Volume: 13 Issue: V Month of publication: May 2025
DOI: https://doi.org/10.22214/ijraset.2025.70277

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Phishing URL Detection Using XGBoost and Custom Feature Engineering

Prof. P. S. Prasad[1], Aishwarya Kalamkar[2], Manasi Nagpure[3], Neha Vaidya[4], Pranal Mohadikar[5], Bhagyashri Tembhurne[6]
[1]Associate Professor, [2, 3, 4, 5, 6]UGStudents, Department of Information Technology, Priyadarshini College of Engineering (Autonomous), Affiliated to RTMN University Nagpur, Maharashtra, India

Abstract: Phishing is a prevalent cyberattack technique that deceives users into revealing sensitive personal and financial information through fake websites. With the exponential growth of online services, phishing attacks have become more sophisticated, necessitating intelligent and automated detection mechanisms. This study introduces a smart phishing URL detection approach that utilizes carefully engineered features—such as lexical patterns, structural elements, and domain-related information—to differentiate between malicious and legitimate web addresses.

A custom feature extraction module was developed to parse URLs and retrieve 13+ critical features, including URL length, directory structure, file name characteristics, presence of IP addresses, SSL certificate availability, information about the Autonomous System Number (ASN) and domain registration details, including creation and expiration dates. The extracted features were used to train an Extreme Gradient Boosting (XGBoost) classifier, selected for its superior performance in imbalanced and noisy datasets. The model was developed and fine-tuned using PyCaret, an automated machine learning library that optimizes classification performance using cross-validation and hyperparameter tuning.

The trained model achieved strong performance across multiple evaluation metrics, highlighting its reliability and effectiveness in accurately identifying phishing URLs. To enhance usability, a web-based application was developed using FastAPI and HTML/CSS, allowing users to submit a URL and receive instant predictions regarding its legitimacy. The system provides an interpretable and scalable framework for real-time phishing detection, suitable for integration into email filters, browsers, and cybersecurity tools. The results affirm that combining feature engineering with a tuned XGBoost classifier offers an effective and deployable solution to mitigate phishing threats in real-world environments.

Keywords: Phishing, URL Detection, Machine Learning, Extreme Gradient Boosting(XGBoost) Classifier, FastAPI, PyCaret, Feature Extraction

I. INTRODUCTION

In today's technology-driven world, online platforms play a vital role in everyday activities, supporting functions such as financial transactions, online shopping, virtual communication, and digital learning. While this digital transformation has brought immense convenience, it has also exposed users to various cybersecurity threats—one of the most prevalent being phishing. Phishing refers to a deceptive practice in which cybercriminals impersonate trusted organizations or individuals to trick users into disclosing sensitive details like login credentials, banking information, or personal identification numbers. These attacks typically occur through deceptive emails, fake websites, or misleading links, often leading unsuspecting users to fraudulent web pages designed to steal sensitive data.

One of the major difficulties in combating phishing attacks is their constantly changing tactics and adaptability. Attackers continuously modify their strategies, making it difficult for conventional defence mechanisms like blacklist-based filters or static rule systems to keep up. Such traditional methods rely on known patterns or previously reported phishing URLs, which renders them ineffective against newly crafted, zero-day phishing links. This limitation emphasizes the necessity for smart and forward-looking detection systems that can recognize phishing threats by analysing the inherent characteristics of the URLs.

This research presents a machine learning-driven approach to phishing detection that focuses on analysing structural, lexical, and domain-specific features of URLs. The proposed system extracts key indicators such as domain length, use of special characters, presence of IP addresses, SSL certificate usage, domain registration details, and redirect behavior. These features are used to train a highly accurate and optimized XGBoost classifier, which learns to distinguish between phishing and legitimate URLs.

To enhance accessibility and usability, the trained model is integrated into a web application built using FastAPI. This user-facing platform allows real-time URL submissions and provides instant classification results, empowering users to verify link legitimacy before interaction. The objective of this study is to develop a reliable, scalable, and intelligent phishing detection system that not only improves upon existing methods but is also ready for real-world deployment across enterprise cybersecurity tools, email client software, and browser security features.

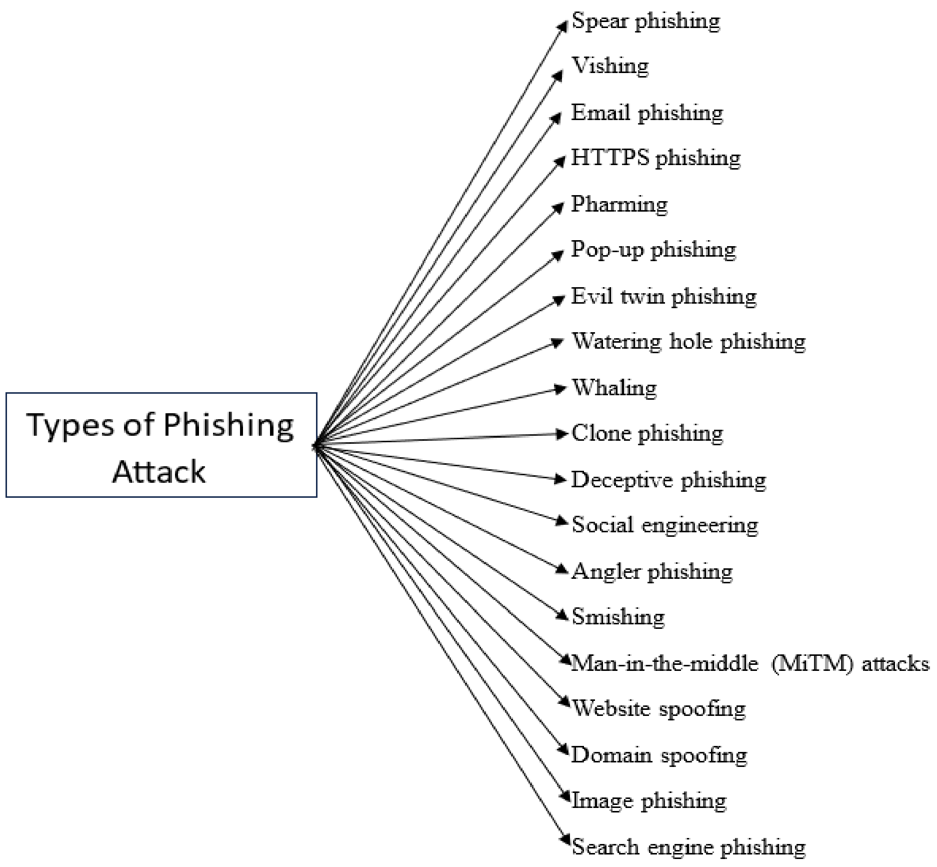


Fig.1: Types of Phishing Attacks

II. LITERATURE REVIEW

Phishing detection has gained significant attention in recent years due to its critical role in protecting users against cyber fraud and identity theft. Numerous researchers have proposed various methods to classify phishing websites using machine learning and feature engineering techniques.

