JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

CYBER THREATS PREDICTION AND ANALYSIS USING MACHINE LEARNING ALGORITHMS

Mridul Kanti Sikder, Himanshu kumar sah
Student
National Institute Of Technology Durgapur

Abstract

In the ever-shifting terrain of cybersecurity, threat prediction and analysis became central to creating a lateral defense mechanism. The work presents a strong comparison of five machine learning algorithms, viz. Random Forest, Support Vector Machine (SVM), Naive Bayes, XGBoost, and Decision Trees for cyber-threat prediction and classification. The past few years have seen the development and maintenance of a large-scale dataset containing various cyber threat indicators and network behavior patterns. After several iterations and optimizations, the Random Forest seemed to do best with the threat predictive classification of 80.88% accuracy and an F1-score rating of 0.84. The model performed extraordinarily well in classifying high-severity threats because, for some threat categories, the precision increases went up to 95%. On top of that, hyperparameter tuning on the Random Forest model was applied in the study, increasing prediction accuracy and reducing false positives even further. The comparative study also offered some insights regarding training time for Random Forest, XGBoost, and SVM models with Naive Bayes being the fastest at 9.8 seconds with decent accuracy. The study contributes to the incidents by establishing a systematic approach to cyber threat prediction and by giving some hints for practical applications for machine learning-based security solutions [1]

Keywords: Machine Learning, Cyber Security, Random Forest, Support Vector Machine, XGBoost, Naive Bayes, Decision Trees, Predictive Analytics, Network Security, Feature Importance, Model Optimization, Classification Algorithms, Threat Detection, Cyber Threat Intelligence, Performance Analysis.

INTRODUCTION

Cyberattacks are getting increasingly sophisticated, cybersecurity is meaningfully critical to reduce malicious attacks to protect the networks, systems, and data from threats, such as viruses, ransomware, and phishing schemes. As systems are increasingly more complex, manual processes of detection are becoming less effective. Machine Learning (ML) solutions utilized for threat alleviation are more effective due to being smarter, faster, and organized. The model used in this study utilized an optimized Random Forest classifier, in total went through supervised ML algorithms applied to real-world instances of cyber threats [1] it showed 80.88% accurateness with acceptable levels of precision that shows the viability of using ML improves automated cybersecurity systems in real-world digital environments that continuously change.

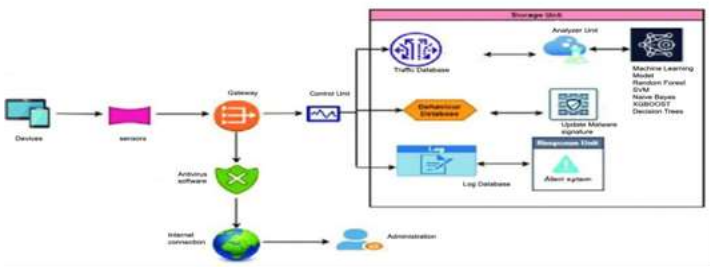


Fig 1: Architecture for threat detection

Literature Review

A number of studies have been conducted in network and IoT intrusion detection based on machine learning and deep learning. Peng Xiao suggested a high-accuracy deep learning-based network malware detection approach with a hybrid framework very applicable to Android security, but it is methodologically opaque and does not provide details for deployment. R. A. Disha and S. Waheed proposed a GIWRF feature selection method that had outstanding performance figures (99.90% accuracy, 99.85% F1) on

contemporary datasets such as TON-IoT and UNSW-NB15 but was restricted to binary classification and targeted datasets. Likewise, S. M. Kasongo and Y. Sun employed XGBoost feature reduction to enhance decision tree accuracy to 90.85% and reduce model complexity while their research was restricted to UNSW-NB15 and had poor identification of infrequent classes. A. Septiadi et al. compared five ML models and concluded that Random Forest worked best on KDD'99 and SVM on UNSW-NB15, but they left out the deep learning architectures and computational cost evaluations. S. S. Dhaliwal et al. utilized XGBoost for the development of an efficient intrusion detection system with 98.98% accuracy on the NSL-KDD dataset and robust generalization, but the old dataset and absence of multiclass testing are significant demerits. A. Hasan et al. used SVM and Random Forest to model IDS on a redundant KDD'99 dataset and reported SVM slightly more accurate and RF quicker, though with a limited study that used a redundant data set and only two classifiers. J proposed a multi-level model that integrated FS, RF, and fuzzy inference. B. Awotunde et al, attaining 99.46% accuracy and severity classification for IoT networks but it had a complicated pipeline and 6.14% false positive rate. Lastly, J. Li et al. compared feature extraction and selection for optimizing IoT IDS and found that feature extraction provided higher accuracy with fewer features—where decision trees had the best performance for FS and MLP for FE—but since only the TON-IoT dataset is used and only a few techniques are employed, the generalizability of their conclusion is limited.

