and Website Logo are employed as embedding strategies in this model. the evaluation of the SPWalk, Deep Walk, and Node2vec findings. In this case, the SPWalk consistently outperforms both Deep Walk and Node2vec.Precision, which is 95%, has been utilized as the parameter to assess performance.

**[10] *Ramana, A. & Rao, K. & Rao, Routhu. (2021)*. “Stop-Phish: an intelligent phishing detection method using feature selection ensemble. Social Network Analysis and Mining”11. 10.1007/s13278-021-00829-w.**

The 11055 webpages that make up the UCI repository's dataset for the experiment are used. The remaining 4898 are phishing websites, leaving 6157 legal websites. To find the best classifier, we applied a variety of machine learning methods and created an ensemble model using the Random Forest, Decision tree, and Boost algorithms. According to our experimental research, we were able to detect phishing attempts with an accuracy of 97.51% using a dataset from UC Irvine (Dataset 1) and 98.45% using a dataset from Mendeley (Dataset 2). Additionally, the proposed model significantly outperformed baseline models.

**[11] *Kalabarige, Lakshmana & Rao, Routhu & Abraham, Ajith &Gabralla, Lubna. (2022).* “MLSELM:Multi-layer Stacked Ensemble Learning Model to detect phishing websites.” IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3194672.**

Multi-layer stacked ensemble learning model consists of estimators at several layers, with predictions from the current layer serving as input to the subsequent layer. The dataset utilized has 10052 records, each with a unique URL and information about how valid or not they are. The suggested model Random Forest is tested using data from Mendeley 2018 (D2), Mendeley 2020 (D1), and UCI (D1) (D3, D4). With the D1 dataset, the suggested model has a detection rate of 97.76% and a D2 dataset accuracy of 98.9%. Finally, D3 and D4 are used to test the approach, with accuracy of 96.79% and 98.43%, respectively.

**[12] *Aljofey, A.; Jiang, Q.; Qu, Q.; Huang, M.; Niyigena, J.-P*. “An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. Electronics2020”, 9, 1514. https://doi.org/10.3390/electronics9091514.**

Authors does not call for retrieving the content of the target website or utilizing any third-party services. Without having any prior knowledge of phishing, it collects data and sequential patterns of URL strings, using the sequential pattern features for quick classification of the actual URL. Comparisons between numerous classical machine learning models and deep learning models are offered for evaluations utilizing a range of feature sets, including hand-crafted, character embedding, character level TF-IDF, and character level count vectors features. The suggested model, according to the experiments, had accuracy of 95.02% on the dataset and 98.58%, 95.46%, and 95.22% on benchmark datasets.

**[13]  *Areti Nagendra Soma Charan, Yu-Hung Chen, Jiann-Liang Chen*, "Phishing Websites Detection using Machine Learning with URL Analysis", 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), pp.808-812, 2022.**

Author uses URLs as a dataset to identify phishing websites. Ten features were retrieved from the dataset, which contains 6000 URLs, and were then used to identify whether a website was phishing. For this study, eight machine learning algorithms were created. According to the performance analysis findings, when compared to other algorithms, the Multilayer Perceptron algorithm had the greatest accuracy (85.41%) and F1 score (85.17%).

**[14] *Safa Alrefaai, Ghina Özdemir, Afnan Mohamed*, "Detecting Phishing Websites Using Machine Learning", 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp.1-6, 2022.**

Author is intended to use machine learning to identify phishing websites. We took data from Kaggle that included 11,430 URLs altogether, 11,430 features, and half of them were phishing sites. The Decision Tree (DT), Random Forest (RF), Boost, Multilayer Perceptron, K-Nearest Neighbors, Naive Bayes, AdaBoost, and Gradient Boosting models were used to train our data, and XG Boost produced the model with the highest accuracy (96.6%).

**[15] *Salloum, S., Gaber, T., Vadera, S., Shaalan, K. (2021).* “Phishing Website Detection from URLs Using Classical Machine Learning ANN (Artificial Neural Network) Model.” In: Garcia-Alfaro, J., Li, S., Poovendran, R., Debar, H., Yung, M. (eds) Security and Privacy in Communication Networks. SecureComm 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 399. Springer, Cham. https://doi.org/10.1007/978-3-030-90022-9\_28.**

The author employed two datasets of 58,645 and 88,647 URLs that were classified as "Phishing" or "Legitimate" to train the system. There are several different machine learning models that are examined, including "XGBOOST, Support Vector Machine (SVM), Random Forest (RF), k-nearest Neighbour (KNN), Artificial neural network (ANN), Logistic Regression (LR), Decision tree (DT), and Gaussian naive Bayes (NB)" classifiers. In tests, ANN demonstrated the best performance with a 97.63% accuracy rate for phishing URL detection.

**[16] *Brij B. Gupta, Krishna Yadav, Imran Razzak, Konstantinos Psannis, Arcangelo Castiglione, Xiaojun Chang," A* novel approach for phishing URLs detection using lexical based machine learning in a real-time environment,” Computer Communications, https://doi.org/10.1016/j.comcom.2021.04.023.**

Authorused the ISCXURL-2016 dataset to test 11964 instances of legal and phishing URLs. The characteristics were taken from the URLs and subjected to training against various machine learning classifiers, with the Random Forest technique yielding the highest accuracy of 98.07%.

**[17] *Youness Mourtaji, Mohammed Bouhorma, Daniyal Alghazzawi, Ghadah Aldabbagh, Abdullah Alghamdi,* "Hybrid Rule-Based Solution for Phishing URL Detection U ", Wireless Communications and Mobile Computing, vol. 2021, Article ID 8241104, 24 pages, 2021. https://doi.org/10.1155/2021/8241104.**

Author used to incorporate 37 features taken from six different methods, including the black listed method, the lexical and host method, the content method, the identity method, the identity similarity method, the visual similarity method, and the behavioural method. Later Different deep learning and machine learning models, including MLP and CNN (Convolution Neural Network), are utilized for training, in addition to machine learning models like CART, SVM, and KNN. The study's conclusions showed that the approach was successful in analysing URL stress from various angles, supporting the model's validity. However, deep learning was able to achieve the highest accuracy level with the provided values of 97.945 for the CNN model and 93.216 for the MLP model.

**[18] *P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar and A. Gururajan,* "URL Tran: Improving Phishing URL Detection Using Transformers," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), 2021, pp. 197-204, Doi: 10.1109/MILCOM52596.2021.9653028.**

Author proposed a model employing transformers. The author compared these models to refined BERT and Roberta models while also considering extra domain-specific pre-training tasks and standard masked language modelling. Combining the information from these studies, the author derived URL Tran, which outperforms existing deep learning-based algorithms in terms of phishing URL identification over a wide range of extremely low false positive rates (FPRs)? When compared to the next best baseline at an FPR of 0.01%, URL Tran, for instance, yields a true positive rate (TPR) of 86.80% as opposed to 71.20%, representing a relative improvement of almost 21.9%.

**[19] *Jalil, S., Usman, M. & Fong, A*. “Highly accurate phishing URL detection based on machine learning. J Ambient Intell Human Compute (2022).” https://doi.org/10.1007/s12652-022-04426-3.**

Authorsuggested a technique based on URL that employs the whole URL, protocol scheme, hostname, path region of the URL, entropy feature, suspicious terms, and brand name matching using TF-IDF technique. Eight different machine learning classifiers were used in the trials, which were conducted on six different datasets. Random Forest significantly outperformed the other classifiers in terms of accuracy across all datasets. On the Kaggle datasets, the suggested framework with just 30 features produced superior accuracy results of 96.25% and 94.65%. The comparison findings demonstrate that the suggested model outperformed the existing methods with accuracy of 92.2%, 91.63%, 94.80, and 96.85% on benchmark datasets.

**[20] *Murthy, K.S.R.C., Bhattacharya, T., Rajagopalan, N. (2022).* “Feature Extraction-Based Phishing URL Detection Using Machine Learning Techniques.” In: Satyanarayana, C., Samanta, D., Gao, XZ., Kapoor, R.K. (eds) High Performance Computing and Networking. Lecture Notes in Electrical Engineering, vol 853. Springer, Singapore. https://doi.org/10.1007/978-981-16-9885-9\_14.**

Author proposed a machine learning technique for feature extraction-based phishing URL detection. The objective of this study is to recognize phishing URLs using address bar and behavioural data. The decision tree method, which the author took into consideration and cross-validated on a few machine learning techniques, is found to be effective in categorizing phishing URLs according to the discovered features. 9581 data points and 41 characteristics make up the dataset used in this study. Finally, author was able to locate phishing URLs with 97% accuracy, which is comparatively greater accuracy than other detection algorithms.

**[21]  *S. -C. Lin, P. -C. Wl, H. -Y. Chen, T. Morikawa, T. Takahashi and T. -N. Lin*, "Sense Input: An Image-Based Sensitive Input Detection Scheme for Phishing Website Detection," ICC 2022 - IEEE International Conference on Communications, 2022, pp. 4180-4186, Doi: 10.1109/ICC45855.2022.98386.**

In this paper author uses to prevent the phishing attacks by using an ensemble machine learning technique called phish haven which can identify AI generated and manufactured phishing URL’s. For feature extraction of URL, he used lexical analysis and introduced HTML encoding to classify URL’s and URL hit approach for easy URL’s. Predicted the phishing URL by unbiased voting mechanism to prevent misclassification when there is equal number of votes. In this proposed model the author divided the URL into five components to extract lexical features from an URL. By this model author gained an accuracy of 98.00% with 100,000 phishing and normal URL’s. This is another benefit of the proposed approach above that which was previously disclosed in as feature engineering takes a lot of effort.

.

**[22] *C. Opara, B. Wei and Y. Chen*, "HTML Phish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, Doi: 10.1109/IJCNN48605.2020.9207707.**

The phishing attack is a social engineering technique that manipulates internet users into revealing confidential information such as personal data that may be exploited for fraudulent purposes to gain money. Machine learning-based anti-phishing techniques typically follow specific approaches such as the required representation of features is firstly extracted, then a phishing detection machine learning model is trained using the feature vectors. Machine learning processing techniques are used to extract specific features such as the number of common phishing words etc. from the components of a web page The impact of phishing attacks on individuals such as identity theft, psychological, and financial costs can be highly destructive or damaging. Cybercrime has recently become common because it is carried out with little technical ability and significant cost and efficient.