Shraddha Parekh et al. [1] introduced a novel detection model based on URL analysis, emphasizing rule-based and pattern-matching techniques to distinguish malicious links. Similar studies, such as by Sanjukta Mohanty [4], have adopted filter-based univariate feature selection to enhance classification performance. Others, like V. S. Lakshmi and M. S. Vijaya [5], explored supervised learning algorithms, showing that decision trees and support vector machines can effectively handle phishing datasets.

IEEE studies such as those by Upendra Shetty et al. [8] and Rakesh Verma et al. [10] focused on URL-based lexical and statistical feature extraction, proving the efficiency of syntactic features like URL length, special characters, and redirection behavior. Garje et al. [9] and Sinha et al. [11] contributed practical evaluations on phishing detection systems using machine learning, while others like D. Sahoo [7] conducted comprehensive surveys highlighting challenges in detecting zero-day phishing links.

More advanced strategies include the use of reinforcement learning [36], deep learning [34][37], and hybrid feature engineering [39], which demonstrate improved accuracy but often require greater computational resources. The study conducted by Jeeva and Rajsingh [37] highlights the use of association rule mining to facilitate interpretable classification in phishing detection. Additionally, Sánchez-Paniagua et al. [38] examined the robustness of phishing detection models against evolving phishing strategies.

Recent works have also introduced domain-specific datasets and real-world testing environments, such as the Kaggle dataset [35], emphasizing the need for both scalability and real-time detection capability. The integration of AI models in dynamic environments, like browser-based filters or cloud-integrated APIs, has been explored in several studies [40][41][42].

Overall, the literature indicates a trend toward integrating lexical, domain, and behavioral features with advanced classification models such as XGBoost, Random Forest, and deep neural networks. This work extends earlier methodologies by introducing a real-time detection framework that combines comprehensive feature engineering with an optimized XGBoost classifier, deployed via FastAPI.

III. PROPOSED SYSTEM

The proposed system aims to detect phishing domains through a robust and automated machine learning pipeline. It integrates domain-specific, lexical, and network-based attributes to determine whether a given URL is legitimate. A key component of this system is the use of an Extreme Gradient Boosting (XGBoost) classifier trained using PyCaret, which enables efficient model selection, tuning, and deployment.

The entire flow begins with the collection and preprocessing of labeled URL data. Feature extraction is performed using a custom-built Extract Features module that analyses attributes such as domain age, URL length, presence of special characters, redirections, TLS/SSL status, and Whois information. The extracted features are structured and passed into the classifier for training and evaluation.

The backend is powered by FastAPI, which allows real-time classification of user-input URLs through a web interface. The trained model (xgb.pkl) is loaded and used to predict whether a given URL is phishing or legitimate. The frontend is designed using HTML and CSS, offering a minimal and responsive user interface.

This project utilizes libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, PyCaret, and Socket/Whois APIs for feature extraction and prediction. The modular architecture ensures ease of integration with email gateways, browsers, or messaging apps for real-time threat detection.

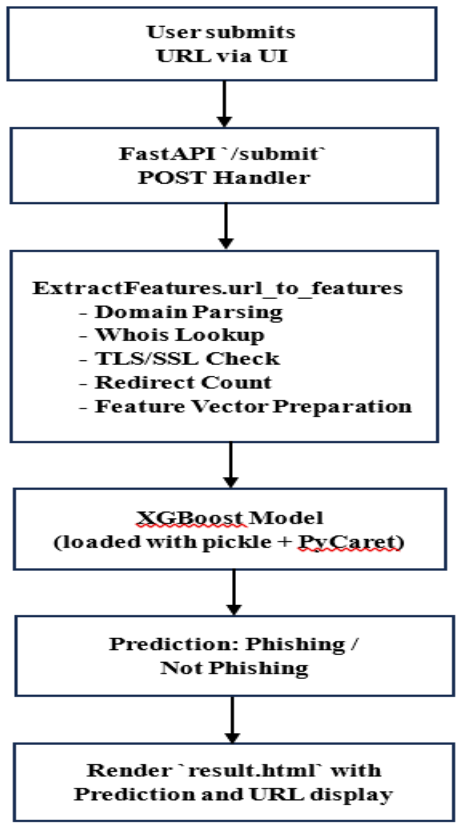


Fig.2:Flowof operations

IV. OBJECTIVES

The primary objective of this project is to design and implement an intelligent phishing URL detection system using Machine Learning (ML) techniques, with a focus on real-time prediction and deployment. The system utilizes the Extreme Gradient Boosting (XGBoost) algorithm trained via PyCaret to effectively classify URLs as either legitimate or phishing attempts. Feature extraction plays a critical role in identifying patterns and indicators that differentiate safe domains from malicious ones.

Key objectives include

1) Implementing the XGBoost algorithm for high-accuracy phishing URL classification.
2) Designing a custom feature extraction module to capture critical URL attributes such as domain age, redirection behavior, SSL usage, and lexical structure.
3) Comparing model performance using a selected set of 13 key features to optimize classification without sacrificing accuracy.
4) Deploying the trained model via a FastAPI backend for real-time phishing detection through a user-friendly web interface.
5) Making use of powerful Python libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, PyCaret, and FastAPI to improve the efficiency of model development, performance assessment, and real-time system implementation.
6) Constantly improving the feature set and model retraining to bolster cybersecurity protections and adjust to changing phishing methods.

V. METHODOLOGY

A. Data Collection and Exploration

A well-organized dataset consisting of 88,647 web URLs was employed in this study, with each record explicitly marked as either a phishing attempt (1) or a legitimate URL (0). The data was sourced from verified repositories and open-source contributions known for containing real-world phishing attempts as well as legitimate web addresses.

Each entry in the dataset represents a single URL and is accompanied by multiple syntactic and semantic features that capture the structure, behavior, and metadata associated with the web address. These features are essential for training machine learning models to distinguish between malicious and benign URLs based on learned patterns.

An initial exploratory analysis revealed a class imbalance: approximately 58,000 legitimate URLs and 31,000 phishing URLs, indicating that phishing samples constitute roughly 35% of the dataset. While this level of imbalance is not extreme, it is sufficient to necessitate careful model evaluation using metrics beyond accuracy (e.g., precision, recall, and AUC) to ensure robust performance, especially in detecting the minority class.

This dataset serves as the foundation for feature engineering, model training, and evaluation throughout the proposed system. Its diversity and real-world representativeness make it suitable for building and validating a phishing detection model that can generalize effectively to new, unseen URLs.

