UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Group Project Final Report

Performance of Machine Learning Models in Detecting Web Attacks

Members

Nguyễn Quang Vinh	BA11-103
Đặng Đức Anh	BA11-002
Lại Hải Anh	BA11-004
Nguyễn Ngọc Linh	BA11-062
Chu Văn Nam	BA11-076

Supervisor

Dr. Nguyễn Minh Hương,

ICT Department, USTH

January 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Acknowledgment

The completion of this project was made possible by the tremendous support and assistance we received throughout its duration. We would like to express our sincere gratitude to the University of Science and Technology of Hanoi (USTH) as well as all the respected professors at the institution, especially Dr. Nguyen Minh Huong, who played a critical role in guiding us through this venture.

Dr. Nguyen Minh Huong provided invaluable guidance that helped us navigate various stages of this project within the designated timeline. Despite not having formal instruction on Machine Learning during our university studies, she offered unwavering support in equipping us with essential knowledge related to artificial intelligence and machine learning concepts. It is thanks to her help that we were able to complete both the project itself and its accompanying report.

Certainly, there were obstacles along the way; however, determination led us to ultimate triumph. This endeavor has greatly enhanced our understanding of data analysis techniques, machine learning models, and practical implementation methods for artificial intelligence tools.

Abstract

An Intrusion Detection System (IDS) is an important solution in a network to detect intrusion and malicious activity. An IDS can be integrated with Artificial Intelligence/Machine Learning to detect new malicious signatures and it can use various models of machine learning. In this project, we aim to find the most suitable models for detecting web attack traffic. We have analyzed a public dataset and applied various methods to divide it into sets that then have been used for training and testing the four accordant models for our project including K-nearest Neighbor, Decision Tree, Random Forest, and Support Vector Machine. Then, we performed two experiments that are binary classification and multiple classification with these models with different ratios of training set and testing set. Lastly, we tried to figure out the metrics such as accuracy, precision, recall, and F1 score from the confusion matrix of each model in these experiments that then were used to compare to find out the most usable model to detect web attack traffic.

1. Introduction

The current state of cybersecurity is characterized by an increasing frequency and complexity of cyber-attacks, highlighting the crucial role played by Intrusion Detection Systems (IDS) in safeguarding network security. Although traditional IDS methods have been successful in identifying known threats, they face a significant challenge when it comes to detecting novel and unforeseen attacks that exploit zero-day vulnerabilities.

The limitations of conventional approaches are due to the inherent nature of zero-day vulnerabilities. Since these vulnerabilities are unknown to vendors, there is a considerable delay before patches can be developed and distributed. This creates a vulnerable window for attackers to exploit without being detected or prevented. Consequently, there is an urgent need for a more adaptive and proactive approach to intrusion detection.

Artificial Intelligence (AI) emerges as a powerful solution to this predicament. By incorporating AI into IDS systems, the process of detection can be automated, enabling systems to learn from patterns and anticipate potential threats associated with zero-day vulnerabilities. The ability of AI technologies to handle vast amounts of data in real time becomes indispensable in the constantly changing digital landscape.

However, integrating AI into IDS introduces another challenge: selecting the most suitable machine learning model for detecting abnormal network traffic. With numerous models available, cybersecurity practitioners find themselves faced with a complex decision-making process. Each model possesses unique advantages as well as limitations, making it necessary to conduct comprehensive evaluations to identify the optimal solution.

This project aims to address this critical problem through a series of experiments involving multiple machine-learning models. The objective is to thoroughly analyze and compare the performance of these models in detecting web attack traffic. By examining metrics such as accuracy, precision, recall, and F1 score, we aim to determine which model not only performs exceptionally well under controlled experimental conditions but also demonstrates practical effectiveness within dynamic real-world network environments.

In conclusion, this introduction emphasizes the evolving landscape of cyber threats along with the shortcomings associated with traditional IDS methods while also recognizing the transformative potential that AI holds for intrusion detection. The subsequent exploration of various machine learning models aims to navigate the intricate terrain of model selection, contributing valuable insights to the ongoing quest for robust and adaptive cybersecurity solutions.

2. Background

2.1. Definition of IDS

According to Jabez et al. (2014) [8], an Intrusion Detection System (IDS) is a software application or device that monitors network activities to identify instances of policy violations or malicious actions. Its main purpose is to generate reports for management rather than focusing on preventing intrusions. In contrast, the primary goal of Intrusion Detection and Prevention Systems (IDPS) is to recognize potential incidents, log relevant information, and report attempted breaches. IDPS serves multiple functions such as identifying security policy issues, deterring individuals from unauthorized access attempts, and documenting existing threats to protect policies.

IDPS has become an integral part of organizational security frameworks due to its wide integration across various industries. It employs different detection methods tailored specifically towards the type of attack vectors being monitored with emphasis on efficient and early detection to minimize any negative impacts.

When it comes to classifying IDSs based on their functionality, they are typically categorized into three types: Network-based IDS (NIDS), Host-based IDS (HIDS), and Distributed IDS (DIDS). Each type focuses on monitoring either the network traffic itself or individual host systems within the network environment.

On the other hand, when classifying IDSs based on their detection technique there are two distinct categories: Signature-based IDSs and Anomaly-based IDSs. Signature-based IDS relies heavily on predefined patterns or signatures associated with known attacks to detect suspicious activity accurately.

In this project's context however, we will be integrating AI/ML models into an Anomaly-Based ID System that utilizes advanced algorithms capable of recognizing abnormal behavior by establishing baselines through continuous learning processes thus enabling effective identification of previously unseen threats while minimizing false positives.

2.2. Components of IDS

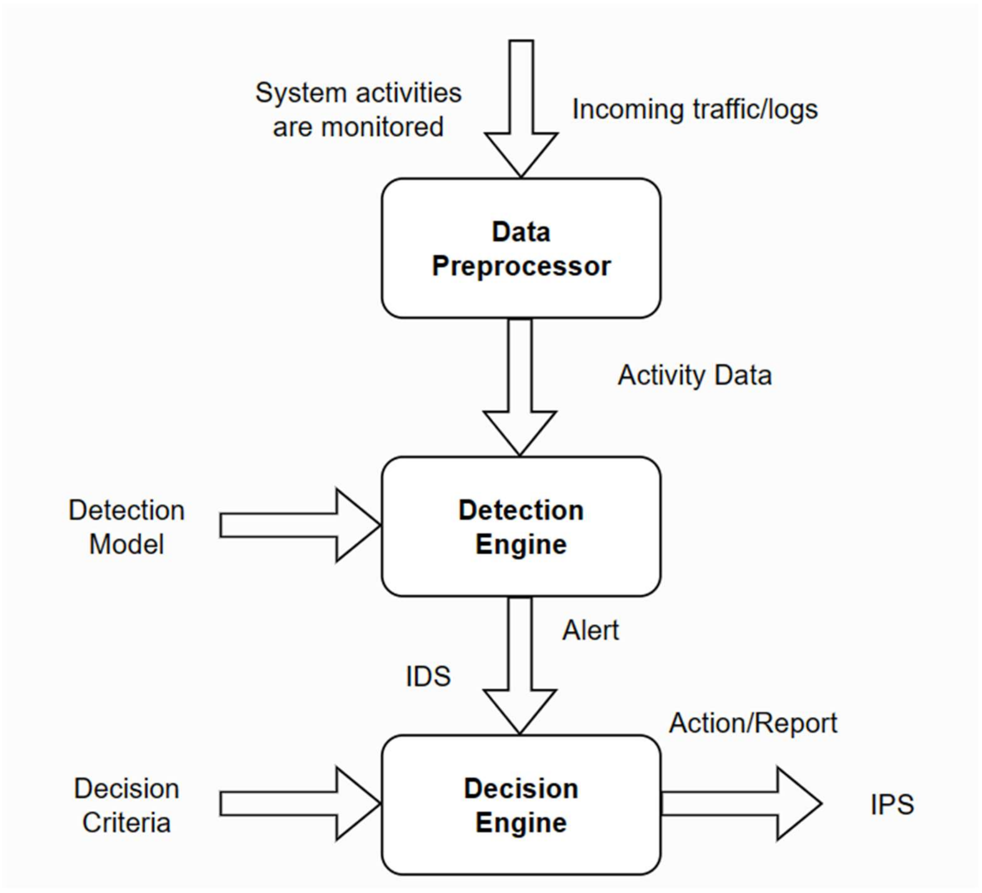

Figure 2-1: Components of Intrusion Detection System

There are 3 main components:

- Data Preprocessor: collects the information on the stream of monitored events.
- Detection Engine: analyze all collected packets to indicate which actions are attacks.
- Decision Engine: responses based on the outcome from the detection engine.

