Machine Learning-Driven Network Security: Behavior Analysis for IoT DDoS Attack Detection

DATA 245 : Machine Learning Tec
Submitted by (Group-4)
Aradhya Alva Rathnakar
Bhavan Kumar Basavaraju
Mahamaya Panda
Namratha Sampath Kumar
Shashi Kumar Kadari Mallikarjuna

***Abstract*—This project aims to enhance IoT network security through machine learning-driven behavior analysis for Distributed Denial of Service (DDoS) attack detection. The proliferation of IoT devices has introduced new vulnerabilities that cyber attackers exploit to launch DDoS attacks from compromised botnets. Using over 11 million records from the Aposemat IoT-23 dataset containing benign and malicious network traffic labels, extensive data preparation including cleaning, transformation, and sampling is performed. Machine learning algorithms such as Random Forest Classifier, Support Vector Machine, and K-Nearest Neighbors are developed and evaluated to classify DDoS and non-DDoS attacks based on network features. Ensemble techniques that combine classifier predictions are also explored. The models demonstrate high accuracy in detecting DDoS attacks from patterns in IoT traffic. Evaluation of individual models and their ensemble techniques reveal Random Forest and KNN outperform SVM. This research thus confirms the potential of machine learning approaches to analyze IoT network behavior and detect security threats like DDoS attacks through network traffic patterns. By providing early detection of such attacks, these techniques can help strengthen security for rapidly growing IoT ecosystems.**

***Index Terms*—IoT security, Machine Learning, DDoS attack detection, Behavior Analysis, Network Traffic Analysis, Ensemble Techniques**

I. Introduction

Internet of Things (IoT) enables easy access to cloud services but faces severe Distributed Denial of Service (DDoS) threats. Many IoT devices have limited security built-in, which makes them vulnerable to attackers to enlist them into botnets. Attackers are then able to remotely control large number of these compromised IoT devices and use them to bombard important systems with junk traffic in DDoS attacks. Currently, IoT networks are under attack regularly as perpetrators exploit vulnerable devices to flood websites, web services, and other internet-connected systems with immense amounts of data traffic. With more internet-connected devices being brought online each day, security experts express concern that the scale and frequency of DDoS attacks spawned from compromised IoT networks will continue intensifying unless manufacturers make security a higher priority and users implement stronger access controls and monitoring on their devices.

A. Problem Statement

In recent years, the Internet of Things (IoT) has witnessed exponential growth, enabling many operations that enhance daily lives. However, this proliferation of IoT also presents new security challenges, particularly when defending against DDoS attacks and other network threats. These malicious attacks can disrupt critical IoT services, leading to downtime, data breaches, and financial losses. A visionary and adaptive approach is essential to safeguard IoT ecosystems from similar pitfalls. This project explores the use of machine learning-driven network security strategies, specifically focusing on behavior analysis techniques, to describe and alleviate DDoS attacks in IoT systems. By utilizing the power of ML algorithms such as Random Forest Classifier, Support Vector Machine, and K-Nearest Neighbour, the aim is to enhance the adaptability of IoT networks and ensure the continued operation of IoT devices and services in the face of evolving cyber pitfalls.

B. Project Background

The project aims to detect DDoS attacks in IoT environments using machine learning. Aposemat IoT-23 dataset is utilized for this project that contains over 11 million records of benign and malicious IoT network traffic, including labels for various attacks, command and control activity, and DDoS. The link to the dataset is provided under the references below. Extensive data cleaning, transformation, and exploratory analysis are performed on the dataset. A variety of machine learning models, such as RFC, KNN, and SVM, are developed and evaluated on the task. Ensemble techniques that combine multiple classifiers are also explored and implemented. The models demonstrate high accuracy in detecting DDoS attacks from IoT network traffic patterns. Thus, the project outcomes show that the machine learning-driven approaches can effectively analyze IoT network behavior and traffic characteristics to identify such threats, thus strengthening security against DDoS crimes targeting IoT networks.

C. Literature Survey

Analytics evolves continuously to adapt to changing needs and requirements, shaping the project idea based on concepts

essential for fast-paced, machine learning data-driven environments. Therefore, a few concepts are incorporated below to enhance the project's resilience against dynamic DDoS attacks.

[1] In the study by Shahid et al. (2018), six different machine learning algorithms are tested for classifying bidirectional flows according to the IoT device they belong to; Random Forest, Decision Tree, SVM, k-Nearest Neighbors, Artificial Neural Network (ANN), and Gaussian Naive Bayes. The Random Forest classifier achieves the best performance with an overall accuracy of 99.9% on the test set, followed by other algorithms with high overall accuracy ranging between 98.6% and 99.9%. The t-distributed Stochastic Neighbor Embedding (t-SNE) visualization technique demonstrates the discriminative power of the selected features to differentiate network traffic generated by different IoT devices. The study also highlights the limitations of the experimental network, such as the small number of devices used and the need for larger datasets to improve the performance of the ANN model. Thus, the results indicate the potential for high-accuracy recognition of IoT devices through passive analysis of network traffic, with the Random Forest classifier showing the most promising performance

[2] Another study by Patel et al. (2018), compares the accuracy of four basic Machine Learning algorithms, Naïve Bayes, K-nearest neighbor (KNN), Decision Tree, and Support Vector Machine (SVM), in classifying network traffic data. The study collects features using the Wireshark tool (which helps to capture network packets and display them at a granular level) and trains and tests the models using Python Libraries. The results show that KNN outperforms Naive Bayes, Decision Tree, and SVM, achieving the highest mean accuracy of 92.42% with K set to 11 in an internet traffic classification task. This study influences the consideration of KNN as a potential candidate in the project's modeling phase.