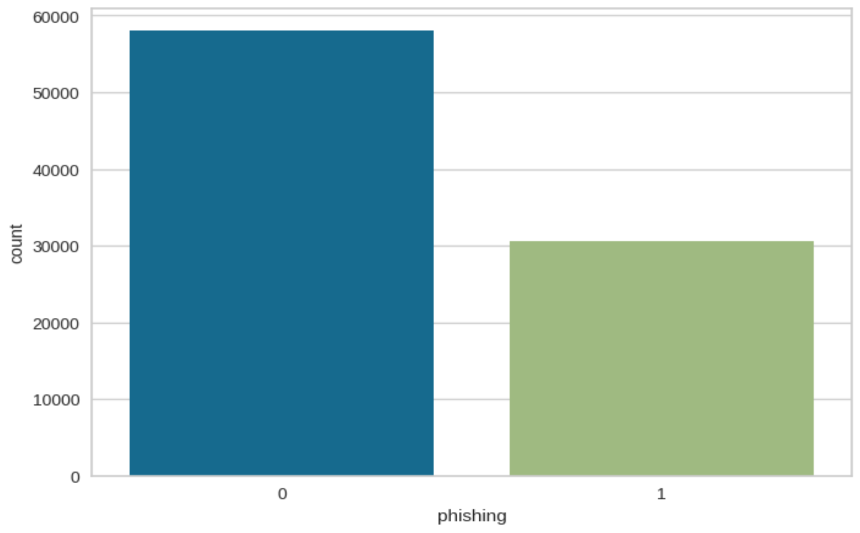


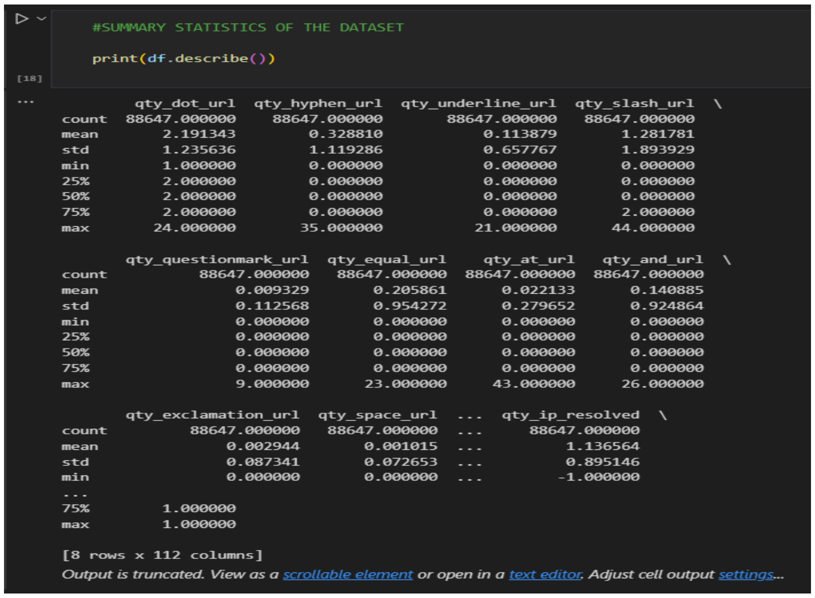
Fig.3: Distribution of Dataset

B. Data Cleaning and Feature Reduction

To enhance the performance and interpretability of the machine learning model, a feature reduction step was performed prior to training. The initial dataset included 112 features, many of which were either redundant, low in variance, or not strongly correlated with the target variable. Retaining such features could introduce noise and lead to overfitting or unnecessary computational overhead.

Features such as url_google_index, domain_google_index, tld_present_params, and other similar indicators were identified as non-contributive or weakly associated with phishing behavior. These were systematically removed based on exploratory analysis and domain relevance.

As a result of this pruning process, the feature space was reduced from 112 to 14 core attributes, preserving only those with high information gain and relevance to phishing detection. This step not only improved model efficiency but also enhanced the clarity of feature importance during evaluation.



```
#SUMMARY STATISTICS OF THE DATASET

print(df.describe())
```

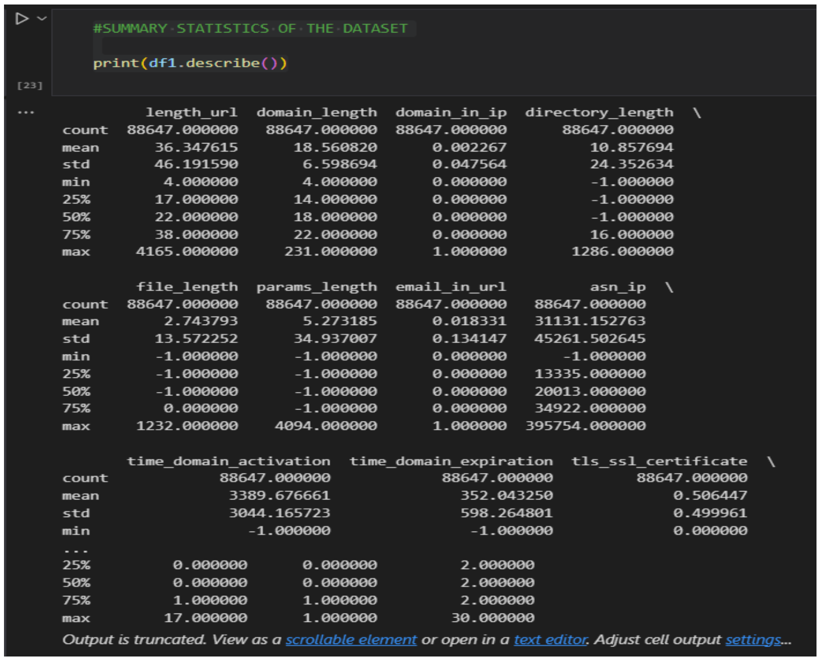
```
[18]
...
        qty_dot_url  qty_hyphen_url  qty_underline_url  qty_slash_url  \
count  88647.000000    88647.000000       88647.000000   88647.000000
mean       2.191343        0.328810           0.113879       1.281781
std        1.235636        1.119286           0.657767       1.893929
min        1.000000        0.000000           0.000000       0.000000
25%        2.000000        0.000000           0.000000       0.000000
50%        2.000000        0.000000           0.000000       0.000000
75%        2.000000        0.000000           0.000000       2.000000
max       24.000000       35.000000          21.000000      44.000000

       qty_questionmark_url  qty_equal_url    qty_at_url   qty_and_url  \
count          88647.000000   88647.000000  88647.000000  88647.000000
mean               0.009329       0.205861      0.022133      0.140885
std                0.112568       0.954272      0.279652      0.924864
min                0.000000       0.000000      0.000000      0.000000
25%                0.000000       0.000000      0.000000      0.000000
50%                0.000000       0.000000      0.000000      0.000000
75%                0.000000       0.000000      0.000000      0.000000
max                9.000000      23.000000     43.000000     26.000000

       qty_exclamation_url  qty_space_url  ...  qty_ip_resolved  \
count         88647.000000   88647.000000  ...     88647.000000
mean              0.002944       0.001015  ...         1.136564
std               0.087341       0.072653  ...         0.895146
min               0.000000       0.000000  ...        -1.000000
...
75%         1.000000
max         1.000000

[8 rows x 112 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Fig.4:Dataset Before Feature Elimination



