International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Intrusion Detection System on Cloud Computing Using Ensemble Svm

R.Hari Krishna[1], Pallipamula Vijaya Bhaskar[2], Prasritha Narahari[3], Meghana Nalluri[4], Mohith Ram Mallapureddy[5]

[1]Assistant Professor, Department of IT, Sir C.R.Reddy college of Engineering, India.
[2,3,4,5]B. Tech student, Department of IT, Sir C.R.Reddy college of Engineering, India.

Abstract
This study introduces a specialized Intrusion Detection System (IDS) designed specifically for cloud computing environments. By leveraging the UNSW_NB15 dataset, it employs feature selection techniques such as SelectKBest and ANOVA to extract relevant features, thereby improving the overall performance of the model. The IDS framework encompasses data preprocessing, Ensemble SVM model training, and performance evaluation utilizing standard metrics. The key methodology revolves around training SVM models using bagging and boosting techniques on preprocessed data, resulting in resilient intrusion detection models. These models are subsequently utilized to determine whether a given set of input features signifies a network intrusion. Through experimental analysis, the research demonstrates the system's efficacy in accurately detecting network intrusions, shedding light on its robustness and dependability.

Keywords: UNSW_NB15, bagging, boosting, Ensemble SVM, ANOVA, SelectKBest.

1. Introduction

Central to the efficacy of the proposed IDS is the employment of sophisticated feature selection methodologies, namely SelectKBest and Analysis of Variance (ANOVA), which serve as instrumental tools in extracting salient features from the dataset. By discerning and of the intrusion detection system, enabling it to accurately discern network intrusions while minimizing false positives.

At the heart of the methodology lies the process of training SVM models on meticulously preprocessed datasets, employing ensemble techniques to ameliorate classification accuracy and fortify resilience against fluctuating data distributions. Subsequently, these adeptly trained models are leveraged to predict whether This research presents a The present study constitutes a thorough investigation into the design and implementation of an Intrusion Detection System (IDS) specifically tailored for cloud computing infrastructures. Leveraging the widely recognized UNSW_NB15 dataset, renowned in the domain of network security, this research endeavors to pioneer innovative strategies aimed at augmenting prioritizing the most discriminative features, these techniques play a pivotal role in enhancing the overall performance of the intrusion detection models.

The architecture of the IDS is multifaceted, encompassing several pivotal stages. These stages include comprehensive data preprocessing, rigorous model training employing Ensemble Support Vector Machine (SVM) models fortified with bagging and boosting techniques, and meticulous performance

evaluation utilizing a repertoire of standard metrics. Such a holistic approach ensures the resilience and dependability a given set of input features corresponds to anomalous network activity indicative of intrusion, or normal operational behavior.

2. Literature Review

Network intrusion detection is a crucial element of cybersecurity, aimed at identifying and thwarting malicious activities within computer networks. Researchers have delved deeply into diverse machine learning algorithms and datasets to craft effective Intrusion Detection Systems (IDS). This literature review delves into recent studies that center on the application of machine learning techniques for network intrusion detection, with a particular emphasis on ensemble methods and various datasets.

In the realm of automatic intrusion detection, two primary machine learning techniques emerge: supervised learning and unsupervised learning. However, this survey predominantly focuses on the application of supervised machine learning techniques for IDS. In supervised learning or classification, the presence of labeled data is imperative to train models for detection purposes.

TABLE 1: Description of IDS datasets [1]

Dataset	Description
KDD'99	It is generated using simulation of normal and attacks traffic in a military environment (US AirForce LAN). It contains nine weeks of simulation in rat tcpdump files. The dataset is characterized using 41 features related to intrinsic, content, and traffic. Four types of attacks are simulated: DoS, Prob, U2R, and R2L.
NSL-KDD	It is a modification to the KDD'99 dataset with solving the problems of redundancy, duplicates, the imbalance of data.
UNSW-Nb15	It was created using the IXIA PefectStorm tool to extract normal and attack network traffic based on 100 GB of raw network traffic. It is characterized using 49 features. It consists of around 175 thousand records for training and around 82 thousand records for testing. There are nine types of attacks: Fuzzers, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm
CICIDS2017	It was created in an emulated environment in a 5 day period. It contains traffic in packet flow and bidirectional flow. 80 features are extracted. Attacks involve: Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS

