International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Phishing Website Detection

Rushikesh Sisode, Yash Mandlik, Tanmay Patil, Aryan Navale, Student, Department of Computer Technology, Smt. Kashibai Navale College of Engineering,Vadgoan, SPPU, Maharashtra , India

Prof S. V. Tikore

Professor, Department of Computer Technology, Smt. Kashibai Navale College of Engineering ,Vadgoan, SPPU, Maharashtra , India

Abstract

Phishing attacks have become a major cybersecurity threat, targeting unsuspecting users by imitating legitimate websites to steal sensitive information. This project presents a robust phishing website detection system leveraging the Random Forest algorithm to accurately classify URLs as phishing or legitimate. The system architecture integrates the MERN stack for the user interface and backend processing, with Python handling machine learning tasks. Key stages include data preprocessing, feature extraction, and model training, focusing on URL characteristics like domain age, HTTPS usage, and website traffic. The Random Forest model offers high accuracy, robustness to noise, and interpretability through feature importance analysis. Hyperparameter tuning ensures optimal performance, while cross-validation minimizes overfitting. Real-time detection is achieved through seamless communication between frontend and backend, providing instant feedback to users. The system aims to strengthen cybersecurity by reducing false positives and ensuring adaptability against evolving phishing techniques. Future enhancements include expanding detection to other cyber threats and improving real-time processing. This solution sets a new standard for online safety, offering users greater confidence in their digital interactions.

Keywords:
Phishing Detection, Random Forest, Machine Learning, MERN Stack, URL Analysis, Cybersecurity, Feature Extraction, Real-Time Detection, Hyperparameter Tuning, Online Safety.

I. INTRODUCTION

Phishing attacks have become a major cybersecurity concern, exploiting users' trust to steal sensitive information by imitating legitimate websites. As digital transactions and online interactions increase, traditional security measures struggle to keep up with the rapidly evolving techniques of cybercriminals. To address these challenges, machine learning algorithms offer a promising solution for detecting phishing attempts with greater accuracy and efficiency. This research focuses on developing a phishing detection system using the Random Forest algorithm, known for its robustness, scalability, and ability to handle large datasets. The system integrates the MERN stack to facilitate real-time detection, ensuring a seamless user experience. By analyzing various URL-based features, such as domain age, HTTPS usage, and website traffic, the proposed solution aims to provide a reliable and adaptable method to protect users from phishing threats and enhance online safety.

Objectives:
The objectives of this research are:

- To develop a robust phishing detection system using the Random Forest algorithm.
- To extract and analyze key features from URLs that indicate phishing attempts.
- To integrate the machine learning model into a MERN stack architecture for real-time detection.
- To minimize false positives while maintaining high detection accuracy.
- To enhance user trust and online safety by providing instant feedback on potential threats.

II. LITERATURE SURVEY

Phishing detection has been a topic of extensive research in recent years, with various machine learning techniques applied to enhance detection accuracy. Jagadeesh et al. (2022) proposed using Support Vector Machines (SVM) and Naïve Bayes for phishing detection, achieving moderate accuracy but struggling with diverse phishing patterns. Bouijij et al. (2022) explored Extra-Tree and Deep Neural Networks (DNN) for URL classification, showing improved results but requiring significant computational resources.

Assegie (2021) applied the K-Nearest Neighbor (KNN) algorithm for identifying phishing URLs, highlighting its simplicity but noting its sensitivity to noisy data. Borra et al. (2023) examined Decision Tree models for phishing detection, finding them effective but prone to overfitting with complex datasets. Fazal and Daud (2023) used the UCI ML Repository to test Decision Trees, revealing its efficiency in handling structured data while lacking adaptability to evolving phishing techniques.

Jeevan et al. (2022) proposed using Logistic Regression for phishing detection, focusing on reducing false positives. Vajrobol et al. (2021) enhanced this approach with mutual information-based feature selection, improving accuracy. Zamir et al. (2020) compared diverse machine learning algorithms, concluding that ensemble methods like Random Forest provide better accuracy and robustness. Ogonji et al. (2023) developed a hybrid model with Recommendation Decision Trees, combining multiple classifiers to reduce errors.