```
#SUMMARY STATISTICS OF THE DATASET

print(df1.describe())
```

```
[23]
...
         length_url  domain_length  domain_in_ip  directory_length  \
count  88647.000000   88647.000000  88647.000000      88647.000000
mean      36.347615      18.560820      0.002267         10.857694
std       46.191590       6.598694      0.047564         24.352634
min        4.000000       4.000000      0.000000         -1.000000
25%       17.000000      14.000000      0.000000         -1.000000
50%       22.000000      18.000000      0.000000         -1.000000
75%       38.000000      22.000000      0.000000         16.000000
max     4165.000000     231.000000      1.000000       1286.000000

        file_length  params_length  email_in_url         asn_ip  \
count  88647.000000   88647.000000  88647.000000   88647.000000
mean       2.743793       5.273185      0.018331   31131.152763
std       13.572252      34.937007      0.134147   45261.502645
min       -1.000000      -1.000000      0.000000      -1.000000
25%       -1.000000      -1.000000      0.000000   13335.000000
50%       -1.000000      -1.000000      0.000000   20013.000000
75%        0.000000      -1.000000      0.000000   34922.000000
max     1232.000000    4094.000000      1.000000  395754.000000

       time_domain_activation  time_domain_expiration  tls_ssl_certificate  \
count            88647.000000            88647.000000         88647.000000
mean              3389.676661              352.043250             0.506447
std               3044.165723              598.264801             0.499961
min                 -1.000000               -1.000000             0.000000
...
25%          0.000000        0.000000          2.000000
50%          0.000000        0.000000          2.000000
75%          1.000000        1.000000          2.000000
max         17.000000        1.000000         30.000000
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Fig.5: Dataset After Feature Elimination

C. Feature Engineering

Feature engineering played a central role in transforming raw URLs into meaningful numerical data suitable for machine learning classification. The goal was to extract attributes that capture the structural, behavioral, and contextual characteristics of URLs that often differentiate phishing attempts from legitimate websites.

A custom feature extraction process was implemented to derive a comprehensive set of indicators from each URL. These features included both syntactic and metadata-driven attributes, carefully selected based on prior research and empirical relevance to phishing behavior.

The key features extracted are as follows:

1) length_url.
2) domain_length
3) directory_length and file_length
4) params_length
5) Boolean Indicators
6) domain_in_ip
7) email_in_url
8) tls_ssl_certificate
9) Behavioral and Domain Metadata
10) time_domain_activation and time_domain_expiration
11) qty_redirects
12) asn_ip
13) qty_char_domain

The final dataset, after feature engineering, included 14 carefully curated attributes that represent the most critical aspects of a URL from a phishing detection perspective. The engineered dataset retained 14 key features for training, as illustrated earlier in Fig. 5.

D. Correlation and Feature Selection

To refine the predictive capabilities of the model and reduce redundancy among input variables, a correlation analysis was conducted on the engineered features. This step helps in identifying attributes that may be linearly dependent or offer overlapping information, which could negatively impact model performance or introduce multicollinearity.

A correlation matrix was generated using Pearson's correlation coefficient to evaluate the relationships between numeric features. Features showing very high correlation (close to +1 or -1) were flagged for review. In such cases, only one of the strongly correlated features was retained based on domain relevance and empirical impact during preliminary model runs.

In addition to correlation analysis, feature importance scoring was later employed during model training using tree-based algorithms such as XGBoost. Features contributing minimal gain to the decision boundaries were marked as low-impact and considered for exclusion.

After completing this selection process, the feature set was optimized to include 14 significant attributes, each offering unique information relevant to phishing behavior—ranging from structural patterns to domain age and redirection behavior.

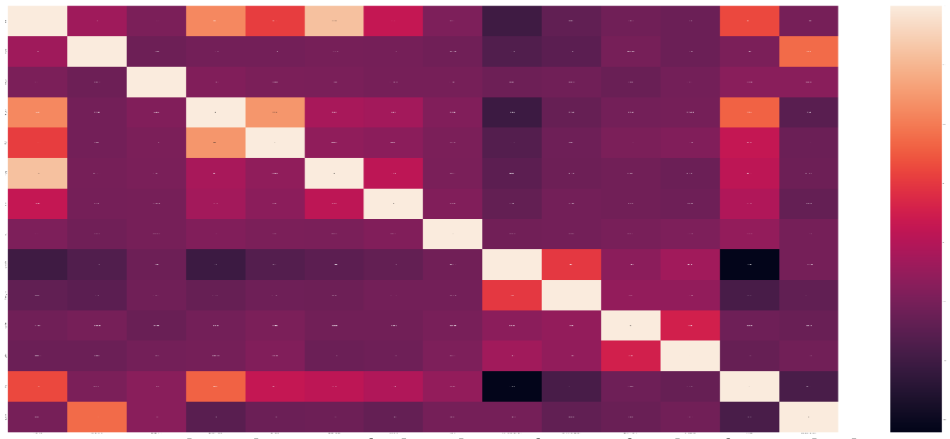


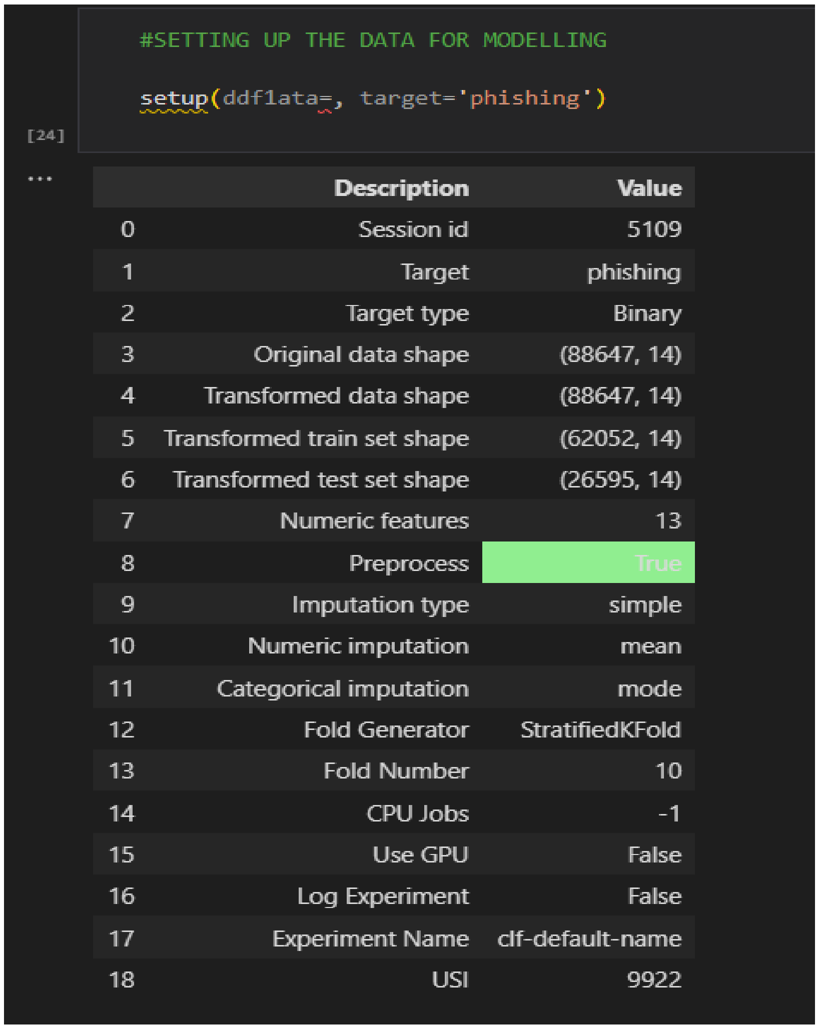
Fig.6: Correlation heatmap of selected URL features for identifying redundancy.

