electronics

Article

Explainable Artificial Intelligence for Intrusion Detection System

Shruti Patil [1,*], Vijayakumar Varadarajan [2,3,4,*], Siddiqui Mohd Mazhar [5], Abdulwodood Sahibzada [5], Nihal Ahmed [5], Onkar Sinha [5], Satish Kumar [1], Kailash Shaw [5] and Ketan Kotecha [1]

1 Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India
2 School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 1466, Australia
3 School of NUOVOS, Ajeenkya D Y Patil University, Pune 412105, India
4 Swiss School of Business and Management, 1213 Geneva, Switzerland
5 Department of Computer Science Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India
* Correspondence: shruti.patil@sitpune.edu.in (S.P.); vijayakumar.varadarajan@gmail.com (V.V.)

Abstract: Intrusion detection systems are widely utilized in the cyber security field, to prevent and mitigate threats. Intrusion detection systems (IDS) help to keep threats and vulnerabilities out of computer networks. To develop effective intrusion detection systems, a range of machine learning methods are available. Machine learning ensemble methods have a well-proven track record when it comes to learning. Using ensemble methods of machine learning, this paper proposes an innovative intrusion detection system. To improve classification accuracy and eliminate false positives, features from the CICIDS-2017 dataset were chosen. This paper proposes an intrusion detection system using machine learning algorithms such as decision trees, random forests, and SVM (IDS). After training these models, an ensemble technique voting classifier was added and achieved an accuracy of 96.25%. Furthermore, the proposed model also incorporates the XAI algorithm LIME for better explainability and understanding of the black-box approach to reliable intrusion detection. Our experimental results confirmed that XAI LIME is more explanation-friendly and more responsive.

Keywords: IDS; CICIDS2017; XAI; LIME; ensemble techniques; intrusion detection system

check for updates

Citation: Patil, S.; Varadarajan, V.; Mazhar, S.M.; Sahibzada, A.; Ahmed, N.; Sinha, O.; Kumar, S.; Shaw, K.; Kotecha, K. Explainable Artificial Intelligence for Intrusion Detection System. *Electronics* **2022**, *11*, 3079. https://doi.org/10.3390/electronics11193079

Academic Editor: Flavio Canavero

Received: 18 July 2022
Accepted: 20 September 2022
Published: 27 September 2022

1. Introduction

As a result of extensive internet use, cyberattacks against financial institutions, government organizations, and energy companies have increased dramatically in recent years. Hackers and intruders attack businesses with large websites. Viruses, malware, worms, fraudulent logins, and spyware are just a few of the ways for them to attack. Organizations need security applications to protect their networks from malicious attacks and misuse. Intrusion detection systems (IDS) can detect and prevent data breaches by detecting and stopping intrusions. Misuse detection and anomaly detection are two types of intrusion detection. Misuse detection relies on information or patterns, whereas anomaly detection relies on behavior [1]. Current intrusion detection systems have a high detection rate, which may result in many false alarms. In an IDS, false positives should be minimized. Various IDS are implemented using different machine learning approaches because they can discover valuable information from datasets. These approaches have the potential to minimize false positives. Machine learning approaches, including association of rules, genetic algorithms, and intrusion detection systems, are frequently implemented using artificial neural networks. Ensemble learning mixes various machine learning methods [2]. Researchers have discovered that an ensemble approach to ML reduces false positives.

With the advancements in technology, the threat to industry also increases. The three major contributors to Industry 4.0 are the internet of things (IoT), AI, and extensive data analysis [3]. IoT-based monitoring systems for secure computer numerical control machines have proven safe against cyber-attacks with increased reliability. Use of IoT also shows remarkable improvements in deep neural networks (DNN) for recognizing intrusion detection in automated guided vehicles (AGVs) [4]. The major drawback of any model is its complexity, with no clear indication about the behavior or cause of the decision of the model. Hence, research interpretation has tilted toward other fields such as computer vision, robotics, bioinformatics, and natural language processing. Recently, the SHapley Additive explanations (SHAP) [5] framework has improved the transparency of IDs, which further helps in cybersecurity by positively identifying the IDs' judgments.

In this research, we worked with the network traffic data set CIC-IDS-2017. The data set passed through a preprocessed pipeline consisting of cleaning, normalization, SMOTE, and feature selection. Later, the extracted feature was classified using various ML methods, such as decision tree (DT), random forest (RF), and support vector machine (SVM). Model accuracy was reported and analyzed using explainable artificial intelligence (XAI), to justify the trustworthiness, ability, and reliability of the AI-based solutions in IDS. XAI [6] is a method that allows humans to understand the results of a model, as models are too difficult to understand and explain due to their black-box concept [7,8], and meant for improving accuracy. Our approach is to provide a solution as a white-box approach for a better understanding of the model and a reliable prediction, so that all stakeholders are on the same page [9,10]. Model and data discrepancies can be detected early on during the modeling process and corrected accordingly. From the experiment, we successfully showed the effect of XAI techniques such as the LIME framework that strengthened the model' accuracy and reliability [11].

Deep neural networks, complex nonlinear models, have mainly been responsible for recent advancements in AI and machine learning [12]. "Black box" models such as ensemble methods or support vector machines are hard to interpret and understand. Some algorithms require many samples, because of their nonlinear characteristics, multiple parameters, and complex transformations [13]. Additionally, the training sets for learning are enormous, making it difficult to explain how the model learns. As a result, the model cannot explain how it is learning from the data and which portion of the data sets significantly influenced the output, ultimately leading to uncertainties within the model and a delayed human response [14]. In addition, there are ethical issues in insensitive, high-profile domains, such as in finance, healthcare, and security. Artificial intelligence and machine learning are becoming increasingly crucial in security solutions and defense. Various government efforts have been implemented to promote responsible use of AI systems and minimize unethical practices related to their use. An algorithm decision can be challenged by requesting proof from the European Union, which promotes the "right of explanation" [15]. There is also the possibility of auditing machine learning systems for bias and discrimination, which could require corrective action. Reasons for bias have been identified by the Department of Defense [16]. XAI provides a solution to this problem by displaying the prediction and explaining the black box. Additionally, for those unfamiliar with XAI, here is a brief description of how it works. A comprehensive AI model can be described in three dimensions [17]:

Explainability: A learning model's explainability involves the ability to clearly explain the processes it undertakes. This study explains how training models work by explaining their inner workings. Critical applications used in real-world service are not only intellectually fascinating but are also weighed against risky factors when human lives are at stake.

Interpretability: Interpretability provides information regarding how the model functions and allows users to draw conclusions from it.

Furthermore, a learning model is considered transparent if it exhibits understandability by itself, without any intervention from the user. Models are transparent when they are inherently comprehensible without additional components [18]. Transparency is a key characteristic of comprehensive learning models, including explainability and interpretability. Academics and the industry have debated interpretability, consistent with which ML and DL models perform better in causality, reliability, and usefulness [19]. Furthermore, a variety of definitions of explanation and interpretability are used in different literature, but these terms are interchangeable. The definition of interpretability suggests that it is a mechanism that allows algorithms to explain decisions and for the inner details of a model to be understood, and consequently, models are explained mathematically [20]. Keeping generality, and following LIME's approach, we will discuss interpretability and explainability interchangeably. Transparency is a key component of AI models, and their decisions require explanations [21].

On the other hand, the threat of adversarial attacks on machine learning algorithms emphasizes the need for algorithmic and functional transparency within machine learning mechanisms. A transparent AI model will serve many purposes, including allowing us to expect positive outcomes from AI, choose whether to trust AI or take human factors into account fully, and address threats to AI-based systems [22]. In addition to helping the responsible intrusion detection system (RIDS) make better and more accurate predictions, the XAI algorithm explains how the classifiers identify specific attacks in a dataset [23].

The rest of the paper is organized as follows: Section 2 discusses the literature review, Section 3 contributes significant studies, Section 4 deals with the methodology, and Section 5 discusses the generation of explanations using the X-AI model (LIME), followed by prediction and accuracy in Section 6. Section 7 deals with the experimental results and discussions, and the conclusions are in Section 8.

2. Literature Review

An overview of the algorithm distributions and major techniques used in intrusion detection research is presented in Figure 1. On the basis of the majority papers, we identified 68 primary studies. This study examines several classification techniques [24]: The majority of experiments were performed using classification approaches, which account for 81% of all experiments. Other techniques mainly include clustering, dataset analysis techniques, prediction techniques, estimation techniques (3%), association techniques, and statistical analysis. The study showed that public datasets accounted for 79% of experiments and experiments on private datasets were 21%, which were compared in the research studies [25]. A comprehensive study of 18 techniques for intrusion detection was conducted: six of these methods were found to be most prevalent: support vector machine, deep neural network, random forest, naive Bayes, decision tree, and K- nearest neighbor. Additionally, some researchers proposed some different techniques.

Ensemble methods to improve machine learning classifier accuracy on IDS included boosting algorithms, ensemble combined feature selection algorithms, and machine learning methods, as mentioned in Table 1.

Peddabachigari et al. [26] modeled intrusion detection systems using hybrid intelligent systems. On the basis of their research, they investigated some new techniques for intrusion detection and examined their performance against the benchmark KDD Cup 99 intrusion dataset. Their research focused on data temporal correlation (DT) and sparse maximum likelihood (SVM). As a next step, they developed a hybrid DT–SVM model, and an ensemble approach with DT, SVM, and DT–SVM models as the base classifiers. The results indicated that DT provides better or equal accuracy than Probe, U2R, and R2L for all classes. If compared with direct SVM, a hybrid DT-SVM approach delivers better performance or offers equal performance for all classes. It is the ensemble approach that offers the best performance for the Probe and R2L classes. According to the ensemble approach, the Probe class achieved a 100% accuracy, suggesting that with proper base classifiers other classes may reach 100% accuracy too. The final proposal was to develop a hierarchical intelligent IDS model that can optimally take advantage of the top-performing individual base classifiers and the ensemble approach.