**[23] S. *Y. Yerima and M. K. Alzaylaee*, "High Accuracy Phishing Detection Based on Convolutional Neural Networks," 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), 2020, pp. 1-6, Doi: 10.1109/ICCAIS48893.2020.9096869.**

Phishing is a social engineering attack which enable criminals to steal credentials, distribute, money and carry out financial fraud. The convolutional layer extracts the optimal features and reduces the dimensions of the convolutional layer features, and fully connected layer are then used for classification. The performance of the CNN depends on number of layers and the number of kernels. A phishing website detection approach that utilizes only the URL of the website to build detection models. Their system combines RNNs and CNN to extract features from the URL strings to detect attacks and type of malwares. They extracted both character level and word-level features based on URL strings and utilized Convolutional Neural Network for training and testing. The CNN based approach presented in this paper utilizes not only URL features but features from other properties of the websites, which increases robustness, efficiency, and scalability.

**[24] *Y. Huang, J. Qin and W. Wen*, "Phishing URL Detection Via Capsule-Based Neural Network," 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2019, pp. 22-26, Doi: 10.1109/ICASID.2019.8925000.**

There are three types of machine learning algorithms that can be applied on URL detection methods, including supervised learning, unsupervised learning, and semi-supervised learning. And the detection methods are based on URL behaviors. The characteristics of URLs can be divided into two main group. They are static and dynamic. The methods of analyzing static behavior of URLs including Lexical, Content, Host and popularity-based. A new URL is accessed a database query is executed If the URL is blacklisted it is considered a malicious and then a warning will be generated otherwise URLs will be considered as safe.

**[25] *L. Zhang and P. Zhang,* "Phish Trim: Fast and adaptive phishing detection based on deep representation learning," 2020 IEEE International Conference on Web Services (ICWS), 2020, pp. 176-180, Doi: 10.1109/ICWS49710.2020.00030.**

Phishing uses social engineering to steal consumer personal identity data, financial account credentials and steal data. The heuristic-based approach detects phishing attempts by establishing a heuristic rule base search. Some phishing sites do not have the same characteristics, resulting in a high false positive rate and poor adaptability using this mechanism and reduce efficiency. Phish Trim model automatically learns the deep representation of URLs through pre-trained models like Bi-LSTM models and LSTM models. The list-based approach like ML or DL or combination of both maintains a list of information about known phishing sites, which has significant limitations and does not have the ability to detect actual attacks or zero-day attacks. The search engine-based approach extracts feature such as text, images, or links from pages and the information obtained from the search results is used for phishing detection.

**[26] Liang*, Y., Wang, Q., Xiong, K., Zheng, X., Yu, Z., & Zeng, D. (2021)*. “Robust Detection of Malicious URLs With Self-Paced Wide & Deep Learning****.” IEEE Transactions on Dependable and Secure Computing, 19(2), 717-730.**

In this paper, a deep learning system Cyber Len is used for detecting harmful or illegitimate URLs effectively and robustly. Cyber Len consists of two branches, the top branch characterizes. Cyber Len combines the heterogeneous features and considers both the sample complexity and sample diversity. The wide components in Cyber Len capture the latent interactions among statistical features and lexical features. Factorization Machine is used for the learning of latent interaction among the lexical features. The dataset used here is a collection of suspicious URLs from phish tank, whitelist, open phish. Along with the comparative analysis of traditional machine learning methods like Random Forest, K nearest Neighbours, Decision Tree, Linear SVC classifier, one class SVM classifier, and wrapper-based features selection, which contain URL metadata and use the data to determine whether a website is legitimate or not.

**[27] *S. -J. Bu and S. -B. Cho*, "Integrating Deep Learning with First-Order Logic Programmed Constraints for Zero-Day Phishing Attack Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics,** **Speech, and Signal Processing (ICASSP), 2021, pp. 2685-2689, Doi: 10.1109/ICASSP39728.2021.9414850**.

Understanding whether a web page is legitimate or not is an incredibly challenging problem, due to its semantics-based attack structure, which exploits the computer user vulnerabilities. The phishing URL classification research can be categorized into three principal areas: blacklist-based detection (which was studied until early 2010), the modelling of lexicon extracted from the text based on machine algorithms, and the text feature extraction based on the latest deep learning algorithms. To define the declarative semantics for phishing attack modelling, we review the relevant methods based on URL representation and its modelling method. As phishing URLs comprise sequences of characters and words existing CNN and LSTM networks are known as the amenable method for URL feature extraction. The combination of the CNN and LSTM for estimating the time-coefficients is already widely used in the field of LSTM feature modelling.

**[28] *P. Yang, G.*** ***Zhao, and P. Zeng*, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," in IEEE Access, vol. 7, pp. 15196-15209, 2019, Doi: 10.1109/ACCESS.2019.2892066.**

The spread of phishing is no longer limited to traditional modalities such as e-mail, SMS. Though the mobile Internet and social networks have brought convenience to users, they have also been employed to spread phishing, such as QR code phishing, spear phishing and spoof mobile applications. Phishing attacks are hosted on websites that have HTTPS and SSL certificates because many users think that HTPPS websites are legitimate. Blacklists and whitelists are widely used in phishing website detection. The current common browsers convenience blacklists and whitelists to protect users from phishing attacks. Google provides a blacklist of malicious websites that is continuously monitored Users can check the security of URL links through Google Safe Browsing APIs. Phishing website detection based on blacklists and whitelists is easy to implement with high running speed and a low false positive rate.

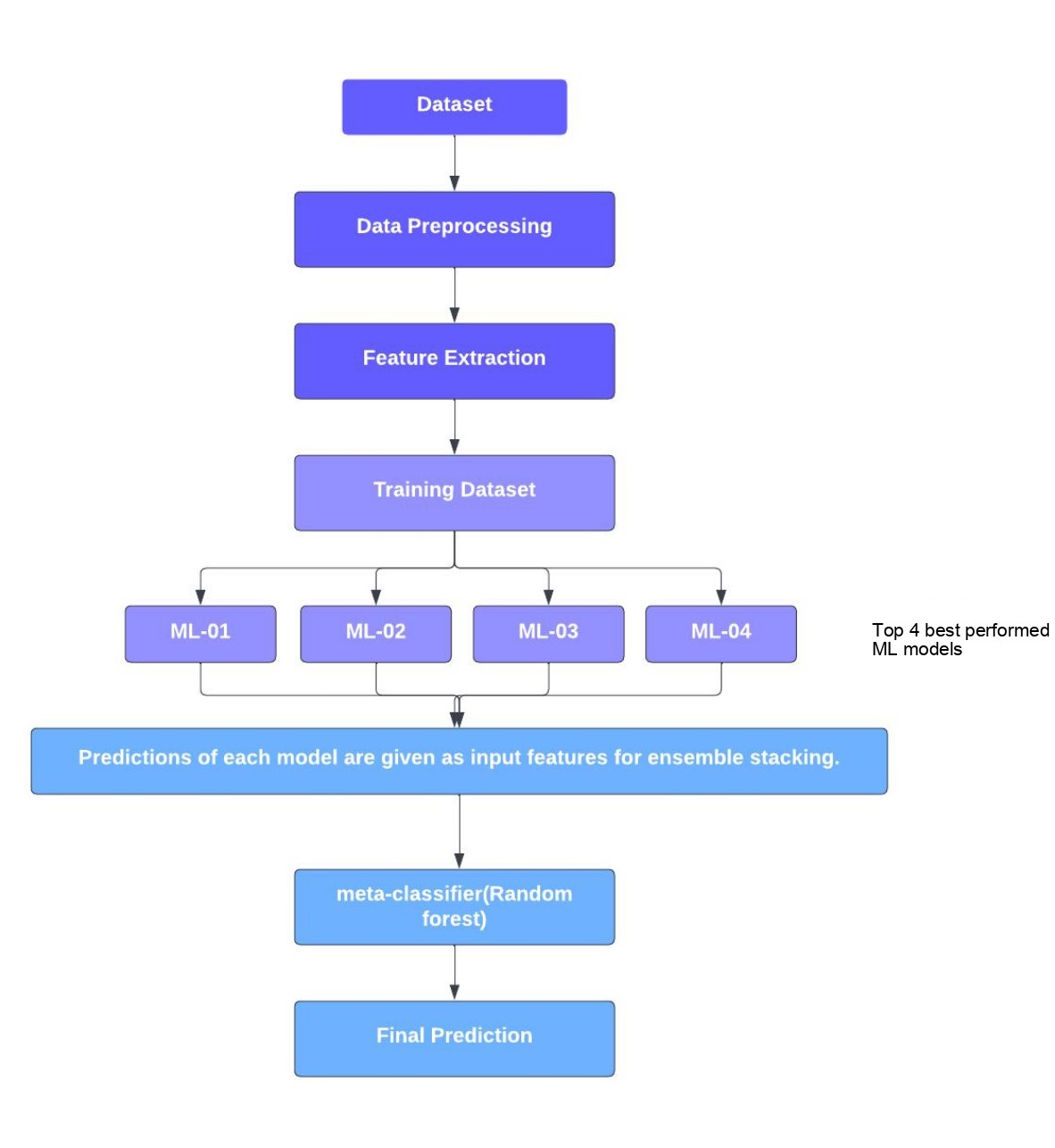
**[29] *Odeh, I. Keshta and E. Abdelfattah*, "Efficient Detection of Phishing Websites Using Multilayer Perceptron", International Journal of Interactive Mobile Technologies (IJIM), vol. 14, no. 11, pp. 22, 2020.**

Phishing is an online fraud that uses swindling websites to attempt to obtain user credentials. A novel approach is used to identify fraud URLs using a multilayer perceptron-equipped neural network. A Multilayer perceptron is a class of feed-forward artificial neural network (FFNN) that consists of more than two layers, the first layer is the input layer and the last one is the output layer and there are some layers between them called hidden layers. As the number of layers is increased, the time complexity is increased. Based on a single attribute evaluator, the proposed model eliminates irrelevant attributes. The next step is to combine attributes and apply the search strategy to remove the redundant data and keep the high correlated attributes. Finally, the system decides if the link is harmful or not. A new phishing detection approach by using a Multilayer perceptron Neural Network. The model applies the processing steps, single attribute evaluator and attribute combine to achieve high accuracy of 98.5% where the Perceptron training ratio is 70%.