[3] In a study by Liang et al. (2021), machine learning algorithms are being explored for their pivotal role in detecting abnormalities and outliers in IoT networks and devices. Security solutions in the IoT domain often leverage SVM, RFC, logistic regression, neural networks, and ensemble supervised learning models, showcasing notable accuracy rates in identifying issues like DDoS attacks, authentication attacks, and malware detection. The data is collected from IoT devices utilizing edge computing and discovers that SVM surpasses RFC and Logistic Regression. Also, SVM achieves an impressive 95.24% accuracy in detecting DDoS attacks. This insightful finding significantly influences decision-making during the project's modeling phase, aiding in the selection of an appropriate algorithm for effective DDoS attack detection. Despite these successes, the broader implementation of machine learning-based security solutions in IoT encounters challenges, including the need for large and accurate datasets, algorithm selection, and the management of outliers and vague data. Further research is deemed essential to enhance the overall effectiveness of machine learning approaches in addressing IoT security challenges.

[4] A paper by Neto et al. (2023) utilizes five machine learning models, namely Logistic Regression, Perceptron, Adaptive Boosting (AdaBoost), Random Forest, and Deep Neural Network, to classify attacks in IoT operations. The results indicate that Logistic Regression, Perceptron, and Adaboost methods exhibit lower accuracy and F1-scores, with average accuracy below 80% and F1-scores less than 50% for all cases. In contrast, both Random Forest and Deep Neural Network maintain high accuracy and F1 scores, even in the face of a slight reduction compared to the eight-class challenge. These models demonstrate the potential of machine learning methods in classifying attacks against IoT operations, highlighting the possibility of using optimized methods and similar strategies to address IoT-specific problems. Thus, their study also provides insights into the nuanced trade-off between accuracy and F1-score in model evaluation for this project.

II. CRISP-DM Methodology

The project adheres to the CRISP-DM methodology, also known as the Cross Industry Standard Process for Data Mining. Its well-defined structure ensures clarity and facilitates a comprehensive understanding of the project. With the iterative process in place, enhancements can be easily incorporated. Furthermore, the methodology enables the application of enhanced project management principles, allowing for effective project planning, resource allocation, and risk assessment. In this project, the six stages in CRISP-DM are Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The CRISP-DM methodology is illustrated in the figure below.

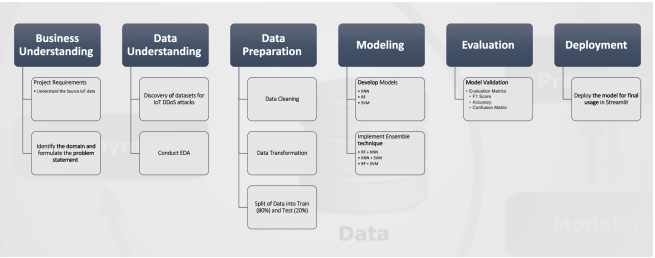

Fig. 1. CRISP-DM Methodology

1) Business Understanding: The intent is to acquire detailed information about the project being executed and the data (IoT data) used to address the problem. The domain needs identification, followed by the formulation of the problem statement. The emphasis is on laying a strong foundation of knowledge about the project's background and objectives, allowing an in-depth understanding of the challenges at hand.

2) Data Understanding: This phase includes collecting data for IoT DDoS attacks, and exploratory data analysis is conducted to identify outliers, inconsistencies, and anomalies within the dataset, aiding in understanding the inherent patterns and characteristics of the data. This preliminary analysis serves as a foundation to make informed decisions and guides in formulating an effective data-cleaning strategy for the next phase.

3) Data Preparation: Data cleaning is the first step of this phase, where all issues are identified as a part of exploratory data analysis is addressed. This involves getting rid of outliers, and inconsistencies and handling anomalies within the dataset.

The goal is to ensure the data's integrity, accuracy, and consistency, thereby laying the groundwork for subsequent stages in the CRISP-DM process. The data is then split into train (80%) and test (20%) accordingly, making it ready for the next phase.

4) Modeling: This phase includes identifying models that would best suit the problem. To classify between DDoS and Non-DDoS, KNN, RF, and SVM are used along with ensemble techniques, including (RF and KNN), (KNN and SVM), and (RF and SVM). Strategic model selection helps achieve accurate and robust classification, essential for effectively addressing the challenges posed by IoT DDoS attacks.