Methodology

The proposed methodology is a systematic approach towards cyber threat prediction with five prominent machine learning algorithms: Random Forest, Support Vector Machine, Naive Bayes, XGBoost, and Decision Tree. Each algorithm being different from one another adds unique strengths to the prediction framework, allowing the combined analysis of cyber threats.

1. Random Forest

Random Forest is a suitable model for this cybersecurity data set due to its ensemble learning characteristic. With GUIDE having 45 diverse attributes of evidence, alerts, and incidents, Random Forest's ability to handle high-dimensional data without overfitting is critical. It can efficiently process both categorical features (such as DetectorIds and entity types) and numerical features (such as timestamps and metrics) [2] . The ability of the algorithm to rank feature importance is particularly useful in knowing the security indicators that are most predictive of incident.[7] In addition, its noise tolerance and ability to handle the imbalanced data set (unequal numbers of true positives, false positives, and benign positives) make it an ideal choice to predict security incidents.

2. Support Vector Machine (SVM)

The choice of SVM was based on its stability in high-dimensional spaces of features, which is particularly valuable to the rich feature set of the GUIDE dataset spanning 33 entity types. The capability of the algorithm in determining optimal separation boundaries between various incident classifications (TP, BP, FP) makes it beneficial for precise incident triage [10] SVM's kernel function enables it to identify non-linear patterns within security data, which is especially important since security attacks tend to be composed of complex, non-linear patterns. The algorithm's being situated high in theoretical framework within statistical learning positions it favorably for the dataset's requirement of high confidence level (99%) in machine-driven decision-making.

3. Naive Bayes

Naive Bayes was chosen due to the fact that it is computationally expensive and probabilistic, and this is a requirement of the vast scale of the GUIDE dataset (13 million evidence points). That it can easily handle missing values - a given feature of security telemetry data - makes it ideal. Probabilistic in nature is the algorithm that best suits security incident prediction, where it is important to know the probability of various types of incidents occurring. In spite of the naivety of its independence assumption about features, Naive Bayes per se really performs very well on text-intensive security data and can handle the multiple categorical features of the entity types and alert types in the dataset quit well.

4.XGBoost

XGBoost was chosen because it has better performance in processing complex data with multi-features and hierarchical organization. The gradient boosting nature of the algorithm renders it better suited for the GUIDE dataset's high demand for prediction precision [7] in detecting security incidents. The integrated missing value management by XGBoost and its capability for efficient processing of large datasets (size of training set: 2.43 GB) also align with security use cases in actual operating environments. Its unbalanced class support - not unusual in security incident data - and feature importance ranking provide security analysts with insightful information.

5. Decision Trees

Decision Trees were included primarily due to their transparency and interpretability of decision-making, necessary in security operations.The capability of the algorithm to generate transparent, logical decision paths fits well with the hierarchical structure of the GUIDE dataset (evidence → alerts → incidents). Decision Trees are able to naturally deal with both categorical and numerical features without the need for preprocessing, and hence are appropriate for the heterogenous feature set in security telemetry data. Their clear rule-based formulation is very informative of the decision-making process that is critical for security analysts to comprehend and verify the predictions of the model.

Machine Learning Algorithms:

1.Random Forest (RN)[3] [4] [1]

- Selected due to its ability to handle complex security information with numerous attributes
- Effective for categorical as well as numerical data in the data set
- Can detect non-linear patterns between security incidents
- Provides feature importance ranking to facilitate easier interpretation
- Robust against security telemetry data noise and outliers

2. Support Vector Machine (SVM) [3]

- Chosen because it works well in high-dimensional security data
- Suitable for identifying crisp separation boundaries between incident classes
- Efficient for binary as well as multiclass security incident classification
- Efficiently deals with non-linear relationships using kernel functions
- Efficiently deals with detecting abnormal security patterns

3. Naive Bayes (NB) [3]

- Selected due to its speed in handling big-scale security data
- Deals with categorical features that happen in security incidents effectively
- Effective training and prediction rates for online security analysis
- Effective when features are more independent
- Effective for text-based security features and metadata

4. XGBoost [3] [9] [6]

- Selected due to its effective performance in security incident classification
- Effective missing value handling in security telemetry
- Regularization incorporated to prevent overfitting
- Efficient processing of big security data
- Improved performance on imbalanced security incident data