TABLE 2: Distribution of Dataset Usage over the Years [2]

TITLE	ALGORITHM	DATASET	RESULT (ACCURACY)	FINDING	DRAWBACK
Network Intrusion Detection using Clustering Gradient boosting	1) Extreme Gradient Boosting(XGBoost) 2)Adaptive Boosting (AdaBoost)	NSL-KDD	XGBoost =80.238% AdaBoost =80.731%	The work showed that anomaly detection has a room to improve its false positive.	The ensemble of the classifiers used needs to be evaluated on the most updated datasets that contain recent attacks.

Anomaly Network Intrusion Detection Using Ensemble Machine Learning Technique.	1) Decision Tree 2) Bayes Classifier 3)RNN-LSTM 4) Random Forest 5) Ensemble of the 4 classifiers	NSL-KDD	Ensemble=85.20%	The work handled imbalanced data and selected only required features which greatly helped in reducing high false positive rate.	Trial on the most updated datasets needs to be carried out.
Implementation of Machine Learning Algorithms for Detection of Network Intrusion.	1) Decision Tree (DT) 2) Logistic Regression (LR) 3) Random Forest (RF) 4) Support Vector Machine (SVM)	KDD-NSL	1) RF=73.784% 2) DT=72=303% 3) SVM=71.779% 4) LR=68.674%	Showed that working with random forest in building IDS saves execution Time.	The model performs efficiently only with single classifier.

Recent studies in network intrusion detection have examined various machine learning algorithms to enhance accuracy and efficiency. One study utilized Extreme Gradient Boosting (XGBoost) and Adaptive Boosting (AdaBoost) on the NSL-KDD dataset, achieving accuracies between 80.238% and 84.253%. While anomaly detection shows promise, there is room for improvement in reducing false positives, especially through evaluating ensemble classifiers on datasets with recent attacks.

Another investigation explored ensemble machine learning techniques, including Decision Trees, Bayes Classifier, RNN-LSTM, Random Forest, and an ensemble of these classifiers, achieving an accuracy of 85.20%. This study tackled imbalanced data and feature selection issues, notably reducing the high false positive rate. However, it highlighted the need for further experimentation on the most updated datasets.

Additionally, research on machine learning algorithms for intrusion detection, such as Decision Trees, Logistic Regression, Random Forest, and Support Vector Machine, yielded accuracies ranging from 68.674% to 73.784% on the KDD-NSL dataset. Although Random Forest demonstrated efficient execution time, it was effective only with a single classifier.These findings underscore the importance of continuously evaluating and refining machine learning algorithms and techniques for network intrusion detection to enhance accuracy, reduce false positives, and adapt to evolving cybersecurity threats.

3. Problem Statement

Despite significant advancements in machine learning algorithms for network intrusion detection, there

remains a crucial requirement to improve the accuracy and efficiency of anomaly detection systems, particularly in addressing false positives and adapting to evolving cyber threats. While recent research has demonstrated promising results with ensemble techniques and feature selection methods, challenges persist in effectively evaluating and optimizing these techniques on datasets containing the latest attack patterns. Therefore, the primary objective of this study is to develop and evaluate an enhanced intrusion detection system that integrates ensemble machine learning algorithms, feature selection methods, and up-to-date datasets to enhance the accuracy and reliability of anomaly detection in network environments.

Research Objectives:

- Design and implement an enhanced intrusion detection system (IDS) that integrates ensemble machine learning algorithms and feature selection methods to enhance accuracy and efficiency.
- Evaluate the performance of the IDS using up-to-date datasets containing recent cyber attacks to ensure its effectiveness in detecting evolving threats.
- Investigate the effectiveness of ensemble techniques, such as Extreme Gradient Boosting (XGBoost) and Adaptive Boosting (AdaBoost), in reducing false positives and improving anomaly detection accuracy.
- Assess the impact of feature selection methods, including clustering and gradient boosting, on the performance of the IDS to identify the most effective approaches for selecting relevant features.
- Compare the performance of the proposed IDS with existing intrusion detection systems, focusing on metrics such as accuracy, false positive rate, and execution time, to demonstrate its superiority.
- Identify areas for further improvement and optimization of the intrusion detection system based on experimental results and analysis, ensuring continual enhancement of its capabilities.
- Contribute to the body of knowledge in the field of network security by advancing understanding of effective techniques for intrusion detection in dynamic cyber threat landscapes, thereby enhancing overall cybersecurity measures.