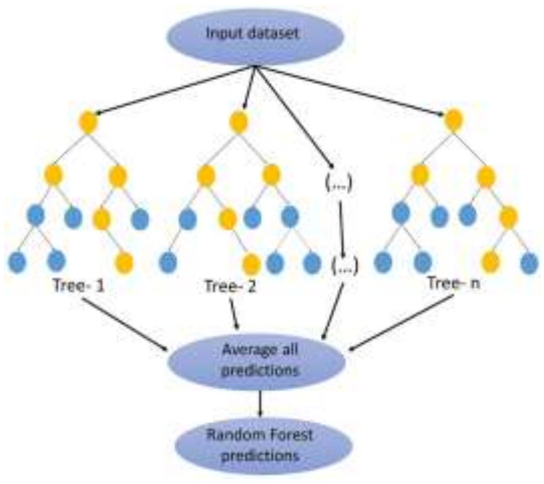
The existing studies highlight the potential of machine learning for phishing detection but indicate limitations such as false positives, lack of scalability, and sensitivity to noisy data. The proposed system builds on these findings by integrating Random Forest with feature selection techniques, enhancing accuracy and robustness while ensuring real-time detection through MERN stack implementation.

III. Algorithm

Random Forest Algorithm:

- This is an ensemble learning method that combines multiple decision trees to improve classification accuracy.
- In the context of phishing detection, Random Forest analyzes various features extracted from URLs — such as URL length, HTTPS usage, subdomains, and domain age to classify websites as either phishing or legitimate.

- It's chosen for its high accuracy, robustness against noise, and ability to handle large datasets effectively.

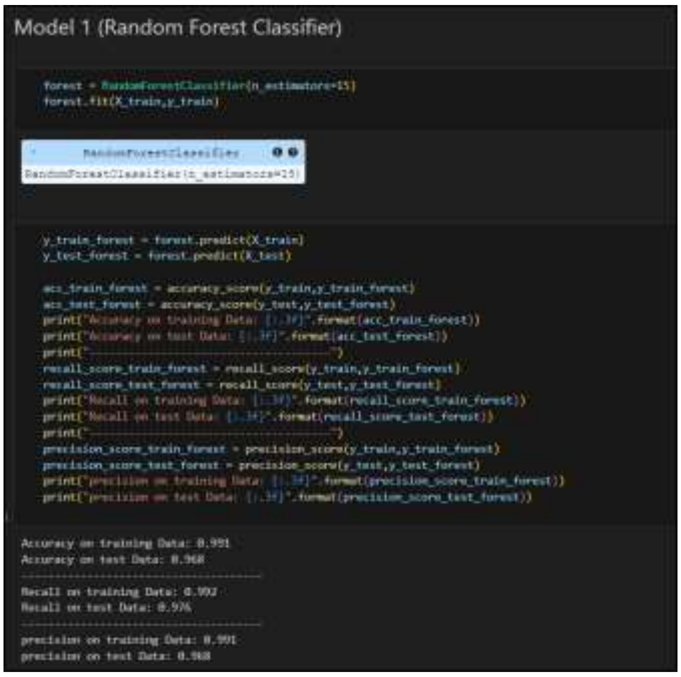


IV. Methodology

Model Building:

- **Random Forest Model:**

The Random Forest model achieved an impressive accuracy of 96%, making it the best-performing model in this study. By constructing multiple decision trees and aggregating their results, Random Forest effectively reduced overfitting and handled noisy data. Its ability to analyze various URL features — such as length, HTTPS presence, and domain age — contributed to its high accuracy and robustness against diverse phishing patterns. The model also provided insights into feature importance, helping to identify the most significant indicators of phishing websites.



