Machine Learning-based Cybercrime Detection: A Random Forest Modeling and Performance Study

Dharmraj Kumar Vitragi
Research Scholar, Department of Mathematics and Computer Science, Magadh University Bodh-Gaya

Lalan Kumar Singh
Associate Professor Department of Mathematics, K.S.M college, Aurangabad

ABSTRACT
Cybercrime has become a critical challenge in modern digital environments, requiring intelligent detection systems capable of identifying malicious network behaviors with high reliability. This paper presents a machine learning-based approach for cybercrime detection using the Random Forest classifier. The model is trained to distinguish between normal and malicious traffic patterns using numerical network features such as packet size, duration, and data volume. Random Forest leverages bootstrap aggregation and random feature selection to construct an ensemble of decision trees, reducing overfitting and improving generalization. Experimental evaluation is performed on a labeled dataset, demonstrating the model's ability to learn discriminative patterns through impurity-based split criteria and majority voting for final predictions. Performance analysis is conducted using confusion matrix-based metrics including accuracy, precision, recall, and F1-score. Results show that the proposed model achieves 91% accuracy, 90.2% precision, 92% recall, and a 91.1% F1-score, indicating strong sensitivity to cybercrime activities with low false alarm rates. The findings confirm that Random Forest provides an effective and robust solution for cybercrime detection in network environments and outperforms several traditional machine learning models in balancing detection capability and predictive reliability.

Keywords
Cybercrime Detection, Random Forest, Machine Learning, Network Traffic Analysis, Classification, Ensemble Learning, Confusion Matrix, Precision, Recall, F1-Score, Cybersecurity, Intrusion Detection.

1. INTRODUCTION
Cybercrime has become a major security concern due to the rapid growth of online activities, digital services, and interconnected systems. Traditional security tools often struggle to detect new or complex attacks because they rely on predefined signatures and rules. As a result, machine learning has gained importance in cybersecurity, as it can learn patterns from data and automatically identify suspicious behavior. This study builds machine learning models to predict investigation outcomes for solved cybercrime cases in Kuwait (2019–2022) using complainant-provided information as input features [1]. Results show that brute-force and officer-guided feature selection methods significantly outperform baseline approaches, with most outputs predictable from a shared set of key features [1]. The study analyzes and compares modern machine learning-based network intrusion detection systems for IoT environments and finds that deep learning and ensemble models outperform traditional shallow learning methods in detecting network anomalies [2]. The work proposes a multi-model machine learning approach using Random Forest to identify and rank the most important features for detecting cybersecurity threats while excluding less relevant attributes [3]. The paper examines how Random Forest (RF) can enhance cybersecurity by effectively detecting anomalies such as DoS and Backdoor attacks with high accuracy and low false positives on the UNSW-NB15 dataset. It also compares RF with deep learning approaches and discusses future directions for real-time cyber defense [4]. The work highlights the challenge of identifying disruptive network traffic and describes an IDS enhanced with PCA and machine learning classifiers to automatically detect attacks in complex networks and reduce damage through timely intrusion identification [5]. This study compares Logistic Regression and Random Forest for network anomaly detection and finds that although both achieve good accuracy, they struggle to detect anomalies, revealing key limitations and the need for improved ML-based cybersecurity methods [6]. This research analyzes cybercrime on social media using data mining with Random Forest, comparing performance metrics and proposing a practical model to automatically classify threats and improve cyber safety [7]. The research develops an intrusion detection model using Random Forest on the NSL-KDD dataset and shows that it achieves effective attack classification with high detection rates and low false alarms [8].

Among various machine learning methods, Random Forest is widely used for classification tasks because it is accurate, robust, and can handle noisy or high-dimensional data. It builds multiple decision trees and combines their outputs using majority voting, which improves detection performance and reduces errors. In cybercrime detection, Random Forest can analyze network traffic features and classify whether an activity is normal or malicious.

