International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v11i8.7951
***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*

Cyber Crime Detection and Prevention Techniques on Cyber Cased Objects Using SVM and Smote

R. Sudha Kishore[1], D. Lalitha Bhaskari[2]
[1]Research Scholar, Department of CS&SE, Andhra University College of Engineering (Autonomous), Andhra University. Visakhapatnam, Andhra Pradesh, India. Email: sdhkishore@gmail.com
[*]Professor, Department of CS&SE, Andhra University College of Engineering (Autonomous), Andhra University. Visakhapatnam, Andhra Pradesh, India. Email: lalithabhaskari@yahoo.co.in

Abstract—Conventional cybersecurity employs crime prevention mechanisms over distributed networks. This demands crime event management at the network level where Detection and Prevention of cybercrimes is a must. A new Framework IDSEM has been introduced in this paper to handle the contemporary heterogeneous objects in cloud environment. This may aid for deployment of analytical tools over the network. A supervised machine learning algorithm like SVM has been implemented to support IDSEM. A machine learning technique Like SMOTE has been implemented to handle imbalanced classification of the sample data. This approach addresses imbalanced datasets by oversampling the minority classes. This will help to solve Social Engineering Attacks (SEA) like Phishing and Vishing. Classification mechanisms like decision trees and probability functions are used in this context. The IDSEM framework could minimize traffic across the cloud network and detect cybercrimes maximally. When results were compared with existing approaches, the results were found to be good, leading to the development of a unique SMOTE algorithm.

Keywords-Intrusion Detection; Security Event Management (IDSEM); Synthetic Minority Oversampling Technique (SMOTE); Support Vector Machine (SVM).

I. INTRODUCTION

Frequently the cybercrimes will be with regard to social and economic issues. But the users may not be aware of this due to lack of experience in detecting them. Cybercrimes may happen very much frequently in distributed environments like cloud systems. Under the cloud environment, there is every possibility that the data across the networks may be stolen without any credentials being fulfilled. This may happen due to active or passive attacks by the attackers watching the networks continuously. These attackers will act as normal users and pretend to be genuine users. The users will execute malicious activates across the networks. They are much aware of the aspect that the large amount of data will not be stored securely due to its scalability and the issues of scalability. Because of this Detection and Prevention capabilities of these attackers over the network is a cumbersome task for the network administrators. It will be more troublesome for cyber cased object considered the fact that the entire details of the object will be known to the attackers with a single transaction on the network. The attackers will be changing the form of stealing the data over the network. This may be related to malicious attacks like vishing and phishing. The vishing and phishing will reduce the performance of the system through their versatile forms. This may lead cybercrimes over the networks where the data with respect to cyber cased object may be stolen effectively by these attackers. It demands the network users to implement certain mechanisms where data related to cyber cased object may not be stolen in any regard.

In recent years, India has witnessed a significant increase in the usage of digital devices, including smartphones, laptops, tablets, and IoT (Internet of Things) devices. This surge in digital adoption has been fueled by factors such as affordable internet connectivity, government initiatives, and the rapid expansion of e-commerce and digital services. However, along with the numerous benefits, the surge in digital device usage has also exposed individuals and organizations to various cybersecurity threats. *The Digital Divide:* Despite the remarkable growth in digital device adoption, India still faces a significant digital divide. While urban areas enjoy relatively better connectivity and access to digital resources, rural areas and marginalized communities often lack reliable internet infrastructure and digital literacy. This digital divide creates disparities in terms of cybersecurity awareness and preparedness, making certain segments of the population more vulnerable to cyber threats.

- *Mobile Device Proliferation:* The proliferation of smartphones in India has been remarkable, with millions of users accessing the internet primarily through mobile devices. However, the rapid adoption of mobile devices brings unique security challenges. Issues such as malware-infected applications, phishing attacks, and insecure Wi-Fi networks pose significant risks to users' personal data and financial security.

 Risks Associated with Online Services: The increasing reliance on online services for banking, e-commerce, healthcare, and education has exposed individuals and organizations to various cyber threats. Data breaches, identity theft, ransomware attacks, and social engineering are some of the risks associated with the use of online services. Additionally, the lack of robust cybersecurity practices among service providers further amplifies the vulnerabilities.

 Cyber Threats to Critical Infrastructure: India's expanding digital infrastructure also encompasses critical sectors such as power grids, transportation, healthcare, and government systems. These sectors face sophisticated cyber threats, including advanced persistent threats (APTs), distributed denial-of-service (DDoS) attacks, and ransomware campaigns. A successful attack on critical infrastructure could have severe consequences for the nation's security and economic stability.

 Mitigation Strategies: To address the cybersecurity challenges arising from the increased usage of digital devices, it is crucial to implement comprehensive mitigation strategies. These strategies include enhancing cybersecurity awareness through public education campaigns, promoting digital literacy, strengthening regulations and standards for data protection, encouraging collaboration between government and industry stakeholders, and investing in cybersecurity research and development.

As India experiences a surge in the usage of digital devices, it is imperative to recognize the associated cybersecurity risks and implement proactive measures to mitigate them. By addressing the digital divide, promoting cybersecurity awareness, and adopting robust security practices, India can safeguard its citizens, organizations, and critical infrastructure from the evolving cyber threats in the digital era.

Certain mechanisms are implemented in cloud system in order to Detect and Prevent the cybercrimes over the network on the cyber cased objects. Few intrusion detection systems (IDS) are incorporated over the cloud networks, but they could able to focus on Detection and Prevention of the cybercrimes to a minimal extent. If huge amount of traffic exists across the network the IDS will fail to detect the cybercrimes over the network. This is because they cannot differentiate boundary values clearly between one cyber cased object to another. This really demands the users to incorporate certain capabilities in the networks where cybercrimes may be handled effectively. In this regard machine learning algorithms with a better classification of differentiation will help the network users to detect and prevent the cybercrimes effectively. In a supervised machine learning where classification and labelling of cyber cased objects will be clearly being notified to improve the detection capabilities of malicious attackers over the network. A learning algorithm like a Support Vector Machine (SVM) along with aspect Intrusion Detection with security event management (IDSEM) has been proposed as a framework to improve the capabilities of Detection and Prevention of cybercrimes over the network. To address the SEA stated above, the SMOTE PROCESS will equalize and organize the data. SMOTE works by selecting samples in the feature space that are near together and drawing a line among them. A random sample of data from the minority class is chosen first. Then, for that example, k of the closest neighbors is found (usually k=5). This will be implicated through the IDSEM framework to enhance the performance of the system.