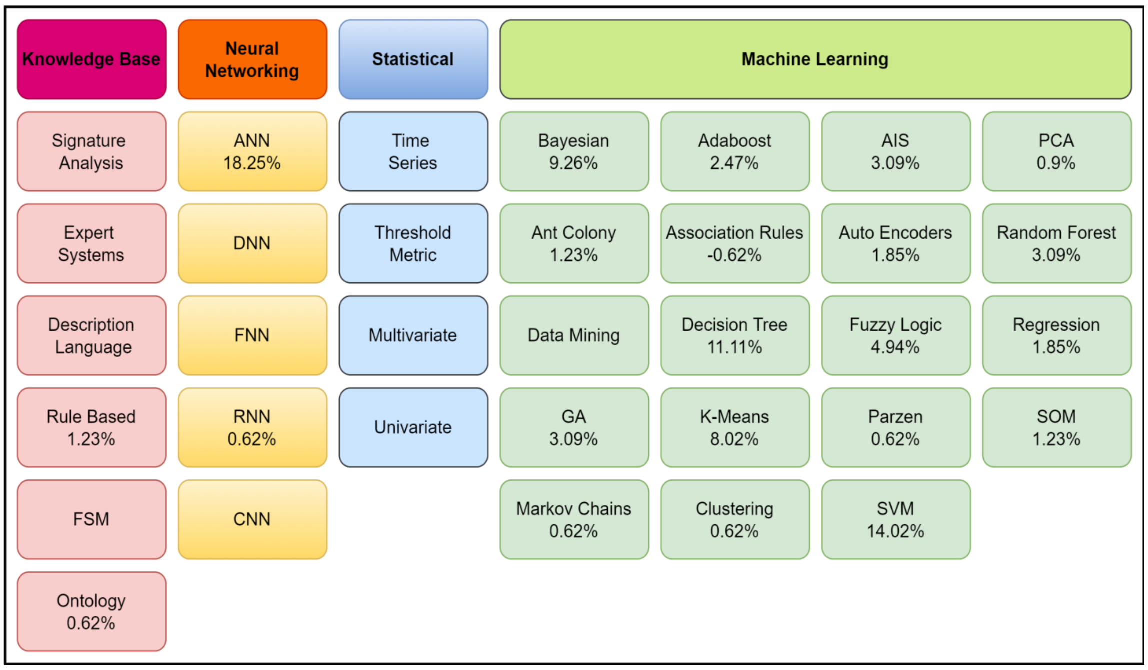


Figure 1. Algorithm distribution.

Li et al. [26] proposed a Ditto framework to achieve fairness and robustness via federated learning systems. They statistically identified heterogeneous networks as spaces where robustness to attacks, as well as fairness, measured as the uniformity of performance across devices, compete for resources. The Ditto framework and a scalable solver are our proposals for addressing these constraints. A class of linear problems was theoretically analyzed to determine whether Ditto can achieve fairness and robustness simultaneously. They demonstrated empirically that Ditto results in a competitive performance relative to recent customization methods, but also generates more accurate, robust, and fair models, compared to standard fair and robust baselines.

Mohseni et al. [27] reviewed the challenges and opportunities in bringing ML safety to open-world tasks and applications. Reviewing and comparing the dependability limitations of ML algorithms in uncontrolled open-world scenarios with conventional safety standards, we first review the shortcomings of ML in uncontrolled open-world scenarios. The third strategy category is run-time error detection. These three strategies were applied in order to meet the ML dependability objective of achieving safe design, to improve model performance and robustness, and increase run-time error detection. As a research direction for ML, ML safety tries to reduce the potential long-term risks associated with ML. In particular, for the next decade, the focus is on cases where general machine learning capabilities outpace safety, or where safety problems are going to become more challenging.

Table 1. Literature Review.

Reference	Dataset Used	Techniques Used	Research Findings
[28]	NSL-KDD	This paper proposes a different explainable AI framework that provides the capabilities for transparency in every stage of the machine learning pipeline, including SHAP, LIME, CEM, ProtoDash, and Boolean Decision Rules. They also employed deep neural networks for intrusion detection.	This approach uses a variety of explainability techniques, these approaches are applied on the NSL-KDD dataset. This is a dataset which is based on KDD99, an older dataset.
[29]	KDDCup-99	This paper proposes different machine learning classifiers on top of kdd99 This research paper also elaborately covered the application of DNNs in intrusion detection comprehensively.	This approach used many classical machine learning classifiers, as well as deep neural networks of layer 1 to 5; however, the approach used the KDD99 dataset.

Table 1. *Cont.*

Reference	Dataset Used	Techniques Used	Research Findings
[30]	KDDCup-99	Using genetic algorithms to detect several types of network intrusions, the paper discusses the implementation of an intrusion detection system (IDS). Utilizing the standard KDD99 benchmark dataset, we implemented our system and obtained a good detection rate.	In intrusion detection, many approaches have been adopted, but none of the systems have been completely error-free. Thus, the quest for betterment continues. KDDcup-99 was yet another limitation to this approach.
[31]	NSL-KDD	An application of self-taught learning (STL) for classification was proposed in this paper. STL employs two stages for deep learning. Accuracy, precision, recall, and F-measure values are among the metrics compared.	The use of a deep-learning-based NIDS with a sparse autoencoderand softmax regression was demonstrated. It is a limitation of this approach that NSL- KDD does not apply.
[32]	CIC-IDS- 2017	Several analysis methods involving artificial neural networks and ML were proposed in this paper for the CIC-IDS-2017 data set.	This paper proposes pros and cons of using CIC-IDS-2017 for domain research and implementation. It was found that the data were not completely reliable in the high required working processes. Many cases where data cells read "NaN" and "Infinity" ceased to exist.
[33]	NSL-KDD	A paper presented different deep learning techniques that have the ability to adapt to dynamic environments. Three models were used: bidirectional long short-term memory, Inception-CNN, and deep belief network.	A number of practical problems with existing intrusion detection were addressed in this work, and different deep learning models were compared to solve the problems. The model has been developed and tested on the NSL-KDD dataset.
[34]	KDD99, NSL- KDD	Ten machine learning approaches were presented for IDS, which include decision tree, hybrid with locally weighted learning, rotation forest, and Bayesian belief network, and then combination of J48 and NB with AdaBoost was implemented for IDS.	The results of this research suggest that the proposed approach utilizing a variety of machine learning methods using the NSL-KDD dataset performed well in all major categories of attacks. However, a low detection rate was reported. It is inevitable to detect minority attacks such as U2R and R2L, which results in large bias in the dataset.
[35]	NSL-KDD	A combined approach is proposed by combining NB with feature vitality based feature selection method for accuracy improvement.	The FVBRM method achieved 97.78% overall classifier accuracy, with 98.7 TPR for DoS, 97% for normal, 98.8% for probe, 96.1 for r2l, and 64% for u2r, which was higher than others compared.
[36]	NSL-KDD	Clustering algorithm was used to group dataset samples into a set representing four attack classes, and then classification was done.	The K-means algorithm took 9.25 s to build cluster models and the mean squared error in this process was 19,308.72.
[37]	DARPA1998, KDD99, NSL-KDD, UNSW-NB15	Survey on machine learning and deep learning algorithms used for intrusion detection.	The paper proposed an IDS taxonomy over various machine learning algorithms used in this field. It discusses the various data sources, i.e., logs, packets, flow, and sessions. The paper emphasized ML for IDs
[38]	Examine 37 cases	The paper examined the contribution of the IDPSs in the SG paradigm, providing an analysis of 37 cases.	Timely detecting of IDs was given stress in the paper. The cases the paper deals with are on the advanced metering infrastructure (AMI), supervisory control, and data acquisition (SCADA) systems, substations, and synchrophasors. Based on a comparative analysis, the limitations and the shortcomings of the current IDPS systems were identified, and appropriate recommendations were provided for future research efforts.
[39]	KDD 99, NSL-KDD, CIC-IDS2017, CIC-IDS2018	Evaluated four deep learning models on four intrusion detection datasets.	Gave a taxonomy and survey of deep learning models for intrusion detection. Evaluated four deep learning models on four intrusion detection datasets. Feed-forward neural networks performed best across all metrics, on all datasets.
[40]	14 different datasets were used	Evaluated the performance of SVM for IDS approaches.	This paper presented a comprehensive study and investigation of the SVM-based intrusion detection and feature selection systems.
[41]	KDDCUP99, NSL-KDD	IoT NIDS based on machine learning techniques.	This paper reviewed existing NIDS implementation tools and datasets, as well as free and open-source network sniffing software. Then, it surveyed, analyzed, and compared state-of-the-art NIDS proposals in the IoT context, in terms of architecture, detection methodologies, validation strategies, treated threats, and algorithm deployments.
[42]	Comparison result on various dataset	The work focused on the newest studies in intrusion detection and intelligent techniques applied to IoT, to keep data secure.	The research focused on rigorous state-of-the-art literature on machine learning techniques applied in internet-of-things and intrusion detection for computer network security.

Table 1 discusses research findings, concentrating on the dataset used to train the machine learning models, the research findings, and limitations of each approach.

3. Significance of the Study

In this study, the intrusion detection system was trained using the CICIDS-2017 dataset. Choosing the right features is an essential part of intrusion detection. According to the review of the literature, some traits are required for all types of attacks, while others are partially required, not required, or just required for specific attacks. There are 76 features in the CICIDS-2017 dataset for training and testing IDS. Based on this research, we picked only 10 significant features from the CICIDS-2017 dataset, to improve classification accuracy. Explanations have been proven to be useful for both professionals and non-experts in selecting between models measuring trust, for improving untrustworthy models, and for obtaining insights into forecasts in the text domain. Hence, the authors produced local interpretable model-agnostic explanations (LIME) observations for DT, RF, and SVM after applying the machine learning models.

4. Methodology

4.1. DATASET

A sufficient number of security events must be supplied for a model to accurately conduct network data feature classification tasks. These factors will have a big impact on the final outcome. As there are now many NIDS datasets with different feature sets that are often independent of one another, it is essential to remember that extracting all of those characteristics once the model is implemented in the field would be impossible. As a result, integrating one feature set across different datasets is critical in model building, to enhance the assessment reliability and, hence, deployment possibilities. NIDS datasets must be similar in their data representation, to facilitate reliable experiments. This research work considered CICIDS2017.

Ref. [34] provided the dataset for implementing the proposed methodology. This dataset consists of both normal and abnormal sample sets of network traffic. It has a realistic network attack date, which is very important for designing mitigation techniques in IDS. This dataset is noteworthy because it contains the most recent cyber-attacks, and the recovered PCAP file reflects accurate real-world data, as represented in Figure 2. There are roughly 80 features and 15 unique classes in this set. The dataset, however, is dispersed across eight CSV files, making it challenging to work with. Another flaw in this dataset is the inclusion of NaN values, while the distribution of most of the classes is not uniform. These parameters directly affect any machine learning model's performance and accuracy.