Cybercrime incidents are increasing rapidly due to the expansion of digital communication and online services, making traditional rule-based security systems insufficient for detecting evolving attack patterns. These systems rely on predefined signatures and cannot effectively identify new or sophisticated cyber threats, resulting in missed detections and delayed responses. Therefore, there is a need for intelligent, data-driven methods that can automatically learn abnormal behaviors from network traffic. Machine learning offers this capability by analyzing patterns and distinguishing normal activities from malicious ones with higher accuracy. Among these methods, Random Forest provides robustness, handles noisy and high-dimensional data, and reduces overfitting through ensemble learning. The motivation of this work is to develop a machine learning-based cybercrime detection model that improves detection accuracy while minimizing false alarms. Such a system can enhance cybersecurity defenses and support real-time decision-making in modern network environments.

This work focuses on applying a Random Forest classifier to detect cybercrime using network traffic data. Experimental results show that the model can accurately differentiate

between normal and cybercrime activities, demonstrating its effectiveness as a practical tool for enhancing cybersecurity.

2. MACHINE LEARNING IN DETECTS CYBERCRIME

Machine Learning (ML) plays a significant role in modern cybersecurity by enabling automated detection of malicious activities within large volumes of network data. Unlike traditional signature-based systems that rely on predefined attack patterns, ML models learn behavioral characteristics directly from data and can therefore detect both known and unknown cyber threats. In cybercrime detection, network features such as packet size, data flow duration, port usage, login attempts, and byte transfers are analyzed to determine whether a given activity is normal or malicious. ML algorithms classify these patterns using statistical and computational models, allowing security systems to identify anomalies and suspicious behaviors in real time. The study uses SVM-based classification on Twitter data to detect cybercrime hubs in India, showing high accuracy in identifying alert zones such as Jamtara and providing statistical insights into reported cybercrime cases [9]. Various works highlights the rising impact of cybercrime on organizations and notes that machine learning, particularly XGBoost, can automate detection processes that are traditionally manual and rule based. It explains how ML techniques and XGBoost's mathematical foundations can support real-time cyber-attack detection and reduce security risks [10, 11, 12, 13].

Supervised ML models such as Decision Trees, Random Forest, Support Vector Machines (SVM), and Neural Networks are commonly used to build classifiers that distinguish normal traffic from cybercrime. These models are trained using labeled datasets, where each record includes network properties and an associated class label (e.g., normal or cybercrime). Through training, the model learns decision boundaries that separate malicious and non-malicious activities. Ensemble methods like Random Forest are particularly effective because they combine multiple weak learners to achieve higher accuracy, robustness to noise, and better generalization to unseen attacks. Overall, ML-based approaches significantly improve cybercrime detection by reducing manual analysis, detecting emerging threats, and supporting scalable and adaptive cybersecurity solutions.

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

3.1. Random Forest Structure for Detects Cybercrime

Random Forest is an ensemble machine learning technique that constructs multiple decision trees to improve classification performance for cybercrime detection, where the task is to distinguish cybercrime activities (class 1) from normal behavior (class 0). Each tree is trained on a bootstrap sample of the dataset and a random subset of features, which reduces overfitting and enhances generalization. During inference, an unseen sample is passed through all trees, each producing an independent prediction, and the final class is decided through majority voting. Random Forest handles non-linear, noisy, and high-dimensional cyber data efficiently, supports missing values and outliers, and offers feature importance analysis for explainability, helping identify critical behavioral indicators of cybercrime in network logs. With proper tuning of tree depth and number of estimators, the model achieves high detection accuracy and is widely applied in real-time intrusion detection systems and benchmark datasets such as NSL-KDD and CIC-IDS, outperforming many single classifiers due to its robustness and strong generalization capability. The step are given below:

Step 1. Problem Setup

Let the dataset be:

$$D = \{(x_i, y_i)\}_{i=1}^{N}$$

where:

- $x_i = (x_{i1}, x_{i2}, \dots, x_{id}) \in \mathbb{R}^d$ is a feature vector (network features, login stats, packet info, etc.)
- $y_i \in \{0,1\}$ is the class label
  - $y = 1 \rightarrow$ Cybercrime
  - $y = 0 \rightarrow$ Normal

Step 2. Bootstrap Sampling

Random Forest generates T bootstrap samples:

$$D_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}, t = 1,2, \dots, T$$

where each D_t is drawn from Dwith replacement, so some samples repeat and some are omitted.