5) Evaluation: In this phase, the developed algorithms are implemented and evaluated using evaluation metrics to assess their performance. Model validation is done, and the results are provided using a specific set of metrics: F1 score, accuracy, and confusion matrix. Evaluation helps in understanding the value and reliability of the information obtained from the data.

6) Deployment: The last phase includes the deployment of the final model implemented in the production environment. This involves integrating the selected model into existing systems, ensuring smooth functionality, and monitoring its performance in real-world circumstances. A successful deployment guarantees that the designed solution is put to use, bringing value to end-users.

III. System Architecture

DDoS attack detection solution development involves the usage of various tools and techniques to enable the user to leverage the implemented solution. Google Drive is leveraged to store the downloaded Aposemat IoT-23 dataset ranging from 2018-2019 which is the data source. Google Colab is used for the solution implementation which enables parallel development by using Git as a version control tool. The raw data is first read using the pandas library on Google Colab, on which exploratory data analysis (EDA) is performed to understand the data and formulate a plan to prepare the data to be used by the machine learning models. After the EDA is performed, the raw dataset is cleansed to handle missing values, duplicate values, rename columns, and drop the unwanted columns. The cleansed dataset is then transformed into a format that the machine learning models can interpret. Categorical values are converted into numerical values, after which binning, data standardization, data regularization, and oversampling of minority classes are performed. The transformed data is then prepared for consumption by the machine learning models by splitting and using 80% of the data for training and the remaining 20% of the data for testing.

Different classification-related machine learning models like Random Forest Classifier (RFC), K-Nearest Neighbors (KNN) algorithm, and Support Vector Machine (SVM) are used for solution implementation and tested for accuracy. Voting-based ensemble technique is used where three different combinations of the above three models (SVM-RFC, RFC-KNN, and KNN-SVM) are developed and the ensemble model with the best prediction accuracy is chosen for deployment and is loaded into the AWS S3 bucket. The ML repository uses the final version of the model and connects it to the front-end application in Streamlit for users to access. The front-end application accepts multiple records of different network parameters in a CSV file and returns the number of DDoS and Non-DDoS attacks which would help the end-user understand the number of DDoS attack attempts on their system.

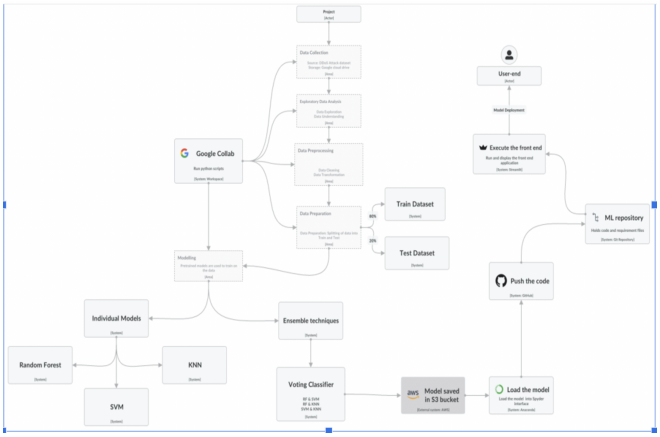

Fig. 2. System Architecture

IV. Data Preparation

In the third phase of the CRISP-DM model, data preparation plays a crucial role in ensuring the quality of data before it can be used for modeling. This phase involves several key steps, including Exploratory Data Analysis, data pre-processing (data cleaning), and data transformation. Exploratory Data Analysis involves exploring and understanding the dataset by analyzing its statistical properties and identifying any potential outliers or patterns. Data pre-processing involves cleaning the data, handling missing values, and transforming the data into a format that can be used for modeling. This step includes techniques such as label encoding and KNN Imputation, data standardization, random over-sampler, and data regularization. Finally, the dataset is split into training, validation, and testing sets to evaluate the model's performance. The workflow for the data preparation can be seen in the below figure.

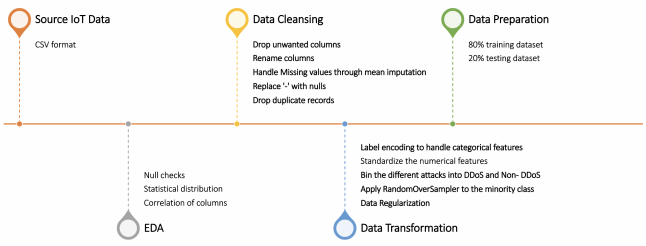

Fig. 3. Data Preparation Workflow

A. Data Exploration

The IoT data sourced from the Aposemat IoT-23 is used for predicting the DDoS and the Non-DDoS attacks. Exploratory Data Analysis is being conducted on the obtained dataset where the goal is to summarize and visualize the main

characteristics of the dataset. It is of vital importance for handling missing data, outlier detection, feature engineering, data distribution, pattern recognition, and data quality checks.

There are a total of 11,448,425 records, and it is seen in the below figure that the datatypes of 14 columns are of the ‘object’ data type, and seven columns are of the ‘float64’ data type from which it is understood that the column types need to be amended to support further data analysis. Numerical columns can be converted from float64 to integer datatypes if they are whole numbers to save memory space and dates should be stored as ‘datetime’ datatype for effective analysis. Renaming of columns, addition of new columns through calculated values, and deletion of unwanted columns are necessary.