Within an Intrusion Detection System (IDS), valuable details about the actions occurring within a system and network are collected as input. This data undergoes processing by a specialized processor, which extracts important activity records necessary for analyzing security. The IDS houses pre-built detection models that examine this activity information through its detection engine. Should any intrusion be identified according to specific rules, the IDS promptly generates an alert signalizing potential threats. Subsequently, the decision-making process is initiated concerning a decision table to determine the most suitable course of action moving forward.

2.3. Anomaly-based IDS

There are two main approaches used in Intrusion Detection Systems (IDS) to detect potential threats and intrusions: Anomaly-based detection and Signature-based detection. Unlike signature-based detection, which relies on a database of predefined signatures, the Anomaly-based IDS (AIDS) is designed to identify abnormal user activity without using pre-established patterns. When observed behavior differs from normal behavior, AIDS sends out an alert or warning signal.

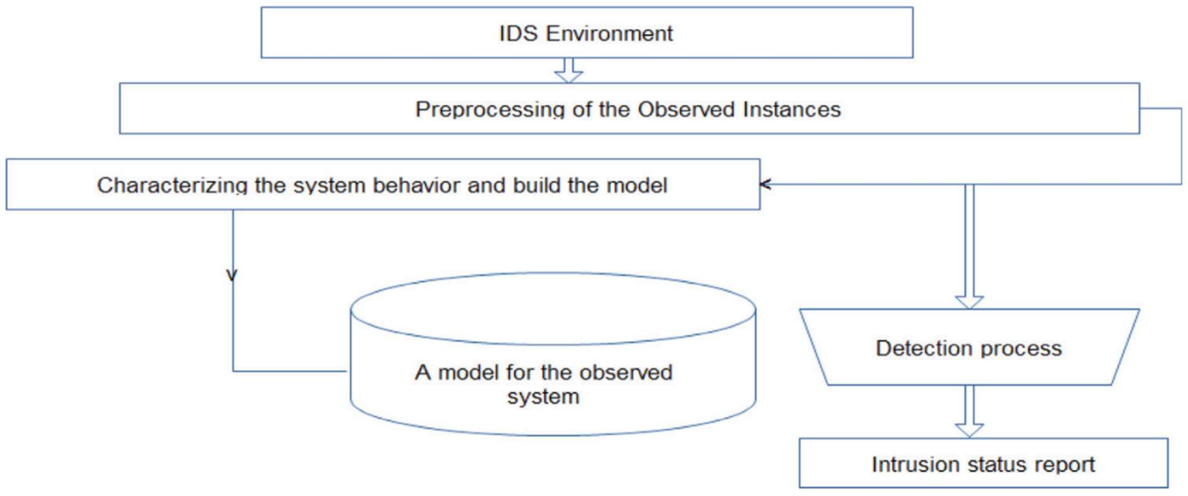
IDS Environment
Preprocessing of the Observed Instances
Characterizing the system behavior and build the model
A model for the observed system
Detection process
Intrusion status report

Figure 2-2: Anomaly-based Intrusion Detection System

In the study conducted by Jyothsna et al. (2019) [9], it was found that Anomaly-based Intrusion Detection Systems (AIDS) follow a sequential process consisting of various functional stages. Initially, attributes are created through preprocessing based on the unique characteristics of the target system. Following this, in the observation stage, a model is generated using behavioral features specific to the system under consideration. This allows for intrusion observations to be made through either automated or manual detection methods. Finally, we arrive at the functional stage known as detection where a comparison is made between observed network traffic and an existing system model to identify any anomalies present.

One advantage of AIDS is its ability to detect zero-day attacks and uncover internal malicious activities effectively making it difficult for cybercriminals to evade detection by relying on customized profiles depicting normal behavior patterns. However, there is also a drawback associated with AIDS which lies in its tendency towards yielding false positives as occasionally new normal activities may be mistakenly flagged as genuine intrusions rather than being correctly identified.

The development process of AIDS generally comprises two main phases: training and testing. In the training phase, a model representing normal behavior is constructed utilizing data from regular traffic profiles while considering all relevant factors affecting their occurrence within this context.

Subsequently comes into play during the testing phase where fresh datasets are utilized to evaluate how well previously unseen intrusions can still be accurately detected and classified by our IDS.

3. Materials and Methods

3.1. Data Preparation

In this project, we use the CICIDS2017 dataset for training and testing in our experiments. This dataset was published by Iman Sharafaldin et al. [1] and it holds significance in the realm of cybersecurity as it has been specifically curated for assessing the efficacy of Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs). It encompasses a diverse collection of benign traffic patterns alongside prevalent attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS.

The five-day data capture period occurred during July 2017 wherein approximately 50GB worth of information was accumulated. This included both PCAP files consisting of raw packet-level data and structured CSV files that provided labeled flow data with intricate details like timestamps, source/destination IPs, ports used, protocols employed, and attack classifications.

For our project's scope, however, we exclusively focused on utilizing the CSV file about web-based attacks transpiring solely on Thursday mornings.

	Normal	Brute-force attack	XSS attack	SQL injection attack
No. of packets	168186	1507	652	21

Table 3-1: Structure of the Dataset

Table 3-1 present the structure of the CSV file with the number of packets in each type of attack compared with the number of normal packets.

3.2. Feature Selection

The training dataset used in this study was carefully labeled by experts and scholars. To extract network traffic features, the researchers employed two tools: CICFlowMeter (CICFlowMeter, 2017) [11] and Habibi Lashkari et al.'s method from 2017 [12]. These tools allowed them to derive a total of 80 features from each Pcap file. To identify the most effective feature set for detecting different attacks among these extracted features, they utilized the RandomForestRegressor class available in sci-kit-learn (Pedregosa et al., 2011) [13]. They thoroughly evaluated the significance of each feature across the entire dataset.

To obtain their result, they calculated an average standardized mean value for each feature within its respective attack class and multiplied it by its corresponding importance value. By doing so, they were able to determine which common features were most influential in labeling instances as specific types of attacks.

According to Iman Sharafaldin et al.'s paper titled "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," presented at Portugal's 4th International Conference on Information Systems Security and Privacy (ICISSP), January 2018 edition [1], some notable web attack-related characteristics include Init Win Forward Bytes, Subflow Forward Bytes, Init Win Backward Bytes, and Total Len Forward Packets.

3.3. Models Selection

3.3.1. Supervised Machine Learning Models

In supervised learning, the model undergoes training by utilizing a dataset that is labeled. This dataset contains both input and output parameters. The labeled dataset employed in supervised learning comprises input features as well as corresponding output labels. Input features refer to the attributes or characteristics of data utilized for making predictions, while output labels represent desired outcomes or targets aimed at being predicted by algorithms.