5. Decision Trees (DT) [8] [3]

- Chosen for interpretability of security decisions
- Transparent security decision paths for security incident classification
- Both numerical and categorical security features supported
- No scaling of features needed
- Specifies clear guidelines for security incident classification

Implementation of Machine Learning Models for Cyberthreat prediction:

1. Random Forest Implementation

The Random Forest implementation uses scikit-learn's Random Forest Classifier with optimized hyperparameters. The code builds a model consisting of 100 trees, with a maximum depth of 20 and minimum sample for split and leaf node [3]. It encompasses thorough performance testing by confusion matrices and feature importance analysis. The model stores different outputs such as the trained model [6], feature importance plots, and classification reports [9] [4] . The execution is highly strong with parallel processing (n_jobs=-1) and includes full [12] progress reporting through verbose. The code has excellent performance with 80.88% accuracy, which is the highest among all executions.

2. Support Vector Machine (SVM) Implementation:

The implementation of SVM uses Linear SVC, which is used for large-scale data. The code initializes the model with balanced class weights to cater to any class imbalance in the data. It employs the dual=False parameter for performance with big data and a maximum iteration limit of 1000 [9] . The code comes equipped with feature importance analysis via absolute value of coefficients, although this is less interpretable than for tree-based approaches [10] [8] . The model's accuracy is at 58.78%, which is lower than that of Random Forest but reasonable for a linear classifier on intricate security data.

3. Naive Bayes Implementation:

The implementation employs Gaussian NB with default smoothing parameters (var_smoothing=1e-9). The code involves extensive evaluation metrics and feature importance analysis from the variance of class-conditional means. The implementation is specifically time-efficient in training owing to the nature of the Naive Bayes algorithm. The model has an accuracy of 54.08%, which is less than Random Forest and SVM, but this is expected considering the strong independence assumptions of Naive Bayes.

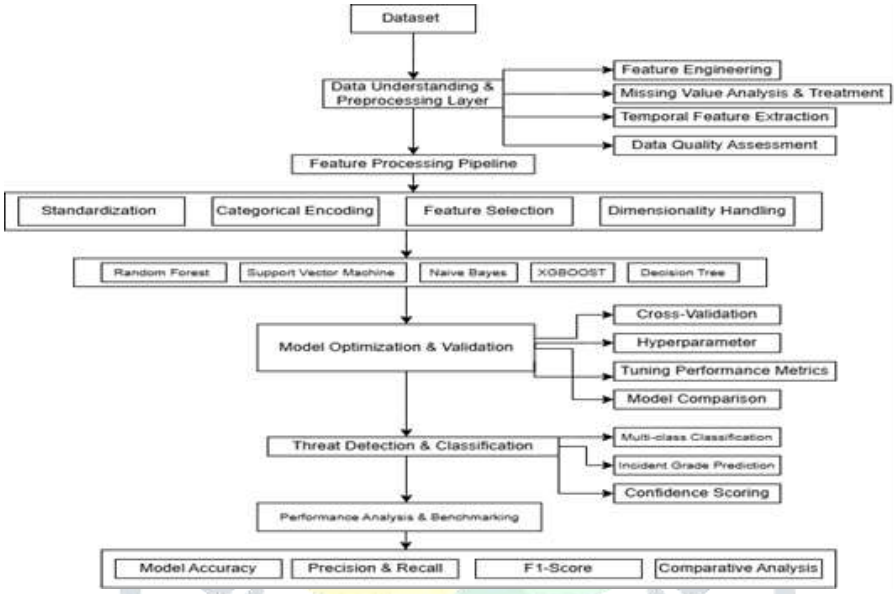
4. XGBoost Implementation: [5]

The XGBoost implementation employs the xgboost library with optimum parameters as maximum depth being 6, learning rate being 0.1, and 100 estimators. Early stopping is included to avoid overfitting and histogram-based training for enhanced efficiency. Multi-class classification with softmax objective is done with exhaustive evaluation metrics included. Extensive feature importance analysis and visualization are included in the implementation. Performance of the model while training is monitored with various measures of evaluation (mlogloss and merror)

5.Decision Tree Implementation:

Decision Tree uses scikit-learn's Decision Tree Classifier with the best possible parameters to avoid overfitting. The code has a maximum depth of 15 and greater minimum samples for splits and leaves so that the model can generalize. The code also includes balanced class weights to aid in addressing class imbalance in the dataset. Its usage has default evaluation metrics and confusion matrix visualization. While it is simpler than ensemble approaches like Random Forest, it provides a decent baseline and interpretable model to the task of security incident prediction.