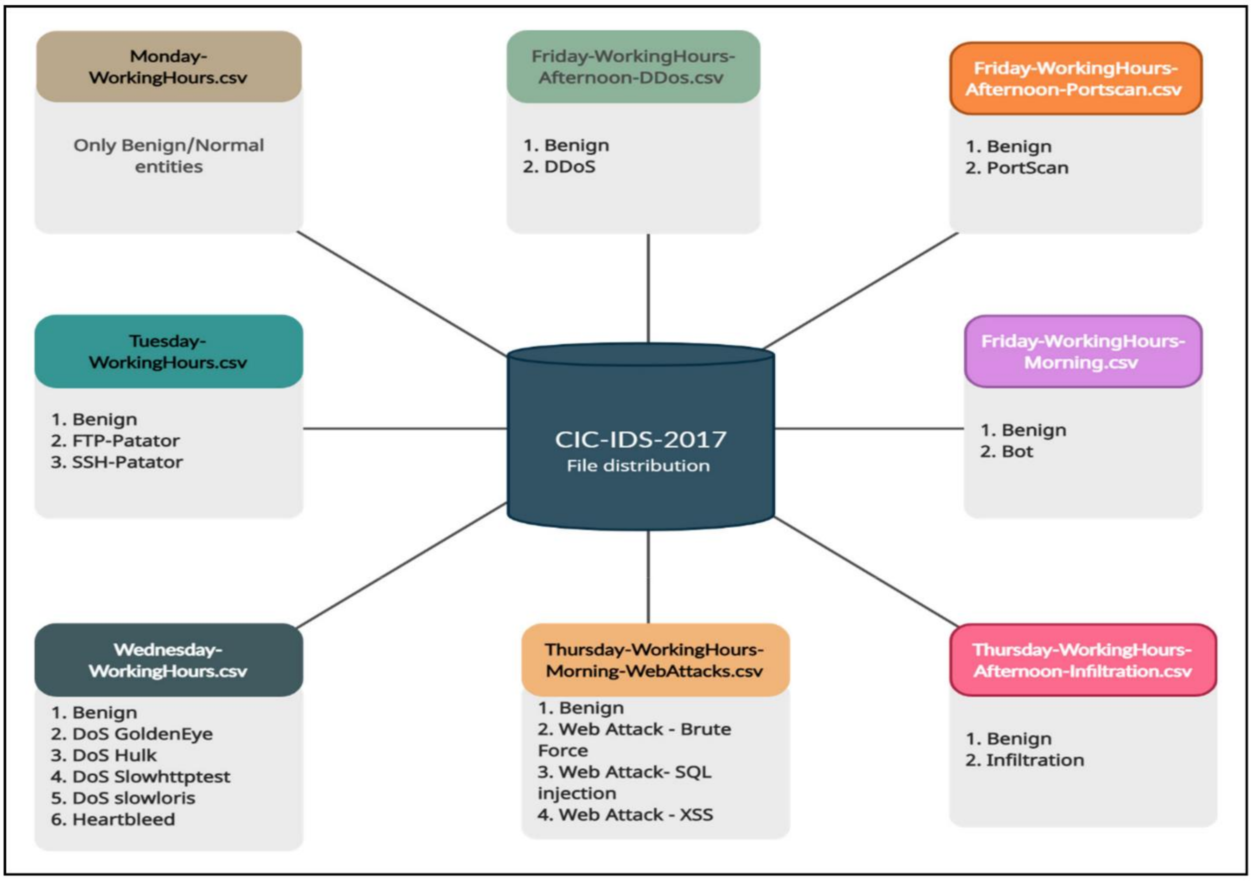

Monday-WorkingHours.csv
Only Benign/Normal entities
Friday-WorkingHours-Afternoon-DDos.csv
1. Benign
2. DDoS
Friday-WorkingHours-Afternoon-Portscan.csv
1. Benign
2. PortScan
Tuesday-WorkingHours.csv
1. Benign
2. FTP-Patator
3. SSH-Patator
CIC-IDS-2017
File distribution
Friday-WorkingHours-Morning.csv
1. Benign
2. Bot
Wednesday-WorkingHours.csv
1. Benign
2. DoS GoldenEye
3. DoS Hulk
4. DoS Slowhttptest
5. DoS slowloris
6. Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.csv
1. Benign
2. Web Attack - Brute Force
3. Web Attack- SQL injection
4. Web Attack - XSS
Thursday-WorkingHours-Afternoon-Infiltration.csv
1. Benign
2. Infiltration

Figure 2. File distribution of the dataset CIC-IDS-2017.

4.2. Feature Selection

It is important to identify the features that make a significant contribution to output prediction in the process of feature selection. Model overfitting and processing time are reduced along with an increased model accuracy [35,36]. After initial data preparation, the authors reduced the number of features in the dataset. After which, they performed feature selection for ML models. To select the appropriate feature selection method, the authors compared the performance of several feature selection techniques.

Correlation heatmap: this shows how the features relate to one another. Moreover, if we determine that a set of features are very closely related by their values, i.e., a correlation can be found between their values, then the set can be replaced or dropped using a single feature. Figures 3 and 4 show the correlation between all the features in the CIC-IDS-2017 dataset.

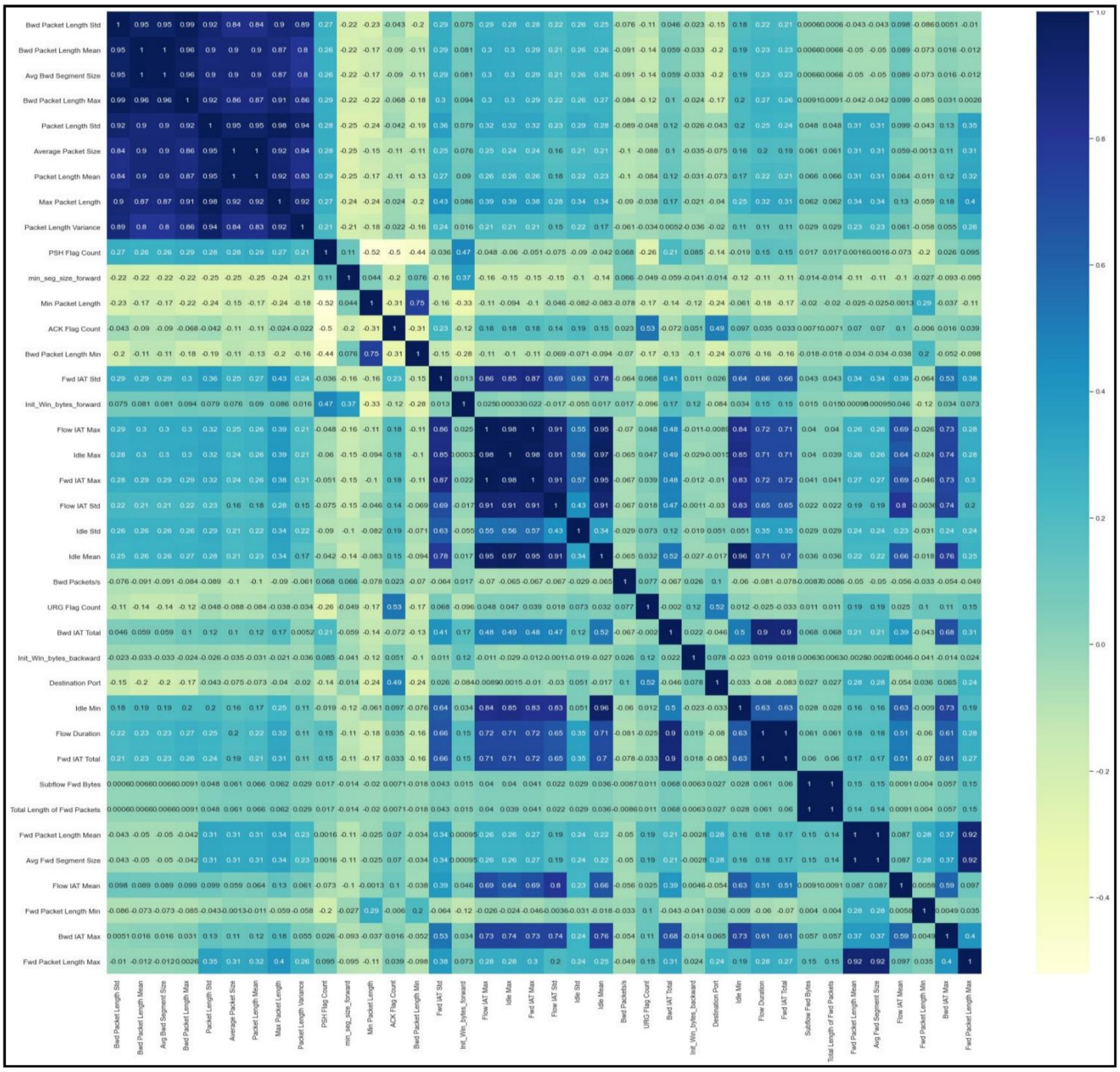


Figure 3. Correlation heatmap of features for the first stage.

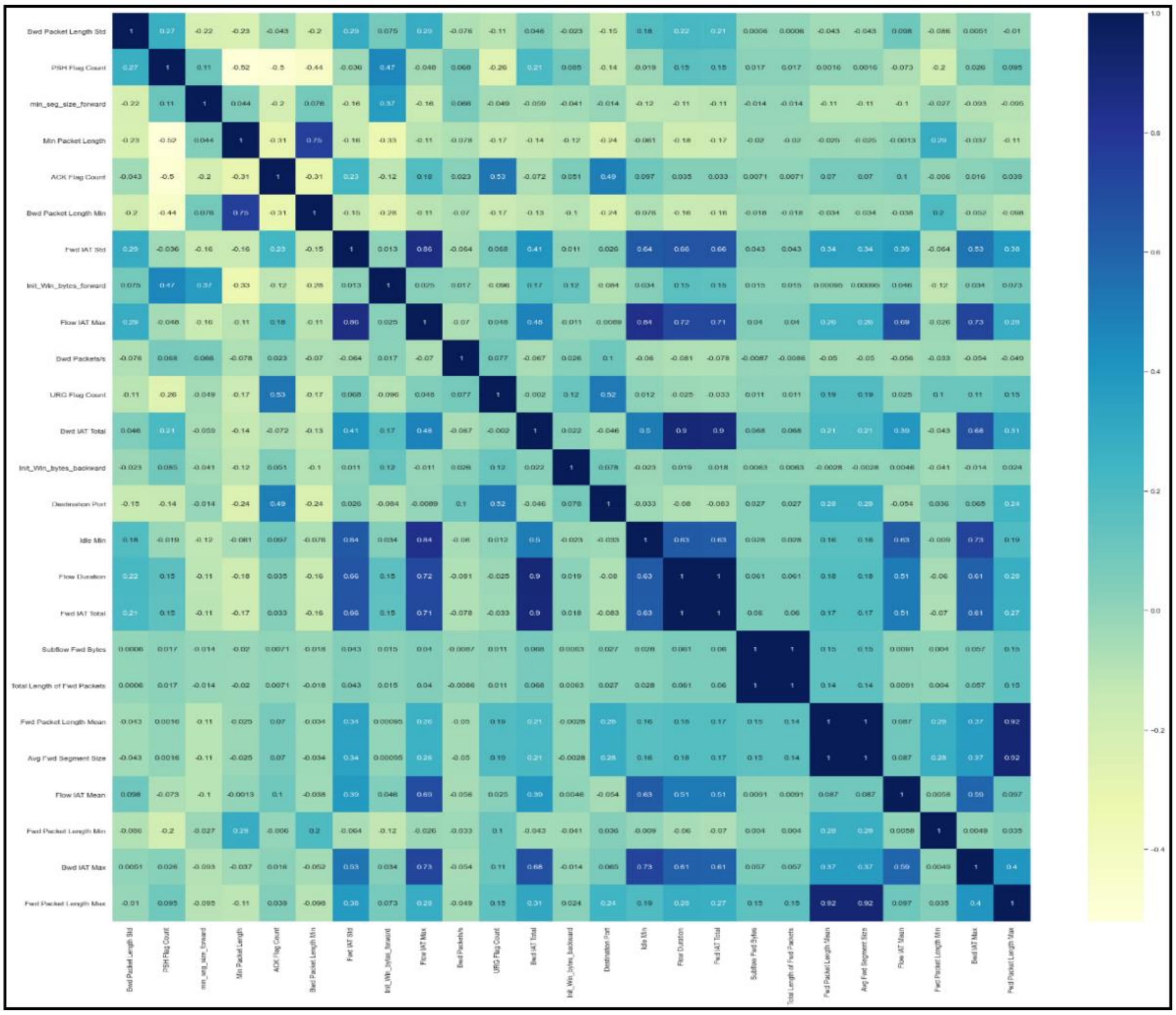


Figure 4. Correlation heatmap of features for the second stage.