A key advantage associated with supervised learning lies in its ability to generate intricate models capable of delivering accurate predictions on novel data points. Nevertheless, large quantities of accurately labeled training data must be available for this approach to yield optimal results. Moreover, the accuracy of the model can be significantly influenced by factors such as the quality and representative nature exhibited within the training data set itself.

3.3.2. Models Selection

3.3.2.1. K-nearest Neighbor model

The K-nearest Neighbor algorithm, also known as KNN, aims to estimate the category or numerical value of a fresh data point by considering its closest neighbors within the training dataset. [5] The appropriate selection of K is contingent upon both the specific characteristics of the data and the nature of the problem at hand. Techniques such as cross-validation are commonly employed to ascertain an optimal value for K that leads to accurate predictions. Furthermore, it should be noted that different distance metrics may be utilized to assess similarity between various data points; these metrics can vary depending on factors like feature type and scale. The distance metric used in the K-NN algorithm is Minkowski distance and the p = 2 by default.

Minkowski distance: $d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$

3.3.2.2. Decision Tree model

Supervised learning algorithms known as decision trees can tackle classification and regression problems. [5] Their operation involves acquiring elementary decision rules from data features, which are subsequently applied to new data points.

To enhance performance, hyperparameter tuning is conducted by adjusting algorithm parameters like maximum tree depth, minimum sample requirement for node splitting, or the split quality criterion.

Pruning comes into play to eliminate redundant or overfitting branches from the tree structure to streamline complexity and boost generalization capabilities.

3.3.2.3. Random Forest model

Random forests are a type of ensemble learning method that combines multiple decision trees to produce a more accurate and robust prediction. [5] Each decision tree in the forest is trained on a random subset of the data and a random subset of the features, to introduce diversity and reduce correlation among the trees.

The number of trees in the forest is a hyperparameter that can be tuned to optimize the performance of the algorithm, as well as other parameters related to the individual trees, such as the maximum depth, the minimum samples per leaf, or the splitting criterion.

The rationale for choosing random forests over individual decision trees is that they can reduce the variance and overfitting of the single trees, by averaging their predictions and exploiting the wisdom of the crowd.

3.3.2.4. Support Vector Machines model

Support Vector Machines (SVMs) represent a type of supervised learning algorithm that proves valuable in solving classification problems. These algorithms excel when the dataset possesses either linear separability or high dimensionality. SVMs operate by seeking the optimal hyperplane, which maximizes the separation between various classes' data points—a margin extending to each class's nearest neighbor. [5]

To enhance linearity within datasets, kernel functions come into play. They alter input data and project it onto higher-dimensional feature spaces—spaces where enhanced linear separability may arise for improved performance. The selection of a suitable kernel function substantially influences an SVM's effectiveness since distinct kernels can capture diverse patterns and relationships present within the dataset.

Linear, polynomial, radial basis function (RBF), and sigmoid stand as prominent examples of commonly employed kernel functions with varying impacts on SVM outcomes. The regularization parameter furnishes another essential component governing an SVM's accuracy versus complexity trade-off: this parameter penalizes models incorporating numerous support vectors or those boasting substantial margins. Inaccurate tuning toward small values could result in overfitting scenarios while excessively large ones might lead to underfitting situations.

3.4. Confusion Matrix

3.4.1. Definition of Confusion Matrix

In the field of machine learning, specifically in statistical classification problems, a confusion matrix, also known as error matrix [14], is an organized table that offers a visual representation of how well an algorithm performs. This type of matrix is typically used in supervised learning scenarios and referred to as an error matrix or matching matrix for unsupervised learning situations.

The structure of the confusion matrix involves rows representing instances belonging to actual classes and columns indicating instances assigned to predicted classes. Both variations are present within relevant literature [15]. The name "confusion" stems from its ability to clearly show if the system mistakenly confuses two different classes by frequently mislabeling one as another.

This kind of contingency table consists of two dimensions - "actual" and "predicted." Additionally, it possesses identical sets of "classes" along both dimensions. Each combination formed between dimension and class serves as a variable within this contingency table framework.

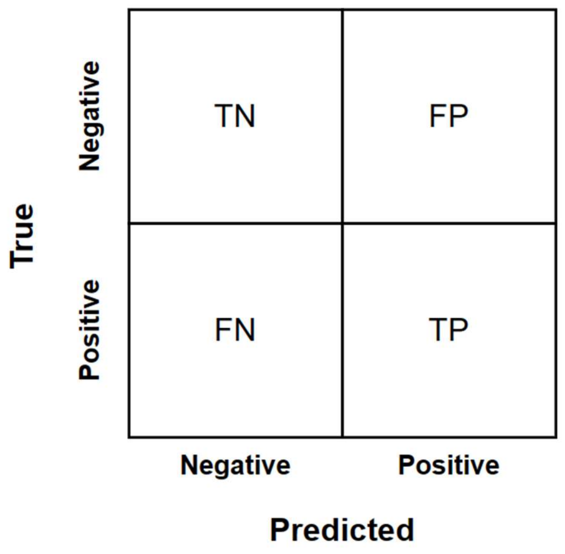

Figure 3-1: Binary Confusion Matrix

Figure 3-1 present an example of binary confusion matrix. The matrix displays the number of instances produced by the model on the test data.

- True positives (TP): occurs when the model accurately predicts a positive data point.
- True negatives (TN): occurs when the model accurately predicts a negative data point.
- False positives (FP): occurs when the model predicts a positive data point incorrectly.
- False negatives (FN): occurs when the model predicts a negative data point incorrectly.

3.4.2. Common Metrics

Metrics are quantitative indicators that are employed to evaluate different facets of a system, operation, or achievement. In the context of artificial intelligence and statistical analysis, metrics hold immense significance as they enable us to assess the effectiveness and performance of models, algorithms, and forecasts in an impartial and standardized manner. In this project, we consider four common metrics that are accuracy, precision, recall, and F1 score:

- Accuracy is determined by dividing the total number of correct instances by the overall number of instances under consideration.
$$Accuracy = \frac{Total\ correct\ instances}{Total\ instances} = \frac{TP + TN}{TP + TN + FP + FN}$$
- Precision is a metric that gauges the level of accuracy in a model's positive predictions. It can be calculated by dividing the number of true positive predictions with the total count of positive predictions made by the model.
$$Precision = \frac{TP}{TP + FP}$$
- Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive instances to the sum of true positive and false negative instances.
$$Recall = \frac{TP}{TP + FN}$$
- F1-score is used to evaluate the overall performance of a classification model. It is the harmonic means of precision and recall.
$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall}$$

3.5. Overfitting and Underfitting

3.5.1. Overfitting

Overfitting occurs when a model becomes excessively familiar with the training data, going as far as to memorize insignificant details or arbitrary fluctuations present in the data. [16] As a result, although the model demonstrates exceptional performance on the specific dataset it was trained on, its accuracy significantly diminishes when faced with new and unseen information.

Indicators of overfitting:

- Discrepancy between high training accuracy and low testing accuracy: The model achieves impressive results when evaluated against its training set but fails to deliver satisfactory outcomes for novel datasets.
- Excessive complexity: The model might exhibit an unnecessary level of intricacy due to an abundance of parameters or features relative to the available amount of training data.
- Inadequate generalization ability: When presented with fresh and unfamiliar instances, the model struggles to apply what it has learned from previous experience effectively.

3.5.2. Underfitting

When a model lacks the complexity to capture the true pattern of data, it is said to be underfitting. [16] Its ability to learn from training data and make accurate predictions on both familiar and unfamiliar data is compromised.