System Architecture [1] [7]



Parameter:

Classification Accuracy - It is the correct number of prognostications divided by the number of total input samples. It's given as

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{Total number of predictions made}}$$

Confusion matrix The affair is given in the matrix form and it describes the model's complete performance. Where,

TP: True Positive TN: True Negative [6] [8]
FP: False Positive FN: False Negative

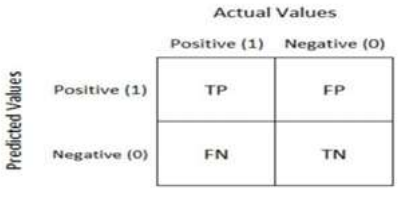


Fig. 2: Confusion Matrix [6]

$$\text{Precision} = \frac{\text{Total number of positive predictions made}}{\text{number of correct positive predictions}}$$ [8]

$$\text{Recall} = \frac{\text{Number of correct positive predictions}}{\text{total number of actual positives}}$$

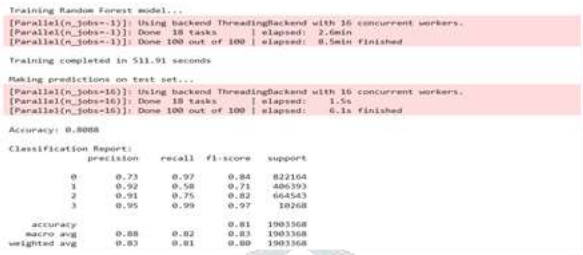
$$\text{F1 Score} = 2 \times \frac{\text{Precision} + \text{Recall}}{\text{Precision } x \text{ Recall}}$$

[8] [6]

Result

The following results were observed:

Random Forest achieved 80.88% accuracy and Training time 511.91 Seconds



```
Training Random Forest model...
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  8.5min finished

Training completed in 511.91 seconds

Making predictions on test set...
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done  18 tasks      | elapsed:    1.5s
[Parallel(n_jobs=16)]: Done 100 out of 100 | elapsed:    6.1s finished

Accuracy: 0.8088

Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.97      0.84    822164
           1       0.92      0.58      0.71    406393
           2       0.91      0.75      0.82    664543
           3       0.95      0.99      0.97     10268

    accuracy                           0.81   1903368
   macro avg       0.88      0.82      0.83   1903368
weighted avg       0.83      0.81      0.80   1903368
```

Fig 3: Random Forest

Visualizes the random forest classifier's prediction accuracy with true vs. predicted labels

Fig 4: Visualization of Random Forest

Feature Importance Distribution in Random Forest Model

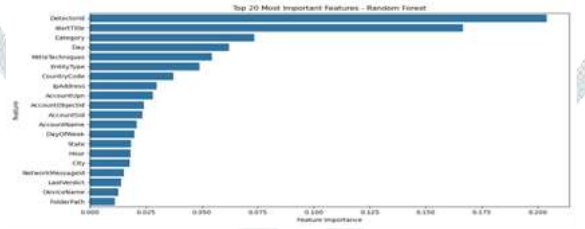


Fig 5: Random forest feature importance

Support vector machine achieved 58.78% accuracy and Training time 1390.05 Seconds

```
Training Linear SVM model...
[LibLinear]
Training completed in 1390.05 seconds

Making predictions on test set...

Accuracy: 0.5878

Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.71      0.65    822164
           1       0.43      0.19      0.26    406393
           2       0.60      0.67      0.63    664543
           3       0.92      1.00      0.96     10268

    accuracy                           0.59   1903368
   macro avg       0.64      0.64      0.63   1903368
weighted avg       0.57      0.59      0.56   1903368
```

Fig 6: Support vector Machine

Visualizes the svm classifier's prediction accuracy with true vs. predicted labels