The objective of this study is to propose a framework to improve the capabilities of Detection and Prevention of cybercrimes over the network in distributed environments like cloud systems.

The motivation for this study is the frequent occurrence of cybercrimes in cloud systems and the lack of experience among users in detecting them. The study aims to address the issue of data being stolen from cloud systems due to active or passive attacks by attackers pretending to be genuine users.

The limitations of this study include the difficulty in detecting and preventing cybercrimes in cloud systems due to scalability issues, and the ability of attackers to change their methods of stealing data over the network. The study also notes that current intrusion detection systems (IDS) incorporated in cloud networks are not sufficient in detecting and preventing cybercrimes. The proposed framework attempts to address these limitations by implementing machine learning algorithms and a Support Vector Machine (SVM) along with aspect Intrusion Detection with security event management (IDSEM) to enhance the performance of the system.

II. SOCIAL ENGINEERING ATTACKS

SEA: Currently, social engineering assaults pose the greatest threat to cybersecurity. They can be detected, but not stopped, according to [1]. To gain access to confidential information, social engineers' prey on unsuspecting individuals. This information is then utilized for illicit purposes or sold to third parties on the black market or on the dark web. Attackers now leverage Big Data to profit from firms' valuable data because of its availability [2]. Massive amounts of data are packaged and sold in bulk on today's markets as goods [3]. Despite the fact that SEA varies from one another, they all follow a similar pattern with similar phases. As a general rule, attackers typically follow a four-phase process: gathering intelligence about their target, building trust, executing the attack and finally disappearing [4]. An attack on a person's social media accounts is depicted in Figure.

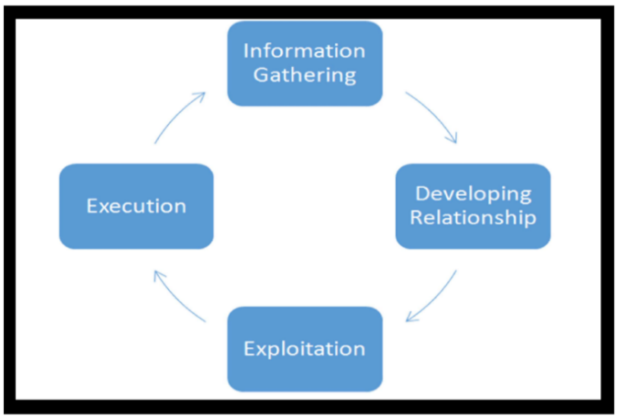


Fig 1: Stages of SEA

It's during this phase of the attack known as information gathering that the attacker chooses a target. During the hook phase, the attacker begins to acquire the victim's trust by communicating with them directly or via email. Victims are emotionally coerced into divulging sensitive information or making security blunders during the paly phase of the attack. The attacker leaves no evidence in the out phase [5]. Human-based and computer-based social engineering attacks are the two forms of social engineering attacks. [6].

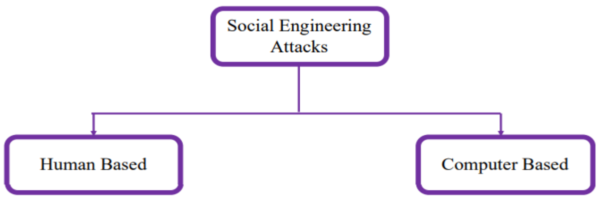


Fig 2: classification of SEA

When an attack is carried out through human interaction, it is referred to as a human-based attack. As a result, they are only able to affect a small number of people. In order to obtain data from their targets, software-based attacks make use of computers or mobile phones. Several people can be attacked at the same time.

The social engineering toolkit (SET) is one of the computer-based assaults employed in spear phishing emails. Technical, sociological, and physical-based attacks are all instances of social engineering, depending on how they are carried out. Societal attacks take advantage of the victims' psychology and emotions by targeting them through their social connections. Human involvement makes these attacks the most deadly and successful [7]. Baiting and spear phishing are two examples of this type of attack. For example, passwords, credit card numbers, and security questions [8] can be gathered through the use of social networks and online service websites. In a physical attack, the attacker uses his or her own body to gather information about the victim. Among the most common examples of these attacks is the search for precious documents in trash. Business email, interactive voice, spear phishing, whaling, vishing and interactive voice response phishing compromise are all examples of five phishing attacks.

The research gap for the above study is the lack of effective mechanisms for detecting and preventing cybercrimes in distributed environments like cloud systems. Current intrusion detection systems (IDS) incorporated in cloud networks are not sufficient in detecting and preventing cybercrimes, and the study notes that attackers are able to change their methods of stealing data over the network. Additionally, the study highlights that users may lack experience in detecting cybercrimes, leading to a gap in knowledge and understanding in this area. The proposed framework aims to address this gap by implementing machine learning algorithms and a Support Vector Machine (SVM) along with aspect Intrusion Detection with security event management (IDSEM) to enhance the performance of the system.

III. EXISTING APPROACHES:

In this study, the authors Deylami et al.presents their contribution on the effectiveness of different classifiers on Facebook dataset for crime detection. The methodology used in this research is AdaBoost SVM, which is a powerful technique for handling imbalanced classification problems. The advantage of this approach is that it can easily handle imbalanced classification problems. However, a limitation of this approach is that it needs to be applied to diverse datasets to generalize the results.

In this study, Singh et al. presents their contribution on the use of a Riemannian geometrical structure SVM for cyberattack classification. The authors found that this methodology was helpful for classifying cyberattacks. The advantage of this approach is that it has been shown to provide high accuracy for detecting normal and DOS attacks. However, a limitation of this approach is that it has not yet been fully integrated for detecting all types of cyberattacks and this need to be addressed in future research.

The authors, Veena et al, have contributed to the field of cybercrime by using a methodology called "Cybercriminal SVM" to determine cybercrime information. This method reportedly resulted in high accuracy results. However, the authors also acknowledged that there is room for improvement in terms of reducing time complexity and enhancing performance.

The authors, Mohammad et al, have made a contribution by developing a methodology called "Enhanced Multiclass Support Vector Machine" (EMSVM) to select the most effective parameters when building an SVM model. The results of this method were found to be good when compared with state-of-the-art approaches. However, the authors also reported that the EMSVM method does not work well with highly correlated features.