Referring to the heatmap of Figures 3 and 4, it can be seen that there is an excellent correlation between the following features: Bwd Packet Length Std, Bwd Packet Length Mean, Avg Bwd Segment Size, Bwd Packet Length Max, Packet Length Std, Average Packet Size, Packet Length Mean, Max Packet Length, and Packet Length Variance. In addition, correlations can be seen in Flow IAT Max, Idle Max, Fwd IAT Max, Flow IAT Std, Idle Std, and Idle Mean.

Finding outliers: A data point that is abnormally dispersed compared to other points is considered an outlier. Basically, it refers to data that differ from the other values in the set. A significant finding could be missed, or real results distorted because of outliers, a problem for many statistical analyses. Figure 5 illustrates some outliers that are present in the CIC-IDS-2017 dataset.

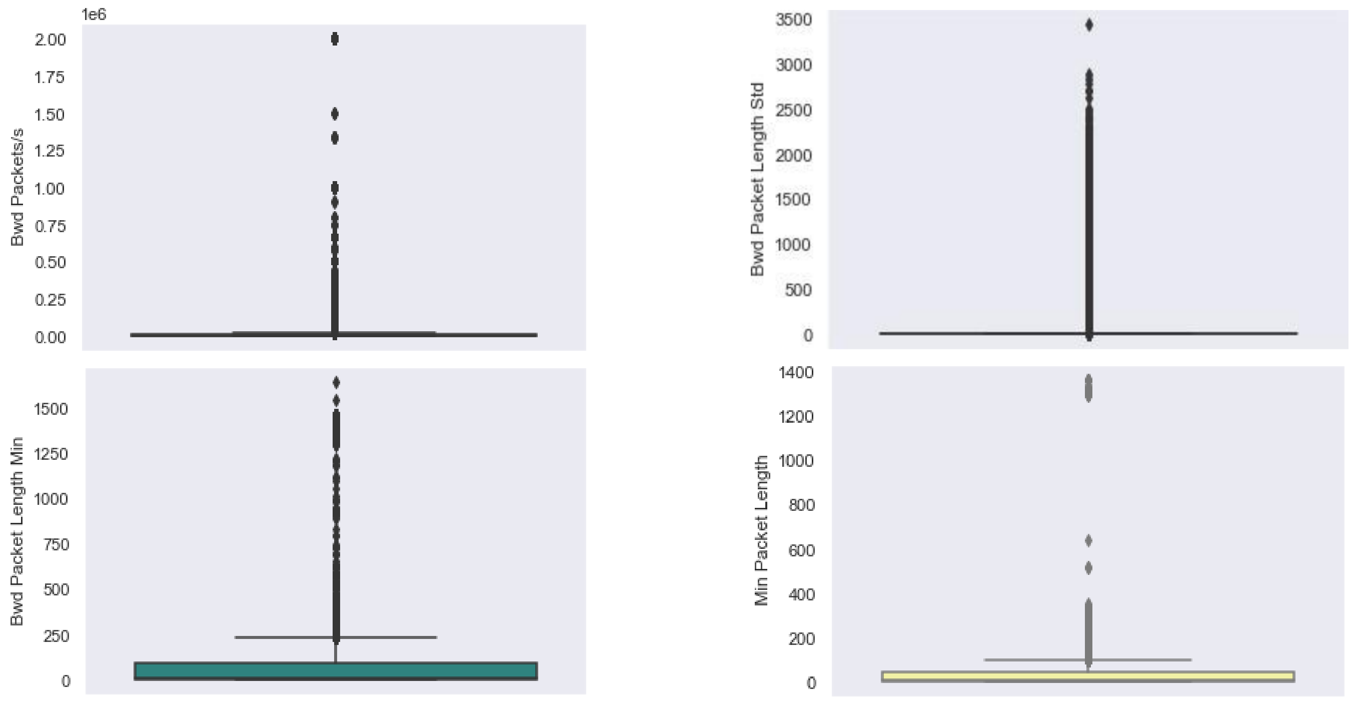


Figure 5. Boxplots of outliers/anomalies.

Some of the features contain outliers, for example Bwd Packert/s, Bwd Packet Length Min, Bwd Packet length Std, and Min Packet length, as shown in the graph above. Our second stage of feature selection included reduction of these features.

4.3. Machine Learning Pipeline

The machine learning pipeline includes steps of data loading, preprocessing, and AI model implementation. Three traditional supervised learning algorithms and one ensemble technique of voting classifier are shown in Figure 6. Each step of the implemented methodology is explained below:

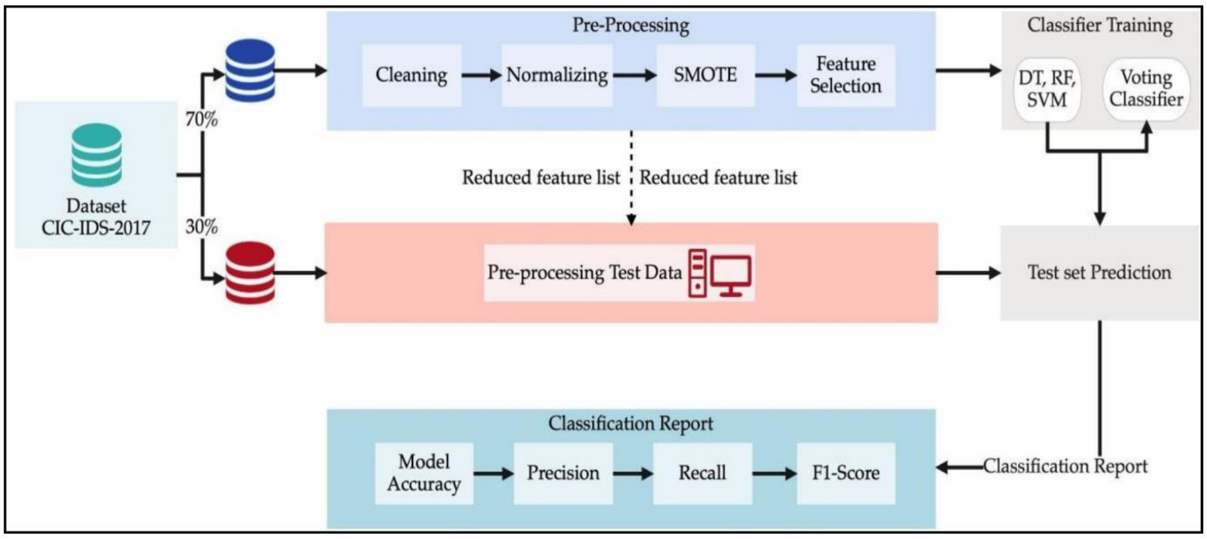


Figure 6. Pipeline model framework.

Data Loading: The first step in any machine learning analysis is to get the raw data into the system. This raw information could be a dataset consisting of log files or a database. Since the dataset consists of eight different files, we first concatenated the set into one single file. In addition, the Pandas library offers a comprehensive toolkit of the data to be loaded, using functions created using sci-kit-learn, an open-source machine learning library written in Python, as well as simulated data generated using simulated data.

Data Preprocessing: StandardScaler was used to standardize the CICIDS dataset: "garbage in, garbage out." In Pandas, None, and NaN are essentially interchangeable as ways of indicating missing data. Hence, null values need to be dropped from a data frame. Since the dataset had a huge volume and consisted of many null values, the drop. null() function was used to remove the rows/columns containing null values.

Train_test_split:

A model is trained on 70% of the data and the rest is used for testing.

Import train_test_split from sklearn model_selection

X_train,X_test,Y_train,Y_test=train_test_split(train_X,y,train_size=0.70,random_state=2)

Balancing the imbalanced data: The dataset was balanced with an imblearn Randomundersampler, which automatically selects randomly selected subsets of data for the targeted classes. Moreover, we used the synthetic minority oversampling technique or SMOTE.

In the research, we split our data into training and testing data at 70:30%. On 70% of training data, we used k-fold cross-validation with k = 5. Training data were split into five parts (train and test), and we trained our algorithm multiple times. The model parameter was trained with four sets of train data and tested with one set of unseen test data. This way, we are able to tune the parameters, as well as prevent the model from overfitting.

4.3.1. Random Forest (RF)

Breiman (2001) developed random forest as an ensemble technique to handle supervised classification. Using supervised learning algorithms, random forests build decision trees with training sets, in order to improve their accuracy. Random forest uses the bagging method to build ensembles of decision trees. RF generates a variety of trees from bootstrapped subsamples (random samples drawn with replacement) of coaching information. It is historically determined, based on finding a split attribute that is simple among a narrower set, what a tree seems to be. Consequently, randomly generated trees are less related, since they make the same kinds of prediction errors and may overfit the model. The trees in less related trees will be incorrect in some cases, but the correct ones, and the trees collectively, should move in the right direction, since as they are closely related, the outputs are summed up for the final prediction. Random forest (RF) is a collaborative model. The first step is selecting features, followed by classifying them. Random forests create multiple decision trees from random subsets of data. The major advantage of random forests over other traditional classifiers is their lower classification errors. When dealing with large datasets, RF requires too much computation.

The random forest recursive feature is eliminated

```
1:  Acquire all the features of the random forest model
2:  Calculate the RMSE and rank the importance of features.
3:  Let i = 1 to n do
4:  Remove the features with the smallest importance
5:  Train the RF model with the tunes subset
6:  Calculate RMSE and measure model performance
7:  Rank the importance of features
8:  End
9:  Identify the optimal length of the features and the rank of each.
```

The main difference between them is the way they combine decision trees. Random forests are built using a bagging method, in which each decision tree is used as a parallel estimator. The n_estimators parameter determines the number of trees used in the ensemble model. The effect of the n_estimators parameter is different in random forests. Increasing the number of trees in random forests does not cause overfitting. After some point, the accuracy of the model does not increase by adding more trees, but it is also not negatively affected by adding excessive trees.

4.3.2. Decision Tree (DT)

This is the most popular and powerful classification and prediction tool. Depending on its purpose, decision trees can represent the result of a test of one or more attributes. Similarly, the internal nodes represent the tests, the branch nodes represent the outcomes, and the leaf nodes represent the classes.