E. Model Setup Using PyCaret

This research employed PyCaret, an open-source machine learning library in Python, to simplify the pipeline and maintain consistency across preprocessing, training, and evaluation phases.PyCaret's setup() function was utilized to automate key preparatory steps, including data splitting, imputation, encoding, and transformation.
Upon initialization, PyCaret automatically inferred the types of each feature (numeric or categorical), handled missing values using mean imputation for numeric fields, and preserved the label column (phishing) as the binary target variable. The dataset was split into 80% training and 20% testing using stratified sampling to maintain class balance across subsets.
Additional configurations applied during setup included:

- 10-fold cross-validation using StratifiedKFold to evaluate models robustly.
- Automatic scaling of numeric features to normalize input distributions.
- Model evaluation metrics set to include Accuracy, Precision, Recall, F1-Score, and AUC.

This setup allowed for a consistent and efficient model comparison and ensured reproducibility across different runs.



```
#SETTING UP THE DATA FOR MODELLING

setup(ddf1ata=, target='phishing')
[24]
```

	Description	Value
0	Session id	5109
1	Target	phishing
2	Target type	Binary
3	Original data shape	(88647, 14)
4	Transformed data shape	(88647, 14)
5	Transformed train set shape	(62052, 14)
6	Transformed test set shape	(26595, 14)
7	Numeric features	13
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	9922

Fig.7:SetUp Data for Data Modelling

F. Model Comparison and Selection

After preparing the dataset and configuring the environment using PyCaret, multiple machine learning classification algorithms were trained and evaluated to identify the most effective model for phishing URL detection. PyCaret'scompare_models() function was used to automatically benchmark several models based on standard performance metrics.
The comparison included a variety of algorithms such as:

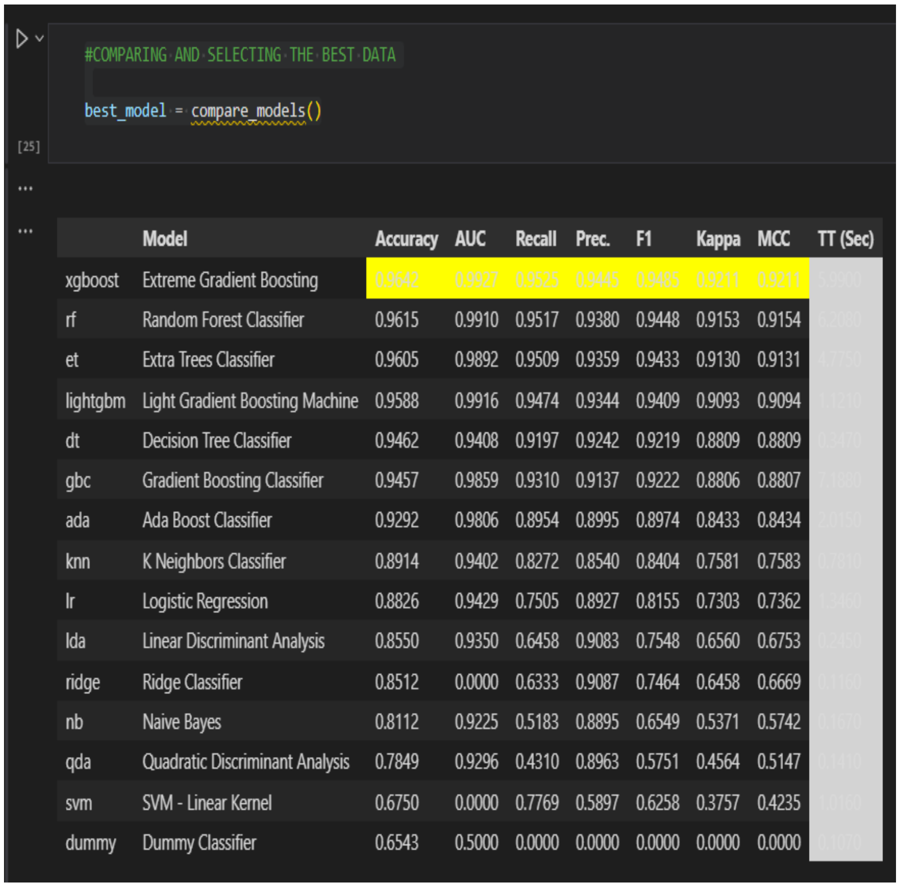
1) Logistic Regression (LR)

2) Decision Tree (DT)
3) Random Forest (RF)
4) K-Nearest Neighbors (KNN)
5) Support Vector Machine (SVM)
6) Naive Bayes (NB)
7) Gradient Boosting Models (GBM, XGBoost, LightGBM, AdaBoost)

Each model's performance was validated using a 10-fold cross-validation procedure on the training data. The metrics considered during comparison included Accuracy, AUC (Area Under the Curve), Precision, Recall, F1-Score, and MCC (Matthews Correlation Coefficient).

The XGBoost classifier had the best overall accuracy and efficacy out of all the models that were tested.

Other high-performing models included Random Forest and Extra Trees Classifier, though they marginally underperformed compared to XGBoost.



```
#COMPARING AND SELECTING THE BEST DATA

best_model = compare_models()
```

[25]

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9642	0.9927	0.9525	0.9445	0.9485	0.9211	0.9211	5.9900
rf	Random Forest Classifier	0.9615	0.9910	0.9517	0.9380	0.9448	0.9153	0.9154	6.2080
et	Extra Trees Classifier	0.9605	0.9892	0.9509	0.9359	0.9433	0.9130	0.9131	4.7750
lightgbm	Light Gradient Boosting Machine	0.9588	0.9916	0.9474	0.9344	0.9409	0.9093	0.9094	1.1210
dt	Decision Tree Classifier	0.9462	0.9408	0.9197	0.9242	0.9219	0.8809	0.8809	0.3470
gbc	Gradient Boosting Classifier	0.9457	0.9859	0.9310	0.9137	0.9222	0.8806	0.8807	7.1880
ada	Ada Boost Classifier	0.9292	0.9806	0.8954	0.8995	0.8974	0.8433	0.8434	2.0150
knn	K Neighbors Classifier	0.8914	0.9402	0.8272	0.8540	0.8404	0.7581	0.7583	0.7810
lr	Logistic Regression	0.8826	0.9429	0.7505	0.8927	0.8155	0.7303	0.7362	1.3460
lda	Linear Discriminant Analysis	0.8550	0.9350	0.6458	0.9083	0.7548	0.6560	0.6753	0.2450
ridge	Ridge Classifier	0.8512	0.0000	0.6333	0.9087	0.7464	0.6458	0.6669	0.1160
nb	Naive Bayes	0.8112	0.9225	0.5183	0.8895	0.6549	0.5371	0.5742	0.1670
qda	Quadratic Discriminant Analysis	0.7849	0.9296	0.4310	0.8963	0.5751	0.4564	0.5147	0.1410
svm	SVM - Linear Kernel	0.6750	0.0000	0.7769	0.5897	0.6258	0.3757	0.4235	1.0160
dummy	Dummy Classifier	0.6543	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1070