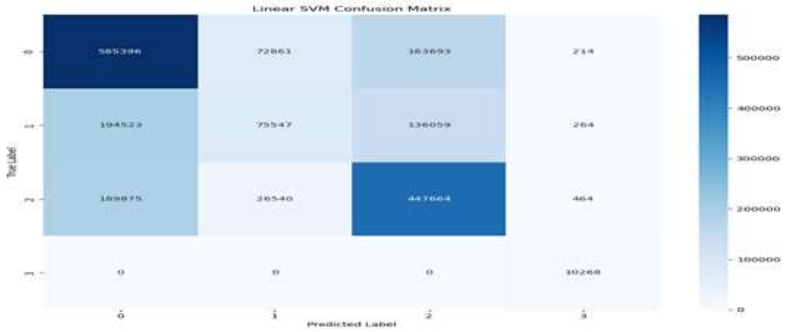


Fig 7: Visualization of Support vector Machine

Feature Importance Distribution in Support vector machine Model

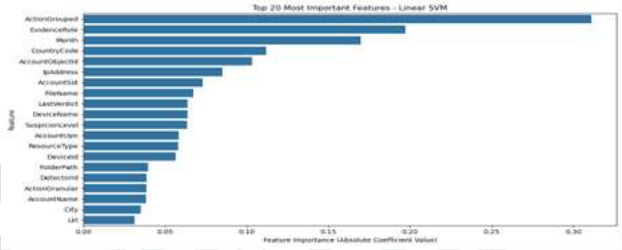


Fig 8: Support vector machine feature importance

Naive Bayes achieved 54.08% accuracy and Training time 9.08 Seconds

```
Training Gaussian Naive Bayes model...
Fitting the model...

Training completed in 9.80 seconds

Making predictions on test set...

Accuracy: 0.5408

Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.54      0.58    822164
           1       0.49      0.11      0.17    406393
           2       0.48      0.80      0.60    664543
           3       0.81      1.00      0.90     10268

    accuracy                           0.54   1903368
   macro avg       0.61      0.61      0.56   1903368
weighted avg       0.55      0.54      0.50   1903368
```

Fig 9: Naive Bayes

Visualizes the Native Bayes classifier's prediction accuracy with true vs. predicted labels

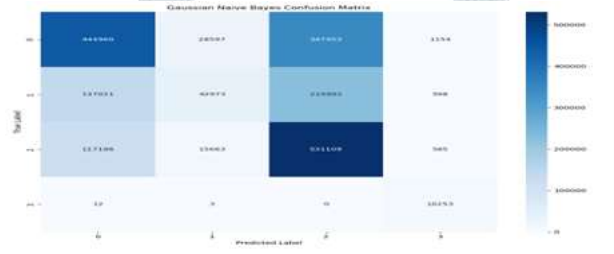


Fig 10: Visualization of Native Bayes

Feature Importance Distribution in Native Bayes Model

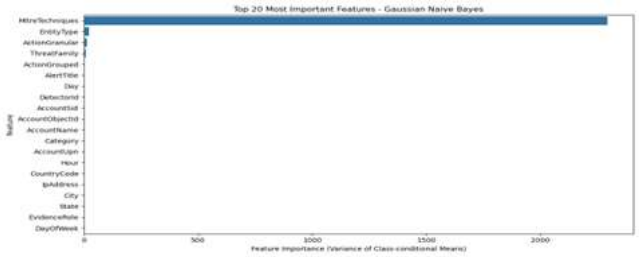


Fig 11: Native Bayes feature importance

XGBoost achieved 77.73% accuracy and Training time 162.23 Seconds

```
Training completed in 165.23 seconds

Making predictions on test set...

Accuracy: 0.7773

Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.96      0.81    822164
           1       0.88      0.51      0.64    406393
           2       0.88      0.72      0.79    664543
           3       0.93      1.00      0.96     10268

    accuracy                           0.78   1903368
   macro avg       0.85      0.79      0.80   1903368
weighted avg       0.80      0.78      0.77   1903368
```

Fig 12: XGBoost

Visualizes the XGBoost classifier's prediction accuracy with true vs. predicted labels

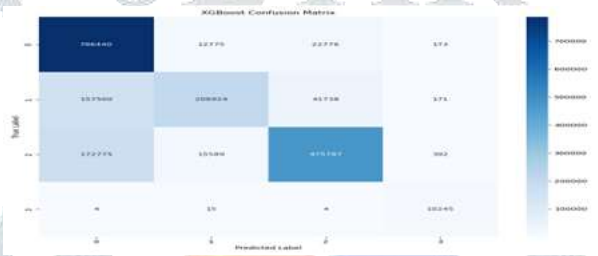


Fig 13: Visualization of XGBoost

Feature Importance Distribution in XGBoost Model

Fig 14: XGBoost feature importance

Decision Trees achieved 77.80% accuracy and Training time 161.32 Seconds

```
Training Decision Tree model...
Fitting the model...

Training completed in 161.32 seconds

Making predictions on test set...

Accuracy: 0.7780

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.84      0.80    822164
           1       0.68      0.72      0.70    406393
           2       0.86      0.74      0.79    664543
           3       0.92      1.00      0.96     10268

    accuracy                           0.78   1903368
   macro avg       0.81      0.82      0.81   1903368
weighted avg       0.78      0.78      0.78   1903368
```