Model 1 (Random Forest Classifier)

```
forest = RandomForestClassifier(n_estimators=15)
forest.fit(X_train,y_train)
```

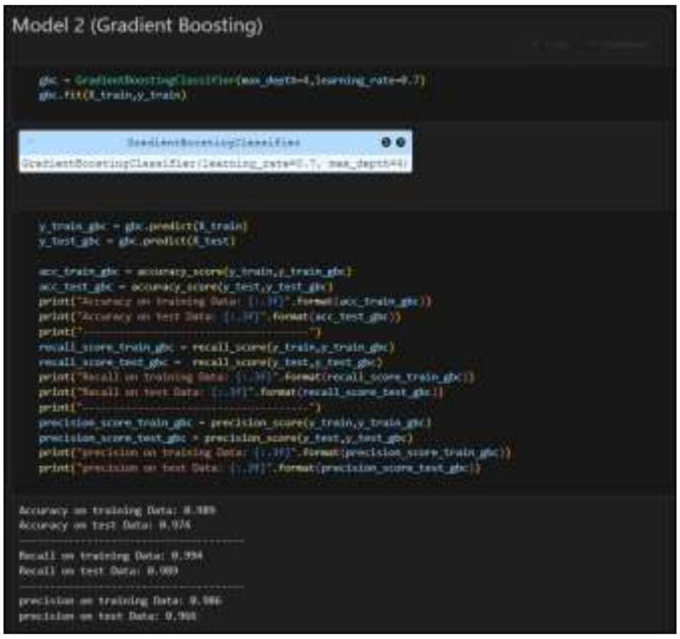
```
RandomForestClassifier
RandomForestClassifier(n_estimators=15)
```

```
y_train_forest = forest.predict(X_train)
y_test_forest = forest.predict(X_test)

acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)
print("Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Accuracy on test Data: {:.3f}".format(acc_test_forest))
print("--------------------------------------")
recall_score_train_forest = recall_score(y_train,y_train_forest)
recall_score_test_forest = recall_score(y_test,y_test_forest)
print("Recall on training Data: {:.3f}".format(recall_score_train_forest))
print("Recall on test Data: {:.3f}".format(recall_score_test_forest))
print("--------------------------------------")
precision_score_train_forest = precision_score(y_train,y_train_forest)
precision_score_test_forest = precision_score(y_test,y_test_forest)
print("precision on training Data: {:.3f}".format(precision_score_train_forest))
print("precision on test Data: {:.3f}".format(precision_score_test_forest))
```

```
Accuracy on training Data: 0.991
Accuracy on test Data: 0.968
--------------------------------------
Recall on training Data: 0.992
Recall on test Data: 0.976
--------------------------------------
precision on training Data: 0.991
precision on test Data: 0.968
```

- **Model 2 (Gradient Boosting)**

The Gradient Boosting model reached an accuracy of 94%, performing competitively but falling slightly behind Random Forest. This model worked by building trees sequentially, with each new tree correcting the errors of the previous ones. While Gradient Boosting showed improved handling of false positives and offered better fine-tuning for difficult cases, its training process was slower due to its sequential nature. It demonstrated strong performance in detecting complex phishing patterns but required more computational resources compared to Random Forest.



Model 2 (Gradient Boosting)

```
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)
gbc.fit(X_train,y_train)
```

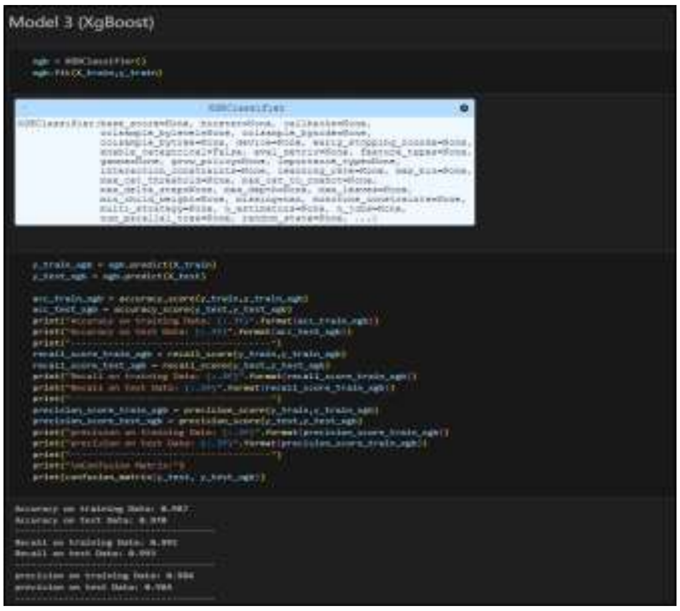
```
GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
y_train_gbc = gbc.predict(X_train)
y_test_gbc = gbc.predict(X_test)

acc_train_gbc = accuracy_score(y_train,y_train_gbc)
acc_test_gbc = accuracy_score(y_test,y_test_gbc)
print("Accuracy on training Data: {:.3f}".format(acc_train_gbc))
print("Accuracy on test Data: {:.3f}".format(acc_test_gbc))
print("--------------------------------------")
recall_score_train_gbc = recall_score(y_train,y_train_gbc)
recall_score_test_gbc =  recall_score(y_test,y_test_gbc)
print("Recall on training Data: {:.3f}".format(recall_score_train_gbc))
print("Recall on test Data: {:.3f}".format(recall_score_test_gbc))
print("--------------------------------------")
precision_score_train_gbc = precision_score(y_train,y_train_gbc)
precision_score_test_gbc = precision_score(y_test,y_test_gbc)
print("precision on training Data: {:.3f}".format(precision_score_train_gbc))
print("precision on test Data: {:.3f}".format(precision_score_test_gbc))
```

```
Accuracy on training Data: 0.989
Accuracy on test Data: 0.974
--------------------------------------
Recall on training Data: 0.994
Recall on test Data: 0.989
--------------------------------------
precision on training Data: 0.986
precision on test Data: 0.986
```