Indicators of underfitting:

- Insufficient accuracy in training and testing: The model exhibits subpar performance when faced with both the known datasets used for training as well as new, unseen datasets.
- Simplicity or inadequacy of the model: It could be that there are too few variables or features within the model's framework, rendering it incapable of grasping the intricate nature of underlying patterns present in the data.
- Failure to identify crucial patterns: Significant trends or patterns inherent in the dataset can elude an underfitted model, causing it to overlook critical information.

4. Implementation

4.1. Data Splitting

In this project, we utilized a comprehensive dataset consisting of 170,366 packets. Among these packets, there were 2,180 attack packets present (with an 8:1 ratio compared to normal ones). To gain further insights into the impact of different ratios of normal and attack packets in the training set, we divided our dataset into three distinct cases for our experiments.

Furthermore, we conducted two separate experiments based on classification types. In the initial experiment, our objective was to train models that could accurately differentiate between normal and abnormal packets using binary classification methods. This entailed teaching the models how to distinguish between regular network traffic and potentially harmful or suspicious activity.

The subsequent experiment focused on multiple classification tasks where the aim was to equip models with skills for categorizing incoming packets according to specific types of attacks, they might contain including Brute-force attacks, XSS attacks, and SQL injection attacks. By doing so, we sought to enhance their ability to detect various forms of threats within network data.

Overall, this research involved a thorough examination of packet datasets through diverse experimental approaches such as varying training set ratios and conducting both binary and multiple classifications. This enabled us to gain valuable insights into distinguishing between normal and attack packets and detecting various attacks from diverse packet types respectively.

Experiment 1:

	No. normal packets	No. abnormal packets
Training set	12000/7200/3600	1200
Testing set	1000	980

Table 4-1: Data Splitting for experiment 1

Experiment 2:

	No. normal packets	No. brute force packets	No. XSS packets	No. SQLi packets
Training set	12000/7200/3600	840	350	10
Testing set	1000	667	302	11

Table 4-2: Data Splitting for experiment 2

Tables 4-1 and 4-2 present the distribution of each packet type within the datasets, showcasing how the ratio between normal and attack packets evolves as we modify the number of normal packets in the training set. This dynamic exploration allows us to discern the models' performance across different scenarios and sheds light on their adaptability to changing training set compositions.

4.2. Environment

Our project has strategically established an environment that optimizes the potential of Visual Studio Code (VSCode) in combination with Jupyter Notebooks for AI model development and training. VSCode, a versatile and widely utilized integrated development environment (IDE), serves as our primary platform. By utilizing the Python extension for VSCode, we can effortlessly develop Python code while benefiting from essential features like linting and IntelliSense. To effectively handle dependencies, we have implemented a virtual environment within this setup. Activating this virtual environment guarantees a clean and isolated workspace dedicated to our project's needs. To facilitate interactive data analysis, we have also installed the Jupyter package which enables convenient execution of Jupyter notebooks. This integration empowers us to seamlessly write code, execute it, and document our analyses all within one unified space. As a result, our workflow is optimized for efficiency while promoting collaboration among team members. With careful attention given to configuring this comprehensive environment, our goal is to maximize productivity when developing and evaluating machine learning models designed specifically for web attack detection purposes.

5. Results and Discussion

5.1. Results

5.1.1. Experiment 1

Ratio 10:1

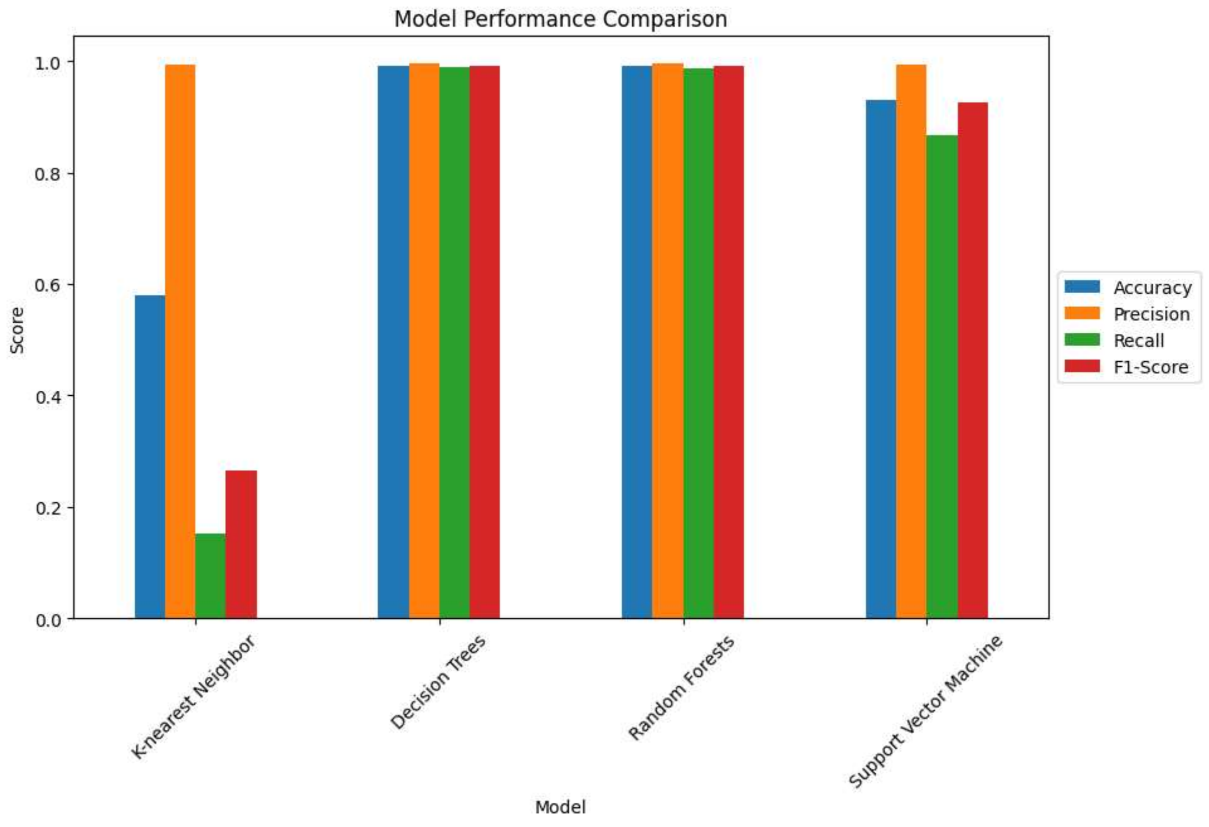

Figure 5-1: Models Performance Comparison of experiment 1 – ratio 10:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.580303	0.993377	0.153061	0.265252
DT	0.991414	0.994861	0.987755	0.991295
RF	0.990404	0.994851	0.985714	0.990261
SVM	0.930808	0.992982	0.866327	0.925341