**[30] *Anurag Pandey, Jay Chadawar, 2022,* “Phishing URL Detection using Hybrid Ensemble Model,” INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 04 (April 2022).**

The dataset considered is a combination of legitimate and malicious URLs of size 20,000. URLs in the dataset are passed to various features which return 0 or 1 depending on the conditions. The returned values are then stored in a csv in a tabular format. The dataset is divided into training and testing data in variable ratios. The classifiers are imported and applied on the dataset and the respective accuracies are calculated. In this work, we will define some numbers of models a variable number of times to generate weak learners. Then finally, the Max Voting Classifier method is used where the class which has been predicted mostly by the weak learners will be the final class prediction of the ensemble model. An accuracy of 85.37% is achieved by this hybrid model.

**3. SYSTEM DESIGN**

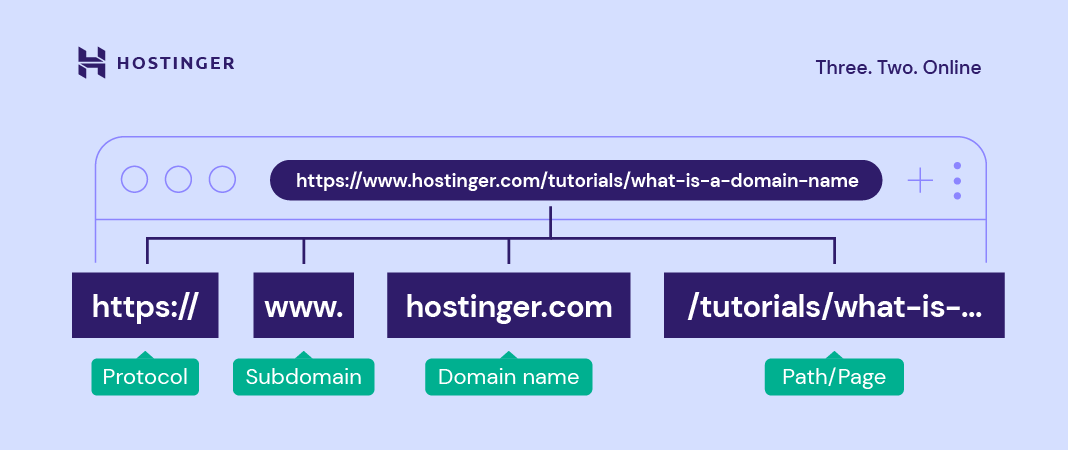


*Figure 3: Flow Chart*

The dataset is preprocessed by removing unnecessary columns after that around 20 features are extracted from each URL but only 17 features are considered for the training of the model using different machine learning algorithms. Around 6 Machine learning algorithms are used for prediction and out of that 4 best accurate machine learning algorithms are given as inputs to ensemble stacking model by taking meta classifier for prediction as Random Forest.

**4. DATASET**

Dataset is a collection of several types of data stored in a digital format. Data is the key component of any Machine Learning project. Datasets primarily consist of images, texts, audio, videos, numerical data points, etc., for solving various Artificial Intelligence challenges such as. Image or video classification. The dataset is created by the team by examining several web portals and the internet, not from any specific website.

  *Figure 4: URL Pattern*

Dataset for this model is a collection of URLs which contain both legitimate and illegitimate.URL is uniform resource locator which contains protocol, host name and page name. URL protocols can be hypertext transfer protocol (HTTP), Hypertext transfer protocol secure (HTTPS), File transfer protocol (FTP). The dataset comprises of 450176 records of phishing URLs and their validity such as whether they are authentic or not. In those records, 104438 phishing URLs outnumber 345738 legal URLs. Two columns in the dataset the URL and its label are present. In the above dataset result 1 means the url is phishing else url is legitimate.

|  |  |  |
| --- | --- | --- |
| **url** | **label** | **result** |
| https://www.google.com | benign | 0 |
| https://www.youtube.com | benign | 0 |
| http://taxcom-online.ru/about/bil.htm | malicious | 1 |
| http://tleg.org/jerry | malicious | 1 |
| http://pacificventurebd.com/biz/ | malicious | 1 |
| https://www.logostelos.info/ | benign | 0 |

*Table 1: Dataset samples*

**5. METHODOLOGY**

**5.1 Data preprocessing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data contains noises, missing values, and in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. Tokenization is used in natural language processing to split paragraphs and sentences into smaller units that can be more easily assigned meaning. The first step of the NLP process is gathering the data (a sentence) and breaking it into understandable parts. Tokenization works in this way: Tokenization substitutes sensitive information with equivalent Non sensitive information. The Non sensitive, replacement information is called a token. Tokenization makes it more difficult for hackers to gain access to cardholder data, as compared with older systems in which credit card numbers were stored in databases and exchanged freely over networks. The main benefits of tokenization include the following. It is more compatible with legacy systems than encryption, it is a less resource-intensive process than encryption, The risk of the fallout in a data breach is reduced. Tokenization stemming is a strategy used to separate the base type of the words by eliminating joins from them. It is very much like chopping down the parts of a tree to its stems. For instance, the stem of the words eating, eats, eaten is eat. Lemmatization procedure is like stemming. The result we will get after lemmatization is called 'lemma', which is a root word as opposed to root stem, the result of stemming. Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning. After lemmatization, we will get a substantial word that implies the same thing. In any case, here stemming, lemmatization cannot be utilized because eliminating the attaches in site causes forecast results wrong. There exists numerous tokens or extraordinary characters which show up most often in any URL series of locales. This trademark include is utilized to separate the URLs into tokens. The pre-processing really looks at the presence of unique characters (//, /, -, @), phish-implied words either in the base URL or way of the whole URL string. Rundown of token words are extricated from URLs and utilized for vectorization.

**5.2 Feature Engineering:**

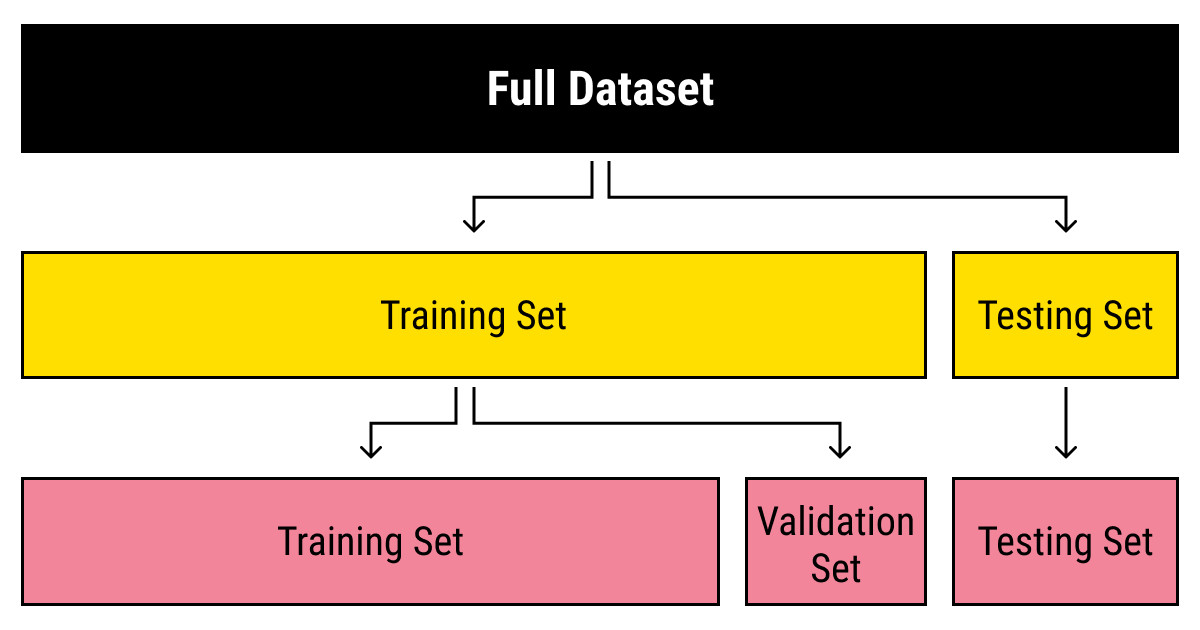
As was previously stated, the issue is not with the algorithms that were used to improve work quality; rather, the proposed model should work with more features without relying on word2vec and TF-IDF; there are other ways to convert features into numbers, such as by using Matlab and feature extraction. The feature extraction logic is used in this proposed model where the features that are been extracted are hostname\_length, path\_length, fd\_length, tld\_length, -, @, %, =, http, https, www, digits, letters, Dir, Ip. By considering the above features the training of the model using various algorithms will be done.

Features which are extracted from the given URL’s used to predict the output for a phishing URL. They are:

Host name\_length: Host name is the string containing domain name of the URL. Each element of a host name must be from 1 to 63 characters long and the entire host name, including the dots, can be at most 253 characters long.

* path\_length: path length is the hop count which is intermediatory nodes between the source and destination.
* fd\_length:
* Tld\_length: It is used in splitting host into domain and sub domain components. It can be in the range of 2 to 63 characters long.
* -: count of hyphens in the URL.
* @: count of the special symbol in URL.
* ?: count of question mark symbol in URL.
* %: count of percentage symbol in URL.
* =: count of equals to symbol in URL.
* http: Hypertext transfer protocol is an application layer protocol.
* https: Hypertext transfer protocol secure is an application layer protocol with SSL certificate which secures the connection between the source and destination.
* www: World Wide Web is an interconnected system of public we page accessible through the internet.
* Digits: counts the number of digits in the URL.
* Letters: counts the number of letters in the URL.
* Dir: Each web page has a URL directory where the URLs are placed hierarchically. It shows the directory in which it is present.
* Ip: Internet protocol address specifies the location. It has versions of IPV4 and IPV6.