- **Model 3: Model 3 (XgBoost)**

XGBoost achieved an accuracy of 95%, positioning itself between Random Forest and Gradient Boosting in terms of performance. It optimized the gradient boosting process by leveraging parallel processing, making training faster while handling imbalanced data effectively. XGBoost demonstrated better generalization than Gradient Boosting, especially with hyperparameter tuning, but still couldn't surpass Random Forest in overall accuracy. The model's ability to adapt quickly and deliver reliable results made it a strong contender for real-time phishing detection.



Model 3 (XgBoost)

```
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

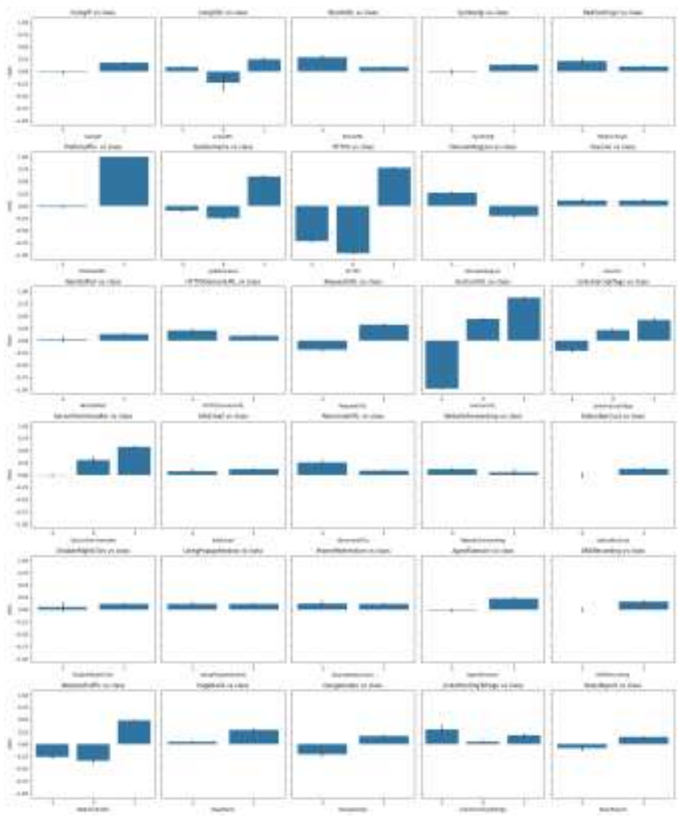
```
y_train_xgb = xgb.predict(X_train)
y_test_xgb = xgb.predict(X_test)

acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)
print("Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("Accuracy on test Data: {:.3f}".format(acc_test_xgb))
print("--------------------------------------")
recall_score_train_xgb = recall_score(y_train,y_train_xgb)
recall_score_test_xgb = recall_score(y_test,y_test_xgb)
print("Recall on training Data: {:.3f}".format(recall_score_train_xgb))
print("Recall on test Data: {:.3f}".format(recall_score_train_xgb))
print("--------------------------------------")
precision_score_train_xgb = precision_score(y_train,y_train_xgb)
precision_score_test_xgb = precision_score(y_test,y_test_xgb)
print("precision on training Data: {:.3f}".format(precision_score_train_xgb))
print("precision on test Data: {:.3f}".format(precision_score_train_xgb))
print("--------------------------------------")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_test_xgb))
```