Table 5-1: Models Performance Comparison of experiment 1 – ratio 10:1

Ratio 6:1

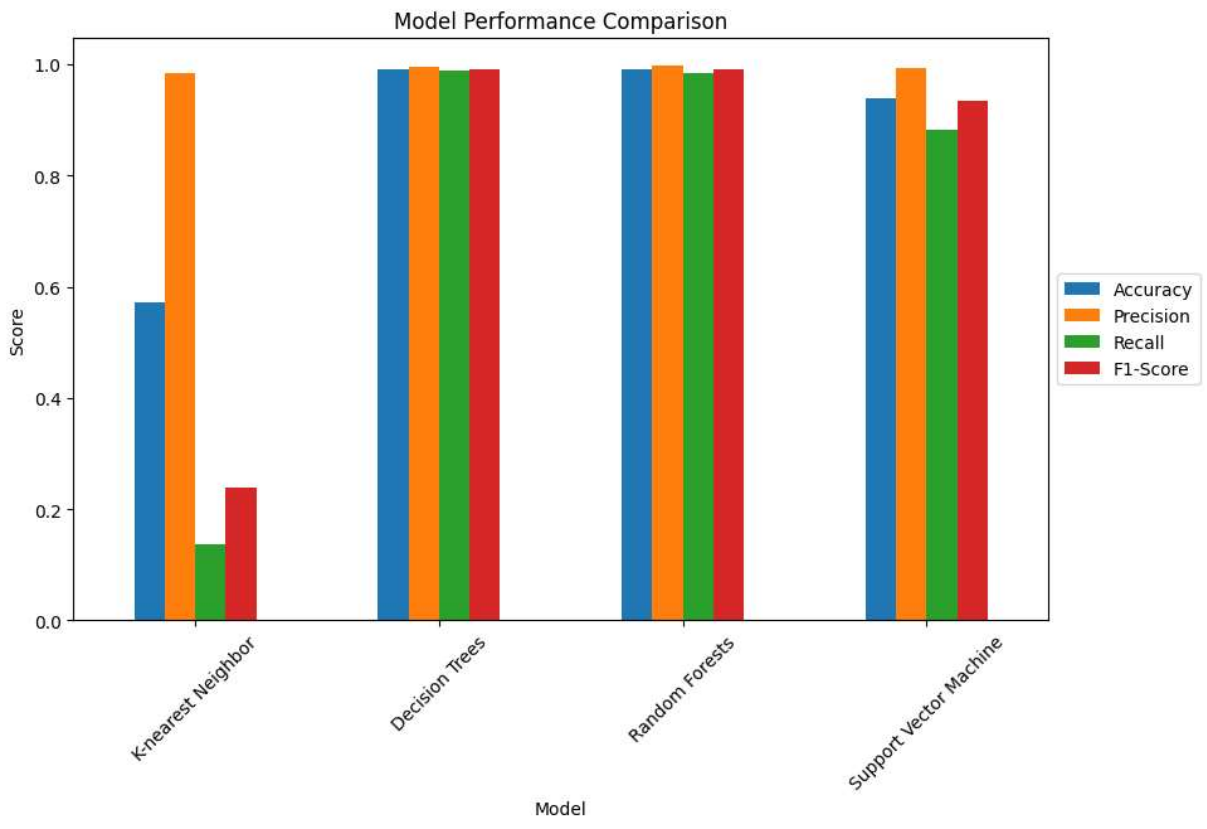

Figure 5-2: Models Performance Comparison of experiment 1 – ratio 6:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.571212	0.985185	0.135714	0.238565
DT	0.991919	0.994867	0.988776	0.991812
RF	0.990404	0.996898	0.983673	0.990241
SVM	0.939394	0.994253	0.882653	0.935135

Table 5-2: Models Performance Comparison of experiment 1 – ratio 6:1

Ratio 3:1

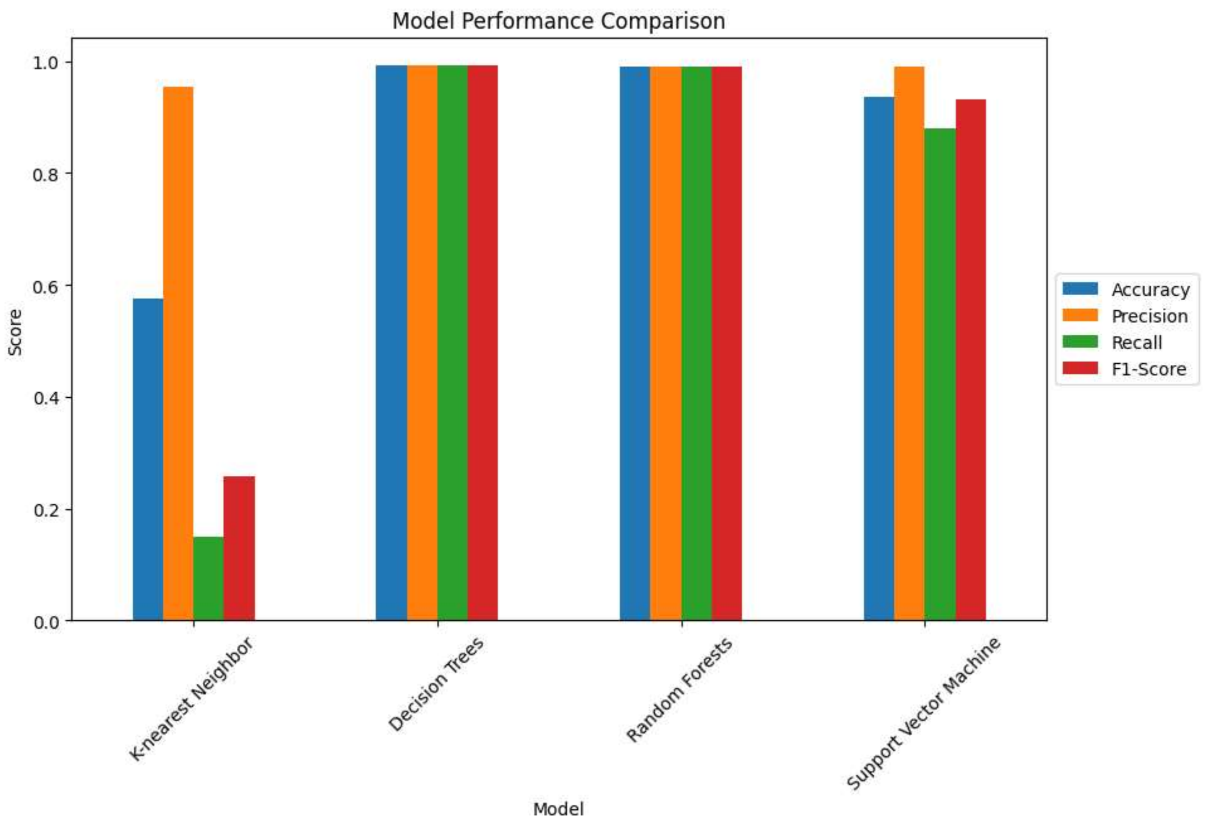

Figure 5-3: Models Performance Comparison of experiment 1 – ratio 3:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.575253	0.954248	0.148980	0.257723
DT	0.991919	0.991837	0.991837	0.991837
RF	0.990404	0.989806	0.990816	0.990311
SVM	0.936869	0.990815	0.880612	0.932469