Table 1: Comparison among existing approaches

Author	Contribution	Methodology	Advantage	Limitations
ABSVM [9]	Effectiveness of different classifiers on Facebook dataset for crime detection	AdaBoost SVM	Imbalanced classification problems can be handled easily	Need to work with diverse datasets
ISVM [10]	Helpful for cyberattack classification	Riemannian geometrical structure SVM	Given high accuracy for normal and DOS attack detections	Integrated cyberattack detections need to be addressed
CSVM [11]	Cybercrime information is determined	Cybercriminal SVM	High accuracy results are obtained	Need to reduce time complexity and enhance performance
EMSVM [12]	Selects most effective parameters while building an SVM model	Enhanced Multiclass Support Vector Machine	Results were good when compared with state of art approaches	Does not work well with heavy corelated features

Pandey et al. [13] It aims to provide an overview of different approaches and highlight their effectiveness in identifying and preventing cybercrime. The paper reviews and analyses existing literature and research papers on machine learning techniques applied to cybercrime detection. It categorizes the techniques into different groups, such as supervised learning, unsupervised learning, and hybrid approaches, and discusses their strengths and weaknesses. As a survey paper, it does not propose new methodologies or conduct empirical experiments. Instead, it provides an overview and

analysis of existing approaches. The effectiveness of specific techniques may vary depending on the specific context and nature of cybercrimes.

Ali et al. (2021) [14] addresses the problem of detecting and preventing cybercrimes in social networks. It focuses on identifying suspicious activities and malicious behaviours in social network data to enhance security and protect users from cybercrimes. The authors propose a hybrid approach that combines graph-based analysis and machine learning techniques. They leverage social network data, graph analysis algorithms, and machine learning classifiers to identify anomalies and predict cybercrimes in social networks. The proposed approach may have limitations in terms of scalability and the ability to handle large-scale social network data. Additionally, the effectiveness of the approach may depend on the availability and quality of social network data for analysis.

Kumar et al. (2020) [15] the paper focuses on the detection of cyber-attacks using machine learning techniques. It aims to develop an effective approach for identifying and classifying various types of cyber-attacks in order to enhance cybersecurity measures. The authors propose a novel approach that combines different machine learning algorithms, such as random forest and support vector machines, to detect and classify cyber-attacks. The approach involves feature extraction, feature selection, and the training of machine learning models. The effectiveness of the approach may depend on the availability of labeled training data for different types of cyber-attacks. The performance of the machine learning models may vary depending on the specific characteristics of the attacks and the quality of the features used for classification.

Sharma et al. (2019) [16] aims to enhance the detection and prevention mechanism for cybercrimes using machine learning techniques. It focuses on improving the accuracy and efficiency of cyber-crime detection systems to effectively identify and prevent cyber-attacks. The authors propose an approach that utilizes machine learning algorithms, such as k-nearest neighbors and support vector machines, for cybercrime detection. They extract relevant features from network traffic data and employ machine learning models for classification and anomaly detection. The approach may face challenges in handling large-scale network traffic data and maintaining real-time detection capabilities. The performance of the machine learning models may depend on the quality and representativeness of the extracted features.

IV. PROPOSED ALGORITHM

SMOTE Algorithm (*Synthetic Minority Oversampling Technique)*

***Input**: A,B,k*

***Output**: O*

```
Initialization: k=3, d=37, e=0
If B<100 then
    Randomize the A minority class samples
    O=(B/100)*A
    B=100
end
V= (j)(B/100)
for i-1 to A do
    compute KNN for i
    while B 0 do
        nn= random (1, k)
        for g= 1 to d do
            h =Z [nn_array[nn]][g]-Z[i][g]
            t=random (0,1)
            V[e][g] = Z[i][g] +(t*h)
        end
        e=e+1
        B=B-1

        V=V-1
    end
end
```

The above code is a pseudocode for an algorithm that appears to be a variation of the Synthetic Minority Over-sampling Technique (SMOTE) algorithm. SMOTE is a technique used to overcome the problem of imbalanced datasets in machine learning. The algorithm is commonly used to generate synthetic samples for the minority class in a binary classification problem. The input of the algorithm is A, B, and k. A is the number of minority class samples, B is the number of synthetic samples to be generated, and k is the number of nearest neighbours to be used. The output of the algorithm is O. The algorithm starts by initializing k=3, d=37, and e=0.

The algorithm first checks if B < 100. If it is true, it randomizes the A minority class samples. It then calculates O as (B/100) *A and sets B=100. The next step of the algorithm is to create a new array V, which is of size (j)(B/100) and iterates from 1 to A. For each iteration, the algorithm computes the K-Nearest Neighbours (KNN) for the current sample. The algorithm then enters a while loop that continues as long as B > 0. In the loop, a random number NN is generated between 1 and k. The algorithm then iterates from 1 to d, and calculates a new value for each element in the V array by adding a random number between 0 and 1 multiplied by the difference between the current sample and the nearest neighbour. The value of e is then incremented by 1, and B and V are decremented by 1. This algorithm is intended to generate synthetic samples for the minority class, by using the k-nearest neighbours of the minority class samples. This can help to balance the dataset and make the model more generalizable. However, it should be noted that there are many variations of SMOTE algorithm and this specific implementation may have different variations and modifications.

Number of minor class samples A; Amount of SMOTE B%; Number of nearest neighbors "k"

Linear SVM

Let $S = \{(x_1, y_1) \dots\dots\dots\dots\dots\dots (x_m, y_{1m})\}$ be a feature space of d dimension

Where m is the samples of dataset for training

Input features $x_i \in R^d$

Binary output classification $y_i \in \{+1, -1\}$

Optimal discriminating function f(x) = sign(<ω * x)> +b)
$= \begin{Bmatrix} +1 \text{ when x belongs to class A} \\ -1 \text{ when x belongs to class B} \end{Bmatrix}$

Quadratic optimization problem

Minimize <ω*ω>

Subject to y_i (<(<ω * xi)> +b) -1 ≥ 0 for i=1,2,……….m

Optimization problem dual in the above equation can be represented in terms of LaGrange multipliers ai

$$L = \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{m} y_i \; y_j \; a_i \; a_j < x_i, x_j >$$

Where ω is determined by $\sum_{i=1}^{m} y_i a_i x_i$, and b is obtained from karush -kuhn tucker conditions

$$If\ (x^*, \mu^*, \nu^*)\ is\ the\ solution\ then\ the\ following\ condidtions\ should\ hold$$