Short URL: In this we can determine the URL as a legitimate by the length of the URL



*Figure 6. Splitting the data set*

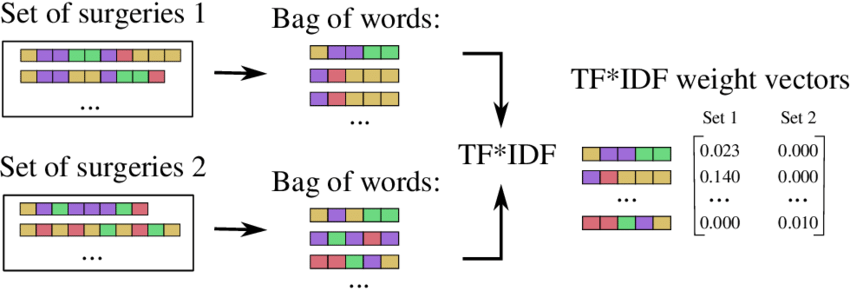
Data splitting is typically done to avoid overfitting. That is an instance where a machine learning model fits its training data too well and fails to reliably fit additional data. In AI, information parting is commonly finished to keep away from overfitting. That is an example where an AI model accommodates its preparation information excessively well and neglects to fit extra information dependably. The first information in an AI model is regularly taken and parted into two sets. The two sets utilized are the training set, the testing set. The train set would contain the data which will be fed into the model. In simple terms, our model would learn from this data. For instance, a Regression model would use the examples in this data to find gradients to reduce the cost function. Then these gradients will be used to reduce the cost and predict data effectively. The training set is the piece of information used to prepare the model. The model ought to notice and gain from the training set, streamlining any of its boundaries. The test set contains the data on which we test the trained and validated model. It tells us how efficient our overall model is and how likely is it going to predict something which does not make sense. There is a plethora of evaluation metrics (like precision, recall, accuracy, etc.) which can be used to measure the performance of our model. The testing set is the part of information that is tried in the last model and is looked at against the past arrangements of information. The testing set goes about as an assessment of the last mode and calculation. The dataset is divided into 70:30 ratio.

**5.3. Vectorization**

In our model we are utilizing two vectorization strategies like TFID and Word2vec.

**5.3.1.TF-IDF**

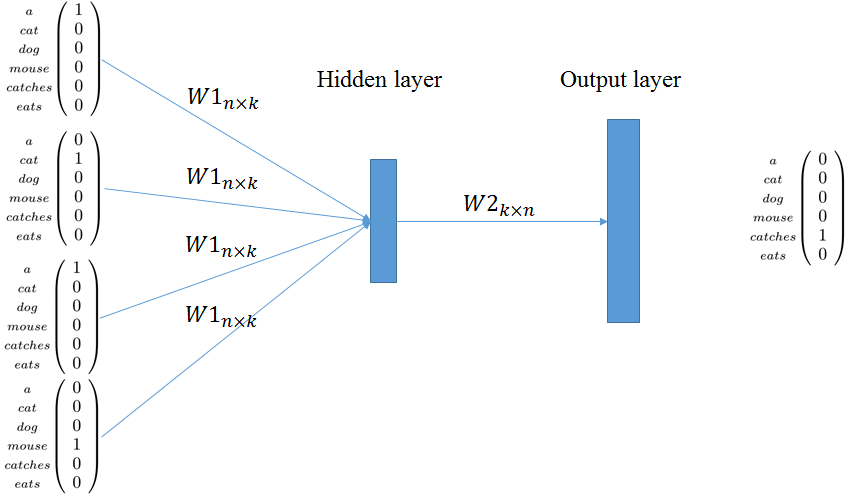
This method is called the TF-IDF stands for “Term Frequency – Inverse Document Frequency”. TF-IDF is a numerical statistic which measures the importance of the word in a document. It is a measure, used in the fields of information retrieval (IR) and machine learning, that can quantify the importance or relevance of string representations (words, phrases, lemmas, etc.) in a document amongst a collection of documents. TF-IDF is a truncation for Term Recurrence Reverse Record Recurrence. This is extremely normal calculation to change message into a significant portrayal of numbers which is utilized to fit machine calculation for prediction. TF-IDF consider overall documents of weight of words. TF-IDF vectorization involves calculating the TF-IDF score for every word in your corpus relative to that document and then putting that information into a vector (see image below using example documents “A” and “B”). Thus, each document in your corpus would have its own vector, and the vector would have a TF-IDF score for every single word in the whole collection of documents. Once you have these vectors you can apply them to various use cases such as seeing if two documents are similar by comparing their TF-IDF vector using cosine similarity. The biggest advantages of TF-IDF come from how simple and easy to use it is. It is simple to calculate, it is computationally cheap, and it is a simple starting point for similarity calculations. TF-IDF will transform the text into meaningful representation of integers or numbers which is used to fit machine learning algorithm for predictions. TF-IDF Vectorizer is a measure of originality of a word by comparing the number of times a word appears in document with the number of documents the word appears in. TF-IDF is used by search engines to better understand the content that is undervalued. For example, when you search for “Coke” on Google, Google may use TF-IDF to figure out if a page titled “COKE” is about: a) Coca-Cola.



*Figure 7. TF-IDF*

**5.3.2. WORD2VEC**

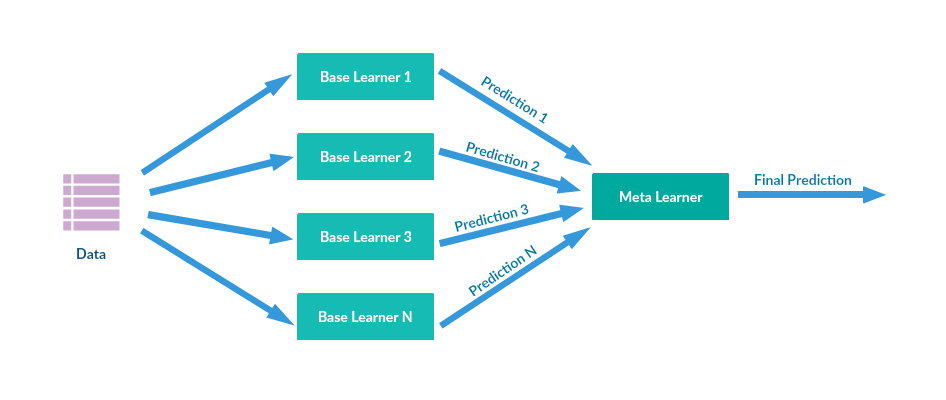
Word2vec is a technique for the conversion of tokens into vectors for each token word2vec generates vectors based on the model in which you trained the Word2vec model. The output of the embedding techniques will be three dimensional vectors. Word2vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large datasets. Embeddings learned through word2vec have proven to be successful on a variety of downstream natural language processing tasks. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. Word2Vec (W2V) is an algorithm that accepts text corpus as an input and outputs a vector representation for each word. There are two flavors of word2vec, such as CBOW and Skip-Gram. The set of sentences (also called corpus), the model loops on the words of each sentence and either try to use the current word w to predict its neighbors (i.e., its context), this approach is called “Skip-Gram”, or it employs each of these contexts to predict the current word w, in that case, the approach is called “Continuous Bag of Words” (CBOW). To limit the number of words in each context at tune the performance of the model, a parameter called “window size” is used.



*Figure 8: Word2Vec*

**5.3.4. STACKING**

Stacking is one of the popular ensemble modelling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future. we implemented six models to check their performance to use the best four algorithms for stacking. The six algorithms that are implemented to check their performance are SVM, TWSVM, decision tree, naive Bayes, logistic regression, KNN. based on the accuracy generated in word2vec the algorithms that are best to predict the output are KNN, SVM, logistic regression, decision tree. In the TF-IDF the best algorithms that has given more accuracy are naive bayes, svm,logistic regression, decision tree. In feature extraction the best algorithms that has predicted the output with best accuracy are naive bayes, logistic regression, KNN, SVM. an ensemble method that enables the model to learn how to use combine predictions given by learner models with meta-models and prepare a final model with accurate prediction.



*Figure 9: Ensemble Stacking*

**6. IMPLEMENTATION**

**6.1. Importing libraries**

**Packages used for Word2ve c**

**Pandas:**

The main scope of the pandas library is the manipulation of data sets, i.e., to edit, change, and replace elements of a Data Frame class object. However, pandas provide a broad range of functions and can also be used for other tasks such as the calculation of descriptive statistics and the visualization of the columns and rows in a data set. Like other Python libraries, packages, and modules, pandas are open source, i.e., freely available for usage, modification, and redistribution.

**Genism:**

Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is the natural language processing (NLP) and information retrieval (IR) community.

**Packages used for TF-IDF:**

**Tfidvectorizer method from sklearn:**

It’s an effective way of distilling and manageable abstraction. The term tf is called as the term frequency and the document will count how many times it shows the document. The df word is called as the document frequency and many times it will show in all the documents. Dividing the tf by modified df means the inverse of DF.

**Packages used for Feature engineering:**

**Urllib;**

Urllib package is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the urlopen function and is able to fetch URLs using a variety of different protocols. Urllib is a package that collects several modules for working with URLs, such as urllib.request for opening and reading, urllib.parse for parsing URLs, urllib.error for the exceptions raised, urllib.robotparser for parsing robot.txt files.

**Tld:**

Extract the top-level domain (TLD) from the URL given. List of TLD names is taken from Public Suffix. Optionally raises exceptions on non-existing TLDs or silently fails (if fail silently argument is set to True).