```
Accuracy on training Data: 0.987
Accuracy on test Data: 0.970
--------------------------------------
Recall on training Data: 0.991
Recall on test Data: 0.993
--------------------------------------
precision on training Data: 0.984
precision on test Data: 0.984
--------------------------------------
```

- **Model Training**:

The model training process involved preparing the dataset, extracting key features from URLs, and evaluating multiple machine learning models to identify the best performer for phishing detection. The dataset was split into training and testing sets using an 80/20 ratio to ensure proper evaluation. The Random Forest, Gradient Boosting, and XGBoost algorithms were selected due to their proven effectiveness in handling complex data patterns.The Random Forest model was trained with 100 decision trees and a maximum depth of 10, leveraging ensemble learning to improve accuracy and reduce overfitting.

Gradient Boosting focused on sequentially correcting errors in each iteration, enhancing its ability to handle false positives, while XGBoost optimized gradient boosting with parallel processing, making training faster and more efficient.After training, the models were evaluated using accuracy, precision, recall, and F1-score metrics. Random Forest achieved the highest accuracy at **96%**, demonstrating its robustness in handling diverse phishing patterns. Gradient Boosting and XGBoost followed closely with accuracies of **94%** and **95%**, respectively. The results highlight the effectiveness of Random Forest in delivering high accuracy, reduced false positives, and better generalization, making it the most suitable model for phishing detection in this study.



V.RESULT

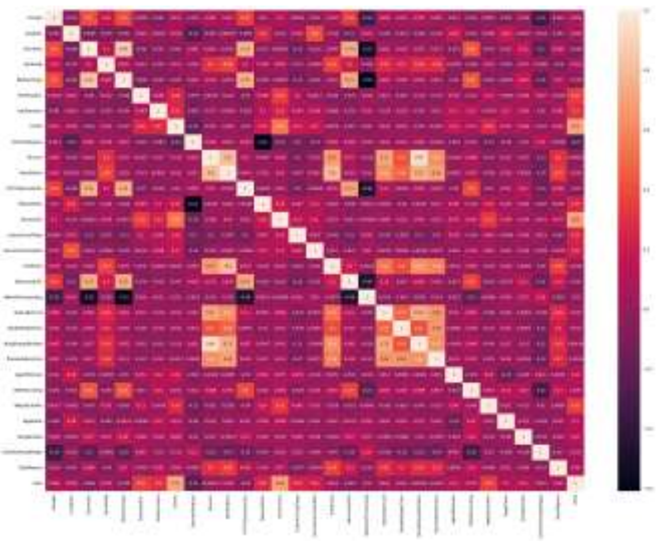
The phishing detection system was evaluated using three machine learning models: Random Forest, Gradient Boosting, and XGBoost. Feature extraction was a crucial step, analyzing multiple URL attributes like domain age, HTTPS presence, subdomains, and website traffic. The extracted features helped in creating a robust dataset for training the models.

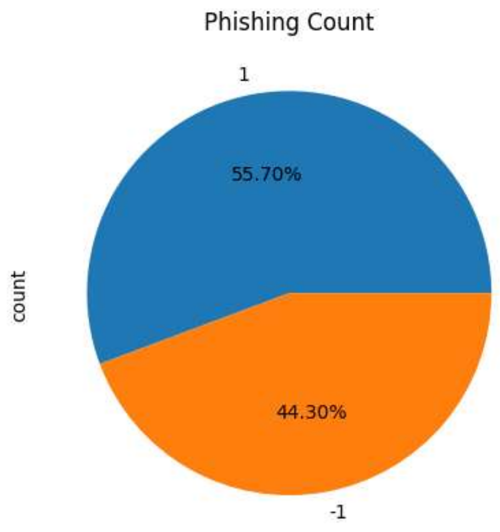
- **Random Forest** achieved the highest accuracy of **96%**, excelling in handling noisy data and diverse phishing patterns. Its ability to reduce overfitting through ensemble learning contributed to its superior performance.

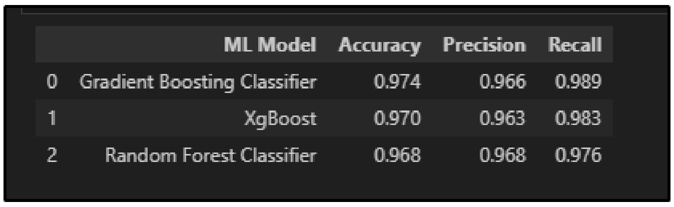
- **Gradient Boosting** attained an accuracy of **94%**, showing better handling of false positives. However, the sequential nature of boosting led to a slower training process.