Table 5-3: Models Performance Comparison of experiment 1 – ratio 3:1

5.1.2. Experiment 2

Ratio 10:1

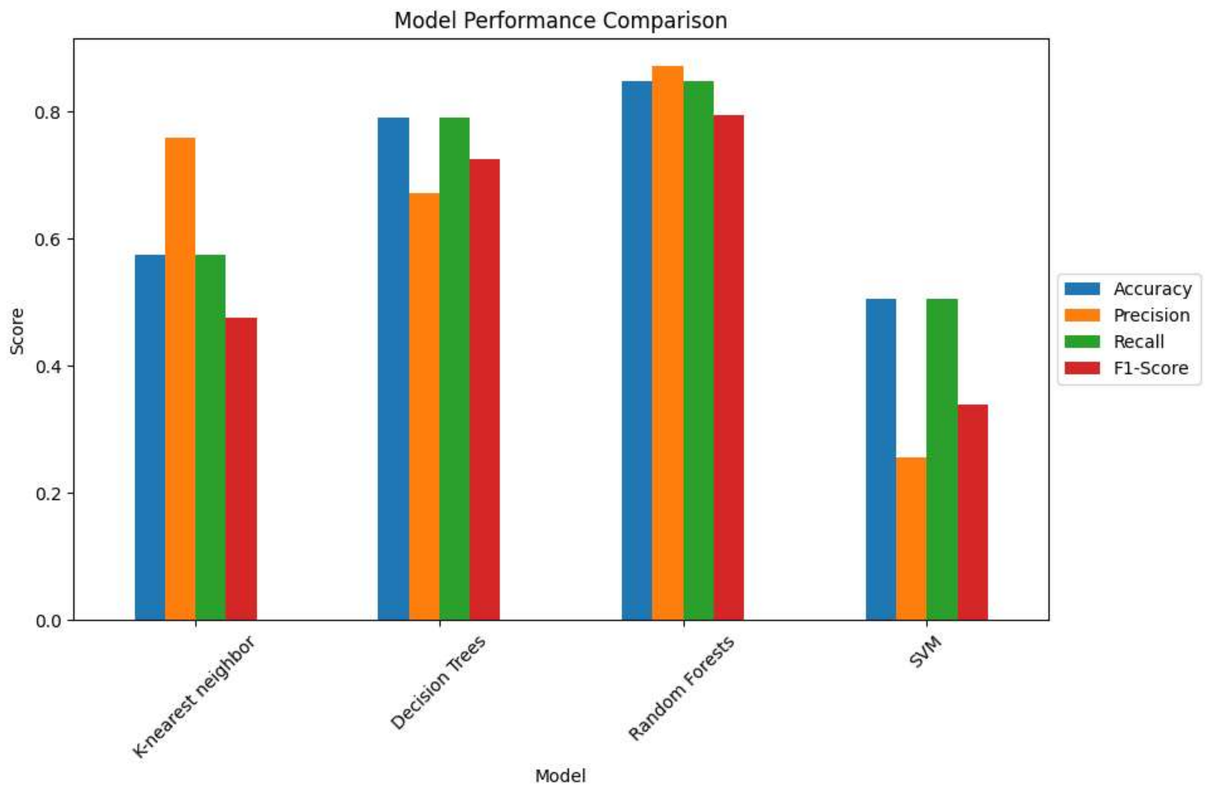

Figure 5-4: Models Performance Comparison of experiment 2 – ratio 10:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.574747	0.759242	0.574747	0.474786
DT	0.790909	0.672585	0.790909	0.726156
RF	0.848990	0.871618	0.848990	0.794700
SVM	0.505051	0.255076	0.505051	0.338960

Table 5-4: Models Performance Comparison of experiment 2 – ratio 10:1

Ratio 6:1

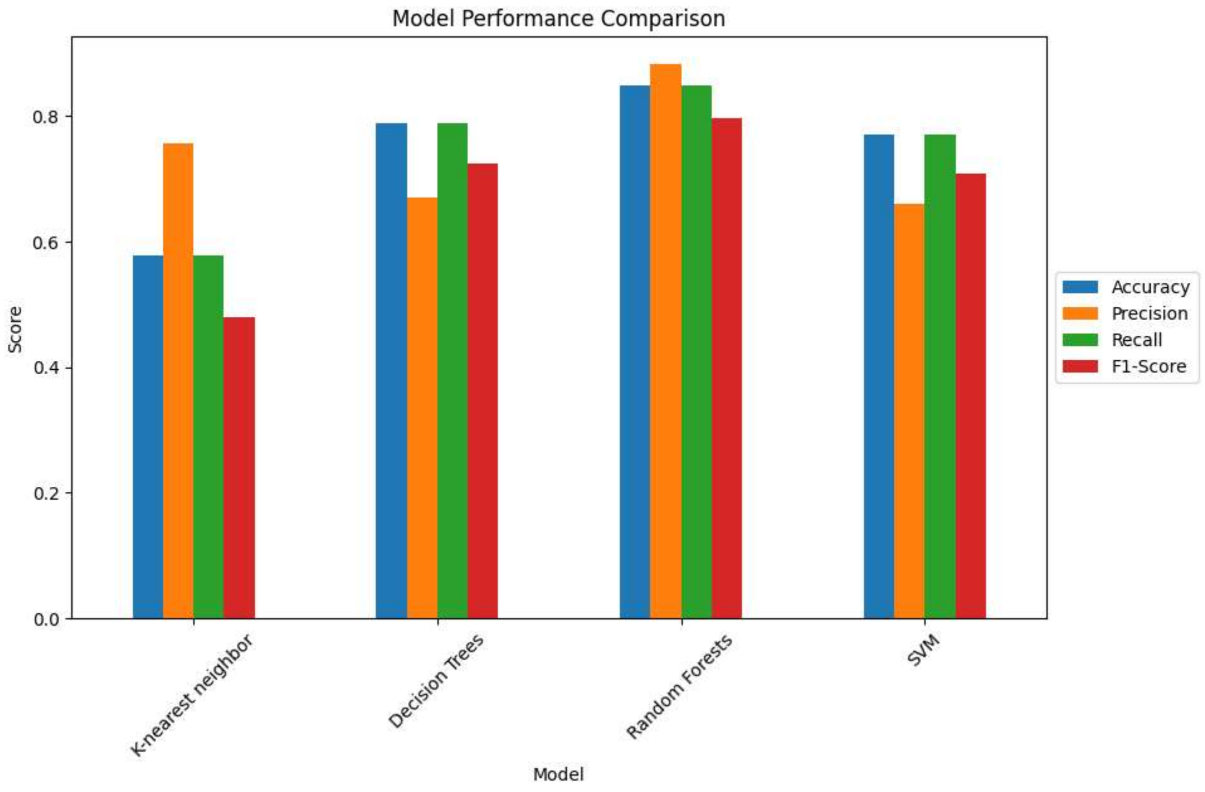

Figure 5-5: Models Performance Comparison of experiment 2 – ratio 6:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.576768	0.756152	0.576768	0.479817
DT	0.788889	0.670557	0.788889	0.724166
RF	0.847980	0.882053	0.847980	0.796093
SVM	0.770202	0.659610	0.770202	0.708798