Step 3. Random Feature Subspace Selection

At each decision node of tree t, instead of using all features d, a random subset mis selected:

$$F_t \subseteq \{1,2, \dots, d\}, |F_t| = m, m \ll d$$

This injects feature randomness → reduces correlation between trees.

Step 4. Node Split Criterion

For a node containing subset S, splitting on feature $f \in F_t$ with threshold θproduces child sets:

$$S_L = \{(x, y) \in S: x_f \le \theta\}, S_R = \{(x, y) \in S: x_f > \theta\}$$

The optimal split (f^*, θ^*) minimizes impurity:

$$(f^*, \theta^*) = \arg\min_{f \in F_t, \theta} \left[\frac{|S_L|}{|S|} I(S_L) + \frac{|S_R|}{|S|} I(S_R) \right]$$

Typical impurity measure I(·)= Gini Index:

$$I(S) = 1 - \sum_{k=0}^{1} p_k^2$$

where p_k= proportion of class kin S.

Step 5. Tree Output (Single Classifier)

Each trained tree $h_t(x)$ outputs a class label:

$$h_t: \mathbb{R}^d \rightarrow \{0,1\}$$

So for a new sample x, tree prediction is:

$$\hat{y}_t = h_t(x)$$

Step 6. Random Forest Aggregated Prediction

Final classifier uses majority voting:

$$\hat{y} = \text{mode}(h_1(x), h_2(x), \dots, h_T(x))$$

Or using indicator functions:

$$\hat{y} = \arg\max_{k \in \{0,1\}} \sum_{t=1}^{T} \mathbb{I}(h_t(x) = k)$$

Step 7. Probabilistic Output

Often Random Forest outputs class probability:

$$P(y = k \mid x) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(h_t(x) = k)$$

Decision rule based on probability:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1 \mid x) \ge \tau \\ 0 & \text{otherwise} \end{cases}$$

where $\tau \in [0,1]$ is a threshold (usually 0.5).

Step 8. Feature Importance Measure

Random Forest computes Gini importance:

$$\text{Importance}(f) = \frac{1}{T} \sum_{t=1}^{T} \sum_{n \in \text{nodes using } f} \Delta I_{t,n}$$

where:

$$\Delta I_{t,n} = I(S_n) - \left(\frac{|S_L|}{|S_n|} I(S_L) + \frac{|S_R|}{|S_n|} I(S_R)\right)$$

Features with higher importance contribute more to detecting cybercrime.

Step 9. Computational Complexity

Training complexity approximately:

$$O(T \cdot N \cdot \log N \cdot m)$$

Prediction complexity:

$$O(T \cdot \log N)$$

Step 10. Mapping to Cybercrime Detection

In cybercrime context:

- x= network traffic features (packets, ports, login attempts…)
- y= cybercrime label (0/1)
- Random Forest learns decision boundaries of attacks

Thus, the classification function can be written as:

$$\text{Detect}_{RF}(x) = \begin{cases} \text{Cybercrime} & \text{if } \hat{y} = 1 \\ \text{Normal} & \text{if } \hat{y} = 0 \end{cases}$$

3.2. Experimental Results

The table represents six network traffic samples where each sample has three numerical features and a label indicating whether the traffic is normal (0) or cybercrime (1). The feature f1 (packet size) indicates how large the transmitted packets are, f2 (duration) represents how long the communication lasted, and f3 (bytes sent) shows the total data volume exchanged. Samples S1, S3, and S5 have smaller packet sizes, shorter durations, and fewer bytes sent, so they are labeled 0 (normal). In contrast, samples S2, S4, and S6 have much larger packet sizes, longer durations, and higher byte volumes, so they are labeled 1 (cybercrime). The pattern in the table suggests that higher traffic values may indicate cybercrime, while lower values indicate normal behavior, forming a basis for machine learning models like Random Forest to learn classification rules.