**Re:**

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

**6.2. Code for the model**

**Word2Vec:**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import gensim

import pandas as pd

df=pd.read\_csv('mainproject.csv')

df.columns=['url','label']

df=df.dropna(axis=0, how='all')

df['url'] = df['url']. map (lambda x:x.replace('//', ' '))

df['url'] = df['url']. map (lambda x:x.replace('/', ' '))

df['url'] = df['url']. map (lambda x:x.replace('-', ' '))

df['url'] = df['url']. map (lambda x:x.replace('.', ' '))

df['url'] = df['url']. map (lambda x:x.replace(',',' '))

x=df['url']

y=df['label']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3, stratify=y,random\_state=100)

from gensim.models import Word2Vec

model = Word2Vec(x\_train)

def get\_w2v\_vector(doc):

tmp = []

for w in doc:

tmp.append(model.wv[w])

return np.mean(tmp, axis=0)

train\_vectors\_w2v = []

for doc in x\_train:

train\_vectors\_w2v.append(get\_w2v\_vector(doc))

test\_vectors\_w2v= []

for doc in x\_test:

test\_vectors\_w2v.append(get\_w2v\_vector(doc))

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score, precision\_score

dt\_model = DecisionTreeClassifier()

dt\_model.fit(train\_vectors\_w2v, y\_train)

dt\_predictions = dt\_model.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test,dt\_predictions))

print (“Precision:”,precision\_score(y\_test,dt\_predictions))

print (“F1-score:”, f1\_score(y\_test,dt\_predictions))

print (“Recall:”,recall\_score(y\_test,dt\_predictions))

*Accuracy:0.82331348334545*

*Precision:0.73220345804583*

*F1-score:0.74000000000111*

*Recall:0.77563482344323*

from sklearn.linear\_model import LogisticRegression

dt\_model1 = LogisticRegression()

dt\_model1.fit(train\_vectors\_w2v,y\_train)

dt\_predictions1 = dt\_model1.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions1))

print (“Precision:”,precision\_score(y\_test, dt\_predictions1))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions1))

print (“Recall:”,recall\_score(y\_test, dt\_predictions1))

*Accuracy:0.8425*

*Precision:0.8420*

*F1-score:0.7234*

*Recall:0.6345*

from sklearn.naive\_bayes import GaussianNB

dt\_model2 = GaussianNB()

dt\_model2.fit(train\_vectors\_w2v, y\_train)

dt\_predictions2 = dt\_model2.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions2))

print (“Precision:”,precision\_score(y\_test, dt\_predictions2))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions2))

print (“Recall:”,recall\_score(y\_test, dt\_predictions2))

*Accuracy:0.6622*

*Precsion:0.4746*

*F1-score:0.3921*

*Recall:0.3412*

from sklearn.neighbors import KNeighborsClassifier

clfl=KNeighborsClassifier(n\_neighbors=1)

clfl.fit(train\_vectors\_w2v ,y\_train)

dt\_predictions3 = clfl.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions3))

print (“Precision:”,precision\_score(y\_test, dt\_predictions3))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions3))

print (“Recall:”,recall\_score(y\_test, dt\_predictions3))

*Accuracy:0.8133*

*Precision:0.7050*

*F1-score0.7213*

*Recall:0.7422*

from sklearn.svm import SVC

svm = SVC (kernel= 'linear', random\_state=1, C=0.1)

svm.fit(train\_vectors\_w2v, y\_train)

y\_pred = svm.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred))

print (“Precision:”,precision\_score(y\_test, y\_pred))

print (“F1-score:”, f1\_score(y\_test, y\_pred))

print (“Recall:”,recall\_score(y\_test, y\_pred))

*Accuracy:0.7892*

*Precision:0.6856*

*F1-score:0.6755*

*Recall:0.6521*

poly\_svc = SVC (kernel='poly', degree=3, C=100)

poly\_svc.fit(train\_vectors\_w2v, y\_train)

y\_pred1 = poly\_svc.predict(test\_vectors\_w2v)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred1))

print (“Precision:”,precision\_score(y\_test, y\_pred1))

print (“F1-score:”, f1\_score(y\_test, y\_pred1))

print (“Recall:”,recall\_score(y\_test, y\_pred1))

*Accuracy:0.8167*

*Precision:0.6798*

*F1-score:0.7234*

*Recall:0.7745*

from sklearn.ensemble import RandomForestClassifier

clf2=RandomForestClassifier(random\_state=1)

clf2.fit (train\_vectors\_w2v, y\_train)

from mlxtend.classifier import StackingClassifier

sclf = StackingClassifier( classifiers = [poly\_svc , svm ,dt\_model1,dt\_model ],use\_probas = True ,meta\_classifier = clf2)

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , test\_vectors\_w2v ,y\_test ,cv = 10, scoring = 'accuracy')

print (" Accuracy: % 0.2f [ % s] "% ( scores.mean () \*100, label))

*Accuracy: 84.50 [StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , test\_vectors\_w2v, y\_test, cv = 10, scoring = 'precision')

print (" Precision: % 0.2f [ % s] "% (scores.mean () \*100, label))

*Precision: 79.23 [StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , test\_vectors\_w2v ,y\_test,cv = 10, scoring = 'f1')

print (" F1-score: % 0.2f [ % s] "% (scores.mean () \*100, label))

*F1-score: 83.98 [StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_train ,y\_test ,cv = 10, scoring = 'recall')

print (" Recall: % 0.2f [ % s] "% (scores.mean () \*100, label))

*Recall: 60.23 [StackingClassifier(RF)]*

**Code for TF-IDF:**

import pandas as pd

df=pd.read\_csv('mainproject.csv')

df.columns=['url','label']

df=df.dropna(axis=0, how='all')

df['url'] = df['url']. map (lambda x:x.replace('//', ' '))

df['url'] = df['url']. map (lambda x:x.replace('/', ' '))

df['url'] = df['url']. map (lambda x:x.replace('-', ' '))

df['url'] = df['url']. map (lambda x:x.replace('.', ' '))

df['url'] = df['url']. map (lambda x:x.replace(',',' '))

x=df['url']

y=df['label']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3, stratify=y,random\_state=100)

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer(ngram\_range=(1,3), lowercase=True)

x\_train=tfidf.fit\_transform(x\_train)

x\_test=tfidf.transform(x\_test)

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score, precision\_score

dt\_model = DecisionTreeClassifier()

dt\_model.fit(x\_train,y\_train)

dt\_predictions = dt\_model.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions))

print (“Precision:”,precision\_score(y\_test, dt\_predictions))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions))

print (“Recall:”,recall\_score(y\_test, dt\_predictions))

*Accuracy:0.8989*

*Precision:0.8002*

*F1-score:0.8102*

*Recall:0.3300*

from sklearn.linear\_model import LogisticRegression

dt\_model1 = LogisticRegression()

dt\_model1.fit(x\_train,y\_train)

dt\_predictions1 = dt\_model1.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions1))

print (“Precision:”,precision\_score(y\_test, dt\_predictions1))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions1))

print (“Recall:”,recall\_score(y\_test, dt\_predictions1))

*Accuracy:0.9156*

*Precision:0.7019*

*F1-score:0.7233*

*Recall:0.7409*

from sklearn.naive\_bayes import GaussianNB

dt\_model2 = GaussianNB()

dt\_model2.fit(x\_train,y\_train)

dt\_predictions2 = dt\_model2.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions2))

print (“Precision:”,precision\_score(y\_test, dt\_predictions2))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions2))

print (“Recall:”,recall\_score(y\_test, dt\_predictions2))

*Accuracy:0.9067*

*Precision:0.8409*

*F1-score:0.7234*

*Recall:0.6390*

from sklearn.neighbors import KNeighborsClassifier

clfl=KNeighborsClassifier(n\_neighbors=1)

clfl.fit(x\_train,y\_train)

dt\_predictions3 = clfl.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions3))

print (“Precision:”,precision\_score(y\_test, dt\_predictions3))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions3))

print (“Recall:”,recall\_score(y\_test, dt\_predictions3))

*Accuracy:0.8506*

*Precision:0.4762*

*F1-score:0.3975*

*Recall:0.3403*

from sklearn.svm import SVC

svm = SVC (kernel= 'linear', random\_state=1, C=0.1)

svm.fit(x\_train, y\_train)

y\_pred = svm.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred))

print (“Precision:”,precision\_score(y\_test, y\_pred))

print (“F1-score:”, f1\_score(y\_test, y\_pred))

print (“Recall:”,recall\_score(y\_test, y\_pred))

*Accuracy:0.8705*

*Precision:0.7834*

*F1-score:0.8655*

*Recall:0.7890*

poly\_svc = SVC (kernel='poly', degree=3, C=100)

poly\_svc.fit(x\_train, y\_train)

y\_pred1 = poly\_svc.predict(x\_test)

accuracy\_score(y\_test, y\_pred1)

precision\_score(y\_test, y\_pred1)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred1))

print (“Precision:”,precision\_score(y\_test, y\_pred1))

print (“F1-score:”, f1\_score(y\_test, y\_pred1))

print (“Recall:”,recall\_score(y\_test, y\_pred1))

*Accuracy:0.8707*

*Precision:0.7239*

*F1-score:0.8567*

*Recall:0.4599*

from sklearn.ensemble import RandomForestClassifier

clf2=RandomForestClassifier(random\_state=1)

clf2.fit(x\_train,y\_train)

from mlxtend.classifier import StackingClassifier

sclf = StackingClassifier( classifiers = [ poly\_svc , dt\_model2 ,dt\_model1, dt\_model ],use\_probas = True ,meta\_classifier = clf2)

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10, scoring = 'accuracy')

print (" Accuracy: % 0.2f [ % s] "% (scores.mean () \*100, label))

*Accuracy:87.08 [StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test,y\_test ,cv = 10, scoring = 'precision')

print (" Precision: % 0.2f [ % s] "% (scores.mean () \*100, label))

*Precision:86.56[StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10, scoring = 'f1')

print (" F1-score: % 0.2f [ % s] "% (scores.mean () \*100, label))

*F1-score:82.34[StackingClassifier(RF)]*

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10, scoring = 'recall')

print (" Recall: % 0.2f [ % s] "% (scores.mean () \*100, label))