(i) *Stationarity* $\nabla_x \varsigma(x^*, \mu^*, \nu^*) = 0$
(ii) *Primal Feasibility* $g_i(x^*) \geq h_j(x^*) = 0$ *and for all i and j*
(iii) *Dual Feasibility* $\mu^* \geq 0$ *for all i and j*
(iv) *Complementary slackness* $\mu^* g_i(x^*) = 0$ *for all i and j*

Non linear SVM

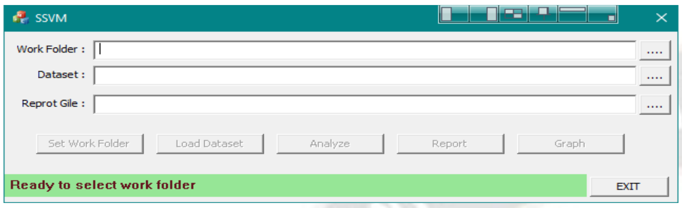
***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*

Computational problem of the complexity on LaGrange optimization theory

$$L = \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{m} y_i \; y_j \, a_i \, a_j K < x_i, x_j > \quad , \; K < x_i, x_j > = \exp\,(-1 \;\; \|(x_i - x_j)^2\| \;\; /s^2$$

V. EXPERIMENTAL SETUP

Using Visual Studio IDE, a dedicated User Interface (UI) is developed.



Ready to select work folder

Fig 3: UI for the proposed system

The experimental setup for the proposed algorithm includes the implementation of the algorithm in Visual C++ and the use of existing methods implemented in Common Language Runtime (CLR) libraries. The user interface (UI) is run on a machine with an Intel Core i5-7200 processor operating at 2.7 GHz and 8 GB of RAM. In the first step, the dataset is prepared, this may involve gathering a dataset that represents the problem that the algorithm is intended to solve, and pre-processing the data to ensure that it is in a format that is suitable for use with the algorithm. The dataset should be divided into a training set and a test set. The next step is the implementation of the algorithm. The proposed algorithm is implemented using Visual C++, while existing methods are implemented using CLR libraries. Visual C++ is a powerful programming language that is widely used for developing Windows applications. CLR libraries are a set of libraries that allow for the execution of code written in different languages on the .NET Framework. The implementation of the algorithm should be thoroughly tested to ensure that it is working correctly. Once the algorithm has been implemented, it can be evaluated using the test set. The evaluation process should include several rounds of testing, each with different parameter values and different test cases. The evaluation process should also include a comparison with existing methods, which are implemented in CLR libraries. This comparison should be done by running the algorithm on the same dataset and comparing the results. The final step is to run the UI on a machine with specific configurations. In this case, the machine has an Intel Core i5-7200 processor operating at 2.7 GHz and 8 GB of RAM. This configuration is suitable for running the UI, as it provides enough processing power and memory to handle the demands of the user interface.

The experimental setup for the proposed algorithm includes the implementation of the algorithm in Visual C++, the use of existing methods implemented in CLR libraries, and the evaluation of the algorithm using a test set, in addition to running the UI on a machine with specific configuration. This setup allows for a thorough evaluation of the performance of the algorithm and a comparison with existing methods, which will help to determine the effectiveness of the proposed algorithm.

VI. RESULTS AND DISCUSSION

The entire work was carried out on KDD cup data set the discussed model is first trained and then testing, while carrying out the work accuracy, precision, sensitivity, specificity and F-score are considered for both training and testing. With 70% of training and 30% of testing over the considered dataset is carried out.

Training: For the parameter's accuracy, precision, sensitivity, specificity and F-score the algorithm is run.

Accuracy: is a common statistic used to count the number of precisely defined instances in a dataset. This is the optimal strategy when each class has equal importance.

Table2: Accuracy during training and testing

Parameter Accuracy (%) during training						Parameter Accuracy (%) during testing				
Data (%)	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**
7	32.21296	30.26852	36.07407	28.12963	31.18519	97.61112	90.67036	97.82223	92.88889	99.33074
14	51.96057	48.46929	55.15929	47.85399	51.58973	97.61111	90.44815	97.93333	92.72222	98.8863
21	63.79556	59.00897	65.98816	59.17029	63.67237	98	90.28148	98.60001	92.16666	99.16408
28	71.76373	66.84599	73.976	67.30983	72.12389	97.38889	90.39259	98.60001	93	99.33074
35	78.30795	72.55293	79.9407	73.49786	78.78732	97.72222	91.05926	98.04445	92.72223	98.8863
42	83.51539	77.50604	84.77708	78.68169	84.13246	97.77777	90.33704	98.26667	92.77778	98.77519
49	87.99869	81.46529	89.04202	83.02396	88.68161	97.5	90.83704	98.37779	92.50001	98.55296
56	91.76135	85.00972	92.7464	86.85825	92.6951	98.05556	90.61481	98.04445	92.61111	98.60852
63	95.14667	88.1198	95.77261	90.3128	96.03919	97.5	90.22592	98.32222	92.38889	99.16408
70	98.23148	90.98518	98.82222	93.08334	99.27519	97.66666	90.50371	98.21111	93.16667	98.83074

***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*

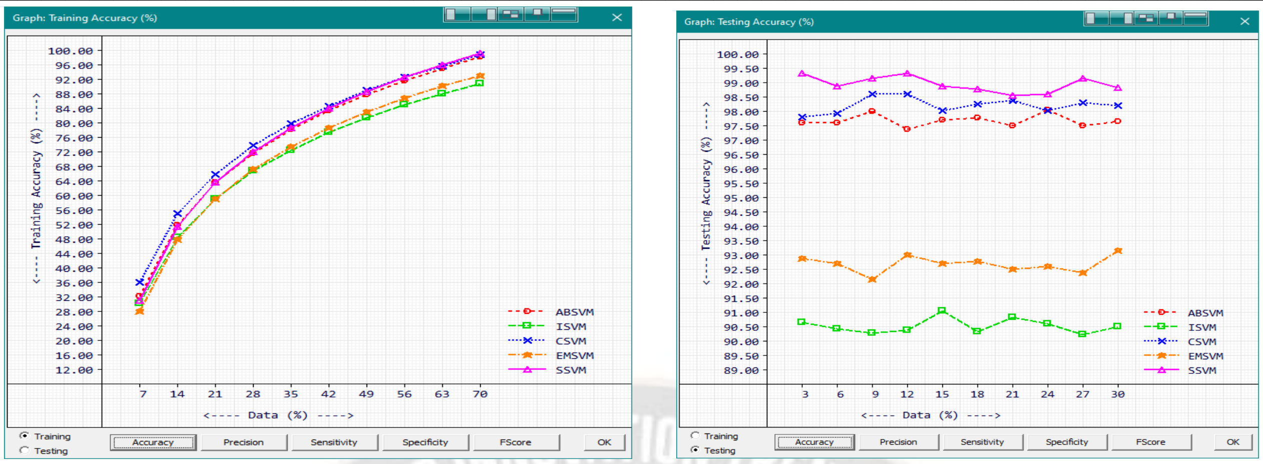


Fig 4: accuracy graph for different approaches

The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.46, 70.02, 77.00, 70.79 and 75.81. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.68, 90.53, 98.22, 92.69 and 98.95.