Fig.8: Model Comparison and Selection

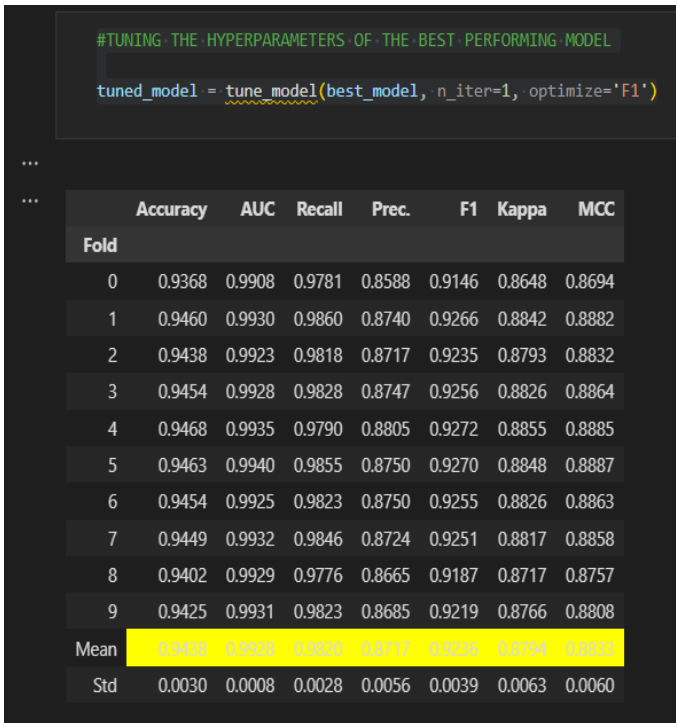
G. Hyperparameter Tuning

To enhance the performance of the chosen XGBoost model, hyperparameter optimization was carried out using the tune_model() function provided by PyCaret. This phase played a critical role in fine-tuning the model's predictive behavior and enhancing its ability to perform well on unseen data.

PyCaret employs Bayesian Optimization and randomized search strategies to explore a defined range of hyperparameters. For XGBoost, the tuning process considered parameters such as:

1) learning_rate: Controls the contribution of each tree.
2) max_depth: Limits the depth of individual trees.
3) n_estimators: Number of boosting rounds.
4) subsample: Fraction of samples used per tree.
5) colsample_bytree: Fraction of features used per tree.

The objective during tuning was to maximize the F1-score, which provides a balance between precision and recall—crucial in phishing detection, where false negatives can have severe consequences.
The tuning process resulted in a noticeable enhancement in performance compared to the initial baseline model. The tuned model achieved near-perfect discrimination between phishing and legitimate URLs while maintaining computational efficiency.
The optimized model was subsequently finalized and stored to enable its integration into the API-driven deployment framework.



```
#TUNING THE HYPERPARAMETERS OF THE BEST PERFORMING MODEL

tuned_model = tune_model(best_model, n_iter=1, optimize='F1')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Fold							
0	0.9368	0.9908	0.9781	0.8588	0.9146	0.8648	0.8694
1	0.9460	0.9930	0.9860	0.8740	0.9266	0.8842	0.8882
2	0.9438	0.9923	0.9818	0.8717	0.9235	0.8793	0.8832
3	0.9454	0.9928	0.9828	0.8747	0.9256	0.8826	0.8864
4	0.9468	0.9935	0.9790	0.8805	0.9272	0.8855	0.8885
5	0.9463	0.9940	0.9855	0.8750	0.9270	0.8848	0.8887
6	0.9454	0.9925	0.9823	0.8750	0.9255	0.8826	0.8863
7	0.9449	0.9932	0.9846	0.8724	0.9251	0.8817	0.8858
8	0.9402	0.9929	0.9776	0.8665	0.9187	0.8717	0.8757
9	0.9425	0.9931	0.9823	0.8685	0.9219	0.8766	0.8808
Mean	0.9438	0.9928	0.9820	0.8717	0.9236	0.8794	0.8833
Std	0.0030	0.0008	0.0028	0.0056	0.0039	0.0063	0.0060

Fig.9: Tuning the Hyperparameters of the Best Performing Model

H. Model Finalization and Deployment

Following the successful tuning and evaluation of the XGBoost model, the final step involved model preservation and real-time deployment integration. The best-performing model was finalized using PyCaret'sfinalize_model() function, ensuring that it was retrained on the complete training dataset with optimal hyperparameters.
The trained model was then serialized and stored as a .pkl file using Python's pickle library, producing a deployable artifact named phishing_url_detector.pkl. This file encapsulates the model, preprocessing steps, and configurations, making it portable and ready for production environments.
For real-time phishing URL detection, the model was integrated into a FastAPI-based web application. The system architecture includes:

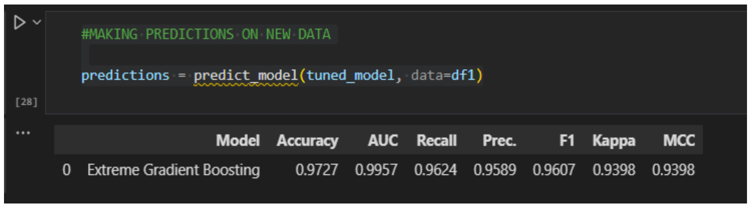
1) A user interface built with HTML and CSS for submitting URLs.
2) A backend server implemented using FastAPI that loads the trained model and handles incoming requests.
3) A custom feature extractor module that dynamically processes input URLs into structured feature vectors.
4) Model inference, which classifies the input URL as "Phishing" or "Legitimate", returning the result to the user.

This real-time system bridges the gap between model development and operational usability, demonstrating the practical impact of the proposed solution.

VI. RESULTS AND DISCUSSION

The performance of the proposed phishing URL detection system was evaluated using multiple metrics, including Accuracy, Precision, Recall, F1-Score, and AUC (Area Under the Curve). These metrics were derived from both cross-validation and hold-out test evaluations within the PyCaret framework.

After training and tuning, the XGBoost classifier emerged as the most effective model among all candidates. The performance outcomes of the final model:



```
#MAKING PREDICTIONS ON NEW DATA

predictions = predict_model(tuned_model, data=df1)
[28]