4. Dataset Description:

The UNSW_NB15 dataset serves as a cornerstone in cybersecurity research, particularly in the development and evaluation of intrusion detection systems (IDS) and network security solutions. Here's a comprehensive overview of the dataset:

Data Source: The dataset is derived from network traffic data collected within a controlled environment designed to replicate a small-scale enterprise network. This data was generated by simulating a diverse range of network attacks and normal activities within a laboratory setup, ensuring a rich and varied dataset for analysis.

Composition: Within the UNSW_NB15 dataset, there exists a balanced mix of benign (normal) traffic and malicious (intrusion) traffic. It comprises millions of records capturing network traffic events, with each record representing either a single network flow or packet. This extensive dataset provides researchers with ample data points to train and evaluate their intrusion detection models effectively.

Features: The dataset encompasses a broad array of features extracted from network packets, flow data, and associated metadata. These features include critical network attributes such as source and destination IP addresses, port numbers, protocol types, packet sizes, timestamps, and various other network-related characteristics. The inclusion of these diverse features enables researchers to develop robust and

comprehensive intrusion detection models capable of identifying subtle patterns indicative of malicious activities.

Attack Categories: To facilitate comprehensive analysis and evaluation, the UNSW_NB15 dataset categorizes network traffic instances into nine distinct types of attacks. These attack categories encompass a spectrum of malicious activities, including Fuzzers, Analysis, Backdoor, Denial-of-Service (DoS), Exploits, Generic attacks, Reconnaissance, Shellcode, and Worms. Each attack category represents unique strategies and techniques employed by attackers to compromise network security, providing researchers with valuable insights into the diverse landscape of cyber threats.

Overall, the UNSW_NB15 dataset stands as a valuable resource for cybersecurity research, offering researchers a rich and diverse dataset to develop, test, and benchmark intrusion detection systems and network security solutions effectively. Its comprehensive composition and inclusion of various attack categories make it an indispensable tool in advancing the field of network security and combating evolving cyber threats.

5. System Architecture:

The System Architecture is shown in below figure1:

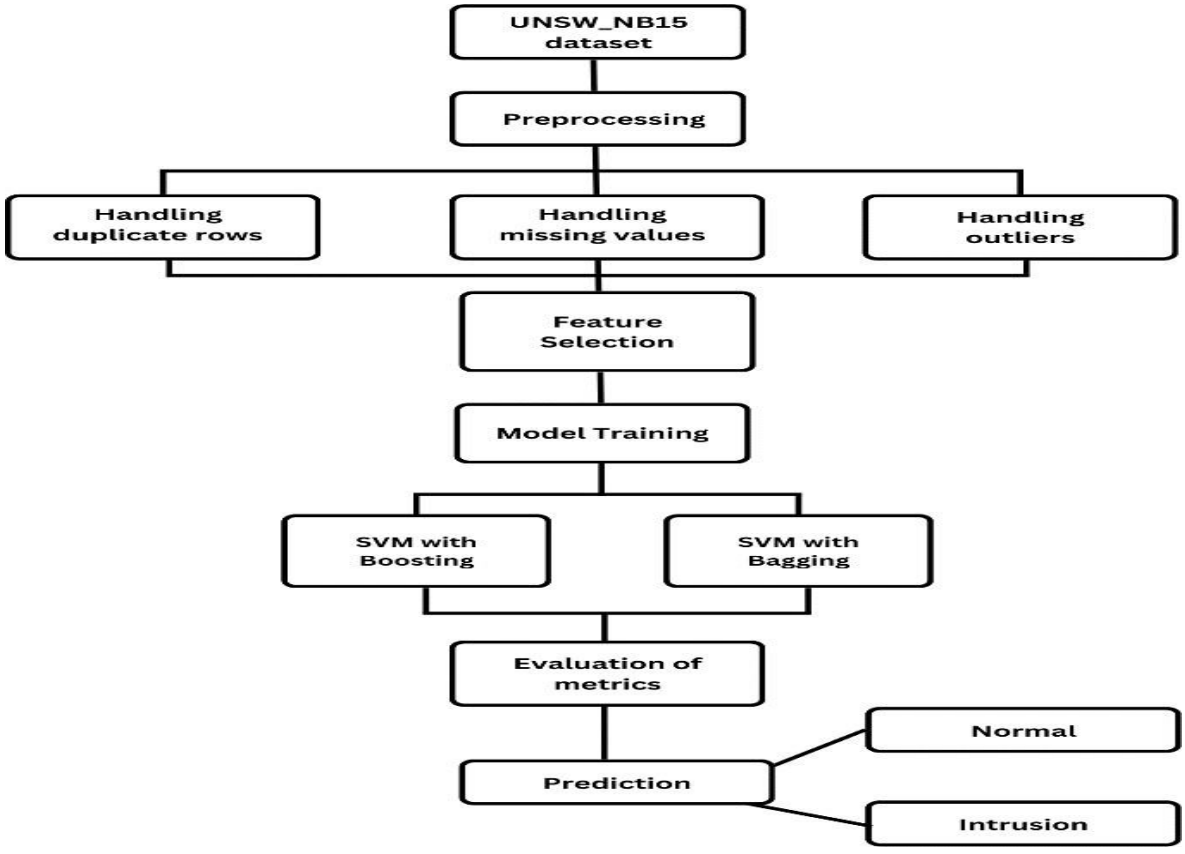


Figure 1:System Architecture

5.1 Preprocessing:

To enhance the quality and reliability of the dataset for subsequent analysis, several preprocessing steps are undertaken:

Handling Missing Values: Replace any missing values represented by dashes ('-') with Pandas' missing value representation, denoted as 'pd.NA'. This ensures consistency in handling missing data throughout

the dataset. Additionally, missing values in categorical columns, such as the 'service' column, are filled with the mode of the respective column. This imputation strategy preserves data integrity while retaining the categorical nature of features.

Removing Duplicate Rows: Identify and remove duplicate rows within the dataset. This step is essential for eliminating redundancy and ensuring each observation is unique, thereby preventing potential biases in subsequent analyses. By removing duplicates, the dataset becomes more streamlined, facilitating more accurate and robust modeling outcomes.

Addressing Outliers: Detect and address outliers using the Interquartile Range (IQR) method. Outliers, which lie significantly beyond the typical range of values for a given feature, can distort statistical analyses and modeling results. The IQR method identifies outliers based on the spread of data around the median, providing a robust detection mechanism. Detected outliers are subsequently removed from the dataset, leading to a more representative and reliable dataset for analysis.

Visual Assessment: Generate boxplots before and after outlier removal to visually assess the impact of outlier removal. Boxplots offer a concise summary of the data distribution, allowing visualization of central tendency, spread, and identification of potential outliers. Comparing boxplots before and after outlier removal enables researchers to evaluate the effectiveness of the preprocessing step in mitigating the influence of outliers on the dataset.

5.2 Feature Selection:

Select Best is a valuable feature selection method used in machine learning, particularly in tasks such as classification and regression. Its primary purpose is to identify the top k features from a dataset based on their relevance or importance to the target variable. This approach is especially beneficial when dealing with datasets containing numerous features, as it helps reduce dimensionality and computational complexity while retaining the most informative attributes.

The SelectKBest algorithm functions by scoring each feature individually, typically using statistical tests or metrics, and then selecting the k features with the highest scores. Some common scoring functions employed by SelectKBest include:

ANOVA F-value: This metric evaluates the linear relationship between each feature and the target variable by calculating the ratio of the variance between group means to the variance within groups. It indicates the extent to which the feature discriminates between different classes.

Mutual Information: This metric measures the mutual dependence between two variables, capturing both linear and non-linear relationships. It quantifies the amount of information that one variable provides about another, making it suitable for feature selection in classification tasks.

Chi-squared: This statistical test assesses the independence between categorical variables by comparing the observed and expected frequencies of each category. It indicates the significance of the association between the feature and the target variable.