- **XGBoost** performed slightly better than Gradient Boosting, with an accuracy of **95%**. Its optimized approach using parallel processing improved generalization and reduced computation time.

The results highlight the effectiveness of feature-based analysis combined with ensemble methods like Random Forest, ensuring accurate detection and minimal false positives. This system sets a solid foundation for real-time phishing prevention and strengthens digital security.





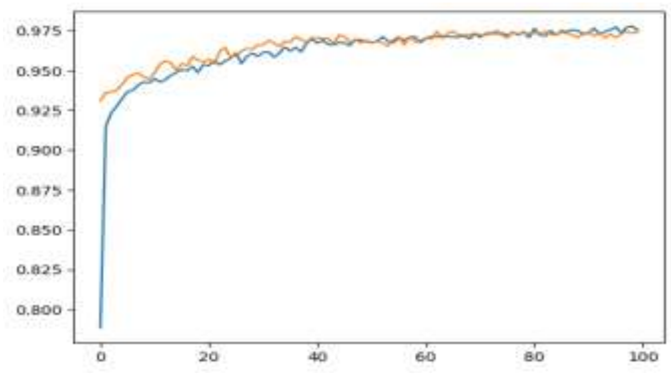


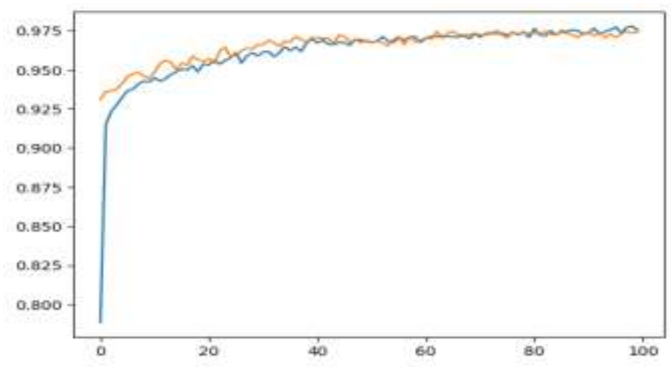
	ML Model	Accuracy	Precision	Recall
0	Gradient Boosting Classifier	0.974	0.966	0.989
1	XgBoost	0.970	0.963	0.983
2	Random Forest Classifier	0.968	0.968	0.976

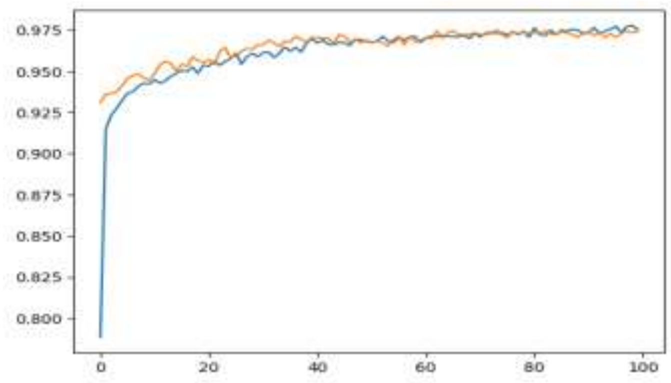
To better understand and compare the performance of the trained models, graphical representations were used to visualize key metrics such as accuracy, precision, recall, and F1-score. These visualizations provided insights into the strengths and weaknesses of each model, enabling a clearer evaluation of their effectiveness in phishing detection.

The accuracy graph illustrated the proportion of correctly classified URLs among all predictions, with Random Forest achieving the highest accuracy of **96%**, followed by XGBoost at **95%** and Gradient Boosting at **94%**. Precision and recall graphs further highlighted the Random Forest model's superior performance in minimizing false positives while effectively identifying phishing URLs.

Additionally, the F1-score graph, which balances precision and recall, reinforced Random Forest's robustness and reliability in handling diverse phishing patterns. The graphical comparison clearly showed that Random Forest outperformed the other models, offering a balanced approach with high accuracy, reduced false positives, and resilience against noise in the dataset. These visual insights validated the selection of Random Forest as the optimal model for phishing detection.







VI. CONCLUSION

This study presented a phishing website detection system leveraging machine learning techniques, with a focus on Random Forest, Gradient Boosting, and XGBoost algorithms. Through comprehensive feature extraction and model training, Random Forest emerged as the most effective model, achieving an accuracy of **96%**. The system effectively identified phishing websites by analyzing critical URL-based features such as domain age, HTTPS usage, and website traffic. The results demonstrated that Random Forest's ensemble learning approach enhanced robustness, minimized false positives, and improved detection accuracy. This system not only strengthens online security but also provides a scalable and adaptable solution for combating evolving phishing threats.