*Recall:45.93[StackingClassifier(RF)]*

**Code for feature engineering:**

import pandas as pd

df = pd.read\_csv('mainproject.csv')

df = df.drop('Unnamed: 0', axis=1)

df = df.drop('label',axis=1)

df.columns=['url','label']

df=df.dropna(axis=0, how='all')

from urllib.parse import urlparse

from tld import get\_tld

import os.path

df['url\_length'] = df['url']. apply (lambda i: len(str(i)))

df['hostname\_length'] = df['url']. apply (lambda i: len(urlparse(i).netloc))

df['path\_length'] = df['url']. apply (lambda i: len(urlparse(i).path))

def fd\_length(url):

df= urlparse(url).path

try:

return len(df.split('/') [1])

except:

return 0

df['fd\_length'] = df['url']. apply (lambda i: fd\_length(i))

df['tld'] = df['url']. apply (lambda i: get\_tld(i,fail\_silently=True))

def tld\_length(tld):

try:

return len(tld)

except:

return -1

df['tld\_length'] = df['tld'].apply(lambda i: tld\_length(i))

df['-']=df['url']. apply(lambda i: i.count('-'))

df['@']=df['url'].apply(lambda i: i.count('@'))

df['?']=df['url'].apply(lambda i: i.count('?'))

df['%']=df['url'].apply(lambda i: i.count('%'))

df['.']=df['url'].apply(lambda i: i.count('.'))

df['=']=df['url'].apply(lambda i: i.count('='))

df['http'] = df['url'].apply(lambda i : i.count('http'))

df['https'] = df['url'].apply(lambda i : i.count('https'))

df['www'] = df['url'].apply(lambda i: i.count('www'))

def digit\_count(url):

digits = 0

for i in url:

if i.isnumeric():

digits = digits + 1

return digits

df['digits'] = df['url'].apply(lambda i: digit\_count(i))

def letter\_count(url):

letters = 0

for i in url:

if i.isalpha():

letters = letters + 1

return letters

df['letters']=df['url'].apply(lambda i: letter\_count(i))

def no\_of\_dir(url):

urldir = urlparse(url).path

return urldir.count('/')

df['dir'] = df['url'].apply(lambda i: no\_of\_dir(i))

import re

def having\_ip\_address(url):

match = re.search(

'(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\. ([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\. ([01]? \\d\\d?|2[0-4]\\d|25[0-5])\\.'

'([01]? \\d\\d?|2[0-4]\\d|25[0-5])\\/)|'

'((0x[0-9a-fA-F] {1,2}) \\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'

'(?[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)

if match:

return -1

else:

return 1

df['ip'] = df['url']. apply(lambda i: having\_ip\_address(i))

def shortening\_service(url):

match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'

'tr\.im|link\.zip\.net',

url)

if match:

return -1

else:

return 1

df['short\_url']=df['url']. apply(lambda i: shortening\_service(i))

x=df[['hostname\_length','path\_length', 'fd\_length','tld\_length','-','@', '?','%', '.', '=', 'http','https','www','digits','letters','dir','ip']]

y=df['label']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,stratify=y,random\_state=100)

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score, precision\_score

dt\_model = DecisionTreeClassifier()

dt\_model.fit(x\_train,y\_train)

dt\_predictions = dt\_model.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions))

print (“Precision:”,precision\_score(y\_test, dt\_predictions))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions))

print (“Recall:”,recall\_score(y\_test, dt\_predictions))

*Accuracy:0.9560*

*Precision:0.9511*

*F1-score:0.9503*

*Recall:0.9599*

from sklearn.linear\_model import LogisticRegression

dt\_model1 = LogisticRegression()

dt\_model1.fit(x\_train,y\_train)

dt\_predictions1 = dt\_model1.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions1))

print (“Precision:”,precision\_score(y\_test, dt\_predictions1))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions1))

print (“Recall:”,recall\_score(y\_test, dt\_predictions1))

*Accuracy:0.9656*

*Precision:0.9650*

*F1-score:0.9625*

*Recall:0.9650*

from sklearn.naive\_bayes import GaussianNB

dt\_model2 = GaussianNB()

dt\_model2.fit(x\_train,y\_train)

dt\_predictions2 = dt\_model2.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions2))

print (“Precision:”,precision\_score(y\_test, dt\_predictions2))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions2))

print (“Recall:”,recall\_score(y\_test, dt\_predictions2))

*Accuracy:0.9721*

*Precision:0.9728*

*F1-score:0.9722*

*Recall:0.9717*

from sklearn.neighbors import KNeighborsClassifier

clfl=KNeighborsClassifier(n\_neighbors=1)

clfl.fit(x\_train,y\_train)

dt\_predictions3 = clfl.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, dt\_predictions3))

print (“Precision:”,precision\_score(y\_test, dt\_predictions3))

print (“F1-score:”, f1\_score(y\_test, dt\_predictions3))

print (“Recall:”,recall\_score(y\_test, dt\_predictions3))

*Accuracy:0.9649*

*Precision:0.9540*

*F1-score:0.9600*

*Recall:0.9500*

from sklearn.svm import SVC

svm = SVC(kernel= 'linear', random\_state=1, C=0.1)

svm.fit(x\_train, y\_train)

y\_pred = svm.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred))

print (“Precision:”,precision\_score(y\_test, y\_pred))

print (“F1-score:”, f1\_score(y\_test, y\_pred))

print (“Recall:”,recall\_score(y\_test, y\_pred))

*Accuracy:0.9564*

*Precision:0.9500*

*F1-score:0.9597*

*Recall:0.9500*

poly\_svc = SVC(kernel='poly', degree=3, C=100)

poly\_svc.fit(x\_train, y\_train)

y\_pred1 = poly\_svc.predict(x\_test)

print (“Accuracy: ”,accuracy\_score(y\_test, y\_pred))

print (“Precision:”,precision\_score(y\_test, y\_pred))

print (“F1-score:”, f1\_score(y\_test, y\_pred))

print (“Recall:”,recall\_score(y\_test, y\_pred))

*Accuracy:0.9865*

*Precision:0.9865*

*F1-score:0.9839*

*Recall:0.9814*

from sklearn.ensemble import RandomForestClassifier

clf2=RandomForestClassifier(random\_state=1)

clf2.fit(x\_train,y\_train)

from mlxtend.classifier import StackingClassifier

sclf = StackingClassifier( classifiers = [ poly\_svc , clf1, dt\_model1,dt\_model2 ],use\_probas = True ,meta\_classifier = clf2)

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10 , scoring = 'accuracy' )

print ( " Accuracy : % 0.2f [ % s ] "% ( scores.mean ( )\*100 , label ) )

Accuracy: 99.74

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test,y\_test ,cv = 10 , scoring = 'precision' )

print ( " Precision : % 0.2f [ % s ] "% ( scores.mean ( )\*100 , label ) )

Precision:99.69

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10 , scoring = 'f1' )

print ( " F1-score : % 0.2f [ % s ] "% ( scores.mean ( )\*100 , label ) )

F1-score:99.60

from sklearn import model\_selection

for clf , label in zip ( [sclf ] , ['StackingClassifier(RF)']):

scores = model\_selection.cross\_val\_score ( clf , x\_test ,y\_test ,cv = 10 , scoring = 'recall' )

print ( " Recall : % 0.2f [ % s ] "% ( scores.mean ( )\*100 , label ) )

Recall:99.55

**7. EXPERIMENTAL SETUP**

In evaluation of the proposed work, we first considered the vectorization techniques such as TF-IDF and word2vec. We tested the performance measures with baseline machine learning model in which ML models are under performed. We also tested the baseline model with ensemble stacking approaches this improves the accuracy but precision, recall and f1-score having some deviation in the performance which shown in table 1,2. In the proposed model, we extracted 17 features using feature extraction techniques. We tested the performance with the baseline machine learning in which we got the better performance and stable performance other matrices which shown inn table 3. To evaluate the performance of the proposed models, standard metrics are used for classification tasks, such as Accuracy, Precision, Recall and F1-Score. Accuracy: It is the number of correctly predicted divided by the total number samples.

**7.1 Evaluation measures:**

To evaluate the performance of the proposed models, standard metrics are used for classification tasks, such as Accuracy, Precision, Recall and F1-Score. Accuracy: It is the number of correctly predicted divided by the total number samples.

𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦 =( 𝑇𝑃 + 𝑇𝑁 )/( 𝑇𝑃 + 𝐹𝑃 + 𝐹𝑁 + 𝑇𝑁 )

Precision: It is the proportion of positive predictions that are truly positives.

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 = 𝑇𝑃 /( 𝑇𝑃 + 𝐹𝑃 )

Recall: It is the proportion of actual Positives that are correctly classified.

𝑅𝑒𝑐𝑎𝑙𝑙 = 𝑇𝑃 /( 𝑇𝑃 + 𝐹𝑁 )

F1-Score: It is the harmonic mean of precision and recall.

𝐹1 − 𝑆𝑐𝑜𝑟𝑒 = (2 × 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 × 𝑅𝑒𝑐𝑎𝑙𝑙)/ (𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + 𝑅𝑒𝑐𝑎𝑙𝑙)

**7.2 Performance of ML models with word2vec:**

*Table 2: Performance of different machine algorithms with Word2vec*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Recall** | **Precision** | **F1 score** |
| Naïve Bayes | 66.22 | 34.12 | 47.46 | 39.21 |
| TWSVM | **81.67** | **77.54** | **67.98** | **72.34** |
| KNN | **81.33** | **74.22** | **70.50** | **72.13** |
| SVM | 78.92 | 65.21 | 68.56 | 67.55 |
| Logisitic Regression | **84.25** | **63.45** | **84.20** | **72.34** |
| Decision Tree | **82.33** | **77.56** | **73.22** | **74.00** |
| Stacking (RF) | **84.50** | **60.23** | **79.23** | **83.98** |

**Evaluation -1:** The model is processed into training and testing and the vectors are trained using above algorithms such as Naïve Bayes, TWSVM, KNN, SVM, LR, RF. We tested the word2vec training set with ML models in which accuracy was improved for ensemble stacking but when consider with other parameters such as recall, precision and f1-score the performance was improved in decision tree, logistic regression, and decision tree, respectively. In the performance of word2vec with baseline ML models are underperformed with proposed models.