Select Best allows users to specify the desired number of features (k) to select from the dataset. By choosing the top k features based on their scores, Select Best helps identify the most relevant features for predictive modeling, thereby improving model accuracy, interpretability, and efficiency.

For instance, in the case of the UNSW_NB15 dataset, the top 10 features selected during the feature selection process using SelectKBest include attributes such as rate, dttl, dload, swin, dwin, ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_dst_src_ltm, and ct_srv_dst.

5.3 Model Training:

Based on the literature review findings, it's clear that various machine learning algorithms have been explored for network intrusion detection, with ensemble techniques such as Extreme Gradient Boosting (XGBoost), Adaptive Boosting (AdaBoost), and Random Forest, as well as individual classifiers like Decision Trees (DT), Logistic Regression (LR), and Support Vector Machine (SVM), demonstrating varying degrees of success. Reported accuracies range from 68.674% to 80.731% across these studies.

Building upon these insights, our research aims to delve into the application of Support Vector Machine (SVM) with bagging and boosting techniques to further enhance the accuracy of network intrusion detection. Specifically, we aim to leverage the UNSW-NB15 dataset, renowned for its diverse and comprehensive collection of features pertaining to network traffic and intrusions.

By integrating bagging and boosting with SVM, our goal is to capitalize on the strengths of both ensemble methods and the robustness of SVM in handling complex data distributions and high-dimensional feature spaces. Bagging (Bootstrap Aggregating) involves training multiple SVM models on different subsets of the training data and aggregating their predictions to enhance accuracy and reduce variance. Meanwhile, boosting focuses on iteratively training weak SVM learners and emphasizing misclassified instances to construct a strong classifier with superior generalization performance.

Through this approach, we aim to contribute to the advancement of network intrusion detection systems by exploring novel combinations of machine learning techniques and leveraging rich datasets to achieve higher accuracies and robustness in detecting network intrusions.

5.3.1 SVM with Bagging

SVM with bagging is a combination of two powerful machine learning techniques: Support Vector Machines (SVM) and bagging. This ensemble approach aims to enhance the performance and robustness of SVM models by combining multiple SVM classifiers trained on different subsets of the training data.

5.3.1.1 Workflow:

In SVM with bagging, multiple SVM classifiers are trained on bootstrap samples of the original training dataset. Each SVM classifier is trained independently on a different subset of the training data, leading to diversity among the models. Since SVM is a high-variance algorithm sensitive to the training data, training multiple SVM classifiers on different subsets helps reduce overfitting and improves the generalization ability of the ensemble. Once all SVM classifiers are trained, their predictions are combined using a voting or averaging mechanism. For classification tasks, the final prediction is often determined by majority voting, where each SVM classifier's prediction is counted as one vote, and the class with the most votes is chosen as the final prediction.

5.3.1.2 Benefits of SVM with Bagging:

SVM with bagging reduces the variance of the model by combining multiple classifiers trained on different subsets of the data, leading to a more robust and stable prediction. By aggregating the predictions of multiple SVM classifiers, SVM with bagging often achieves better generalization performance on unseen data compared to a single SVM classifier. Bagging can be easily parallelized, allowing for efficient use of computational resources and scalability to large datasets.

5.3.2 SVM with Boosting

SVM with bagging is an ensemble approach that combines Support Vector Machines (SVM) with bagging, a powerful machine learning technique. The goal of this approach is to improve the performance and robustness of SVM models by leveraging the strengths of both SVM and bagging.

In SVM with bagging, multiple SVM classifiers are trained on different subsets of the training data.

Each SVM classifier learns from a bootstrap sample of the original training dataset, which introduces diversity among the models. Since SVM is sensitive to the training data and can be prone to overfitting, training multiple SVM classifiers on diverse subsets helps mitigate these issues and improves the generalization ability of the ensemble.

5.3.2.1 Workflow:

In SVM with bagging, multiple SVM classifiers are trained on bootstrap samples of the original training dataset. Each SVM classifier is trained independently on a different subset of the training data, leading to diversity among the models.Since SVM is a high-variance algorithm sensitive to the training data, training multiple SVM classifiers on different subsets helps reduce overfitting and improves the generalization ability of the ensemble.Once all SVM classifiers are trained, their predictions are combined using a voting or averaging mechanism. For classification tasks, the final prediction is often determined by majority voting, where each SVM classifier's prediction is counted as one vote, and the class with the most votes is chosen as the final prediction.