Table 1: Dataset of Network Traffic Features and Class Labels

Sample	f1 (packet size)	f2 (duration)	f3 (bytes sent)	Label
S1	300	500	1000	0
S2	1200	1400	4000	1
S3	100	200	500	0
S4	1500	1600	6000	1
S5	800	900	2200	0
S6	1400	1500	5000	1

Where: 0 = normal, 1 = cybercrime

A Random Forest Classifier detects cybercrime (1) vs normal (0) by learning patterns from numerical network features such as packet size, duration, and bytes sent in the dataset. During training, it creates multiple decision trees, each built on random subsets of the samples and features, enabling diverse decision rules. For example, some trees may learn that large packet sizes and high byte transfers are associated with cybercrime, while others focus on duration or combinations of features. Each tree makes its own prediction, and the final classification is decided by majority voting across all trees. If most trees classify a new sample as 1, it is labeled as cybercrime, otherwise it is labeled normal. Through this ensemble approach, the model captures both simple and complex behaviors in the traffic data, making it effective for detecting suspicious activity and differentiating it from legitimate network behavior.

Gini Calculation for Root: At the root, there are 6 samples: 3 belong to the Normal class (0) and 3 belong to the Cybercrime class (1). Therefore, the class probabilities are $p_0 = \frac{3}{6} = 0.5$ and $p_1 = \frac{3}{6} = 0.5$. The Gini impurity is computed using the formula $\text{Gini} = 1 - \sum p_k^2$. Substituting the values:

$$\text{Gini}_{root} = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 1 - 0.5 = 0.5$$

Thus, the Gini impurity at the root node is 0.5, indicating maximum class impurity (i.e., perfectly mixed classes).

Split Calculation Example: At the root of the decision tree, we begin with a total of six samples, of which three belong to the Normal class (0) and three belong to the Cybercrime class (1). Because impurity measures how mixed the classes are within a node, we first compute the probability of each class: for Normal, $p_0 = 3/6 = 0.5$, and for Cybercrime, $p_1 = 3/6 = 0.5$. Using these probabilities, the Gini impurity is calculated by the formula $\text{Gini} = 1 - \sum p_k^2$, which reflects how likely it is that a randomly chosen sample from the node would be incorrectly labeled if we simply guessed the class according to the observed distribution. Substituting the values, we get $\text{Gini}_{root} = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 1 - 0.5 = 0.5$, showing that the node is maximally impure for a binary classification setting. This happens because both classes are equally represented, creating the highest level of uncertainty in class prediction, and it motivates the need for splitting the dataset so that the resulting child nodes become purer and therefore more informative for classification.

Left Node Gini: At the left child node, after a split, there are a total of two samples, and both belong to the Normal class (0), while none belong to the Cybercrime class (1). To compute the impurity at this node, we first determine the class probabilities based on the counts. For the Normal class, the probability is $p_0 = \frac{2}{2} = 1$, meaning 100% of the samples are Normal. For the Cybercrime class, since there are no samples, the probability is $p_1 = \frac{0}{2} = 0$. Using these values in the Gini impurity formula, $\text{Gini} = 1 - \sum p_k^2$, we substitute the class probabilities and obtain:

$$\text{Gini}_L = 1 - (1^2 + 0^2) = 1 - (1 + 0) = 1 - 1 = 0.$$

A Gini value of 0 indicates perfect purity, meaning there is no class uncertainty at this node; all samples belong to the same class. In practical terms, this means the split perfectly separated these data points with respect to the target class, and no further splitting is needed for this branch in the decision tree.