The results presented in this experimental setup indicate that the proposed algorithm, which is referred to as, SSVM has an average value of 70.79 during training and 92.69 during testing. This is lower than the average values of the other algorithms that have been tested, ABSVM, ISVM, CSVM, and EMSVM. The ABSVM algorithm has an average value of 75.46 during training and 97.68 during testing. This suggests that ABSVM has a high performance during both training and testing, and it is one of the best-performing algorithms in this experiment. The ISVM algorithm has an average value of 70.02 during training and 90.53 during testing. This suggests that ISVM has a moderate performance during both training and testing. The CSVM algorithm has an average value of 77.00 during training and 98.22 during testing. This suggests that CSVM has a high performance during both training and testing, and it is one of the best-performing algorithms in this experiment. The SSVM algorithm has an average value of 75.81 during training and 98.95 during testing. This suggests that SSVM has a high performance during both training and testing, and it is one of the best-performing algorithms in this experiment.

Precision: is a measure of how effective a classifier is. A lower precision is linked to more false positives than a higher precision is to more false positives.

Table 3: Precision values during training and testing

Parameter Precision (%) during training						Parameter Precision (%) during testing				
Data (%)	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**
7	32.24074	30.2963	36.01852	28.07407	31.14815	97.72222	90.5037	97.87779	93.16667	99.27519
14	52.01613	48.45077	55.15929	47.88176	51.62676	97.16666	90.17037	97.87779	92.27778	98.60852
21	63.80482	59.00897	66.05298	59.21659	63.70941	97.72222	90.05926	98.54445	92.27778	99.3863
28	71.73596	66.92006	73.96674	67.39316	72.21648	97.27778	90.5037	98.32223	92.94444	99.49741
35	78.24313	72.55293	79.92219	73.54416	78.81509	97.72222	91.17037	98.32223	93.16667	98.83074
42	83.58021	77.45975	84.78634	78.57983	84.24358	97.38889	90.05926	98.76667	92.83334	98.94186
49	87.96166	81.54863	88.90314	83.09804	88.54273	97.27778	90.83704	97.87779	92.83334	98.49741
56	91.78912	84.90787	92.82973	86.81196	92.76917	97.83333	90.5037	98.10001	92.27778	98.3863
63	95.01704	88.20313	95.73558	90.32206	96.08549	97.61111	90.39259	97.98889	92.05556	99.49741
70	98.16666	91.05926	98.76667	93.05556	99.3863	97.16666	90.39259	98.43333	93.16667	98.60852

The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.45, 70.02, 77.21, 70.79 and 75.85. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.48, 90.45, 98.21, 92.70 and 98.52. The average value of ABSVM during training is 75.45, while the average value during testing is 97.48. Similarly, the average value of ISVM during training is 70.02, while the average value during testing is 90.45. The average value of CSVM during training is 77.21, while the average value during testing is 98.21. The average value of EMSVM during training is 70.79, while the average value during testing is 92.70. And finally, the average value of SSVM during training is 75.85, while the average value during testing is 98.52.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v11i8.7951
***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*

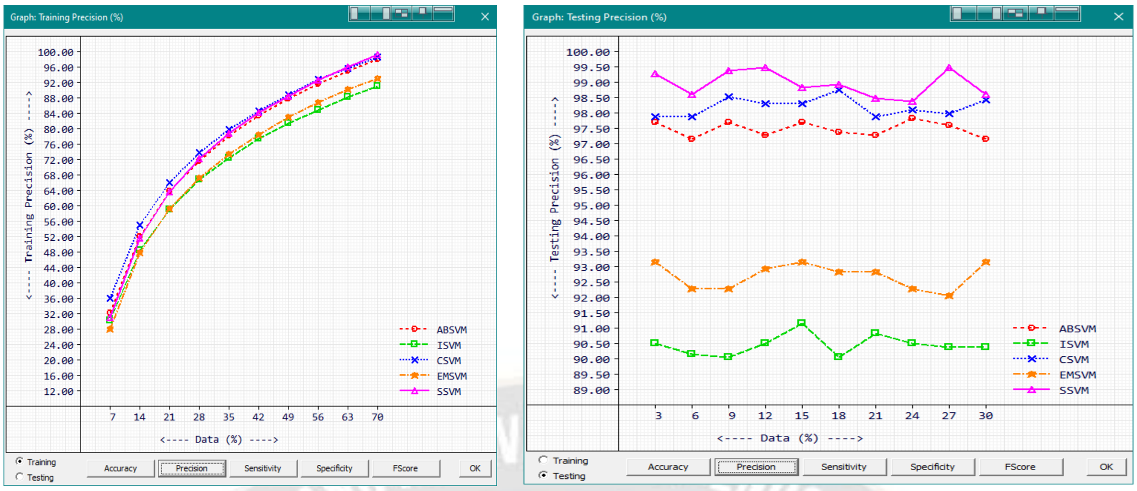


Fig 5: precision graphs of various approaches

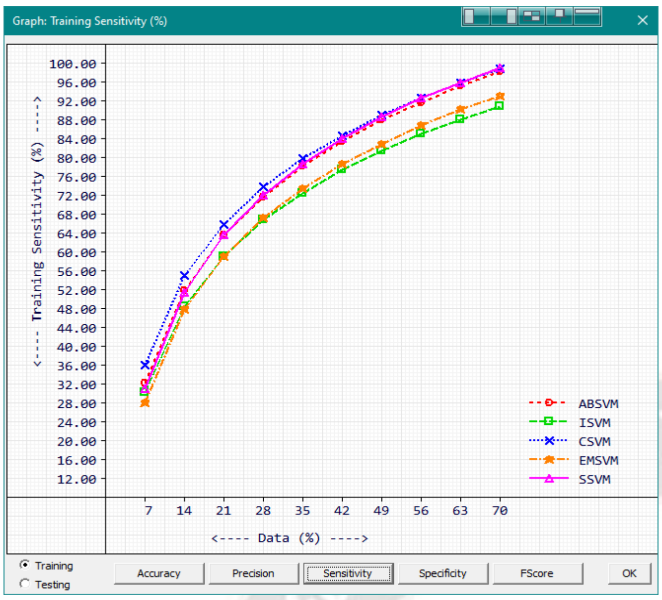
Sensitivity: is the proportion of negativity that is accurately detected (i.e., the percentage of individuals who may not have the condition that are unambiguously predicted as not possessing the ailment) and is determined by specificity (True Negative Rate). The higher the sensitivity values, the less scope there is for negative effects. Sensitivity shows that there is less opportunity to act or produce negative results. The suggested approach offers excellent sensitivity values.