A decision tree allows an individual or organization to utilize the costs, possibilities, and advantages of every action against each other. It can be used by individuals or organizations to compare potential outcomes. Sometimes they are used in developing an algorithm to calculate the best alternative mathematically, or they can be promoted in informal discussions. There is usually one node in a decision tree that branches out to possible outcomes. These nodes branch out into other prospects, giving it its arborescent form. A node can be categorized into three kinds: a likelihood node, a call node, and a finish node.

The following are some assumptions we mad when using decision trees:

- The whole training set is considered the root in the beginning.
- Categorical values are preferred for feature values.
- Recursively, records are distributed based on attribute values.
- According to some statistical methods, nodes or root attributes are ranked according to their importance in a tree.

A decision tree is built upon iteratively asking questions to split data points. The max_depth parameter controls the depth of the tree. The model stops splitting when the max depth is reached. There is no standard or optimal value for max depth; it solely depends on data. In general, the tree depth is more profound, and the chances of overfitting increase when the depth is small; then, the model may not capture information about the dataset.

4.3.3. Support Vector Machine (SVM)

A supervised classification task is performed by supporting vector machines (SVMs), an algorithm originally introduced by Boser, Guyon, and Vapnik in 1992. A nonlinear implicit mapping of input vectors into high-dimensional feature houses can produce an optimum hyperplane that can separate instances of different categories in conjunction with linear classification (the kernel trick) (Hooman et al. 2016). If there are millions of samples, then the computation time becomes extremely expensive.

Based on support vector machines (SVMs), which have an edge over the typical data sets, a decision boundary was constructed. The support vector machine algorithm seeks a hyperplane (where N is the number of characteristics) that distinguishes between data points. This approach minimizes the threat of classification, rather than pursuing an optimal classification. When the number of features m, is large and the number of data points n, is small (m > n), then this technique is useful. Hyperplanes are used to classify data points. Data points can be classified into different classes, based on where they fall on the hyperplane. Moreover, the number of features determines the dimensions of the hyperplane. There are only two input features for a line to be a hyperplane. Support vectors are data points that determine the position and orientation of a hyperplane. These support vectors maximize the classifier's margin. If the support vectors are removed, the position of the hyperplane will change.

Using SVM to recursively eliminate features. Initially, the data subset G is 1, 2, and n. R is the smallest weight criterion applied to the output ranked list.

1: Set R={}

2: Repeat steps 3 to 8 until G contains no empty cells.

4: Incorporate G into your SVM training

5: Estimate the weight vector

5: Identify your ranking criteria.
Then, arrange them in order of importance. Newrank = Sort(Rank)
6: Update the feature rank list by Update R = R + G(Newrank)
7: Remove features ranked at the lowest level
8: Update G = G – G(Newrank)

In the high dimensional feature space of two ad hoc classification problems between two classes, support vector machine (SVM) is the most accurate and robust model and classification algorithm.

SVM is mostly used in supervised ML. SVM creates a decision boundary that can solve the optimization problem, to separate different class labels. The correct placing of the decision boundary is neither too close nor too far for the correct prediction of labeled data points. The parameters c and gamma control the trade-off between the decision boundary placement. When c is low, the low penalty of misclassification is low, so the decision boundary with a large margin can be chosen at the expense of a more significant number of misclassifications.

Similarly, when c is large, SVM tries to minimize the number of misclassified examples due to a high penalty, resulting in a decision boundary with a smaller margin. The penalty is not the same for all misclassified examples. It is directly proportional to the distance to the decision boundary. On the other hand, low gamma values indicate a large similarity radius, resulting in more points being grouped. For high gamma values, the points need to be very close to each other to be considered in the same group (or class). Therefore, models with very large gamma values tend to over fit. As the gamma decreases, the regions separating different classes become more generalized.

4.3.4. Voting Classifier

The voting classifier (VC) is a machine learning model that utilizes an ensemble of several models and identifies the output class with the highest probability. Hard and soft are the two hyperparameters in voting classification. It makes judgments based on predictions of others if set to hard, and if set to soft, it will use the weighted approach.

5. Generation of Explanations Using X-AI Model (LIME)

Figure 7 demonstrates our proposed model's system architecture; adding the LIME model to the designed machine learning pipeline to improve the explainability of the model. Local interpretable model-agnostic explanation (LIME) [42] can explain many machine learning algorithms for regression predictions in a faithful way, using the feature value change of a data sample to transform individual feature values into the predictor's contribution. An explainer can provide a local interpretation of each data sample. For example, interpretable models in LIME often employ linear regression or decision trees that are trained by using tiny perturbations (e.g., adding random noise, removing certain words, and hiding parts of an image) in the model [43,44]. The quality of these models is increasing, and they are being used to resolve a good part of business victimization information. There is also a persistent trade-off between model accuracy and interpretability. Generally, if accuracy needs to be improved, it is recommended to use sophisticated algorithms such as material, boosting, random forest, call trees, and SVM, which are "blackbox" algorithms [45–47]. The Kaggle competition is full of winning entries that rely on algorithms, including XGBoost, because they do not have to explain the method of generating predictions to business users. However, in business settings, simpler models that are easier to understand, such as regression toward the mean, provision regression, call trees, etc. are used, although the predictions are less accurate. The business case for predictions could be made intuitively by using black-box algorithms that can provide us with a high accuracy and interpretability [48,49]. Machine learning models can be deployed more widely by organizations that trust their predictions. The question remains, however, how can trust be built in machine learning models? The model diagram after applying local interpretable model-agnostic explanations (LIME) provides a clear explanation of the issue with black-box classifiers. The LIME approach is a way to understand a machine learning black-box model by perturbing the input data samples and seeing how the predictions change. With LIME, this technique can be applied to any machine learning black-box model. Here are the main steps:

- A TabularExplainer is initially initialized with the data used to train it (or with the statistics of the training data if no training data are present), details about the features, and several class names (if classification is possible).
- In the class explain_instance, a method called explain_instance accepts a reference to the instance for which an explanation is needed, along with the trained model's prediction method and the number of features to be included in the explanation.

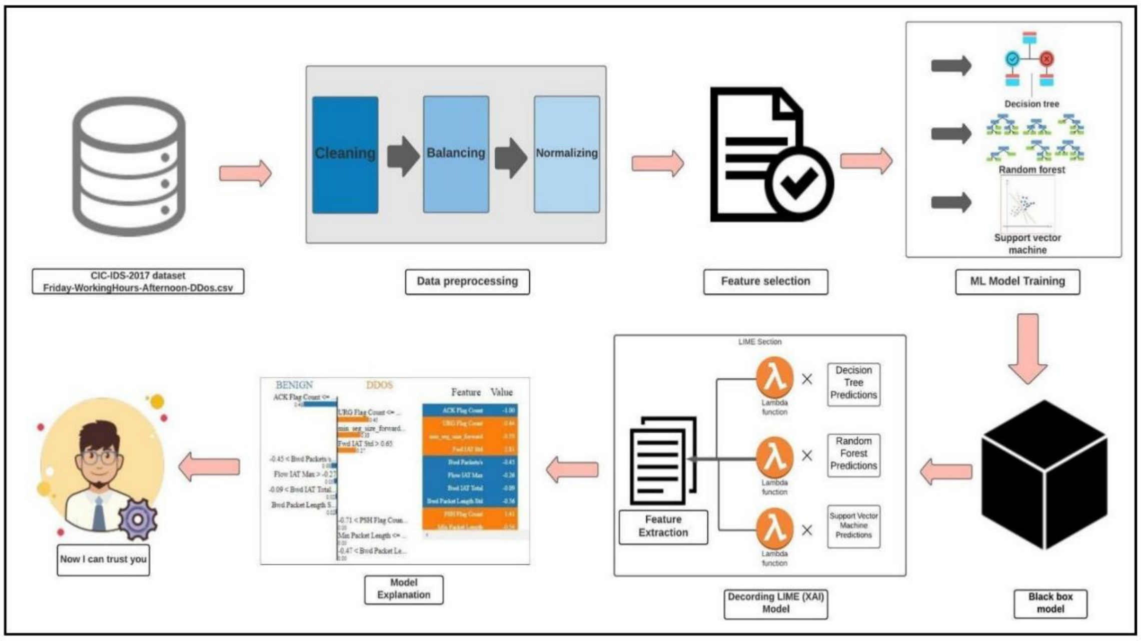


Figure 7. Explainable AI Framework.

Creating a LIME explanation.

explainer=lime. Lime_tabular. LimeTabularExplainer (X_train, feature_names=feature_names,class_names=['BENIGN','DDOS'], categorical_features=cat_columns,categorial_names=feature_names_cat,kernel_width=3) This explanation is divided into three parts:

1. The prediction probabilities are displayed in the left section.
2. The middle section presents 13 of the most important features. The binary classification task would be done in two colors, orange and blue. Class 1 attributes are in orange, and class 0 attributes are in blue. Horizontal bars indicate the relative importance of these features based on floating-point numbers.
3. The color-coding is consistent across sections. A list containing the actual values of the top 13 variables.

Through the LIME framework, classification models, regression models, or supervised models can be developed more efficiently. This work was introduced by Carlos Guestrin, Sameer Singh, and Marco Ribeiro in 2016 [46]. The decision framework is based on the assumption that all complex models are linear at a local level. The LIME model simulates how the complex model behaves at one location, in order to explain the predictions at another location based on the simple model. Tableau, text, and image data types are all supported by LIME. This work illustrates the interpretations LIME made from CICIDS2017 data.

An explanation of the LIME algorithm

- If an explanation is needed, n times of perturbation must occur without a small change in value. Using this fake data, LIME constructs a local linear model around the perturbed observation.
- Outcomes of perturbed data are predicted.
- Measure the distance between each perturbed observation and the original observation.
- Calculate the similarity score based on distance.
- Then determine m features to best represent the predictions from the perturbed data.
- The perturbed data are fitted with a simple model using the selected features.
- The coefficients (feature weights) of the simple model describe the observations.

The pseudo-code for pipeline model building is as follows:

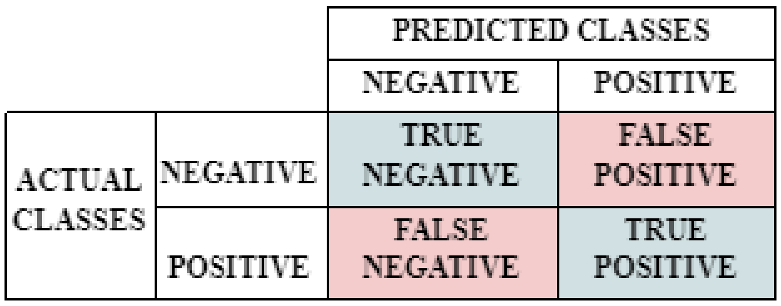
```
IMPORT important libraries including Scikit Learn
IMPORT the dataset
PRE-PROCESSING to impute missing values, replace NaN values, and Infinity values in the dataset.
SCALE the data
STORE various Machine Learning models in a variable 'models'
SET scoring equal to accuracy
SET name as name of the machine learning models
FOR name, Model in models:
    Store value of model_selection using 10 splits in a variable
    Calculate and store results using cross validation score methods of the model selection
    Append results in list of existing results
    Print model accuracy and classification report
END FOR
```

6. Prediction and Evaluation

6.1. Confusion Matrix

A confusion matrix is a performance measurement for machine learning classification problems, where the output can be two or more classes. This confusion matrix appears in the figure below as a table containing four different combinations of predicted and actual values. These combinations can be true negative, false positive, true positive, or false negative.