#	Column	Dtype
0	#types	object
1	t	object
2	uidstring	object
3	addr	float64
4	port	object
5	haddr	float64
6	pport	object
7	enum	object
8	sstring	object
9	interval	object
10	bcount	object
11	rcount	object
12	connstring	object
13	obool	object
14	rbool	float64
15	mbcount	object
16	hstring	float64
17	opcount	float64
18	obcount	float64

Fig. 4. Columns with ’object’ and ’float64’ Datatypes

The count of null values in the column list is observed to gauge the amount of possible discrepancies in the data and enhance the quality of data before further processing. It is seen in the below figure that most of the columns have two null values except the ‘t’ and ‘#types’ columns which have one and zero null values.

```
#types          0
t               1
uidstring       2
addr            2
port            2
haddr           2
pport           2
enum            2
sstring         2
interval        2
bcount          2
rcount          2
connstring      2
obool           2
rbool           2
mbcount         2
hstring         2
opcount         2
obcount         2
rpcount         2
ipbcount        2
dtype: int64
```

Fig. 5. Null Values Distribution in Columns

1) Correlation Matrix: Correlation matrix analysis reveals relationships among attributes related to network attacks. It describes how each numerical attribute is correlated with others in terms of a correlation index that represents the relation as a magnitude. The attributes ’obcount’ and ’rp count’ show the highest correlation in the dataset, whereas ’opcount’ and ’haddr’ exhibit the least correlation. The color of each tile in the plot helps identify the depth of correlation and reference values are provided in the side scale. Understanding this correlation is crucial for shaping effective feature engineering in attack detection. The Correlation Matrix Heatmap can be seen in the below figure.

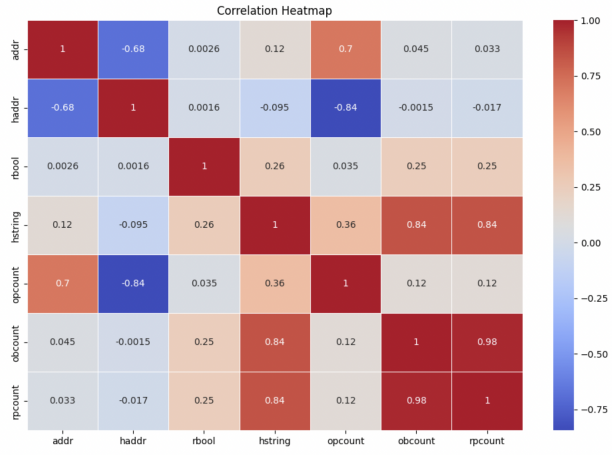

Fig. 6. Correlation Matrix Heatmap

2) Pair Plot: The Pair plot visualization offers insights into relationships and distributions in the dataset. The plot helps to understand the distribution of data values corresponding to other attribute values and is useful for identifying potential correlations and patterns related to DDoS attack activity. The tilt in the values when compared with different attributes explains the influence of the attributes on that respective value and gives insights into the dependency ratio. The Pair plot can be seen in the below figure.

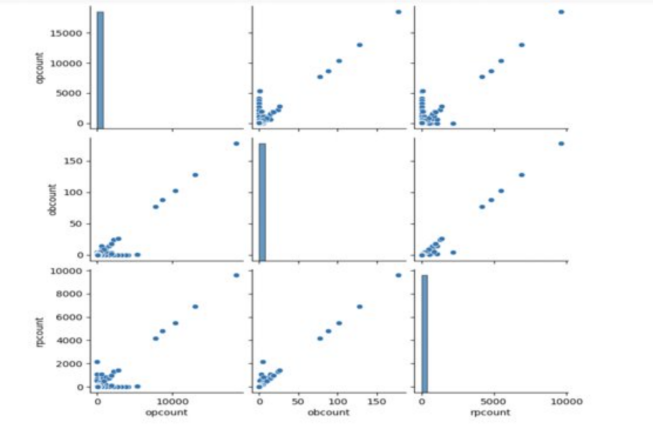

Fig. 7. Pair Plot

3) Bar Plot: The bar plot given below categorizes data into ’Non DDoS’ and ’Malicious DDoS’. The ’Non DDoS’

count is 11,408,840, and the 'Malicious DDoS' count is 39,584, indicating a class imbalance. It can be derived from the plot below that there is a tilt in category class towards 'Non DDoS' which can lead to improper model training or overfitting. The high population of a single class data value will pollute the entire analysis leading to an unfair accuracy. Hence, distribution is crucial for building effective DDoS detection models.