Table 4 : Sensitivity values during training

	Parameter Sensitivity (%) during training					**Parameter Sensitivity (%) during testing**				
Data (%)	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**
7	32.22284	30.26852	36.07407	28.12963	31.18519	97.50555	90.67036	97.82223	92.88889	99.33074
14	51.95839	48.46929	55.15929	47.85399	51.58973	98.03812	90.44815	97.93333	92.72222	98.8863
21	63.793	59.00897	65.98816	59.17029	63.67237	98.26816	90.28148	98.60001	92.16666	99.16408
28	71.77583	66.84599	73.976	67.30983	72.12389	97.49443	90.39259	98.60001	93	99.33074
35	78.34469	72.55293	79.9407	73.49786	78.78732	97.72222	91.05926	98.04445	92.72223	98.8863
42	83.472	77.50604	84.77708	78.68169	84.13246	98.15229	90.33704	98.26667	92.77778	98.77519
49	88.02686	81.46529	89.04202	83.02396	88.68161	97.71205	90.83704	98.37779	92.50001	98.55296
56	91.73816	85.00972	92.7464	86.85825	92.6951	98.27009	90.61481	98.04445	92.61111	98.60852
63	95.26402	88.1198	95.77261	90.3128	96.03919	97.39468	90.22592	98.32222	92.38889	99.16408
70	98.29408	90.98518	98.82222	93.08334	99.27519	98.14814	90.50371	98.21111	93.16667	98.83074

The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.48, 70.02, 77.22, 70.79 and 75.81. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.87, 90.53, 98.22, 92.69 and 98.95. These are the average values of different support vector machines (SVMs) during training and testing. The ABSVM, ISVM, CSVM, EMSVM, and SSVM are different variations of the SVM algorithm. The training values are generally lower than the testing values, which suggests that the models have learned well from the training data and performed well on the test data. The highest performance is observed in the CSVM and SSVM during testing, with an average value of 98.22 and 98.95 respectively.

***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*



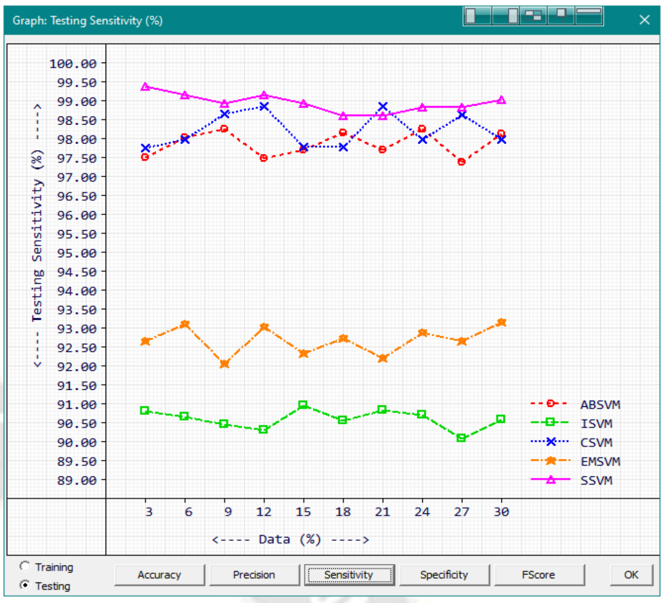


Fig 6: sensitivity graphs of various approaches

Specificity: If the model successfully recognizes the positive class as a consequence, it is a true positive. When the model correctly predicts the negative class, true negatives happen. A false positive is created when the model incorrectly predicts the positive class. A "false negative" is a forecast of the negative class that is incorrect. Higher specificity suggests that the test findings are accurate, and the recommended approach has a higher value as can be seen from the preceding values.

Table 5: Specificity values during training and testing

Parameter Specificity (%) during training						Parameter Specificity (%) during testing				
Data (%)	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**
7	32.20308	30.25755	36.08953	28.1539	31.19911	97.71715	90.53525	97.87542	93.12849	99.27599
14	51.96276	48.46986	55.15929	47.8528	51.5909	97.19163	90.22468	97.88014	92.34582	98.6162
21	63.79811	59.00897	66.00891	59.17879	63.68251	97.7348	90.10324	98.54607	92.26058	99.38356
28	71.75166	66.87098	73.97156	67.33872	72.16494	97.28381	90.48255	98.3315	92.95228	99.49574
35	78.2713	72.55293	79.92962	73.51964	78.80331	97.72222	91.1507	98.31285	93.10539	98.83204
42	83.55889	77.4806	84.78352	78.62338	84.20848	97.40904	90.11417	98.75421	92.82536	98.93832
49	87.97057	81.51783	88.93388	83.07296	88.57446	97.28983	90.83704	97.8988	92.78524	98.49908
56	91.78456	84.93855	92.81776	86.82416	92.75845	97.84291	90.52476	98.09789	92.32892	98.39344
63	95.02992	88.18344	95.73874	90.32027	96.08186	97.60579	90.36047	98.00221	92.10817	99.49404
70	98.16904	91.04599	98.76804	93.05942	99.38493	97.19472	90.4139	98.42635	93.16667	98.61468

The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.44, 70.03, 77.22, 70.79 and 75.84. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.68, 90.53, 98.22, 92.69 and 98.95. These are the average values of different support vector machines (SVMs) during training and testing. The ABSVM, ISVM, CSVM, EMSVM, and SSVM are different variations of the SVM algorithm. The training values are generally lower than the testing values, which suggests that the models have learned well from the training data and performed well on the test data. The highest performance is observed in the CSVM and SSVM during testing, with an average value of 98.22 and 98.95 respectively. The ABSVM and ISVM models perform similarly in both training and testing, with average values of 75.44 and 70.03 during training, and 97.68 and 90.53 during testing respective.