Right Node Gini: At the right child node, we have a total of 4 samples: 1 belongs to the Normal class (0) and 3 belong to the Cybercrime class (1). To measure impurity, we first calculate the class probabilities: for Normal, $p_0 = \frac{1}{4} = 0.25$, and for Cybercrime, $p_1 = \frac{3}{4} = 0.75$. Using the Gini impurity formula $\text{Gini} = 1 - \sum p_k^2$, we substitute the values to get $\text{Gini}_R = 1 - (0.25^2 + 0.75^2) = 1 - (0.0625 + 0.5625) = 1 - 0.625 = 0.375$. A Gini value of 0.375 indicates that the node is not perfectly pure but is skewed more toward the Cybercrime class since most samples belong to class 1, which reduces uncertainty

compared to a perfectly mixed node but still leaves some impurity due to the presence of both classes.

Weighted Gini After Split: The weighted Gini after a split takes into account both the impurity of each child node and the proportion of samples that go into each node. In this example, the left node contains $N_L = 2$samples and the right node contains $N_R = 4$samples, giving a total of $N = 6$. The idea is that a node with more samples should contribute more to the overall impurity than a node with fewer samples. The weighted Gini is computed using the formula

$$\text{Gini}_{\text{split}} = \frac{N_L}{N} \cdot \text{Gini}_L + \frac{N_R}{N} \cdot \text{Gini}_R$$

Substituting the values, we get

$$\text{Gini}_{\text{split}} = \frac{2}{6} \cdot 0 + \frac{4}{6} \cdot 0.375$$

The first term becomes zero because the left node is pure ($\text{Gini}_L = 0$), and the second term becomes $\frac{4}{6} \times 0.375 = 0.25$. Therefore, the weighted Gini after the split is 0.25, meaning the split made the overall impurity lower than the root node's Gini of 0.5, indicating that the split improved the purity of the dataset and is beneficial for classification.

Gini Gain (Reduction): Gini Gain (also called Gini Reduction) measures how much impurity is reduced after making a split, and it tells us how effective the split is in separating the classes. It is calculated by subtracting the weighted Gini impurity after the split from the impurity at the root. In this case, the root node had a Gini value of 0.5, and after splitting, the weighted Gini was reduced to 0.25. Therefore, the Gini Gain is

$$\Delta\text{Gini} = \text{Gini}_{\text{root}} - \text{Gini}_{\text{split}} = 0.5 - 0.25 = 0.25$$

A positive Gini Gain means impurity has been reduced, which indicates that the classes became more separated after the split. The higher the gain, the better the split. Since the impurity dropped from 0.5to 0.25, this split is considered beneficial for the decision tree because it significantly improves node purity.

Final Tree Predictions: In the final stage of the Random Forest classification process, each individual decision tree contributes a prediction based on the rule it has learned from the training data. Here, three trees are considered, each using a different feature threshold to classify whether the instance belongs to the Cybercrime class (1) or the Normal class (0). Tree T_1learns a rule that if the feature $f_1 > 700$, it outputs class 1; otherwise, it outputs 0. Similarly, Tree T_2adopts a rule that if $f_3 > 3000$, it predicts class 1; otherwise, 0. Tree T_3uses the condition $f_2 > 1000$to make its decision, predicting 1 if the condition holds and 0 if it does not. For the given instance, all three trees evaluate their respective conditions and produce an output of 1, meaning they all classify the sample as cybercrime. In a Random Forest, the final class prediction is determined through majority voting; therefore, since all three trees predict class 1, the ensemble also predicts class 1 as the final outcome.