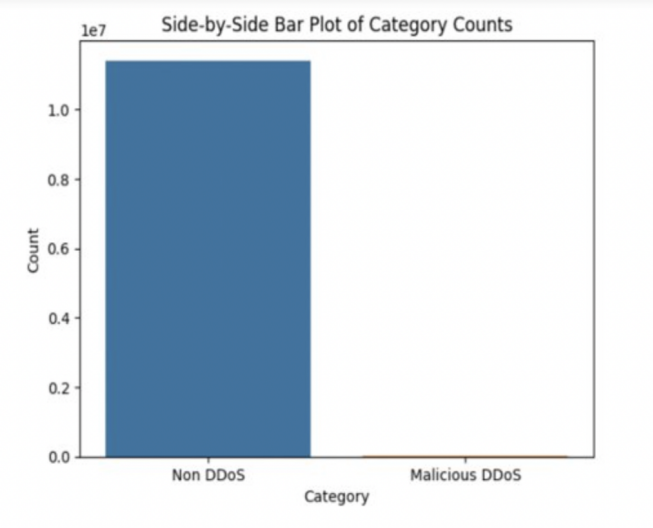

Fig. 8. Bar Plot

4) Pair Box Plot: The Pair box plot given below displays the statistical distribution of attribute values that are necessary for modeling. It gives insights about potential outliers, where the data points are focused on mean, mode, and distribution. For hstring, opcount, obcount, and rpcount, the values are closer to the median which indicates they follow a normal distribution curve. The values for addr are spread out evenly since it is a unique identifier and indicates it follows a uniform distribution curve. Box plots are versatile and can be used for single or multiple variables, making them a valuable tool in exploratory data analysis.

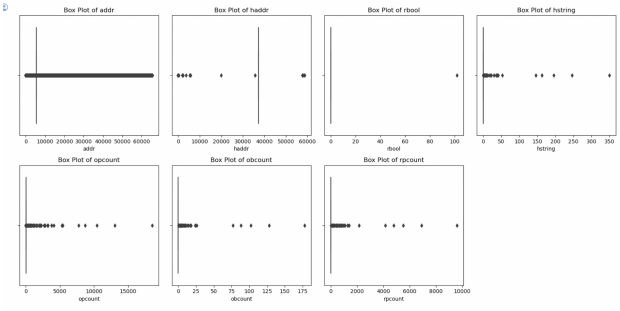

Fig. 9. Pair Box Plot

B. Data Cleaning

The IoT data that is sourced from the Aposemat IoT-23 and is used for predicting the DDoS and the Non-DDoS attacks. The raw data is first cleansed based on the exploratory data analysis before it can be transformed into the machine learning model interpretable format. Most of the data cleansing can be done using the pandas library. The below figure shows the sample of the raw dataset before initiating the cleansing process.

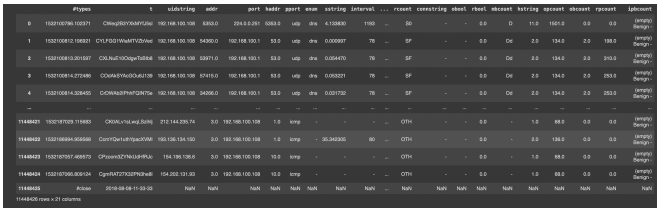

Fig. 10. Sample Raw Data

The columns in the raw data as shown in the above figure are not clear and hence need to be renamed appropriately to the human interpretable form. The column #types is renamed to 'type', t is renamed as 'ts_time', uidstring is renamed as 'uid_string', addr is renamed as 'id.orig_addr', port is renamed as 'id.orig_port', haddr is renamed as 'id.resp_haddr', pport is renamed as 'id.resp_pport', enum is renamed as 'proto_enum', sstring is renamed as 'service_string', interal is renamed as 'duration_interval', bcount is renamed as 'orig_bytes_count', rcount is renamed as 'resp_bytes_count', connstring is renamed as 'conn_state_string', obool is renamed as 'local_orig_bool', rbool is renamed as 'local_resp_bool', mbcount is renamed as 'missed_bytes_count', hstring is renamed as 'history_string', opcount is renamed as 'orig_pkts_count', obcount is renamed as 'orig_ip_bytes_count', rpcount is renamed as 'resp_pkts_count', and ipbcount is renamed as 'Category'. The below figure shows the sample data after renaming the columns.

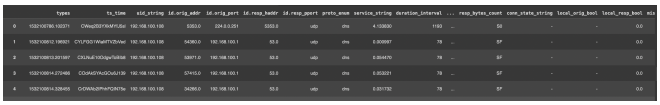

Fig. 11. Sample Raw Data After Renaming Columns

After careful consideration of the necessary columns for modeling purposes, the unnecessary columns which include id.resp_pport, resp_bytes_count, ts_time, proto_enum, conn_state_string, local_orig_bool, service_string, duration_interval, orig_bytes_count, local_resp_bool, and types are dropped. The below figure shows the available columns after dropping the unnecessary columns.

```
Index(['uid_string', 'id.orig_addr', 'id.orig_port', 'id.resp_haddr',
       'missed_bytes_count', 'history_string', 'orig_pkts_count',
       'orig_ip_bytes_count', 'resp_pkts_count', 'Category'],
      dtype='object')
```