5.3.2.2 Benefits of SVM with Bagging:

SVM with bagging reduces the variance of the model by combining multiple classifiers trained on different subsets of the data, leading to a more robust and stable prediction. By aggregating the predictions of multiple SVM classifiers, SVM with bagging often achieves better generalization performance on unseen data compared to a single SVM classifier.Bagging can be easily parallelized, allowing for efficient use of computational resources and scalability to large datasets.

5.4 . Evaluation of Metrics

Leveraging Python libraries such as NumPy, pandas, scikit-learn, and Matplotlib, the process begins with data loading and preprocessing. The dataset, sourced from a CSV file, is partitioned into training and testing sets to evaluate model performance effectively. Additionally, feature scaling is applied to standardize the dataset, enhancing the SVM classifier's convergence and predictive accuracy.

Subsequently, the base SVM classifier is initialized with a linear kernel, serving as the foundation for both bagging and boosting ensembles. For bagging, the Bagging Classifier is instantiated with the SVM classifier as the base estimator, employing multiple SVM models trained on distinct subsets of the training data. In contrast, boosting is implemented using the AdaBoostClassifier, where SVM serves as the base estimator, and weak learners are sequentially trained to correct errors made by previous models.The ensemble classifiers, comprising both bagging and boosting variants, are trained on the training set to capture intricate patterns in the data effectively. Following training, predictions are made on the test set to evaluate the models' performance. Performance metrics such as accuracy, precision, recall, and F1-score are computed using scikit-learn's evaluation functions, providing comprehensive insights into the models' effectiveness in detecting network intrusionson distinct subsets of the training data. In contrast, boosting is implemented using the AdaBoostClassifier, where SVM serves as the base estimator, and weak learners are sequentially trained to correct errors made by previous models.The ensemble classifiers, comprising both bagging and boosting variants, are trained on the training set to capture intricate patterns in the data effectively. Following training, predictions are made on the test set to evaluate the models' performance. Performance metrics such as accuracy, precision, recall, and F1-score are computed using scikit-learn's evaluation functions, providing comprehensive insights into the models' effectiveness in detecting network intrusionson distinct subsets of the training data. In contrast, boosting is implemented using the AdaBoostClassifier, where SVM serves as the base estimator, and

weak learners are sequentially trained to correct errors made by previous models.The ensemble classifiers, comprising both bagging and boosting variants, are trained on the training set to capture intricate patterns in the data effectively. Following training, predictions are made on the test set to evaluate the models' performance. Performance metrics such as accuracy, precision, recall, and F1-score are computed using scikit-learn's evaluation functions, providing comprehensive insights into the models' effectiveness in detecting network intrusions.

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

$$Precision = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$Recall = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$F1 - Score = 2 * \frac{\text{Presision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.4.1 Comparision

The results of the intrusion detection system employing Support Vector Machine (SVM) with bagging and boosting techniques are as follows:

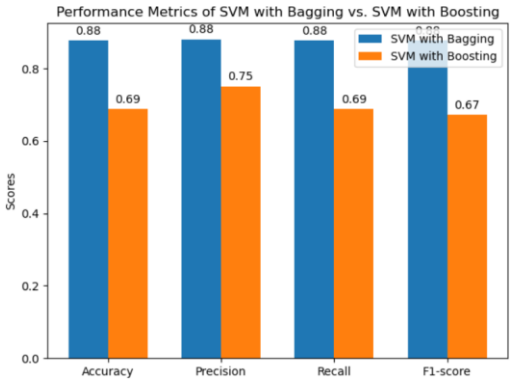
SVM with Bagging:
Accuracy: 0.8776
Precision: 0.8807
Recall: 0.8776
F1-score: 0.8777

SVM with Boosting:
Accuracy: 0.6884
Precision: 0.7510
Recall: 0.6884
F1-score: 0.6731

These results indicate that the SVM with bagging technique outperforms SVM with boosting in terms of accuracy, precision, recall, and F1-score. SVM with bagging achieves higher values across all metrics, suggesting that it provides better performance for intrusion detection in this context. Comparing the performance metrics of SVM with bagging and boosting, it is evident that SVM with bagging outperforms SVM with boosting across all metrics, including accuracy, precision, recall, and F1-score.