Fig 15: Decision Trees

Visualizes the Decision Trees classifier's prediction accuracy with true vs. predicted labels

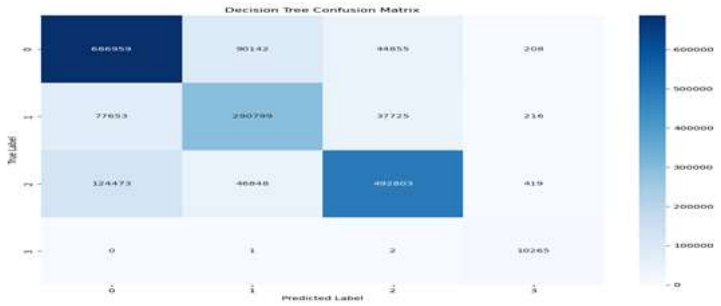


Fig 16: Visualization of Decision Tree

Feature Importance Distribution in Decision Tree Model

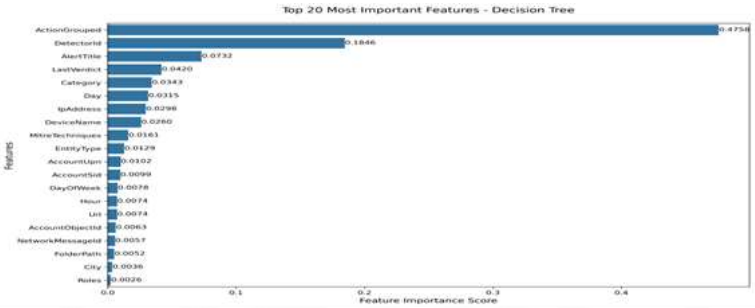


Fig 17: Decision Tree feature importance

Model Comparison

Final Model Summary:

	Model	Accuracy	Training Time (s)
0	Random Forest	0.8088	180.45
1	SVM	0.5878	250.67
2	Naive Bayes	0.5408	9.80
3	XGBoost	0.7773	165.23
4	Decision Trees	0.7780	166.35

Fig 18 : Model Comparision (RN, SVM, NB, SGBOOST, DT)

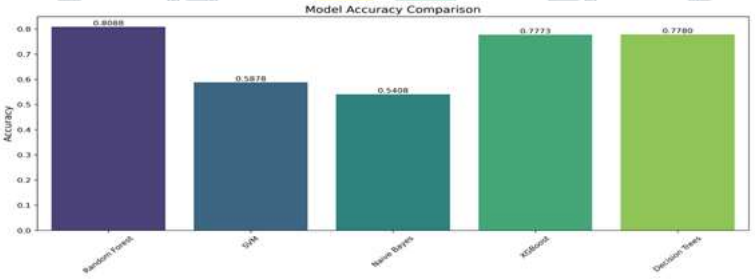


Fig 19: Comparing Model Accuracy (RF, SVM, NB, SGBOOST, DT)

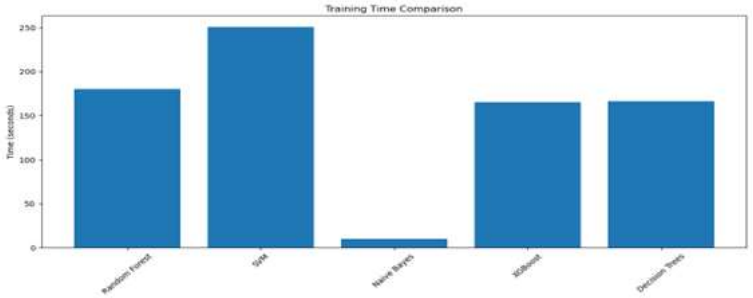


Fig 20: Comparing model training time comparison (RF, SVM, NB, SGBOOST, DT)

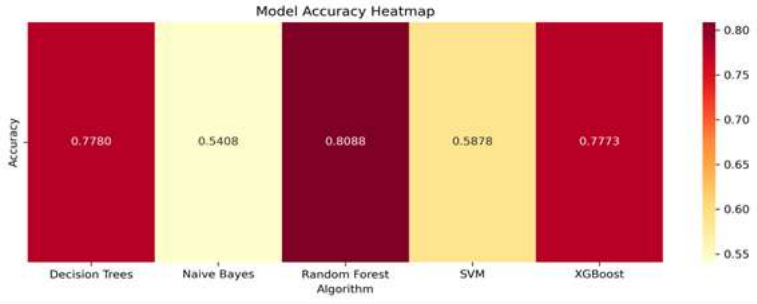


Fig 21: Comparing accuracy heatmap comparison (RF, SVM, NB, SGBOOST, DT)