Fig. 12. Available Columns After Dropping Unnecessary Columns

The dataset is then checked for blank rows to ensure there are no empty records and is handled by deleting the blank rows. There are a few column values that have the value '-' which is a blank value, and is replaced with a null value to

reduce the storage space needed to store the cleansed data. The dataset is also checked for duplicate rows and is handled by dropping the duplicate records. The below figure shows the sample cleansed dataset after handling blank records, duplicate records, and replacing '-' with nulls.

	uid_string	id.orig_addr	id.orig_port	id.resp_haddr	missed_bytes_count	history_string	orig_pkts_count	orig_ip_bytes_count	resp_pkts_count	Category
0	192.168.100.108	5353.0	224.0.0.251	5353.0	D	11.0	1501.0	0.0	0.0	(empty) Benign -
1	192.168.100.108	54360.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	198.0	(empty) Benign -
2	192.168.100.108	53971.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	310.0	(empty) Benign -
3	192.168.100.108	57415.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	253.0	(empty) Benign -
4	192.168.100.108	34266.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	253.0	(empty) Benign -

Fig. 13. Cleansed Dataset

C. Data Transformation

After the raw data is cleansed, the cleansed dataset needs to be transformed into a format that can be interpreted by the machine learning models like SVM, KNN, and Random Forest Classifier that will be used to classify DDoS and Non-DDoS attacks based on network-related features. The below figure shows the cleansed dataset that needs to be transformed.

	uid_string	id.orig_addr	id.orig_port	id.resp_haddr	missed_bytes_count	history_string	orig_pkts_count	orig_ip_bytes_count	resp_pkts_count	Category
0	192.168.100.108	5353.0	224.0.0.251	5353.0	D	11.0	1501.0	0.0	0.0	(empty) Benign -
1	192.168.100.108	54360.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	198.0	(empty) Benign -
2	192.168.100.108	53971.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	310.0	(empty) Benign -
3	192.168.100.108	57415.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	253.0	(empty) Benign -
4	192.168.100.108	34266.0	192.168.100.1	53.0	Dd	2.0	134.0	2.0	253.0	(empty) Benign -

Fig. 14. Dataset Before Transformation

1) Label Encoder and KNN Imputation: The dataset has a categorical column called missed_bytes_count that needs to be converted into a numerical value. Label encoder is used to convert the categorical values in the missed_bytes_count column where each unique value is assigned a unique numerical value. The fit_transform function is used along with the label encoder which belongs to the sklearn preprocessing library to convert the categorical values into numerical values. After performing the label encoding, this column is checked for any null values. The null values in the missed_bytes_count column are handled using the KNN Imputer. The KNN Imputer replaces the missing values with the mean value of the KNN. In this case, the mean value of the five nearest neighbors is used to replace the null value. The below figure shows the sample dataset after label encoding and KNN Imputation.

	uid_string	id.orig_addr	id.orig_port	id.resp_haddr	missed_bytes_count	history_string	orig_pkts_count	orig_ip_bytes_count	resp_pkts_count	Category
0	192.168.100.108	5353.0	224.0.0.251	5353.0	1.0	11.0	1501.0	0.0	0.0	(empty) Benign -
1	192.168.100.108	54360.0	192.168.100.1	53.0	3.0	2.0	134.0	2.0	198.0	(empty) Benign -
2	192.168.100.108	53971.0	192.168.100.1	53.0	3.0	2.0	134.0	2.0	310.0	(empty) Benign -
3	192.168.100.108	57415.0	192.168.100.1	53.0	3.0	2.0	134.0	2.0	253.0	(empty) Benign -
4	192.168.100.108	34266.0	192.168.100.1	53.0	3.0	2.0	134.0	2.0	253.0	(empty) Benign -

Fig. 15. Dataset After Label Encoding and KNN Imputation

2) Data Standardization: Data standardization is a crucial preprocessing step that is used to bring numerical features within a consistent scale. In this context, the StandardScaler technique is utilized, which transforms the data in such a way that the mean of the features becomes zero, and the standard deviation becomes one. This process ensures that all numerical variables share a standardized scale, preventing features with inherently larger values from disproportionately influencing the machine learning algorithm. The motivation behind this standardization lies in the desire to provide equal weight to all features during model training, promoting fair consideration by the algorithm. SVM and KNN are known to be sensitive to variations in input feature scales. By standardizing the data, these models become less susceptible to the dominance of features with larger scales, enhancing their overall performance and interpretability.

The significance of data standardization becomes evident, especially when dealing with SVM and KNN models. SVMs aim to find an optimal hyperplane for classification, and KNN relies on distance metrics to identify nearest neighbors. In both cases, the scale of features directly impacts the model's ability to discern patterns and make accurate predictions. By standardizing the dataset, these models can operate more effectively, as the standardized features contribute more uniformly to the learning process. The provided figure visually represents the dataset after undergoing the standardization process for specific columns, namely 'history_string,' 'orig_pkts_count,' 'orig_ip_bytes_count,' and 'resp_pkts_count,' offering a clear depiction of the standardized values for these variables.