After training the machine learning model using a decision tree, random forest, SVM, and voting classifier with the given dataset and dealing with the various attacks represented by confusion matrices, a prediction is made. A confusion matrix provides information that is later used to calculate the accuracy, precision, and recall, to generate the evaluation reports of each classifier, as presented in Figure 8.



		PREDICTED CLASSES	
		NEGATIVE	POSITIVE
ACTUAL CLASSES	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

Figure 8. Confusion matrix.

6.2. Accuracy

Based on a confusion matrix, an accuracy calculation can be made using the formula below:

$$Acc = \frac{TN + TP}{(TN + TP + FN + FP)}$$

6.3. Precision

Precision is calculated by using a confusion matrix, as follows:

$$Precision = \frac{TP}{(TP + FP)}$$

6.4. Recall

Calculation of recall is completed using the confusion matrix, with the following formula:

$$recall = \frac{TP}{(TP + FN)}$$

7. Experimental Results and Discussions

Three promising supervised machine learning models were used in this experiment: decision trees, random forests, and support vector machines (SVM). Later, a voting classifier model, which is an ensemble technique, was used as a built-in continuation to achieve the best possible results.

7.1. Result Comparison

All of the feature models showed almost the same accuracy, with very minor differences. With reference to the comparison of the results below, we were able to achieve an accuracy of 96.25 using a voting classifier, as shown in Table 2.

Table 2. Classification Comparison Report.

MODEL	ACCURACY	PRECISION	RECALL	F1-SCORE
Decision Tree	0.95551413679	0.88	0.88	0.88
Random Forest	0.9603538146	0.89	0.88	0.88
Support vector machine	0.9557731796	0.89	0.8	0.8
Voting classifier	0.9625651556	0.89	0.89	0.89

The bar-graph visualization in Figure 9 represents the comparison results of each machine learning model used in this study. The results that are closest to 1 on a scale of 0 to 1 show the best performance. Based on the results above, it appears that the voting classifier performed the best in our experiment, among all the predicted matrices.

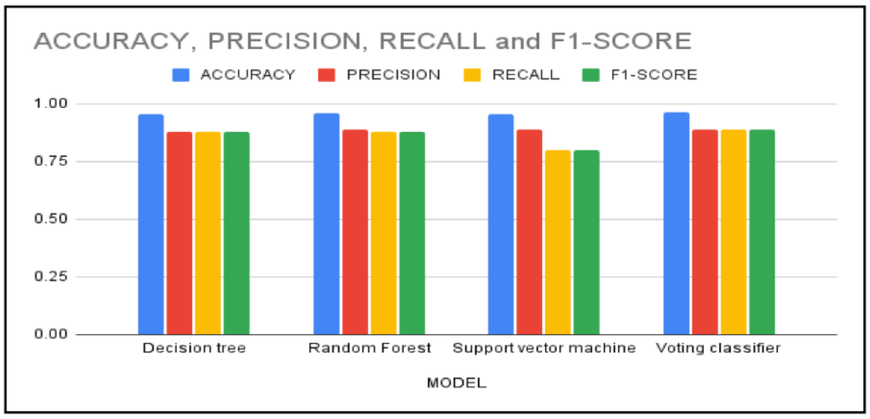


Figure 9. Comparison and results of the models.

7.1.1. Decision Tree

Figure 10 shows a multi-class confusion matrix representation for the decision tree classifier, which shows a summary of the predicted results. By examining the plot, one can understand the relationships between true-positive and false-positive, and true-negative and false-negative, predictions for multi-class classification of 0 to 7. With the confusion matrix, we calculated the model's accuracy and misclassification, both of which are displayed in the figure itself.

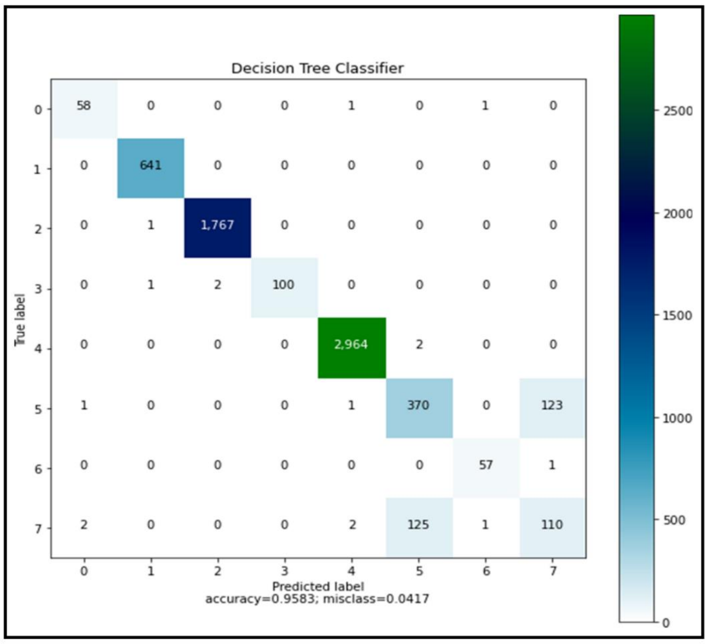


Figure 10. Confusion matrix for decision tree.

7.1.2. Random Forest

Figure 11 depicts the confusion matrix associated with the random forest model.

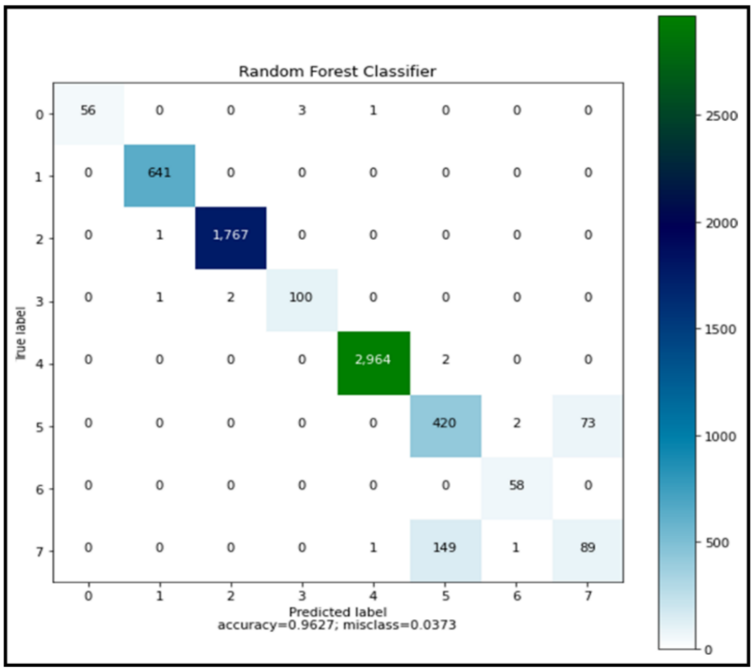


Figure 11. Confusion matrix for a random forest.

The number of correct and incorrect predictions is summarized with count values and broken down by each class. In the figure itself, the confusion matrix allows us to calculate the model's accuracy and misclassification.

7.1.3. Support Vector Machine

Similarly, in Figure 12, we have visualized the prediction results of the support vector machine model as a confusion matrix. A count value is computed for each class for the number of correct and incorrect predictions. Using this confusion matrix, a model accuracy of 96% was achieved.

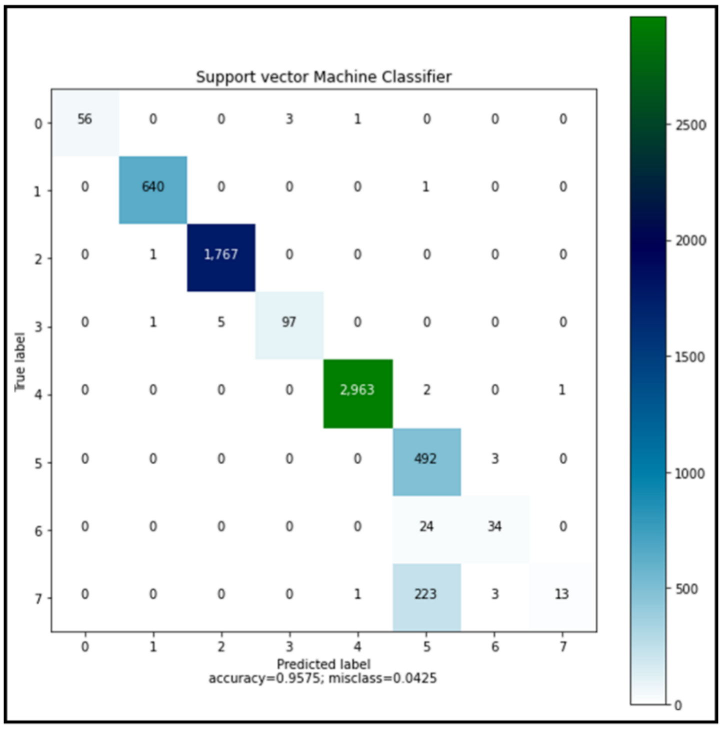


Figure 12. Confusion matrix for the support vector machine.

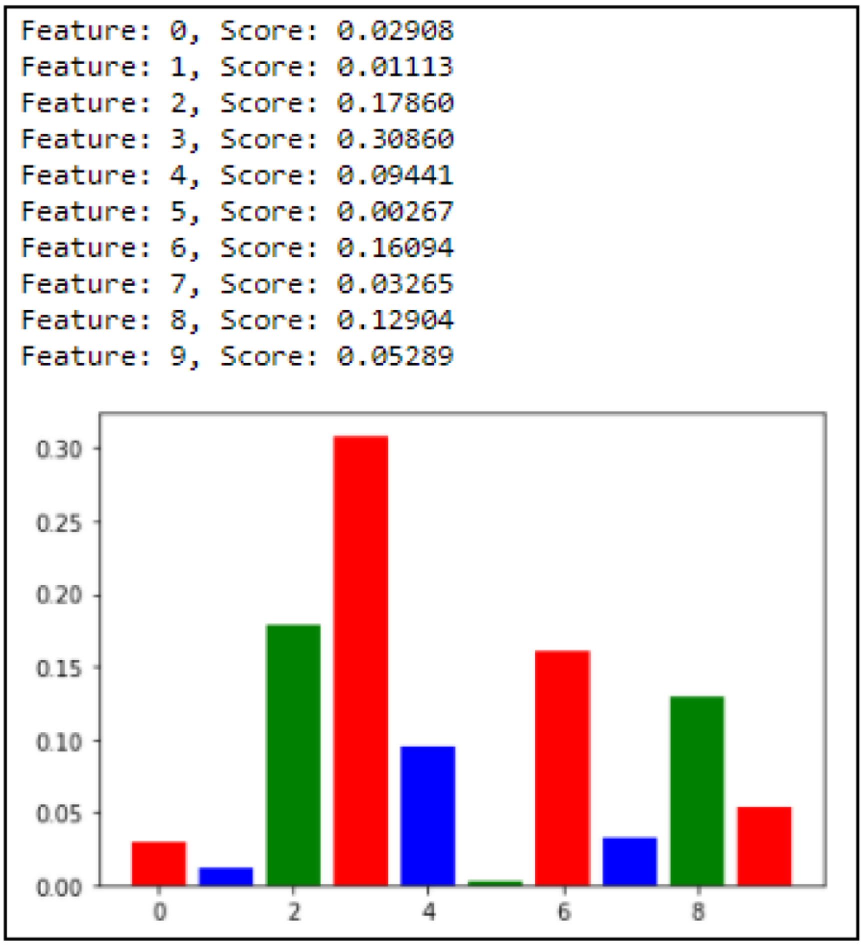
7.1.4. Voting Classifier

The initial extensive experiment involved a feature selection method that targeted the top 10 features in the dataset, as listed in Table 3. Later, we utilized these features to train the machine learning models that are listed below. After successfully training all the machine learning models, we conducted an experiment to plot the feature contribution of each feature for a specific machine learning model.