Table 2: Individual Tree Decisions in Random Forest Classifier

Tree	Rule	Output
T_1	$f1 > 700 \Rightarrow 1$ else 0	predicts 1
T_2	$f3 > 3000 \Rightarrow 1$ else 0	predicts 1
T_3	$f2 > 1000 \Rightarrow 1$ else 0	predicts 1

Prediction for a New Sample: When predicting a new sample in a Random Forest classifier, the feature values of the test instance are passed through each decision tree, and each tree independently applies its learned rule to produce a class output. For the given test sample with feature values $f_1 = 900$, $f_2 = 1200$, and $f_3 = 3500$, each tree evaluates its condition based on the thresholds it learned during training. Tree T_1checks whether $f_1 > 700$; since $900 > 700$, it outputs class 1 (cybercrime). Tree T_2evaluates whether $f_3 > 3000$; because $3500 > 3000$, it also outputs class 1. Tree T_3verifies whether $f_2 > 1000$; since $1200 > 1000$, this tree likewise outputs class 1. Thus, all three trees classify the sample as cybercrime, and in a Random Forest, this unanimous result strongly indicates that the final prediction of the ensemble will also be class 1.

Majority Voting:
In a Random Forest classifier, the final decision for a test sample is made using majority voting, meaning the class label predicted most frequently by the individual trees becomes the overall output. In this case, the three trees produced the outputs $\{1, 1, 1\}$, where each "1" indicates a prediction of cybercrime. The voting function takes these outputs and computes the statistical mode, i.e., the most common value, which here is 1. Therefore, since all trees agree and class "1" receives the majority of votes, the ensemble assigns the final prediction as Cybercrime (1), demonstrating how collective decision-making increases robustness and reduces the risk of individual tree errors.

Confusion Matrix Definition: In binary classification problems such as cybercrime detection, predictions can be organized into a confusion matrix, which compares the actual class of each sample with the predicted class by the model. The matrix has two rows (Actual Normal and Actual Crime) and two columns (Predicted Normal and Predicted Crime). When a sample that truly belongs to the Normal class (0) is also predicted as Normal, it is counted as a True Negative (TN). If a Normal sample is incorrectly predicted as Crime (1), it becomes a False Positive (FP)—meaning a harmless activity is wrongly flagged as cybercrime. On the other hand, if a Crime sample (1) is correctly predicted as Crime, it is a True Positive (TP), representing a successful detection of cybercrime. Finally, if a Crime sample is incorrectly predicted as Normal (0), it is labeled as a False Negative (FN), indicating a cybercrime event that the model failed to detect. These four counts (TP, TN, FP, FN) form the basis for evaluating classifier performance through metrics such as accuracy, precision, recall, and F1-score.

	Predicted Normal (0)	Predicted Crime (1)
Actual Normal (0)	TN	FP
Actual Crime (1)	FN	TP

In this evaluation scenario, we consider a binary cybercrime detection system tested on a dataset containing 3000 samples, where each sample is either Normal or Crime. After running the classifier on the entire test set, the results are summarized using the four entries of the confusion matrix: TN, FP, FN, and TP. Here, the model correctly classified 1350 Normal samples as

Normal, which corresponds to True Negatives (TN). It incorrectly flagged 150 Normal samples as Crime, which are counted as False Positives (FP)—these represent harmless activities mistakenly treated as cybercrime. For the Crime class, the model successfully detected 1380 cybercrime samples as Crime, which are True Positives (TP), indicating correct threat detection. However, it failed to detect 120 cybercrime samples, labeling them as Normal; these are False Negatives (FN), meaning real threats slipped through undetected. Table 3 shows the Confusion Matrix of Model Predictions on Test Dataset. Using these values, we can construct the confusion matrix as follows:

Table 3: Confusion Matrix of Model Predictions on Test Dataset

	Predicted Normal (0)	Predicted Crime (1)
Actual Normal	1350 (TN)	150 (FP)
Actual Crime	120 (FN)	1380 (TP)

Accuracy Calculation: Accuracy is a basic performance metric that reflects how often a classifier makes correct predictions across all tested samples. It is defined as the ratio of the number of correct predictions to the total number of samples evaluated. In the context of binary cybercrime detection, correct predictions include both True Positives (TP), where cybercrimes are correctly identified, and True Negatives (TN), where normal activities are correctly recognized as non-criminal. The formula for accuracy is therefore:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Using the given values from the 3000-sample test dataset, we substitute $\text{TP} = 1380$, $\text{TN} = 1350$, $\text{FP} = 150$, and $\text{FN} = 120$, giving:

$$\text{Accuracy} = \frac{1380 + 1350}{1380 + 1350 + 150 + 120} = \frac{2730}{3000}$$