	uid_string	id.orig_addr	id.orig_port	id.resp_haddr	missed_bytes_count	history_string	orig_pkts_count	orig_ip_bytes_count	resp_pkts_count	Category
0	192.168.100.108	5353.0	224.0.0.251	5353.0	1.0	59.653202	22.270777	-0.016117	-0.012590	Non DDoS
1	192.168.100.108	54360.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	41.504154	Non DDoS
2	192.168.100.108	53971.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	64.988373	Non DDoS
3	192.168.100.108	57415.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	53.036583	Non DDoS
4	192.168.100.108	34266.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	53.036583	Non DDoS
...	...	...	...	...	...	...	...	...	...	...
11426015	192.168.100.108	5526.0	9.142.38.44	37215.0	29.0	-0.014204	-0.086892	-0.016117	-0.012590	Non DDoS
11426016	192.168.100.108	5526.0	77.6.97.156	37215.0	29.0	-0.014204	-0.086892	-0.016117	-0.012590	Non DDoS
11426017	192.168.100.108	5526.0	169.172.173.135	37215.0	29.0	-0.014204	-0.086892	-0.016117	-0.012590	Non DDoS
11426018	192.168.100.108	5526.0	192.168.68.18	37215.0	29.0	-0.014204	-0.086892	-0.016117	-0.012590	Non DDoS
11426019	192.168.100.108	5526.0	43.224.126.126	37215.0	29.0	-0.014204	-0.086892	-0.016117	-0.012590	Non DDoS

11403688 rows × 10 columns

Fig. 16. Dataset After Data Standardization

After standardizing the data, the categorical values in the 'category' column are converted into a numerical format to facilitate interpretation by machine learning models. In this transformation, 'Malicious DDoS' values are replaced by one, while 'Non-DDoS' values are replaced by zero. This conversion allows the models to effectively process and understand the target variable. The provided figure below illustrates a sample of the dataset after this categorical-to-numerical conversion, showcasing the transformed 'category' column with values of zero.

	uid_string	id.orig_addr	id.orig_port	id.resp_haddr	missed_bytes_count	history_string	orig_pkts_count	orig_ip_bytes_count	resp_pkts_count	Category
0	192.168.100.108	5353.0	224.0.0.251	5353.0	1.0	59.653202	22.270777	-0.016117	-0.012590	0
1	192.168.100.108	54360.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	41.504154	0
2	192.168.100.108	53971.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	64.988373	0
3	192.168.100.108	57415.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	53.036583	0
4	192.168.100.108	34266.0	192.168.100.1	53.0	3.0	5.952537	1.351589	22.326843	53.036583	0

Fig. 17. Sample Dataset After Numerical Conversion of "Category" Column

3) RandomOverSampler: For training a classification-based machine learning model, there needs to be enough data for different classes, for the model to learn and predict accurately. From below figure 18, it is seen that there are not many records for the Malicious DDoS attack category and hence RandomOverSampler technique is used to increase the number of Malicious DDoS attack records which is the minority class. The RandomOverSampler increases the records belonging to the minority class by randomly selecting instances from the minority class and replicating them until there is the same number of records for both categories. After performing oversampling, the number of Malicious DDoS values increased to the same value as the number of Non-DDoS values which is

11,408,839. The below figure shows the distribution of the target class output after over-sampling in the form of a bar graph.

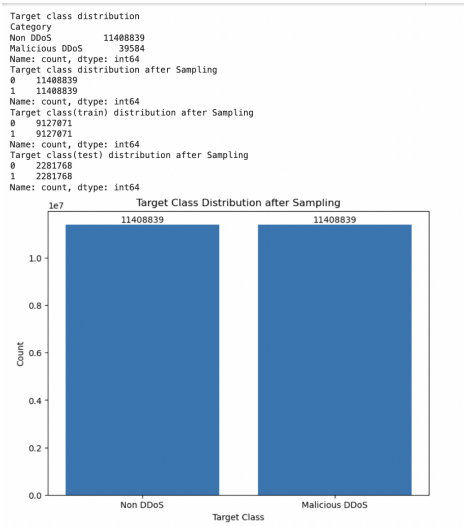

Fig. 18. Distribution of Values in Category Column

4) Data Regularization: Data Regularization is performed on the dataset to prevent overfitting of the machine learning models by using L1 Regularization which is also called the Lasso Regularization. It tends to force some of the coefficient values to be zero leading to a subset of features that are considered to be relevant. The below figure shows the coefficient of the different features in the form of a bar graph.

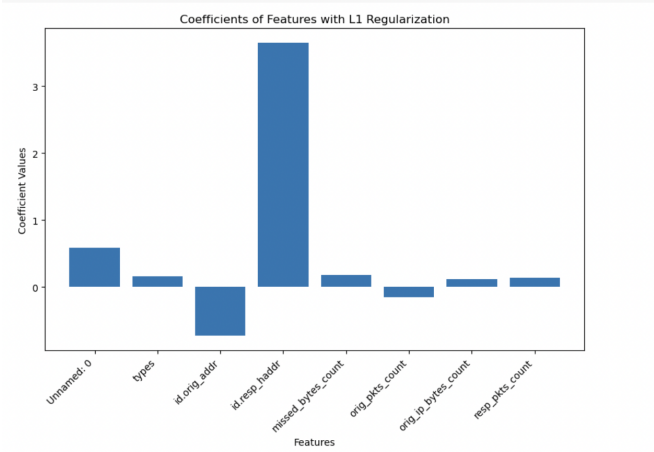

Fig. 19. Bar Graph Shows the Coefficient of Features With L1 Regularization

Once the dataset is transformed using the above data transformation techniques, it is prepared for modeling purposes by using 80% of the data for training purposes and the remaining 20% for testing purposes.