Table 3. Top 10 features in dataset.

1	Bwd Packet Length Min
2	Bwd Packet Length Std
3	Flow IAT Max
4	Fwd IAT Std
5	Bwd IAT Total
6	Bwd Packets/s
7	Min Packet Length
8	PSH Flag Count
9	ACK Flag Count
10	URG Flag Count

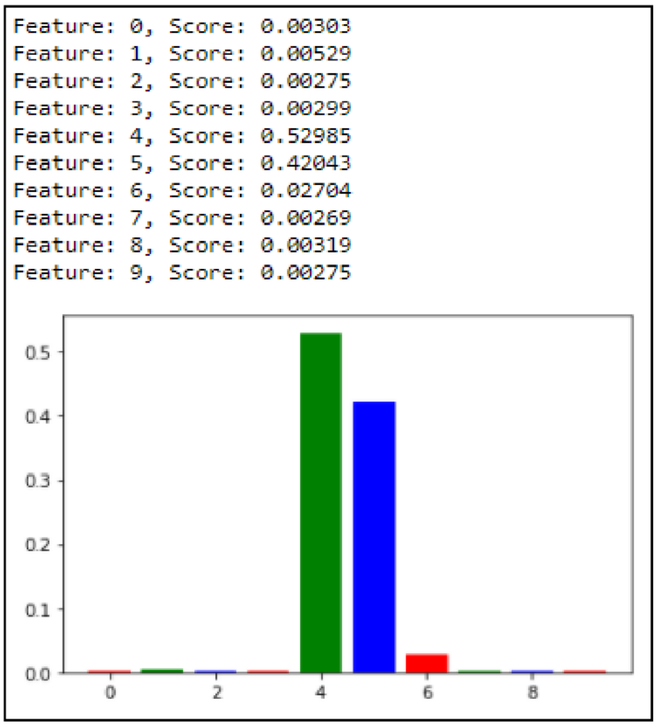
1. A decision tree was used to determine the importance of features on a classification problem involving 1000 samples and 10 features. The importance of split points is estimated using decision tree algorithms, such as classification and regression trees (CARTs), which use standard metrics, such as Gini and entropy, to select split points. Similarly, algorithms such as random forest and stochastic gradient boosting can be applied to ensembles of decision trees. It is possible to use this algorithm as the BasisTreeRegressor and BasisTreeClassifier classes of scikit-learn. After a model has been fitted, the feature_importances property can be accessed, to retrieve the relative importance scores for each input feature. Figure 13 represents the calculated scores for each feature.



```
Feature: 0, Score: 0.02908
Feature: 1, Score: 0.01113
Feature: 2, Score: 0.17860
Feature: 3, Score: 0.30860
Feature: 4, Score: 0.09441
Feature: 5, Score: 0.00267
Feature: 6, Score: 0.16094
Feature: 7, Score: 0.03265
Feature: 8, Score: 0.12904
Feature: 9, Score: 0.05289
```

Figure 13. Feature importance graph for decision tree.

2. The random forest method was used to determine the significance of features in regression problems, as represented in Figure 14. Random forest was implemented as the Random Forest Classifier class and Random Forest Regressor class in scikit-learn, to determine the feature importance. After a model has been fitted, its property of feature importance can be accessed, to retrieve the relative importance of each feature. The bagging and extra trees algorithms can also be used with this approach.

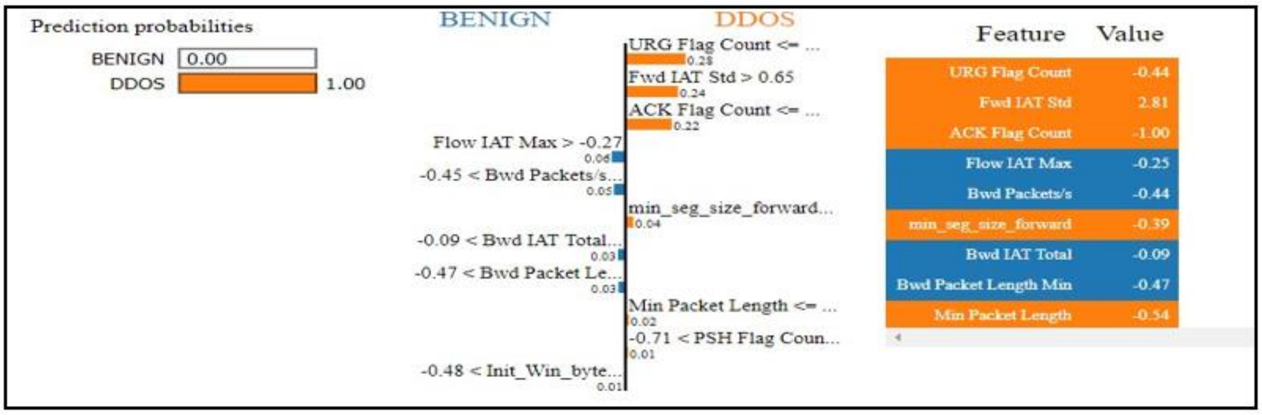


```
Feature: 0, Score: 0.00303
Feature: 1, Score: 0.00529
Feature: 2, Score: 0.00275
Feature: 3, Score: 0.00299
Feature: 4, Score: 0.52985
Feature: 5, Score: 0.42043
Feature: 6, Score: 0.02704
Feature: 7, Score: 0.00269
Feature: 8, Score: 0.00319
Feature: 9, Score: 0.00275
```

Figure 14. Feature importance graph for random.

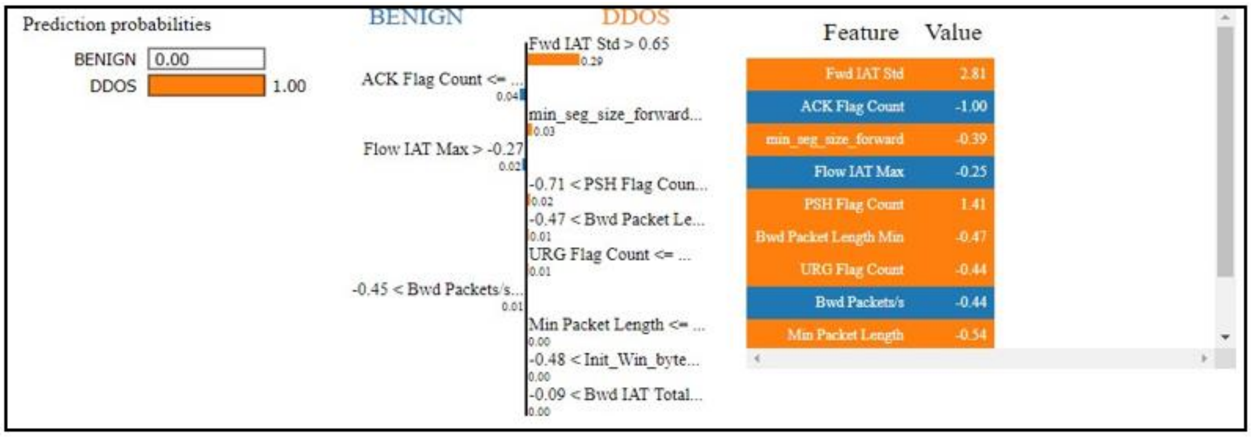
The explanation of the LIME observations for all the algorithms, decision tree, random forest, and support vector machine, are represented in Figures 15–17, respectively. The prediction probabilities are considered for DDOS and BENIGN classes. LIME explains why the probability was assigned in the first place. To calculate the prediction, the probability values are compared with the actual class of the target variable. The observation ID value of 2 was considered for this research study in the validation set. All three algorithms assigned a higher probability to type 2, which is the actual value. However, the probability ranged from −1 to 1.0, with random forest assigning the maximum probability. In addition, the weights assigned by the different algorithms to each feature were quite different. For example, the value of Fwad LAT Std > 2.81 was assigned a weight of −0.26 for the decision tree, a weight of 0.65 for random forest, and 0.65 for SVM. Each feature was then color-coded, to indicate whether it contributed to the prediction of DDOS (Orange), NOT1, or BENIGN (blue) in the feature-value-table. The Feature-Value table by itself shows the actual values of the features for that particular record (here observation or Id = 2).

LIME observations for the different machine learning models:



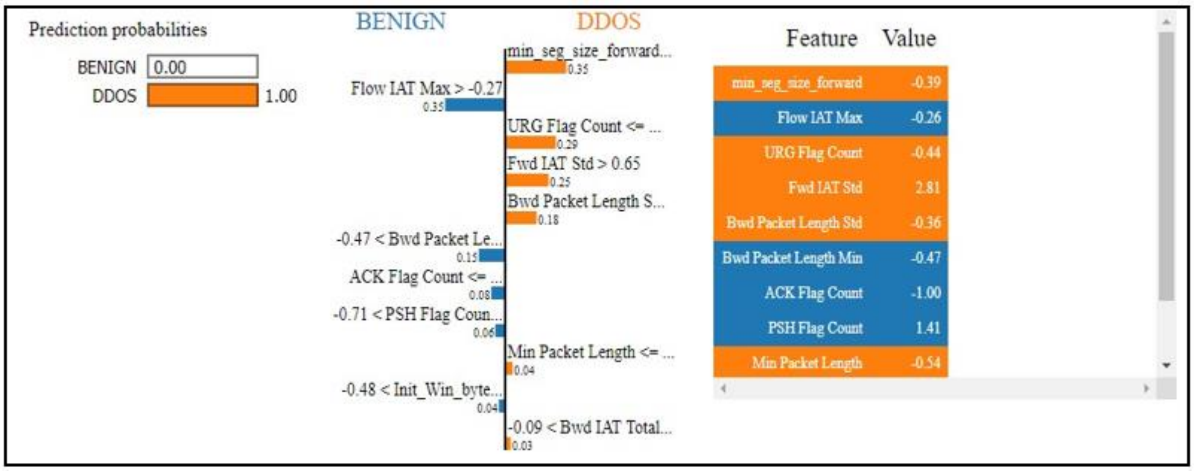
Feature	Value
URG Flag Count	-0.44
Fwd IAT Std	2.81
ACK Flag Count	-1.00
Flow IAT Max	-0.25
Bwd Packets/s	-0.44
min_seg_size_forward	-0.39
Bwd IAT Total	-0.09
Bwd Packet Length Min	-0.47
Min Packet Length	-0.54

Figure 15. LIME observations for decision tree.