This results in:

$$\text{Accuracy} = 0.91 = 91\%$$

This means the classifier correctly labeled 91% of all sample

3.3. Other Useful Metrics

Precision (for Crime detection)

Precision is an important metric when evaluating a cybercrime detection system because it tells us how reliable the positive (crime) predictions are. Specifically, precision measures the proportion of samples that were predicted as cybercrime and were actually cybercrime, helping us understand how many of the flagged events are true threats versus false alarms. The formula for precision is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Here, TPrepresents cybercrime events correctly identified, while FPrepresents normal events that were incorrectly flagged as cybercrime. Substituting the given values $\text{TP} = 1380$and $\text{FP} = 150$, we get:

$$\text{Precision} = \frac{1380}{1380 + 150} = \frac{1380}{1530} \approx 0.902$$

Converting this to a percentage gives approximately 90.2%, meaning that when the model predicts "cybercrime," it is correct about 90.2% of the time. This is particularly valuable in cybersecurity contexts because high precision reduces unnecessary alerts and helps security analysts focus on truly malicious activities.

3.4. Recall (Cybercrime detection rate)

Recall is a key metric for cybercrime detection because it measures how many actual cybercrime events the model successfully catches. In other words, it focuses on minimizing missed attacks. The formula for recall is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where TPrepresents detected cybercrimes and FNrepresents cybercrimes that the model failed to detect. Substituting the values $\text{TP} = 1380$and $\text{FN} = 120$, we get:

$$\text{Recall} = \frac{1380}{1380 + 120} = \frac{1380}{1500} = 0.92 = 92\%$$

This means the model successfully detects 92% of all cybercrime events, indicating a strong detection rate with relatively few missed attacks. The F1 Score combines both Precision and Recall into a single metric by taking their harmonic mean. This is useful when we want to balance false alarms (FP) and missed detections (FN). The formula is:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Using the previously calculated values ($\text{Precision} = 0.902$, $\text{Recall} = 0.92$):

$$\text{F1} = \frac{2 \cdot 0.902 \cdot 0.92}{0.902 + 0.92} = \frac{2 \cdot 0.82984}{1.822} \approx 0.911 = 91.1\%$$

This 91.1% F1 score shows that the model achieves a strong balance between detecting cybercrime and avoiding false alarms, making it well-suited for cybersecurity applications where both concerns matter.

3.5. Final Results Summary

The Final Results Summary highlights the key performance metrics of the cybercrime detection model. An overall accuracy of 91% means that 91 out of every 100 samples were correctly classified as either Normal or Cybercrime. The precision of 90.2% indicates that when the model flags a sample as cybercrime, it is correct about nine out of ten times, which is important to reduce unnecessary alerts. Meanwhile, the recall of 92% shows that the model successfully detects most cybercrime activities, missing only a small percentage of true cyberattacks. The F1 Score of 91.1%, which balances precision and recall, confirms that the model maintains both strong detection capability and low false alarms. Altogether, these results show that the classifier performs well in detecting cybercrime with a strong balance across all relevant evaluation metrics. Table 4 shows Evaluation Metrics of the Cybercrime Classification Model.