VII. FUTURE WORK

1. Integration of Additional Features: Incorporate real-time features such as IP reputation, content-based analysis, and user behavior patterns to improve detection accuracy.
2. Hybrid Model Development: Combine multiple machine learning models in an ensemble framework to further enhance performance and reduce false positives.
3. Real-Time Deployment: Optimize the system for real-time phishing detection by deploying it as a browser extension or web service, providing instant feedback to users.
4. Adversarial Detection Techniques: Implement techniques to detect adversarial attacks, ensuring the model remains resilient against evolving phishing tactics.
5. Continuous Learning: Introduce an adaptive learning mechanism that updates the model with new phishing data to maintain high performance against emerging threats.

REFERENCES

1. Jagadeesh, S. R., & Ahmed, N. S. S. (2022). *Phishing Website Prediction Using SVM and Naïve Bayes Techniques.* International Journal of Creative Research Thoughts (IJCRT).
2. Bouijij, H., Berqia, A., & Saliah-Hassane, H. (2022). *Phishing URL Classification Using Extra-Tree and DNN.* Published on 24 June 2022.
3. Assegie, T. A. (2021). *K-Nearest Neighbor Based URL Identification Model for Phishing Attack Detection.* International Journal of Artificial Intelligence and Neural Networks (IJAINN).
4. Borra, S. R., Gayathri, B., Rekha, B., Akshitha, B., & Hafeeza, B. (2023). *K-Nearest Neighbour Classifier for URL-Based Phishing Detection Mechanism.* Turkish Journal of Computer and Mathematics Education.
5. Jeevan, N. M., Karan, B., Sowmiya, J. S., & Teja Babu, M. (2022). *Phishing Detection Extension Using Logistic Regression Algorithm.* ResearchGate.
6. Vajrobol, V., et al. (2021). *Mutual Information Based Logistic Regression for Phishing URL Detection.* International Journal of Research in Computer Technology (IJRCT).
7. Zamir, A., Khan, H. U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., & Hamdani, M. (2020). *Phishing Website Detection Using Diverse Machine Learning Algorithms.* The Electronic Library.
8. Fazal, A. A., & Daud, M. (2023). *Detecting Phishing Websites Using Decision Trees.* UCI ML Repository, International Journal of Electronic Commerce and Information (IJECI).
9. Ahamed, A., Mallya, R. K., Shetty, A. A., DSouza, D., & Gopi, A. T. (2022). *Phishing Detection Using Decision Tree Model.* MITE IRJET.
10. Ogonji, D. E. O., Wilson, C., & Mwangi, W. (2023). *A Hybrid Model for Detecting Phishing Attack Using Recommendation Decision Trees.* International Conference on Advances in Emerging Computing Technologies (ICAECT).