Feature	Value
Fwd IAT Std	2.81
ACK Flag Count	-1.00
min_seg_size_forward	-0.39
Flow IAT Max	-0.25
PSH Flag Count	1.41
Bwd Packet Length Min	-0.47
URG Flag Count	-0.44
Bwd Packets/s	-0.44
Min Packet Length	-0.54

Figure 16. LIME observations for random forest.



Feature	Value
min_seg_size_forward	-0.39
Flow IAT Max	-0.26
URG Flag Count	-0.44
Fwd IAT Std	2.81
Bwd Packet Length Std	-0.36
Bwd Packet Length Min	-0.47
ACK Flag Count	-1.00
PSH Flag Count	1.41
Min Packet Length	-0.54

Figure 17. LIME observations for support vector machine.

8. Conclusions and Future Work

The paper proposes an intrusion detection system using machine learning algorithms, such as decision trees, random forests, and SVM (IDS) experiments. After training these models, an ensemble technique, voting classifier, was added, and this achieved an accuracy of 96.25%. This paper contends that effective human–machine interactions require trust. LIME is a modular, extensible approach that describes predictions in a clear, understandable way. For the selection of representative models, an explanation of prediction is very helpful. It is used for deciding between models, assessing trust, improving untrustworthy models, and gaining insights into predictions for experts and non-experts of the system. This work proposes using an ensemble of machine learning models and then applying a LIME explainable framework to understand the model's prediction. The ensemble of ML models showed an improved accuracy of 96.25 for the IDS prediction, and the LIME explanation graphs showcased the prediction performance of the decision tree, random forest, and SVM algorithms. This work could be further extended to the application of explainability to deep learning-based analysis of IDS, and different XAI models could also be tried, such as SHAP, DeepLift, and AIX360.

We are also planning to create an app for real-time data analysis and prediction performance evaluation. In the future, we will implement the concept of explainable AI on various complex datasets, such as ToN_IoT [48,49].

Author Contributions: Conceptualization, S.P., S.K. and K.K.; Data curation, S.M.M., A.S. and O.S.; Formal analysis, V.V., A.S., N.A. and K.S.; Funding acquisition, V.V.; Investigation, V.V. and N.A.; Methodology, S.P., S.M.M., A.S. and S.K.; Project administration, S.K. and K.K.; Resources, N.A. and

O.S.; Software, S.M.M., A.S., N.A., O.S. and K.S.; Supervision, S.P. and K.K.; Validation, V.V., K.S. and K.K.; Visualization, A.S. and O.S.; Writing – original draft, S.M.M. and O.S.; Writing – review & editing, V.V., S.K., K.S. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, M.; Zheng, K.; Yang, Y.; Wang, X. An explainable machine learning framework for intrusion detection systems. *IEEE Access* **2020**, *8*, 73127–73141. [CrossRef]
2. Vigneswaran, R.K.; Vinayakumar, R.; Soman, K.; Poornachandran, P. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; pp. 1–6.
3. Tran, M.-Q.; Elsisi, M.; Liu, M.-K.; Vu, V.Q.; Mahmoud, K.; Darwish, M.M.F.; Abdelaziz, A.Y.; Lehtonen, M. Reliable deep learning and IoT-based monitoring system for secure computer numerical control machines against cyber-attacks with experimental verification. *IEEE Access* **2022**, *10*, 23186–23197. [CrossRef]
4. Elsisi, M.; Tran, M.-Q. Development of an IoT architecture based on a deep neural network against cyber attacks for automated guided vehicles. *Sensors* **2021**, *21*, 8467. [CrossRef]
5. Scott, S.-l.l.; Lundberg, M. A unified approach to interpreting model predictions. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
6. Ribeiro, M.T.; Singh, S.; Guestrin, C. “Why should I trust you?”: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
7. Ribeiro, M.T.C. Lime. 2020. Available online: https://github.com/marcotcr/lime (accessed on 17 July 2022).
8. Sahu, S.K.; Sarangi, S.; Jena, S.K. A detail analysis on intrusion detection datasets. In Proceedings of the 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 21–22 February 2014.
9. AI Explainability 360 (v0.2.0). 2019. Available online: https://github.com/Trusted-AI/AIX360 (accessed on 17 July 2022).
10. Mane, S.; Rao, D. Explaining network intrusion detection system using explainable AI framework. *arXiv* **2021**, arXiv:2103.07110t.
11. Ando, S. Interpreting Random Forests. 2019. Available online: http://blog.datadive.net/interpreting-random-forests/ (accessed on 17 July 2022).
12. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
13. Chen, J.; Li, K.; Tang, Z.; Bilal, K.; Yu, S.; Weng, C.; Li, K. A parallel random forest algorithm for big data in a spark cloud computing environment. *IEEE Transact. Parallel Distrib. Syst.* **2016**, *28*, 919–933. [CrossRef]
14. DeJong, G. Generalizations based on explanations. *IJCAI* **1981**, *81*, 67–69.
15. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 581–585.
16. Hooman, A.; Marthandan, G.; Yusoff, W.F.W.; Omid, M.; Karamizadeh, S. Statistical and data mining methods in credit scoring. *J. Dev. Areas* **2016**, *50*, 371–381. [CrossRef]
17. Islam, S.R.; Eberle, W.; Bundy, S.; Ghafoor, S.K. Infusing domain knowledge in ai-based ”black box” models for better explainability with application in bankruptcy prediction. *arXiv* **2019**, arXiv:1905.11474.
18. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection systems. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), New York, NY, USA, 3–5 December 2016; pp. 21–26.
19. Li, Z.; Sun, W.; Wang, L. A neural network-based distributed intrusion detection system on a cloud platform. In Proceedings of the IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, Hangzhou, China, 30 October–1 November 2012; Volume 1, pp. 75–79.
20. Lipovetsky, S.; Conklin, M. Analysis of regression in game theory approach. *Appl. Stoch. Models Bus. Ind.* **2001**, *17*, 319–330. [CrossRef]
21. Lundberg, S. Shap vs. Lime. 2019. Available online: https://github.com/slundberg/shap/issues/19 (accessed on 17 July 2022).
22. Ferdiana, R. A systematic literature review of intrusion detection system for network security: Research trends, datasets and methods. In Proceedings of the 4th International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 10–11 November 2020; pp. 1–6.
23. Peddabachigari, S.; Abraham, A.; Grosan, C.; Thomas, J. Modeling intrusion detection system using hybrid intelligent systems. *J. Netw. Comput. Appl.* **2007**, *30*, 114–132. [CrossRef]
24. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and robust federated learning through personalization. In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 6357–6368.

25. Mohseni, S.; Wang, H.; Yu, Z.; Xiao, C.; Wang, Z.; Yadawa, J. Practical machine learning safety: A survey and primer. *arXiv* **2021**, arXiv:2106.04823.
26. Kishore, R. Evaluating Shallow and Deep Neural Networks for Intrusion Detection Systems Cyber Security. Doctoral Dissertation, Amrita School of Engineering, Amritapuri, India, 2020.
27. Hoque, M.S.; Mukit, M.; Bikas, M.; Naser, A. An implementation of intrusion detection system using genetic algorithm. *arXiv* **2012**, arXiv:1204.1336.
28. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access* **2021**, *9*, 22351–22370. [CrossRef]
29. Laqtib, S.; El Yassini, K.; Hasnaoui, M.L. A technical review and comparative analysis of machine learning techniques for intrusion detection systems in MANET. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 2701. [CrossRef]
30. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transact. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
31. Mukherjee, S.; Sharma, N. Intrusion detection using naive Bayes classifier with feature reduction. *Procedia Technol.* **2012**, *4*, 119–128. [CrossRef]
32. Kumar, V.; Chauhan, H.; Panwar, D. K-means clustering approach to analyze NSL-KDD intrusion detection dataset. *Int. J. Soft Comput. Eng.* **2013**, *4*, 2231–2307.
33. Sharafaldin. *Intrusion Detection Evaluation Dataset (CICIDS2017), Canadian Institute for Cybersecurity, January, 2018*. Available online: https://www.unb.ca/cic/datasets/ids2017.html (accessed on 17 July 2022).
34. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
35. Radoglou-Grammatikis, P.I.; Sarigiannidis, P.G. Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. *IEEE Access* **2019**, *7*, 46595–46620. [CrossRef]
36. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]
37. Mohammadi, M.; Rashid, T.A.; Karim, S.H.; Aldalwie, A.H.M.; Tho, Q.T.; Bidaki, M.; Rahmani, A.M.; Hosseinzadeh, M. A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. *J. Netw. Comput. Appl.* **2021**, *178*, 102983. [CrossRef]
38. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
39. Da Costa, K.A.; Papa, J.P.; Lisboa, C.O.; Munoz, R.; de Albuquerque, V.H.C. Internet of things: A survey on machine learning-based intrusion detection approaches. *Comput. Netw.* **2019**, *151*, 147–157. [CrossRef]
40. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A deep learning approach to network intrusion detection. *IEEE Transact. Emerg. Topics Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
41. Štrumbelj, E.; Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* **2014**, *41*, 647–665. [CrossRef]
42. Ribeiro, M.T.; Singh, S.; Guestrin, C. Model-agnostic interpretability of machine learning. *arXiv* **2016**, arXiv:1606.05386.
43. Adadi, A.; Berrada, M. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [CrossRef]
44. Gunning, D.; Stefik, M.; Choi, J.; Miller, T.; Stumpf, S.; Yang, G.Z. XAI—Explainable artificial intelligence. *Sci. Robot.* **2019**, *4*, eaay7120. [CrossRef]
45. Tjoa, E.; Guan, C. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transact. Neural Netw. Learn. Syst.* **2020**, *32*, 4793–4813. [CrossRef]
46. Wolf, C.T. Explainability scenarios: Towards scenario-based XAI design. In Proceedings of the 24th International Conference on Intelligent User Interfaces, Marina del Ray, CA, USA, 17–20 March 2019; pp. 252–257.
47. Das, A.; Rad, P. Opportunities and challenges in explainable artificial intelligence (XAI): A survey. *arXiv* **2020**, arXiv:2006.11371.
48. Byrne, R.M.J. Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning. *IJCAI* **2019**, 6276–6282.
49. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Hartog, F.T.H.d. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet Things J.* **2022**, *9*, 485–496. [CrossRef]