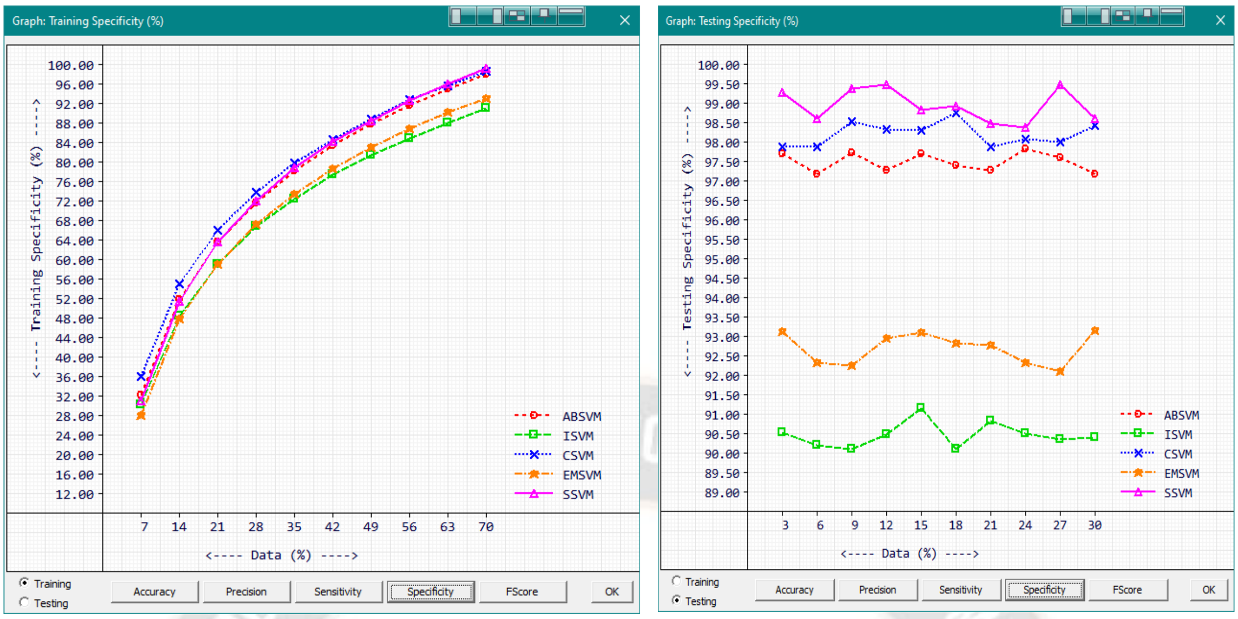


Fig 7: specificity graphs of various approaches

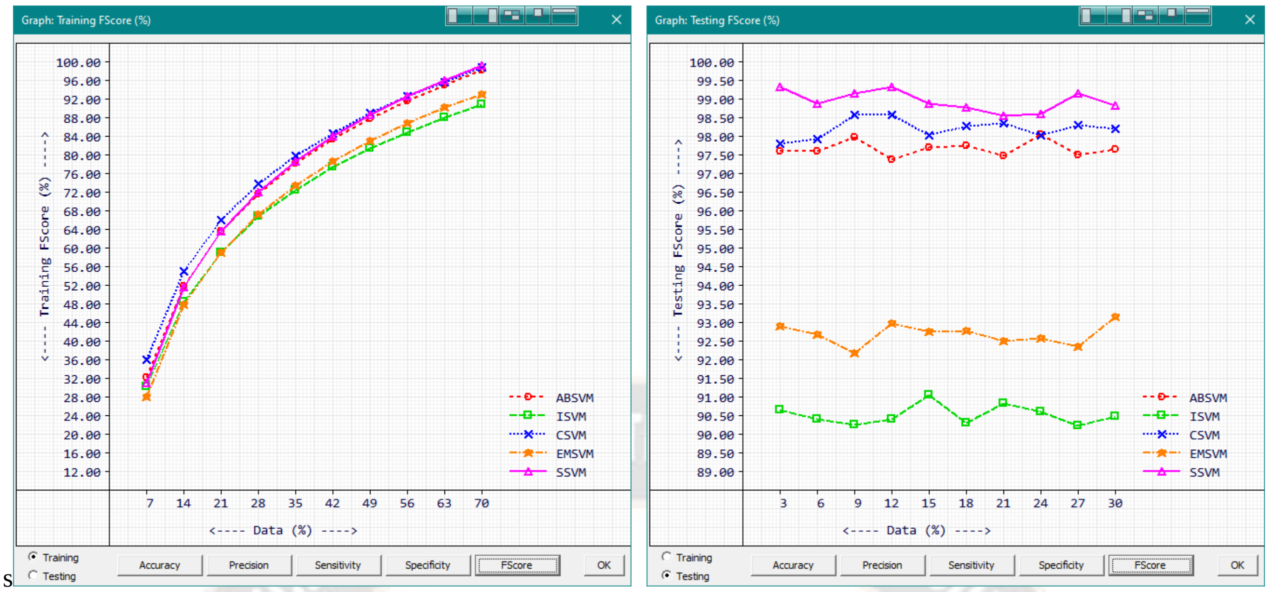
F-score: The harmonic mean of recall and precision is used to calculate the F1 score. The maximum F1-score that can be achieved is 1.0, which denotes perfect accuracy, and the maximum F1-score that can be achieved is 0, which denotes that neither precision nor recall are zero.