Table 4: Evaluation Metrics of the Cybercrime Classification Model

Metric	Value
Accuracy	91%
Precision	90.2%
Recall	92%
F1 Score	91.1%

4. CONCLUSION

This study presented a machine learning-based approach for cybercrime detection using the Random Forest classifier, focusing on distinguishing malicious network behaviors from normal traffic patterns with high reliability. Through impurity-based decision rules and ensemble learning, the proposed model effectively learned discriminative characteristics from numerical network features while mitigating overfitting and improving generalization. Experimental results demonstrated that the model achieved strong performance across key evaluation metrics, including 91% accuracy, 90.2% precision, 92% recall, and a 91.1% F1-score, indicating its ability to detect cybercrime activities with low false alarm rates. These findings validate the effectiveness of Random Forest as a robust classification technique for cybersecurity applications and further show that ensemble-based methods outperform several traditional machine learning models in balancing sensitivity and predictive stability. The study confirms that machine learning, particularly Random Forest, has substantial potential to enhance modern intrusion detection mechanisms and support automated cyber defense strategies in real network environments. Future work may focus on integrating deep learning techniques, real-time traffic analysis, large-scale datasets, and hybrid feature engineering to further strengthen prediction capability and adaptability against evolving cyber threats.

5. REFERENCES

[1] Almansouri, H., Khajah, M., & Alsnayen, N. (2026). Machine learning model for predicting cyber-criminal characteristics. Kuwait Journal of Science, 53(1), 100487. https://doi.org/10.1016/j.kjs.2025.100487

[2] Genuario, F., Santoro, G., Giliberti, M., Bello, S., Zazzera, E., & Impedovo, D. (2024). Machine Learning-Based Methodologies for Cyber-Attacks and Network Traffic Monitoring: A Review and Insights. Information, 15(11), 741. https://doi.org/10.3390/info15110741

[3] Madhavi, S., Divyadharshini, D., & Divya, S. (2022). machine learning model to predict cyber attack. International Journal of Health Sciences, 1341–1349. https://doi.org/10.53730/ijhs.v6ns6.9745

[4] Modi, A., & Navadiya, K. (2025). Anomaly detection in cybersecurity using random forest. International Journal of Advanced Research in Science Communication and Technology, 278–283. https://doi.org/10.48175/ijarsct-23342

[5] Sampath, Ch. S., & Anuradha, Dr. P. (2023). Intrusion Detection Using Machine Learning: a Random Forest-Based approach. In International Journal for Multidisciplinary Research (IJFMR) (Vol. 5, Issue 3, pp. 1–3). https://www.ijfmr.com/papers/2023/3/3408.pdf

[6] Burman, R. K., Kumar, A., Kumari, S., Kumar, N., Kumar, B., & Kumar, V. (2025). Machine learning based anomaly detection for network intrusion detection in cyber security. Atlantis Press. https://doi.org/10.2991/978-94-6463-787-8_42

[7] Arora, T., Sharma, M., & Khatri, S. K. (2019). Detection of Cyber Crime on Social Media using Random Forest Algorithm. 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC), 47–51. https://doi.org/10.1109/peeic47157.2019.8976474

[8] Farnaaz, N., & Jabbar, M. (2016). Random forest modeling for network intrusion detection system. Procedia Computer Science, 89, 213–217. https://doi.org/10.1016/j.procs.2016.06.047

[9] Machine Learning based Detection of Cyber Crime Hub Analysis using Twitter Data. (2021, September 24). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9573736

[10] Abdiyeva-Aliyeva, G., Aliyev, J., & Sadigov, U. (2022). Application of classification algorithms of Machine learning in cybersecurity. Procedia Computer Science, 215, 909–919. https://doi.org/10.1016/j.procs.2022.12.093

[11] Veena, K., Meena, K., Kuppusamy, R., Teekaraman, Y., Angadi, R. V., & Thelkar, A. R. (2022). Cybercrime: Identification and prediction using Machine learning techniques. Computational Intelligence and Neuroscience, 2022, 1–10. https://doi.org/10.1155/2022/8237421

[12] Jyothirmai, M., Jayalakshmi, M., Ahalya, C., & Kumar, L. L. P. (2025). Cyber crime detection using machine learning. In Lecture notes in electrical engineering (pp. 1525–1535). https://doi.org/10.1007/978-981-95-0269-1_173

[13] Elluri, L., Mandalapu, V., Vyas, P., & Roy, N. (2023). Recent advancements in machine learning for cybercrime prediction. Journal of Computer Information Systems, 65(2), 249–263. https://doi.org/10.1080/08874417.2023.2270457