                       Model  Accuracy     AUC  Recall   Prec.      F1   Kappa     MCC
0  Extreme Gradient Boosting    0.9727  0.9957  0.9624  0.9589  0.9607  0.9398  0.9398
```

Fig.10: Performance of XGBoost Model

These results reflect the model's strong ability to distinguish between phishing and legitimate URLs with minimal false positives or false negatives. The high AUC score (0.9957) indicates excellent classification capability, especially in handling the imbalanced distribution between phishing and legitimate URLs.

Further analysis using cross-validation confirmed that the model remained stable across different folds, with low standard deviation values in performance metrics. This suggests a strong generalization ability and low variance, making the system suitable for real-world deployment.

The feature importance scores from the XGBoost model demonstrated that significant domain-related variables (such as time_domain_expiration, domain_in_ip, tls_ssl_certificate) and lexical features (like url_length, character_count_domain) had a considerable impact on the model's predictions. These findings align with existing cybersecurity research that emphasizes domain age, SSL usage, and URL complexity as critical indicators of phishing behavior.

These results validate the effectiveness of the engineered features and the chosen classification approach. The integration of this model into a real-time API-based system further demonstrates its practical viability for proactive phishing detection in modern web environments.

VII. CONCLUSION

This study presents an effective and deployable solution for detecting phishing URLs using machine learning techniques, with a strong emphasis on feature engineering, model optimization, and real-time deployment. The system could precisely distinguish between legitimate and malicious URLs by extracting and analyzing a comprehensive set of lexical, structural, and domain-related features.

The XGBoost classifier outperformed the other methods in terms of overall performance, with an accuracy of 97.27% and an AUC of 0.9957. These results confirm its robustness and suitability for security-critical applications. The selected features—particularly those related to domain behavior and URL structure—proved highly predictive, validating their importance in phishing detection.

Furthermore, the integration of the trained model into a FastAPI-based web application illustrates the system's real-world applicability. The deployed solution enables users to receive phishing predictions instantly, making it useful for email filtering, web browser plugins, and cybersecurity gateways.

In summary, it demonstrates that with thoughtful feature selection, proper model tuning, and deployment architecture, it is possible to build a high-performance phishing detection system that is both accurate and scalable.

REFERENCES

[1] Shraddha Parekh, Dhwanil Parikh, Srushti Kotak, Smita Sankhe, "A New Method for Detection of Phishing Websites: URL Detection," IEEE Xplore Compliant Conference, Part Number: CFP18BAC-ART; ISBN: 978-1-5386-1974-2, 2018.
[2] "Feature Selection for Machine Learning Based Detection of Phishing Websites," IEEE, 2017.
[3] Feature Selection for Machine Learning Based Detection of Phishing Websites," IEEE, Available: http://ieeexplore.ieee.org/abstract/document/8090317/?reload=true.
[4] Sanjukta Mohanty, "Predicting Phishing URL Using Filter Based Univariate Feature Selection Technique," IEEE Conference, 2022.
[5] V. S. Lakshmi, M. S. Vijaya, "Efficient Prediction of Phishing Websites using Supervised Learning Algorithms," Procedia Engineering, vol. 30, pp. 798–805, 2012.
[6] Hasane Ahammad Shaik, "Phishing URL Detection Using Machine Learning Methods," ResearchGate, Jan. 2022.
[7] D. Sahoo, "Malicious URL Detection Using Machine Learning: A Survey," 2022.
[8] Upendra Shetty D. R., Anusha Patil, Mohana, "Malicious URL Detection and Classification Analysis using Machine Learning Models," IEEE Xplore, Part Number: CFP23CV1-ART; ISBN: 978-1-6654-7451-1, 2023.

[9] Aniket Garje, Namrata Tanwani, SammedKandale, Twinkle Zope, Sandeep Gore, “Detecting Phishing Websites Using Machine Learning,” International Journal of Creative Research Thoughts (IJCRT), vol. 9, no. 11, pp. –, Nov. 2021, ISSN: 2320-2882.

[10] Rakesh Verma, et al., “What’s in a URL: Fast Feature Extraction and Malicious URL Detection,” Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy, pp. 55–63, 2017.

[11] Dipayan Sinha, Dr. Minal Moharir, Prof. Anitha Sandeep, “Phishing Website URL Detection using Machine Learning,” International Journal of Advanced Science and Technology, vol. 29, no. 3, pp. 2495–2504, 2020.

[12] Sri Hari Nallamala, Dr. Pragnyaban Mishra, Dr. Suvarna Vani Koneru, “Breast Cancer Detection using Machine Learning Way,” International Journal of Recent Technology and Engineering (IJRTE), vol. 8, issue 2S3, pp. –, July 2019, ISSN: 2277-3878.

[13] Sri Hari Nallamala, Dr. Pragnyaban Mishra, Dr. Suvarna Vani Koneru, “Pedagogy and Reduction of K-NN Algorithm for Filtering Samples in the Breast Cancer Treatment,” International Journal of Scientific & Technology Research (IJSTR), vol. 8, issue 11, pp. –, Nov. 2019, ISSN: 2277-8616.

[14] N. B. Naidu, Sri Hari Nallamala, Chukka Swarna Lalitha, Syed Seema Anjum, “Pertaining Formal Methods for Privacy Protection,” International Journal of Grid & Distributed Computing, vol. 13, no. 1, pp. –, Mar. 2020, ISSN: 2005-4262.

[15] Kranthi Madala, Sushma Chowdary Polavarapu, Sri Hari Nallamala, “Automatic Signal Indication System through Helmet,” International Journal of Advanced Science and Technology, vol. 29, no. 5, pp. –, Apr./May 2020, ISSN: 2005-4238.

[16] Sri Hari Nallamala, Dr. D. Durga Prasad, J. Ranga Rajesh, Dr. Pragnaban Mishra, Sushma Chowdary P, “A Review on Applications, Early Successes & Challenges of Big Data in Modern Healthcare Management,” TEST Engineering and Management Journal, vol. 83, issue 3, pp. –, May–June 2020, ISSN: 0193-4120.

[17] K. B. Prakash, Rama Krishna E, N. B. Naidu, Sri Hari Nallamala, Dr. Pragyaban Mishra, P. Dharani, “Accurate Hand Gesture Recognition using CNN and RNN Approaches,” International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 3, pp. –, May–June 2020, ISSN: 2278-3091.