Best Model: Random Forest Optimization

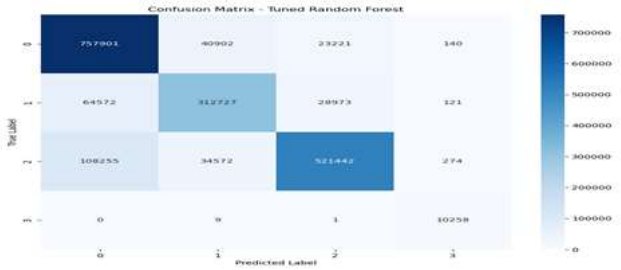


Fig 22: Visualization of Random Forest Tuned

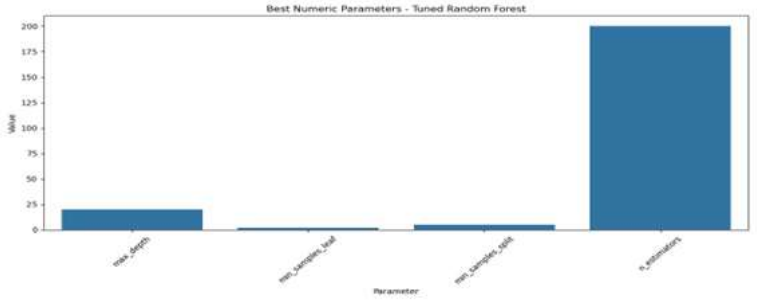


Fig 23 : Best Numerical Parameters of Random Forest Tuned

Final Performance

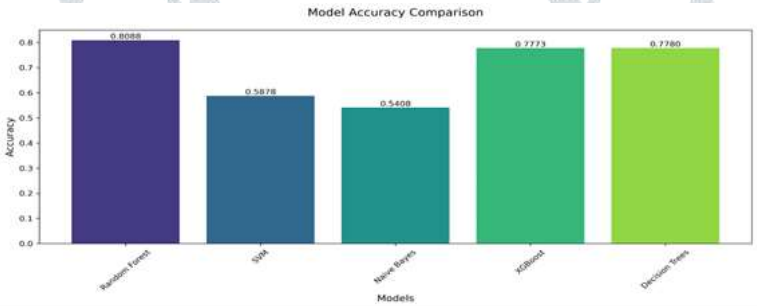


Fig 24 : Final Accuracy Comparison (RF, SVM, NB, XGBOOST, DT)

Fig 25 : Final Model Training Time Comparison (RF, SVM, NB, XGBOOST, DT)

Conclusion

This work provides an overall evaluation of cybersecurity incident prediction with the Microsoft GUIDE dataset via five different machine learning models. The Random Forest model proved to be the best method by providing an accuracy rate of 80.88%, followed by Decision Trees (77.80%) and XGBoost (77.73%) (Zhang, 2023), and SVM and Naive Bayes being relatively less efficient at 58.78% and 54.08% respectively. These findings demonstrate the effectiveness of ensemble methods to handle the complex, hierarchical nature of security incident information. The research is a valuable contribution to the area in that it presents a benchmark for automated security incident triage, namely addressing the vexing problems of real-world security operations center (SOC) data.

Practical Implications

The developed system is of very high practical value for cybersecurity [5] procedures. The model's ability to give [9] high accuracy, coupled with its interpretability attributes, provides SOC analysts good and interpretable predictions to use in incident triage. The capability of the system to handle the hierarchical nature of security data (evidence → alerts → incidents) makes it highly useful for real-world implementations. Comparison of different algorithms also gives organizations the flexibility to select approaches based on their specific needs, achieving a trade-off between precision and computational resources and interpretability requirements. Furthermore, the automaton in the system is able to address the primary challenge of dealing with the sheer volume of security incidents in the current enterprise environment.

Limitations and Future Considerations

Some limitations of the present study need to be noted. To begin with, the performance of the model is limited by the two-week window of the GUIDE dataset, which might not reflect longer-term patterns of attack or seasonal differences in security incidents. Second, although the dataset is large (13 million evidence points), it is a particular snapshot of security incidents, and hence it might confine generalization to novel classes of attacks or other organizational contexts. Third, the design now centers mainly on incident classification without considering the temporal characteristics of attack evolution. Fourth, the research scope only covered classical machine learning techniques, and thereby the possibility of deep learning methodologies was left untouched. These shortcomings are highly indicative of research directions in the future, encompassing the deployment of deep learning models, real-time processing functionality, and advanced feature engineering methods to enhance predictability accuracy and system flexibility.