**7.3 Performance of ML models with TF-IDF:**

*Table 3: Performance of different machine algorithms with TF-IDF*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Recall** | **Precision** | **F1 score** |
| Naïve Bayes | **90.67** | **63.90** | **84.09** | **72.34** |
| TWSVM | **87.07** | **45.99** | **72.39** | **85.67** |
| KNN | 85.06 | 34.03 | 47.62 | 39.75 |
| SVM | 87.05 | 78.90 | 78.34 | 86.55 |
| Logisitic Regression | **91.56** | **74.09** | **70.19** | **72.33** |
| Decision Tree | **89.89** | **33.00** | **80.02** | **81.02** |
| Stacking (RF) | **87.08** | **45.93** | **86.56** | **82.34** |

**Evaluation -2:** We tested the dataset with different ML models using TD-IDF vectorization. The model logistic regression given 91.56% accuracy which was better than the word2vec model. The other matrices such as precision, recall and F1-score also improved compared with word2vec model. We also tested the performance with ensemble stacking approaches which is not stable with other ML models.

**7.4 Performance of ensemble stacking with ML models:**

*Table 4: Performance of different machine algorithms with proposed method*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Recall** | **Precision** | **F1 score** |
| Naive Bayes | **97.21** | **97.17** | **97.28** | **97.22** |
| Twin SVM | **98.65** | **98.14** | **98.65** | **98.39** |
| KNN | **96.49** | **95.00** | **95.40** | **96.00** |
| SVM | 95.64 | 95.00 | 95.00 | 95.97 |
| Logistic Regression | **96.56** | **96.50** | **96.50** | **96.25** |
| Decision Tree | 95.60 | 95.99 | 95.11 | 95.03 |
| **Proposed work Stacking (RF)** | **99.74** | **99.55** | **99.69** | **99.60** |

**Evaluation -3**: In this evaluation, we tested the model performance with our proposed feature-based stacking ensemble. We chose the four best performed stacking models for level-0. In meta layer we chosen the random forest at level-1 for better prediction of the model. The performance matrices also stable for each machine learning where Twin SVM performed better than the other ML models. The proposed ensemble stacking with feature-based approach performed better than the other ML models shown in table [4].

**8. RESULTS**

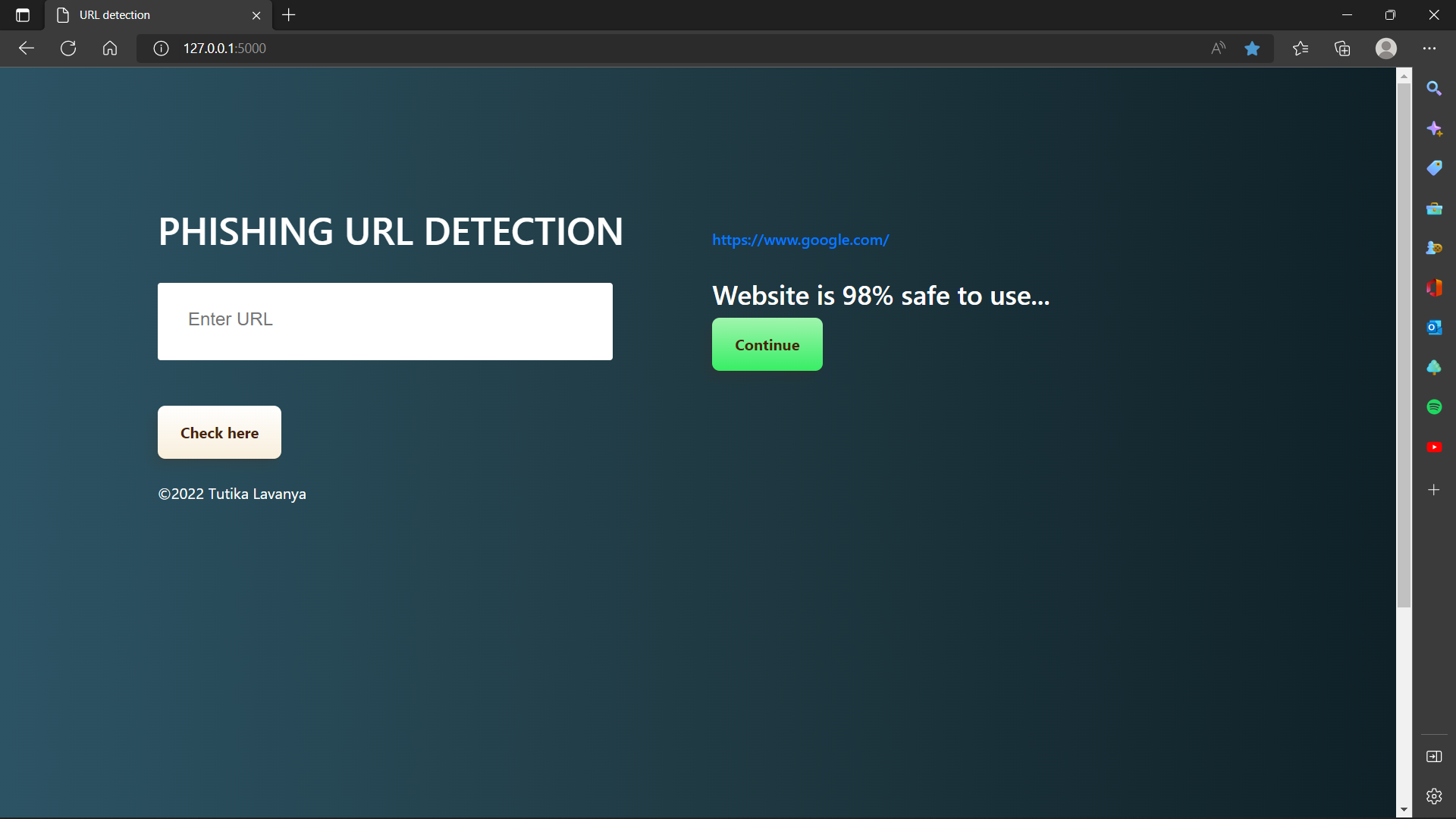
**8.1. Comparative analysis**

The proposed model was compared with existing literature in which Rao, R.S., Pais authors proposed Twin SVM which performed 98.05% accuracy. M. H. Alkawaz authors proposed ML models which performed 96% accuracy and A. El Aassal authors proposed ML based techniques which performed 99%. The proposed work significantly improves with existing works. This ensemble stacking work results in high accuracy with other state of art models.

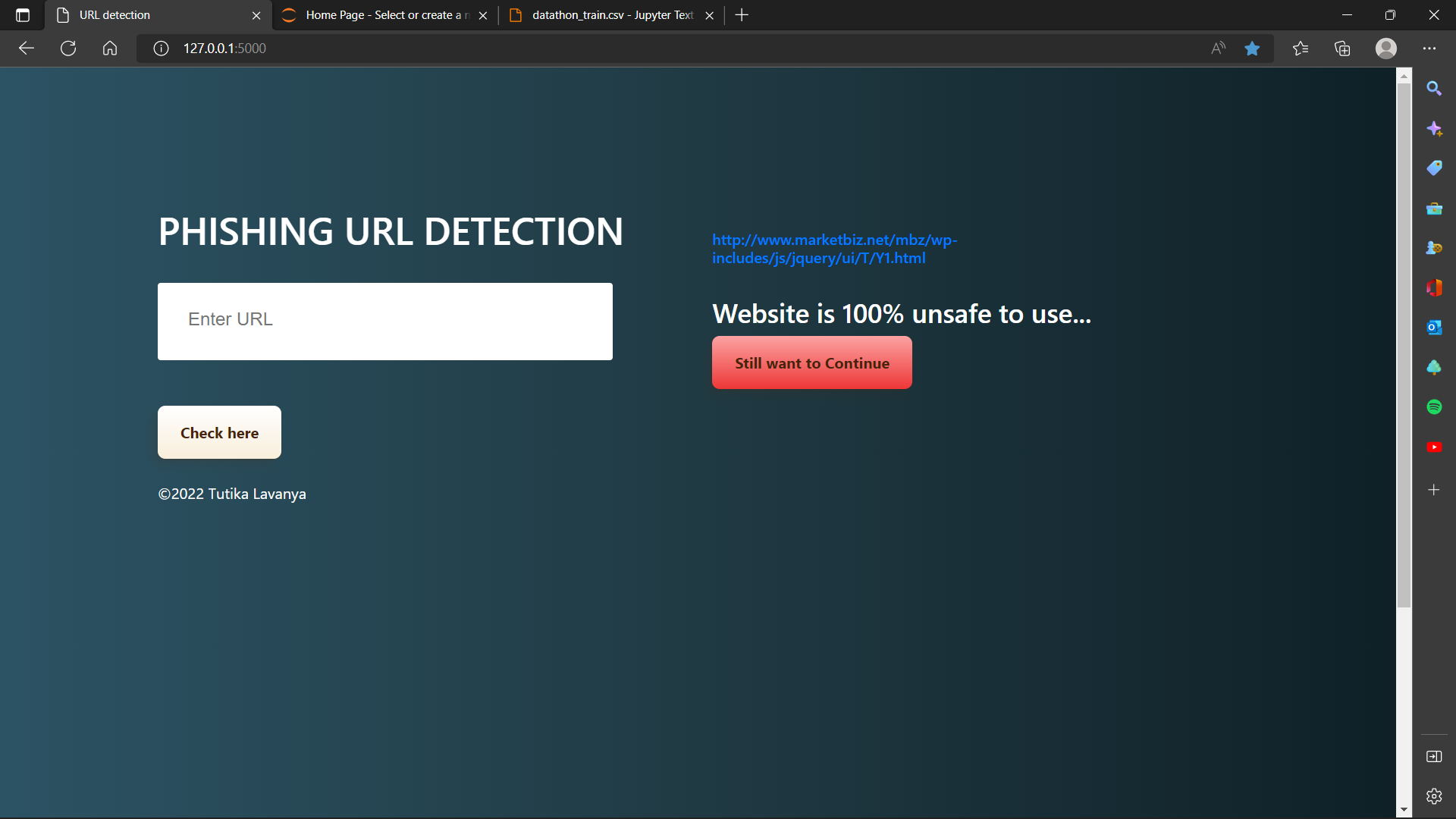
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Authors** | **Algorithm** | **Accuracy** | **Recall** | **Precision** | **F1 score** |
| M. H. Alkawaz etc..[5] | Decision Tree | 96.80 | - | - | - |
| Random Forest | 96.84 | - | - | - |
| SVM | 96.40 | - | - | - |
| A. El Aassal etc..[8] | AutoSk | 97.00 | 90.02 | 92.46 | 96.53 |
| Boosting | 97.70 | 99.00 | 98.01 | 98.00 |
| SVM | **99.01** | **99.00** | **98.02** | **89.01** |
| Rao, R.S., Pais etc..[6] | TWSVM | 98.05 | 98.33 | 97.74 | 98.03 |
| SVM | 97.73 | 97.86 | 97.60 | 97.73 |
| PSVM | 95.03 | 95.49 | 94.67 | 95.08 |
| **Proposed work Stacking (RF)** | | **99.74** | **99.55** | **99.69** | **99.60** |