Table 5-5: Models Performance Comparison of experiment 2 – ratio 6:1

Ratio 3:1

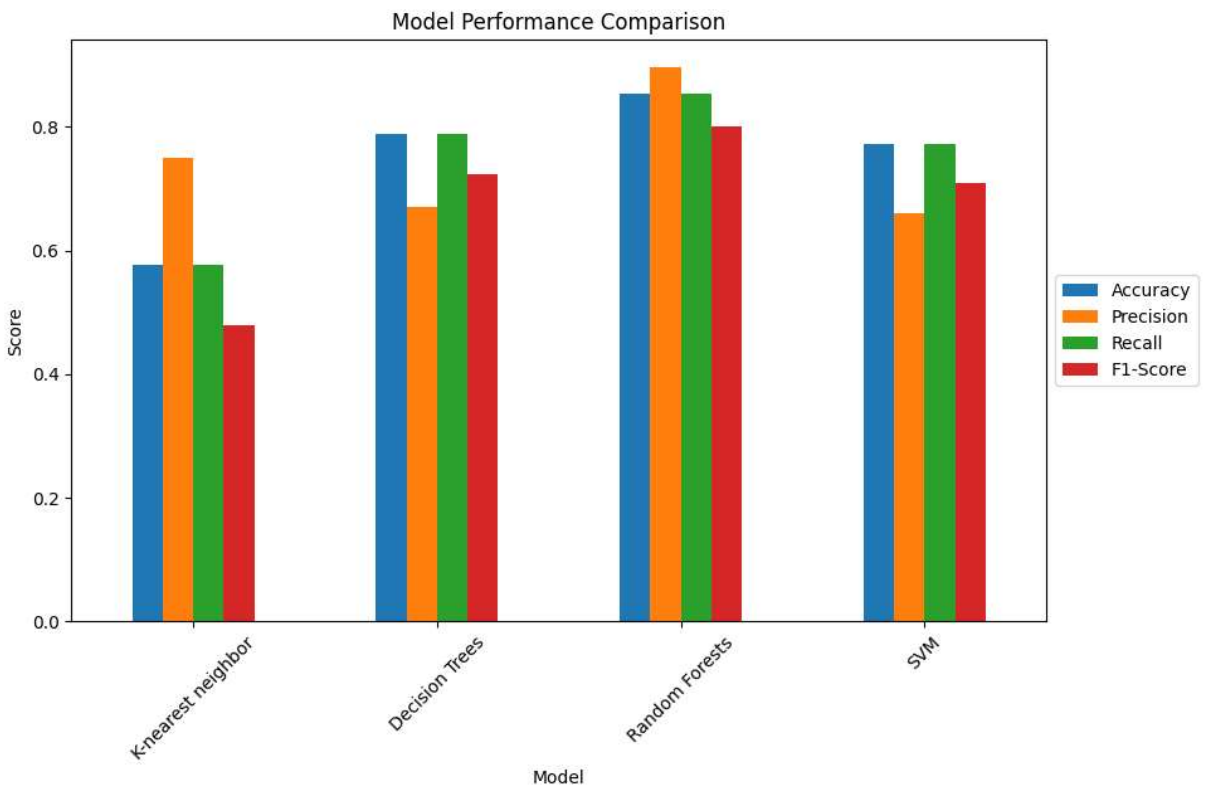

Figure 5-6: Models Performance Comparison of experiment 2 – ratio 3:1

Model	Accuracy	Precision	Recall	F1-Score
KNN	0.576263	0.748914	0.576263	0.478094
DT	0.788384	0.669857	0.788384	0.723612
RF	0.853030	0.896174	0.853030	0.800182
SVM	0.771717	0.659891	0.771717	0.709872

Table 5-6: Models Performance Comparison of experiment 2 – ratio 3:1

5.1.3. Fail Experiments

During the initial stage of our project, we faced a significant obstacle while carrying out our first experiment. The difficulty arose from the selection of datasets and resulted in an unforeseen consequence known as overfitting within our machine-learning models. The reason for the problem lies in an uneven ratio between normal and abnormal instances found within the chosen datasets. Due to this disproportionate prevalence of normal instances, it unintentionally skewed how our models learned, causing them to heavily lean towards predicting the majority class. This phenomenon of overfitting hampered their ability to generalize when encountering new data, making their outcomes unreliable. Realizing the importance of having a well-balanced dataset, we have since adjusted our approach by ensuring a fairer distribution between normal and abnormal instances for future experiments. We aim to foster more robust evaluations that yield meaningful results.

5.1.3.1. Recommendation and Mitigation

It is crucial to address the issue of dataset correction by revisiting how the data is split and rectifying any mistakes. This can be done by ensuring that both normal and abnormal instances are distributed evenly, which will result in a more precise evaluation of the model's performance. After properly splitting the dataset, it is necessary to conduct a comprehensive reevaluation of how well the models perform. It is important to focus on various metrics such as Precision, Recall, and F1-Score to gain a deeper understanding of their capabilities and limitations.

5.1.3.2. Lessons Learned

The significance of being meticulous in experimental design is highlighted by this occurrence, emphasizing the need for vigilance. Particularly during the crucial stage of dataset preparation and splitting, it is essential to ensure thoroughness. When interpreting results, one must acknowledge the potential influence that irregularities within datasets can have and exercise caution when concluding experiments conducted with compromised configurations.

5.2. Discussion

5.2.1. Key Observation

To accurately classify network abnormalities, special parameters are needed for each type of attack. This necessitates the use of recall, precision, and f1-score metrics to evaluate the effectiveness of models in detecting abnormal network traffic.

Recall plays a crucial role in network traffic analysis as it measures how well a model identifies actual positive results. A high recall score indicates that the model is adept at detecting most anomalies present in traffic. Detecting these unusual or malicious activities is essential since failure to do so can result in severe damage. Therefore, an evaluation based on high recall becomes imperative for comprehensive threat detection capabilities.

Precision, on the other hand, calculates the proportion of cases classified by a model as anomalous which are true positives. This means that when a model labels something as suspicious or abnormal, there is a higher likelihood that it truly is so. High precision helps minimize false alarms which consume valuable resources and disrupt operations.

The F1-score acts as a middle ground between precision and recall by considering both false positive (normal requests labeled as abnormal) and false negative (unusual requests labeled as normal) results into account – especially important when dealing with uneven distributions between normality and abnormality within data sets. A high F1 score indicates a balance between precision and recall while also highlighting how effective the model is at identifying anomalies.

Achieving equilibrium among these metrics holds great significance within network security were having high recall but low precision leads to excessive false alarms; conversely, if one has high precision but low recall, threats may be missed by IDS systems altogether resulting in potential harm. Additionally, there exists other noteworthy metrics such as Specificity (True Negative Rate), AUC (Area Under ROC Curve), and Matthews Correlation Coefficient (MCC).

For our experiment purposes, we chose Precision, Recall, and F-Score due to their wide usage across various machine learning domains. These chosen metrics prove suitable not only for binary classification problems alone but multiclass ones too, as well as being easily interpretable and understood. They offer valuable insights into different aspects of model performance.

5.2.2. Evaluation

5.2.2.1. Experiment 1

After conducting a thorough examination of the outcomes obtained from our experiment, it is evident that the four models we employed (KNN, Decision Trees, Random Forests, and SVM) consistently demonstrated satisfactory performance. This remained consistent even when subjected to varying ratios (10:1, 6:1, and 3:1) in the binary classification experiment. From this observation alone, several potential factors can be identified as possible explanations for such reliability:

- Imbalance Insensitivity: Machine learning models demonstrate a reduced susceptibility to imbalanced class distribution when it comes to binary classification. In situations like these, the performance of machine learning models remains strong even in the presence of imbalanced data, particularly if there is sufficient representation of the minority class (abnormal instances).
- Feature Relevance: It is possible that the importance of selected features for binary classification tasks may not vary significantly when different class ratios are considered. Despite this, it should be noted that the chosen features were still

effective in identifying web attacks and played a crucial role in achieving good performance from the models, regardless of any imbalance ratio present.

- Dataset Characteristics: The dataset did not face any significant obstacles due to changes in class ratio. The impact of altering the proportion of classes did not pose any notable challenges. This can be attributed to the fact that normal and abnormal instances were easily distinguishable based on distinct features that were carefully chosen. As a result, the performance of models remained consistently strong throughout.
- Binary Nature of the Task: When it comes to classifying data, binary tasks are usually simpler than multi-class ones. The straightforward nature of identifying the difference between normal and abnormal cases can result in consistent accuracy levels.

K-Nearest Neighbor (KNN): The underperformance of KNN may be attributed to several factors. The selection of the K value plays a pivotal role, as a small K might render the model sensitive to noise, while a large K might overlook essential patterns. Additionally, KNN may struggle to effectively capture the intricate relationships between features crucial for distinguishing anomalies in web traffic.

Decision Trees (DT) and Random Forests (RF): One of the reasons why Decision Tree and Random Forest models excel in binary classification is due to their effective feature selection process. By utilizing the Random Forest model for this task, we can identify relevant features that greatly enhance the models' capacity to detect patterns and achieve precise predictions. Moreover, thanks to their ensemble nature, Random Forests exhibit robustness and ability to generalize well, which further enhances their performance when it comes to binary classification tasks.