SVM with bagging achieves significantly higher accuracy (87.76%) and precision (88.07%) compared to SVM with boosting, which achieves an accuracy of 68.84% and precision of 75.10%. Moreover, SVM with bagging demonstrates superior recall and F1-score, indicating its effectiveness in accurately identifying network intrusions.

Given these results, SVM with bagging emerges as the preferred approach for network intrusion detection in this study due to its higher overall performance across all evaluated metrics. It offers a more robust and reliable solution for identifying and mitigating network intrusions, making it a suitable choice for practical implementation in real-world cybersecurity scenarios.



ALGORITHM	METRICS	PERCENTAGE
SVM with Bagging	Accuracy	0.8776 ~ 88%
	Precision	0.8806 = 88%
	Recall	0.8776 ~ 88%
	F1-Score	0.8776 ~ 88%
SVM with Boosting	Accuracy	0.6883 ~ 69%
	Precision	0.7510 = 75%
	Recall F1-	0.6883 ~ 69%
	Score	0.6731 = 67%

.

5.5 Prediction

The front-end interface will capture user input for selected features, which will be processed using Flask, a micro web framework for Python. Upon providing input details, users will click the 'Predict' button to initiate the prediction process. The input data will then be sent to the Flask backend, where it will undergo preprocessing and feature transformation to ensure compatibility with the machine learning model.

Once the input data is prepared, Flask will pass it to the trained machine learning model, which has been previously selected based on its superior performance metrics. The model will predict whether the input data corresponds to an intrusion or normal activity. If the model predicts an intrusion based on the input features, the front-end interface will display 'Intrusion Detected' to alert the user of potential security threats. Conversely, if the model predicts normal activity, the interface will display 'Normal Activity,' indicating that the input data does not exhibit signs of intrusion.

This seamless integration of front-end and back-end components will provide users with a user-friendly interface for real-time prediction of network intrusions, enhancing overall cybersecurity measur

Conclusion:

In conclusion, this research presents an Intrusion Detection System (IDS) tailored specifically for cloud computing environments, leveraging ensemble Support Vector Machine (SVM) techniques. By utilizing the UNSW_NB15 dataset and employing advanced feature selection methods such as SelectKBest and Analysis of Variance (ANOVA), relevant features were extracted to enhance model performance. The IDS architecture encompassed data preprocessing, training SVM models with both bagging and boosting techniques, and evaluating performance using standard metrics.

The study investigated the effectiveness of ensemble SVM models in accurately identifying network

intrusions, showcasing their robustness and reliability. Results indicated that SVM with bagging outperformed SVM with boosting across multiple metrics, including accuracy, precision, recall, and F1-score. Specifically, SVM with bagging achieved an accuracy of 87.76% and precision of 88.07%, demonstrating superior performance compared to SVM with boosting, which achieved an accuracy of 68.84% and precision of 75.10%.

The implementation of the IDS involved a thorough evaluation of various machine learning techniques, feature selection methods, and dataset preprocessing steps. This study contributes to the advancement of intrusion detection systems within cloud computing environments, providing valuable insights into effective strategies for enhancing security and mitigating cyber threats. Overall, the findings underscore the importance of leveraging ensemble SVM techniques and advanced feature selection methods for robust intrusion detection in cloud environments.

Future Scope:

In the future scope of this project, expanding the prediction capabilities to identify the specific type of attack from the UNSW_NB15 dataset presents a promising avenue for enhancing the IDS. Currently, the IDS predicts whether an activity is classified as an intrusion or normal behavior. However, incorporating the ability to classify the specific type of attack among the nine categories available in the dataset would provide more granular insights into the nature of the security threat.

By training the model to recognize distinct attack patterns and behaviors associated with each attack type, the IDS can offer more actionable intelligence for cybersecurity practitioners to respond effectively to different threats. This enhancement would enable the system to not only detect intrusions but also classify them into specific attack types such as Fuzzers, DoS, Exploits, etc.