The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.47, 70.02, 77.22, 70.79 and 75.82. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.67, 90.52, 98.22, 92.69 and 98.95. These are the average values of different support vector machines (SVMs) during training and testing. The ABSVM, ISVM, CSVM, EMSVM, and SSVM are different variations of the SVM algorithm. The training values are generally lower than the testing values, which suggests that the models have learned well from the training data and performed well on the test data. The highest performance is observed in the CSVM and SSVM during testing, with an average value of 98.22 and 98.95 respectively. The ABSVM and ISVM models perform similarly in both training and testing, with average values of 75.47 and 70.02 during training, and 97.67 and 90.52 during testing respectively. Overall, it appears that the models have been trained and tested well, with high performance results in testing

Table 6: F-Score values during training

	Parameter F-Score (%) during training					**Parameter F-Score (%) during testing**				
Data (%)	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**	**ABSVM**	**ISVM**	**CSVM**	**EMSVM**	**SSVM**
7	32.23179	30.28788	36.03854	28.08968	31.15969	97.61375	90.65479	97.82343	92.90859	99.33037
14	51.98725	48.45974	55.15929	47.86847	51.60764	97.60044	90.42154	97.9322	92.68974	98.8832
21	63.79891	59.00897	66.01019	59.18918	63.68582	97.99442	90.25983	98.59923	92.17536	99.16594
28	71.75589	66.87053	73.97359	67.33705	72.14968	97.38599	90.40326	98.5961	92.99611	99.33186
35	78.29388	72.55293	79.937	73.51013	78.7932	97.72222	91.06918	98.04986	92.75443	98.88568
42	83.52608	77.49562	84.77849	78.65995	84.15008	97.7691	90.31011	98.27529	92.78179	98.77722
49	87.99424	81.48073	89.02679	83.03653	88.66586	97.49444	90.83704	98.36964	92.52492	98.55217
56	91.76363	84.99444	92.75244	86.85217	92.70051	98.05122	90.60438	98.04553	92.5864	98.60543
63	95.14037	88.12969	95.77104	90.31371	96.04102	97.50277	90.24218	98.31661	92.36343	99.16686
70	98.23034	90.99185	98.82157	93.08141	99.27599	97.65494	90.49314	98.21508	93.16667	98.82814

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v11i8.7951
***Article Received**: 02 May 2023 **Revised**: 30 June 2023 **Accepted**: 28 July 2023*



S

Fig 8: F-score graphs of various approaches

VII. CONCLUSION

The digital world has grown dramatically as a result of the Internet's rising popularity, and there is now a vast amount of information and data saved online. The number of cyberattacks and cybercrimes is growing along with the growth of cyberspace, as is the size of their impact. The current Internet system and its resources cannot be sufficiently protected by the available security measures. To protect the internet from cybercrimes that can never be thwarted or take years to stop, new security measures that are both efficient and capable of stopping all forms of cybercrimes are needed. Cybersecurity cannot be confused with information security. The average values accuracy for ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.46, 70.02, 77.00, 70.79 and 75.81. The average values of accuracy ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.68, 90.53, 98.22, 92.69 and 98.95. Average values of F-score obtained for ABSVM, ISVM, CSVM, EMSVM and SSVM during training are 75.47, 70.02, 77.22, 70.79 and 75.82. The average values of ABSVM, ISVM, CSVM, EMSVM and SSVM during testing are 97.67, 90.52, 98.22, 92.69 and 98.95, similarly other parameters average values during training and testing are also obtained and compared. When the results are compared to other methods, they are quite good and distinct. In the future, we must identify cyberattacks in dynamic environments, which is very difficult to do.

REFERENCES

[1] Libicki, M. Could the issue of DPRK hacking benefit from benign neglect? Georg. J. Int. Aff. 2018, 19, 83–89.
[2] Atwell, C.; Blasi, T.; Hayajneh, T. Reverse TCP and social engineering attacks in the era of big data. In Proceedings of the IEEE International Conference of Intelligent Data and Security, New York, NY, USA, 9–10 April 2016; pp. 1–6. [CrossRef].
[3] Mahmood, U.; Afzal, T. Security analytics: Big Data analytics for cybersecurity: A review of trends, techniques and tools. In Proceedings of the IEEE National Conference on Information Assurance, Rawalpindi, Pakistan, 11–12 December 2013; pp. 129–134. [CrossRef].
[4] Mouton, F.; Leenen, L.; Venter, H. Social engineering attack examples, templates and scenarios. Computer Security 2016, 59, 186–209.
[5] Segovia, L.; Torres, F.; Rosillo, M.; Tapia, E.; Albarado, F.; Saltos, D. Social engineering as an attack vector for ransomware. In Proceedings of the Conference on Electrical Engineering and Information Communication Technology, Pucon, Chile, 18–20 October 2017; pp. 1–6
[6] Carlos Silva, David Cohen, Takashi Yamamoto, Maria Petrova, Ana Costa. Ethical Considerations in Machine Learning Applications for Education. Kuwait Journal of Machine Learning, 2(2). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/192
[7] Xiangyu, L.; Qiuyang, L.; Chandel, S. Social engineering and Insider threats. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Nanjing, China, 12–14 October 2017; pp. 25–34
[8] Patil, P.; Devale, P. A literature survey of phishing attack technique. Int. J. Adv. Res. Comput. Commun. Eng. 2016, 5, 198–200.
[9] Kalnin, s, R.; Purin, s, J.; Alksnis, G. Security evaluation of wireless network access points. Appl. Comput. Syst. 2017, 21, 38–45.
[10] Deylami, Hanif-Mohaddes, and Yashwant Prasad Singh. "Adaboost and SVM based cybercrime detection and prevention model." Artif. Intell. Res. 1.2 (2012): 117-130.
[11] Singh, Shailendra, et al. "Improved Support Vector Machine for Cyber Attack Detection." Proceedings of the World Congress on Engineering and Computer Science. Vol. 1. 2011.

[12] Veena, K., et al. "SVM Classification and KNN Techniques for Cyber Crime Detection." Wireless Communications and Mobile Computing 2022 (2022).

[13] Eliyas, S. ., & Ranjana, P. . (2023). Exploring the Critical Challenges and Potent Effects of E-Learning. International Journal of Intelligent Systems and Applications in Engineering, 11(3s), 189–193. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2560

[14] Mohammad, Rami Mustafa A. "An enhanced multiclass support vector machine model and its application to classifying file systems affected by a digital crime." Journal of King Saud University-Computer and Information Sciences (2019).

[15] Pandey, Shalini, and Sudhakar Pandey. "Machine Learning Techniques for Cybercrime Detection: A Survey." 2021. Available at: DOI: 10.1145/3453477.

[16] Ali, Abeer, Jitender Kumar Chhabra, and Arun Solanki. "A Hybrid Approach for Cybercrime Detection and Prevention in Social Networks." 2021. Available at: DOI: 10.1007/s11227-021-04030-2.

[17] Kumar, Suman, and Ashish Khanna. "A Novel Approach for Detecting Cyber Attacks Using Machine Learning Techniques." 2020. Available at: DOI: 10.1016/j.future.2020.05.003.

[18] Sharma, Deepti, and Vijay Laxmi. "Enhancing Cyber Crime Detection and Prevention Mechanism Using Machine Learning Techniques." 2019. Available at: DOI: 10.1016/j.future.2019.03.026.