*Table 5: Comparison between proposed and existing works*

**8.2. Graphical User Interface (GUI):**



*Figure 10: Legitimate URL detection*

 *Figure 10: Phishing URL detection*

**9. CONCLUSION**

Phishing detection technologies are essential for ensuring users have a safe online experience, preventing online users from being targets of online fraud, preventing personal information from being given to an attacker. The purpose of this work is to research and evaluate earlier efforts to determine the best classification scheme for detection of phishing URLs. by using extracted features of a URL in an ensemble stacking model. By detecting phishing URLs accurately, the damage caused by attackers can be decreased.

**10. LIMITATIONS AND FUTURE SCOPE**

The suggested model is only applicable to current URLs. Future changes to the structure properties used to train the model, such as the emergence of new IP addresses and top-level domains (tlds), may occur. The URL patterns and extensions, such as shortening services, may change in the future, and these issues will need to be addressed for this model.

**BIBLIOGRAPHY**

**1.** <https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>

2. <https://www.kaggle.com/code/poigal/cnn-on-word2vec-word-embedding/notebook>

3.<https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794>

4. [TF IDF | TFIDF Python Example](https://youtu.be/UvsQPsrZTK4)

**REFERENCES**

[1]. S. Ariyadasa, S. Fernando and S. Fernando, "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML," in IEEE Access, vol. 10, pp. 82355-82375, 2022, Doi: 10.1109/ACCESS.2022.3196018.

[2]. M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki and V. González-Castro, "Phishing URL Detection: A Real-Case Scenario Through Login URLs," in IEEE Access, vol. 10, pp. 42949-42960, 2022, Doi: 10.1109/ACCESS.2022.3168681.

[3]. Anurag Pandey, Jay Chadawar, 2022, Phishing URL Detection using Hybrid Ensemble Model, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 04 (April 2022).

[4]. Maheshwari, Shantanu & Janet, B & Kumar, R. (2021). Malicious URL Detection: A Comparative Study. 1147-1151. 10.1109/ICAIS50930.2021.9396014.

[5]. M. H. Alkawaz, S. J. Steven, A. I. Hajamydeen and R. Ramli, "A Comprehensive Survey on Identification and Analysis of Phishing Website based on Machine Learning Methods," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2021, pp. 82-87, Doi: 10.1109/ISCAIE51753.2021.9431794.

[6]. Rao, R.S., Pais, A.R. & Anand, P. A heuristic technique to detect phishing websites using TWSVM classifier. Neural Computer & Application 33, 5733–5752 (2021), https://doi.org/10.1007/s00521-020-05354-z.

[7]. P. L. Indrasiri, M. N. Halgamuge and A. Mohammad, "Robust Ensemble Machine Learning Model for Filtering Phishing URLs: Expandable Random Gradient Stacked Voting Classifier (ERG-SVC)," in IEEE Access, vol. 9, pp. 150142-150161, 2021, Doi: 10.1109/ACCESS.2021.3124628.

[8]. A. El Aassal, S. Baki, A. Das and R. M. Verma, "An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs," in IEEE Access, vol. 8, pp. 22170-22192, 2020, Doi: 10.1109/ACCESS.2020.2969780.

[9]. X. Liu and J. Fu, "SPWalk: Similar Property Oriented Feature Learning for Phishing Detection," in IEEE Access, vol. 8, pp. 87031-87045, 2020, Doi: 10.1109/ACCESS.2020.2992381.

[10]. Ramana, A. & Rao, K. & Rao, Routhu. (2021). Stop-Phish: an intelligent phishing detection method using feature selection ensemble. Social Network Analysis and Mining. 11. 10.1007/s13278-021-00829-w.

[11].Kalabarige, Lakshmana & Rao, Routhu & Abraham, Ajith & Gabralla, Lubna. (2022). MLSELM:Multi-layer Stacked Ensemble Learning Model to detect phishing websites. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3194672.

[12]. Aljofey, A.; Jiang, Q.; Qu, Q.; Huang, M.; Niyigena, J.-P. An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. Electronics 2020, 9, 1514. https://doi.org/10.3390/electronics9091514.

[13]. Areti Nagendra Soma Charan, Yu-Hung Chen, Jiann-Liang Chen, "Phishing Websites Detection using Machine Learning with URL Analysis", 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), pp.808-812, 2022.

[14]. Safa Alrefaai, Ghina Özdemir, Afnan Mohamed, "Detecting Phishing Websites Using Machine Learning", 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp.1-6, 2022.

[15]. Salloum, S., Gaber, T., Vadera, S., Shaalan, K. (2021). Phishing Website Detection from URLs Using Classical Machine Learning ANN (Artificial Neural Network) Model. In: Garcia-Alfaro, J., Li, S., Poovendran, R., Debar, H., Yung, M. (eds) Security and Privacy in Communication Networks. SecureComm 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 399. Springer, Cham. https://doi.org/10.1007/978-3-030-90022-9\_28.

[16]. Brij B. Gupta, Krishna Yadav, Imran Razzak, Konstantinos Psannis, Arcangelo Castiglione, Xiaojun Chang, A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment, Computer Communications, https://doi.org/10.1016/j.comcom.2021.04.023.

[17]. Youness Mourtaji, Mohammed Bouhorma, Daniyal Alghazzawi, Ghadah Aldabbagh, Abdullah Alghamdi, "Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network", Wireless Communications and Mobile Computing, vol. 2021, Article ID 8241104, 24 pages, 2021. https://doi.org/10.1155/2021/8241104.

[18]. P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar and A. Gururajan, "URLTran: Improving Phishing URL Detection Using Transformers," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), 2021, pp. 197-204, doi: 10.1109/MILCOM52596.2021.9653028.

[19]. Jalil, S., Usman, M. & Fong, A. Highly accurate phishing URL detection based on machine learning. J Ambient Intell Human Comput (2022). https://doi.org/10.1007/s12652-022-04426-3.

[20]. Murthy, K.S.R.C., Bhattacharya, T., Rajagopalan, N. (2022). Feature Extraction-Based Phishing URL Detection Using Machine Learning Techniques. In: Satyanarayana, C., Samanta, D., Gao, XZ., Kapoor, R.K. (eds) High Performance Computing and Networking. Lecture Notes in Electrical Engineering, vol 853. Springer, Singapore. <https://doi.org/10.1007/978-981-16-9885-9_14>.

[21] S. -C. Lin, P. -C. Wl, H. -Y. Chen, T. Morikawa, T. Takahashi and T. -N. Lin, "Sense Input: An Image-Based Sensitive Input Detection Scheme for Phishing Website Detection," ICC 2022 - IEEE International Conference on Communications, 2022, pp. 4180-4186, Doi: 10.1109/ICC45855.2022.98386.

[22]. C. Opara, B. Wei and Y. Chen, "HTML Phish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, Doi: 10.1109/IJCNN48605.2020.9207707.

[23]. S. Y. Yerima and M. K. Alzaylaee, "High Accuracy Phishing Detection Based on Convolutional Neural Networks," 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), 2020, pp. 1-6, Doi: 10.1109/ICCAIS48893.2020.9096869.

[24]. Y. Huang, J. Qin and W. Wen, "Phishing URL Detection Via Capsule-Based Neural Network," 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2019, pp. 22-26, Doi: 10.1109/ICASID.2019.8925000.

[25] L. Zhang and P. Zhang, "Phish Trim: Fast and adaptive phishing detection based on deep representation learning," 2020 IEEE International Conference on Web Services (ICWS), 2020, pp. 176-180, Doi: 10.1109/ICWS49710.2020.00030.

[26] Liang, Y., Wang, Q., Xiong, K., Zheng, X., Yu, Z., & Zeng, D. (2021). “Robust Detection of Malicious URLs With Self-Paced Wide & Deep Learning.” IEEE Transactions on Dependable and Secure Computing, 19(2), 717-730.

[27] S. -J. Bu and S. -B. Cho, "Integrating Deep Learning with First-Order Logic Programmed Constraints for Zero-Day Phishing Attack Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2021, pp. 2685-2689, Doi: 10.1109/ICASSP39728.2021.9414850.

[28] P. Yang, G. Zhao, and P. Zeng, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," in IEEE Access, vol. 7, pp. 15196-15209, 2019, Doi: 10.1109/ACCESS.2019.2892066.

[29]. Odeh, I. Keshta and E. Abdelfattah, "Efficient Detection of Phishing Websites Using Multilayer Perceptron", International Journal of Interactive Mobile Technologies (IJIM), vol. 14, no. 11, pp. 22, 2020.

[30] Anurag Pandey, Jay Chadawar, 2022, “Phishing URL Detection using Hybrid Ensemble Model,” INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 04 (April 2022).