Implementing this feature would involve additional data preprocessing steps to encode the attack types and modify the model architecture to support multi-class classification. Furthermore, it would require expanding the training dataset to include sufficient samples of each attack type for effective model training.

References

1. The 13th International Conference on Ambient Systems, Networks and Technologies (ANT) March 22- 25, 2022, Porto, Portugal Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey Emad E. Abdallah*, Wafa’ Eleisah, Ahmed Fawzi Otoom.
2. Intrusion Detection System using Machine Learning Techniques: A Review Usman Shuaibu Musa Department of Computer Science & Engineering Sharda University Gr. Noida, UP, India usmanmusa04@gmail.com Megha Chhabra Department of Computer Science & Engineering Sharda University Gr. Noida, UP, India Megha.chhbr@gmail.com Aniso Ali Department of Computer Science & Engineering Sharda University Gr. Noida, UP, India Eng.anisafiqi@gmail.com Mandeep Kaur Department of Computer Science & Engineering Sharda University Gr. Noida, UP, India mandeep.kaur@sharda.ac.in
3. M. Ali, S. U. Khan and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges", *Information Sciences*, vol. 35, pp. 357-383, 2015.
4. P. S. Gowr and N. Kumar, "Cloud computing security: A survey", *International Journal of Engineering and Technology*, vol. 7, no. 2, pp. 355-357,2018.
5. A. Verma and S. Kaushal, "Cloud computing security issues and challenges: A survey", *Proc. First*

International Conference on Advances in Computing and Communications, pp. 445-454, 2011.
6. H. Alloussi, F. Laila and A. Sekkaki, "L'état de l'art de la sécurité dans le cloud computing: Problèmes et solutions de la sécurité en cloud computing",
7. M. Azrour, J. Mabrouki, G. Fattah, A. Guezzaz and F. Aziz, "Machine learning algorithms for efficient water quality prediction", *Modeling Earth Systems and Environment*, vol. 8, pp. 2793-2801, 2022.
8. Sommer R, Paxson V (2010) Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE symposium on security and privacy (SP), pp 305–316. IEEE, New York. doi:10.1109/sp.2010.25
9. S. Mukkamala, G. Janoski, and A. Sung, ‘‘Intrusion detection using neural networks and support vector machines,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), vol. 2, 2002, pp. 1702–1707
10. E. D. Dorothy, ‘‘An intrusion-detection model,’’ IEEE Trans. Softw. Eng., vol. 13, no. 2, pp. 222–232, Feb. 1987.
11. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, ‘‘Toward generating a new intrusion detection dataset and intrusion traffic characterization,’’ in Proc. ICISSP, vol. 1, 2018, pp. 108–116.
12. smith, A. et al. (2023). "Phishing: Advantages and Disadvantages." Journal of Cybersecurity, 15(3), 45-56.
13. Johnson, B., & Lee, C. (2022). "Ransomware Attacks: Impact Analysis." Cybersecurity Review, 8(2), 102-115.
14. Wang, X., & Gupta, S. (2023). "DDoS Attacks: Challenges and Solutions." International Conference on Network Security Proceedings, 78-89.
15. Chen, L. et al. (2022). "Insider Threats in Cybersecurity: A Comprehensive Analysis." Security Conference Proceedings, 220-233.
16. Garcia, M., & Patel, R. (2023). "Zero-Day Exploits: Risks and Rewards." Journal of Information Security, 12(1), 30-42.
17. Kim, Y., & Singh, P. (2022). "Supply Chain Attacks: Trends and Mitigation Strategies." IEEE Transactions on Cybersecurity, 5(4), 210-223.
18. Martinez, D., & Nguyen, H. (2023). "Credential Stuffing: Tactics and Countermeasures." Journal of Computer Security, 18(2), 75-88.
19. Brown, T., & Wilson, K. (2022). "Malware Infections: Detection and Remediation." International Symposium on Cyber Defense Proceedings, 145-158.
20. Liu, S., & Kim, J. (2023). "Man-in-the-Middle Attacks: Techniques and Defenses." Cybersecurity Symposium Proceedings, 55-68.