Support Vector Machines (SVM): The lower performance of SVM models in binary classification compared to Decision Tree (DT) and Random Forest (RF) models may be attributed to the specific characteristics of the dataset and the nature of the SVM algorithm. SVM relies on finding an optimal hyperplane to separate classes, and its performance can be influenced by the distribution of data and the choice of kernel function. In some cases, SVM might struggle with non-linear separations or complex decision boundaries, whereas DT and RF, especially with suitable feature selection and ensemble methods, can adapt more effectively to intricate patterns in the data, leading to superior performance in binary classification tasks.

5.2.2.2. Experiment 2

In Experiment 2, we continue to focus on Precision, Recall, and F1-Score as crucial metrics for evaluating our model's effectiveness in detecting various web attacks. By prioritizing these metrics, we aim to provide a clear understanding of our model's effectiveness in detecting various types of web attacks. The observed differences in the performance of the SVM model across different class ratios in the multiple classification experiment can be influenced by various factors.

- Sensitivity to Imbalance: The performance of the SVM model can be influenced by several factors when dealing with imbalanced class distributions. In situations where there is a high ratio (10:1) between normal and abnormal instances, the model may struggle to effectively learn patterns from the minority class (abnormal), as it tends to prioritize the majority class (normal). However, reducing the class ratio (6:1 and 3:1) could potentially improve performance by providing more balanced training examples for learning.
- Impact on Model Generalization: Another factor that affects SVM models is their ability to generalize well to unseen data. A more balanced representation during training through lower class ratios contributes to better generalization over both normal and abnormal cases within the overall distribution of instances.
- It is important to highlight that the consistency in performance seen across different class ratios for models such as KNN, Decision Trees, and Random Forests can be attributed to their inherent qualities of being robust and adaptable when dealing with imbalanced datasets. On the other hand, SVM stands out due to its unique optimization objective and decision boundary formulation. This distinctiveness may explain why it shows a higher sensitivity towards class imbalances, which could account for the differences in observed performance levels.

K-Nearest Neighbor (KNN): It's important to note that KNN models perform differently in multiple classification scenarios compared to binary classification experiments. The recall metric for KNN increases because it needs to distinguish between more than two classes – normal vs Bruce-Forst vs XSS vs SQLi. This broader perspective considers true positives and false negatives across all classes when assessing its ability to accurately identify each one. Consequently, the increased number of classes can lead to variations in recall values for KNN in multiple classifications compared to its binary counterpart.

Decision Trees (DT) and Random Forests (RF): The performance of the RF model outperforms that of the DT model in this experiment due to overfitting. Decision Trees tend to overfit, especially when confronted with intricate datasets containing four labels. Such trees might inadvertently incorporate irrelevant information or narrow patterns present solely within the training data, resulting in poor generalization capabilities toward unfamiliar data instances. On the other hand, Random Forests excel at mitigating overfitting issues by combining predictions from numerous trees and therefore offer more reliable and resilient performance outcomes.

Support Vector Machines (SVM): Despite the enhancement in the SVM model's performance resulting from modifying the dataset ratio, it continues to exhibit inferiority compared to DT and RF models due to an unresolved issue like experiment 1.

6. Conclusion and Future Work

In our group project, we conducted two experiments that aimed to evaluate the effectiveness of various machine learning models in identifying web attack traffic within an Intrusion Detection System (IDS). With the rising threat of cyber-attacks, it is crucial to develop robust security measures, especially when dealing with zero-day vulnerabilities.

To carry out our experiments, we utilized the CICIDS2017 dataset, which is specifically designed for assessing IDSs and IPSs in the cybersecurity domain. Our choice of features for web attacks was based on Iman Sharafaldin et al. work from 2018. We focused on supervised machine learning models such as K-Nearest Neighbor (KNN), Decision Trees (DT), Random Forests (RF), and Support Vector Machines (SVM).

The results demonstrated that both Decision Tree and Random Forest models exhibited exceptional performance metrics, indicating their suitability for detecting abnormal network traffic. However, we encountered challenges like overfitting and inconsistencies in model performance across different dataset ratios. These findings underscore the importance of careful model selection, hyperparameter tuning, and monitoring model behavior to ensure reliable detection in real-world scenarios.

Moving forward, several areas will be addressed to overcome these challenges and enhance the capabilities of our developed IDS:

- Improving Model Performance: We plan to explore more advanced techniques such as boosting or stacking to further enhance model performance.
- Exploring Other Models: Investigating convolutional neural networks or recurrent neural networks as potential architectures for anomaly detection can provide valuable insights into improving accuracy.
- Focusing on Specific Attack Types: To broaden our understanding and address specific threats effectively, we aim to study how well models perform against different types of web attacks by developing specialized models tailored towards individual attack categories.
- Considering Real-Time Implementation: Assessing the feasibility and overcoming obstacles associated with implementing the best-performing models in real-time settings will be a significant focus area for future efforts.

By addressing these aspects through continued research and development, we aim to create an IDS that is highly effective in real-world scenarios involving web attack detection.

7. References

1. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. (January 2018). *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal. https://www.unb.ca/cic/datasets/ids-2017.html

2. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019, July 17). *Survey of intrusion detection systems: Techniques, datasets, and challenges - cybersecurity* SpringerLink. https://link.springer.com/article/10.1186/s42400-019-0038-7
3. Microsoft. (2021, November 3). *Working with Jupiter notebooks in visual studio code.* RSS. https://code.visualstudio.com/docs/datascience/jupyter-notebooks
4. GeeksforGeeks. (2023, August 8). *Machine learning tutorial.* GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/
5. Supervised machine learning. (2023, August 24). *Supervised machine learning tutorial.* GeeksforGeeks. https://www.geeksforgeeks.org/supervised-machine-learning/
6. N. D. Marom, L. Rokach and A. Shmilovici. (2010). *Using the confusion matrix for improving ensemble classifiers.* 2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel, Eilat, Israel, 2010, pp. 000555-000559, doi: 10.1109/EEEI.2010.5662159.
7. Scikitlearn. (2024, January 6). *Supervised learning.* Scikitlearn. https://scikitlearn.org/stable/supervised_learning.html
8. Jabez, J., & Muthukumar, B. (2014). *Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach*. Procedia Computer Science, *48*, 338-346. https://doi.org/10.1016/j.procs.2015.04.191
9. Jyothsna, Veeramreddy (2019). *Computer and Network Security [Working Title] || Anomaly-Based Intrusion Detection System.* 10.5772/intechopen.78497(), – . doi:10.5772/intechopen.82287
10. Thapa, Suman & Mailewa, Akalanka. (2020). *The Role Of Intrusion Detection/Prevention Systems In Modern Computer Networks.*
11. CICFlowMeter (2017). Canadian institute for cybersecurity (cic).
12. Habibi Lashkari, A., Draper Gil, G., Mamun, M. S. I., and Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. In *In Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, pages 253–262.
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duch-esnay, E. (2011). *Scikit-learn: Machine learning in Python*.
14. Stehman, Stephen V. (1997). *Selecting and interpreting measures of thematic classification accuracy*. Remote Sensing of Environment. 62 (1): 77–89. Bibcode:1997RSEnv..62...77S. doi:10.1016/S0034-4257(97)00083-7.
15. Powers, David M. W. (2011). *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation*. Journal of Machine Learning Technologies. 2 (1): 37–63. S2CID 55767944.
16. Koehrsen, W. (2018, July 12). *Overfitting vs. Underfitting: A Complete Example - Towards Data Science*. Medium. Retrieved from https://towardsdatascience.com.