[18] Sushma Chowdary P, Kranthi Madala, M. Sailaja, Sri Hari Nallamala, “Investigation on IoT System Design & Its Components,” Journal of Advanced Research in Dynamical & Control Systems, vol. 12, issue 6, pp. –, June 2020, ISSN: 1943-023X.

[19] Sri Hari Nallamala, Bajjuri Usha Rani, Anandarao S, Dr. Durga Prasad D, Dr. Pragnyaban Mishra, “A Brief Analysis of Collaborative and Content-Based Filtering Algorithms Used in Recommender Systems,” IOP Conference Series: Materials Science and Engineering, vol. 981, no. 2, 022008, Dec. 2020, ISSN: 1757-899X.

[20] Manukonda Vinay, Gonugunta B. S. Venkatesh, Malempati V. Priyanka, Dogiparthi V. Sai, Dr. Sri Hari Nallamala, “Deep Learning Based Face Mask Detection for User Safety from COVID-19,” International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), vol. 10, issue 5, pp. –, May 2022, e-ISSN: 2320-9801, p-ISSN: 2320-9798.

[21] P. R. Vyshnavi, M. V. N. S. Niharika, M. Summayya, P. Pravallika, Dr. Sri Hari Nallamala, “Liver Disease Prediction Using Machine Learning,” International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), vol. 11, issue 6, pp. –, June 2022, e-ISSN: 2319-8753, p-ISSN: 2320-6710.

[22] Y. Vineela Devi, T. Akshara, S. Mohitha, V. Venkatesh, N. Sri Hari, “Precision Farming by Analysing Soil Moisture and NPK Using Machine Learning,” IJIRSET, vol. 11, issue 6, pp. –, June 2022.

[23] Dr. N. Sri Hari, M. Ramya Sri, Mythri P., N. Sai Harshitha, M. V. N. S. Kumar, “Detection of COVID-19 Using Deep Learning,” IJFANS International Journal of Food and Nutritional Science, vol. 11, issue 12, pp. –, Dec. 2022, P-ISSN: 2319-1775, Online-ISSN: 2320-7876.

[24] Dr. N. Sri Hari, P. Vanaja, M. Ajay Kumar, M. D. V. S. Akash, K. Sivaiah, “Multi Disease Detection Using Machine Learning,” IJFANS International Journal of Food and Nutritional Science, vol. 11, issue 12, pp. –, Dec. 2022.

[25] Dr. N. Sri Hari, Shaik Nelofor, Siramdasu L. Vardhan, Sura R. P. Reddy, Sakhamuri Devendra, “CycleGAN Age Regressor,” International Journal for Innovative Engineering and Management Research, vol. 12, issue 4, pp. 45–51, Apr. 2023, ISSN: 2456-5083.

[26] Sudheer Mangalampalli et al., “Fault-Tolerant Trust-Based Task Scheduling Algorithm Using Harris Hawks Optimization in Cloud Computing,” Sensors, vol. 23, no. 18, 8009, 2023. DOI: https://doi.org/10.3390/s23188009

[27] K. Sudharson et al., “Hybrid Quantum Computing and Decision Tree Based Data Mining for Improved Data Security,” 7th Int. Conf. on Computing, Communication, Control and Automation (ICCUBEA), Aug. 2023. IEEE, ISBN: 979-8-3503-0426-8.

[28] G. S. Gandhi, K. Vikas, V. Ratnam, K. Suresh Babu, “Grid Clustering and Fuzzy Reinforcement Learning Based Energy-Efficient Data Aggregation Scheme for Distributed WSN,” IET Communications, vol. 14, no. 16, pp. 2840–2848.

[29] K. V. Prasad, G. S. Gandhi, S. Balaji, “Inexpensive Colour Image Segmentation by Using Mean Shift Algorithm and Clustering,” Int. J. of Graphics and Image Processing, vol. 4, no. 4, pp. 260–266.

[30] P. S. K. V. Maddumala, S. G. Gundabatini, P. Anusha, “Classification of Cancer Cells Detection Using Machine Learning Concepts,” Int. J. of Advanced Science and Technology, vol. 29, no. 3, pp. 9177–9190.

[31] S. G. Gundabatini, S. B. Kolluru, C. H. V. Ratnam, N. N. Krupa, “DAAM: WSN Data Aggregation Using Enhanced AI and ML Approaches,” Lecture Notes in Electrical Engineering (LNEE), vol. 976, June 2023.

[32] S. G. Gundabatini, E. Rayachoti, R. Vedantham, “Recurrent Residual Puzzle-Based Encoder Decoder Network (R2-PED) Model for Retinal Vessel Segmentation,” Multimedia Tools and Applications, 2023. https://doi.org/10.1007/s11042-023-16765-0

[33] S. G. Gundabatini, E. Rayachoti, R. Vedantham, “EU-net: An Automated CNN Based Ebola U-net Model for Efficient Medical Image Segmentation,” Multimedia Tools and Applications, 2024. https://doi.org/10.1007/s11042-024-18482-8

[34] M. Sánchez-Paniagua, E. Fidalgo Fernández, V. González-Castro, E. Alegre, W. Al-Nabki, “Phishing URL Detection: A Real-Case Scenario Through Login URLs,” IEEE Access, Apr. 2022.

[35] Dataset: “Phishing Website Detector,” [Online]. Available: https://www.kaggle.com/eswarchandt/phishing-website-detector

[36] M. Chatterjee, A. Siami Namin, “Detecting Phishing Websites through Deep Reinforcement Learning,” 2019 IEEE 43rd Annual COMPSAC, pp. –, 2019.

[37] S. C. Jeeva, E. B. Rajsingh, “Intelligent Phishing URL Detection Using Association Rule Mining,” Human-centric Computing and Information Sciences, vol. 6, no. 10, 2016. https://doi.org/10.1186/s13673-016-0064-3

[38] M. Sánchez-Paniagua et al., “Impact of Current Phishing Strategies in Machine Learning Models for Phishing Detection,” in 13th Int. Conf. on Computational Intelligence in Security for Information Systems (CISIS 2020), Springer, vol. 1267, pp. –, 2021.

[39] P. C. R. Chinta, C. S. Moore, L. M. Karaka, M. Sakuru, V. Bodepudi, S. R. Maka, "Building an Intelligent Phishing Email Detection System Using Machine Learning and Feature Engineering," Eur. J. Appl. Sc. Eng. Technol., vol. 3, no. 2, pp. 41–54, Mar.–Apr. 2025. DOI: 10.59324/ejaset.2025.3(2).04
[40] E. Gandotra, D. Gupta, "An Efficient Approach for Phishing Detection Using Machine Learning," in Multimedia Security, Springer, 2021. https://doi.org/10.1007/978-981-15-8711-5_12
[41] A. Begum, S. Badugu, "A Study of Malicious URL Detection Using Machine Learning and Heuristic Approaches," in ICETE 2019, Springer, vol. 4, pp. –, 2020. https://doi.org/10.1007/978-3-030-24318-0_68
[42] R. Patgiri, H. Katari, R. Kumar, D. Sharma, "Empirical Study on Malicious URL Detection Using Machine Learning," in ICDCIT 2019, Springer, vol. 11319, pp. –, 2019. https://doi.org/10.1007/978-3-030-05366-6_31
[43] Sri Hari Nallamala, Kommu Namitha, Kunchanapalli Raviteja, Kadiyam Sai Sumanth, Jyothi Sri Kota, "Phishing Website Detection Using Machine Learning," International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 12, no. 4, pp. 1387–1392, Apr. 2024. DOI: 10.22214/ijraset.2024.59261