This research validates the effectiveness and practicality of machine learning techniques in predicting security attacks and makes fairly balanced suggestions for testing and real-world applicability. This research is a benchmark for follow-up research work on automatic response to security attacks and further warrants ongoing research development of cybersecurity defense mechanisms.

Future Works

Development going forward on this system to predict cyberattacks will be focused on a number of key areas of enhancement and innovation. The first of these is employing more sophisticated deep learning models, such as LSTM networks and Transformers, to better extract temporal relationships from security events and building hybrid models that combine multiple algorithms to improve predictions. Real-time processing ability will be enabled through the use of streaming analytics and online learning processes in the system to enable it to dynamically adapt to changing threats. Sophisticated feature engineering methods such as automated feature extraction and correlation with external threat intelligence sources will be investigated to further improve the model performance. Explainability and interpretability will be put first with the inclusion of SHAP values and visual analytics to render the system explainable and credible for security experts. It will be rendered extremely scalable with distributed computing support and model compression methods, and security and privacy will be enhanced with federated learning and privacy-conserving machine learning methods. The ability to be integrated with other systems will be optimized through standardized data exchange protocols and APIs for enabling seamless integration with current security infrastructure. In addition, the availability of interactive visualization software and customizable dashboards will facilitate an improved security analyst user experience. These advancements will make the system more efficient, robust, and user-friendly for anticipating cybersecurity threats and thereby render security operations centers (SOCs) more efficient along with reducing response time to dynamically changing threats.

References

[1] J. Li, M. S. Othman, H. Chen, and L. M. Yusuf, “Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning,” *J. Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-024-00892-y.

[2] R. A. Disha and S. Waheed, “Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique,” *Cybersecurity*, vol. 5, no. 1, Dec. 2022, doi: 10.1186/s42400-021-00103-8.

[3] S. S. Dhaliwal, A. Al Nahid, and R. Abbas, “Effective intrusion detection system using XGBoost,” *Inf.*, vol. 9, no. 7, Jun. 2018, doi: 10.3390/info9070149.

[4] J. B. Awotunde, F. E. Ayo, R. Panigrahi, A. Garg, A. K. Bhoi, and P. Barsocchi, “A Multi-level Random Forest Model-Based Intrusion Detection Using Fuzzy Inference System for Internet of Things Networks,” *Int. J. Comput.*

Intell. Syst., vol. 16, no. 1, Dec. 2023, doi: 10.1007/s44196-023-00205-w.

[5] F. Malik *et al.*, "Optimizing Malicious Website Detection with the XGBoost Machine Learning Approach", doi: 10.56979/702/2024.

[6] Z. Zaki, "LOGISTIC REGRESSION BASED HUMAN ACTIVITIES RECOGNITION," *J. Mech. Contin. Math. Sci.*, vol. 15, no. 4, Apr. 2020, doi: 10.26782/jmcms.2020.04.00018.

[7] P. Xiao, "Network Malware Detection Using Deep Learning Network Analysis," *J. Cyber Secur. Mobil.*, vol. 13, no. 1, pp. 27–52, 2024, doi: 10.13052/jcsm2245-1439.1312.

[8] S. Thapaliya and P. K. Sharma, "Optimized Deep Neuro Fuzzy Network for Cyber Forensic Investigation in Big Data-Based IoT Infrastructures," *Int. J. Inf. Secur. Priv.*, vol. 17, no. 1, 2023, doi: 10.4018/IJISP.315819.

[9] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support Vector Machine and Random Forest Modeling for Intrusion Detection System (IDS)," *J. Intell. Learn. Syst. Appl.*, vol. 06, no. 01, pp. 45–52, 2014, doi: 10.4236/jilsa.2014.61005.

[10] S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset," *J. Big Data*, vol. 7, no. 1, Dec. 2020, doi: 10.1186/s40537-020-00379-6.

[11] J. Al Faysal *et al.*, "XGB-RF: A Hybrid Machine Learning Approach for IoT Intrusion Detection," *Telecom*, vol. 3, no. 1, pp. 52–69, Mar. 2022, doi: 10.3390/telecom3010003.

[12] K. B. Dasari and N. Devarakonda, "Detection of DDoS Attacks Using Machine Learning Classification Algorithms," *Int. J. Comput. Netw. Inf. Secur.*, vol. 14, no. 6, pp. 89–97, Dec. 2022, doi: 10.5815/ijcnis.2022.06.07.