V. Model Selection

In choosing the models for the project, a deliberate choice has been made to benefit from the models such as Random Forest Classifier (RFC), Support Vector Machine (SVM), and K-Nearest Neighbour (KNN).

A. Random Forest Classifier

The Random Forest Classifier (RFC) is one of the powerful ensemble learning techniques that integrates predictions of multiple decision trees which aids with enhancing the predictive accuracy while mitigating overfitting issues. The Random Forest Classifier is used to test data to forecast labels, and the predicted results are compared to the true labels. RF is suitable for the dataset that is used as it is capable of handling imbalanced datasets, which is a common problem in binary classification. Implementing this model proves to be beneficial and stands out well in capturing the complex relationships within the IoT-23 dataset.

B. K-Nearest Neighbor

The K-nearest neighbor (KNN) algorithm uses an instance-based learning approach, which means it memorizes the training instances and classifies new instances based on the similarity to the training data. The data points are classified based on the majority class of the respective k-nearest neighbor making it efficient for variations in the dataset. KNN is efficient with pattern recognition making it suitable for the IoT dataset that consists of patterns where similar instances tend to have similar patterns. It is effective in capturing non-linear relationships in the data. This is valuable when the decision boundary between DDoS and non-DDoS instances is not a simple linear separation. It serves as an effective model for datasets with numerous features as it remains efficient in high-dimensional spaces.

C. Support Vector Machine

Support Vector Machine (SVM) in the context of IoT datasets is crucial for achieving accurate and robust classification. It is designed to find a hyperplane that maximizes the margin between various classes. SVM can handle both linear and non-linear relationships and is particularly effective for binary classification tasks (NonDDos and DDoS), dividing data into two distinct categories. In scenarios as such where the dataset involves classifying instances into two classes, SVM provides a strong framework for making accurate predictions. Since the dataset also exhibits class imbalances, where certain classes have fewer instances, SVM's ability to maximize the margin helps avoid overfitting to the majority class.

D. Ensemble Techniques

A voting classifier makes combined predictions of individual classifiers. The method is being implemented to make predictions for the following combination along with the voting classifier for RF and KNN, KNN and SVM, and RF and SVM. The ensemble approach leverages the power of combining better performance and provides optimal results when compared to individual accuracies. The pickle method provides a serialized representation of the stored machine learning model.

The voting Classifier is classified as an ensemble method since it combines predictions from various models, assigning

greater weight to those with higher confidence scores. This approach offers a flexible framework for leveraging the strengths of different model types, including SVM, Random Forest, and neural networks, all under one umbrella. The Voting Classifier proves robust against noise due to the inherent variance in prediction results, and its voting parameters can be adjusted for optimal performance. Thus, the Voting Classifier serves as an effective benchmark, demonstrating superiority over more complex ensemble methods by utilizing simple models and weighted summation. As a classification ensemble technique, it stands out for its intuitiveness, requiring less mathematical knowledge compared to stacking or blending methods. Despite its simplicity, the Voting Classifier excels in flexibility, accuracy, and ease of use.

VI. Model Evaluation

Model evaluation is the process of assessing the performance and quality of a machine learning model. It involves measuring the model's ability to make accurate predictions on unseen data and understanding its strengths and weaknesses. Evaluation metrics are used to quantify the model's performance and compare different models or variations of the same model. In the context of classifying DDoS and non-DDoS instances, the evaluation metrics utilized for this project are recall, precision, F1 score, and support.

Recall: It is used to assess the model's ability to capture all instances of DDoS attacks. A high recall indicates that the model is effective at identifying DDoS attacks, reducing the risk of false negatives, which are instances of DDoS attacks incorrectly classified as non-DDoS.

Precision: It measures the accuracy of positive predictions made by the model. A high precision indicates that the model's positive predictions for DDoS attacks are accurate, minimizing the occurrence of false positives, which are instances of non-DDoS incorrectly classified as DDoS.

F1 Score: The F1 score is a measure that strikes a balance between precision and recall, providing an overall assessment of the model's performance. It takes into account both false positives (precision) and false negatives (recall) and is particularly useful when the class distribution is imbalanced.

Support: Support refers to the total number of data instances that are processed to predict DDoS or non-DDoS records. It provides an understanding of the data distribution and can be useful for interpreting the model's performance.

A. Individual Model Evaluation

1) K-Nearest Neighbor: The achieved results, with perfect precision, recall, and F1-score values of 1.0, indicate flawless classification by the model on the dataset. These scores imply that the model not only correctly identified all instances of each class but also avoided any false positives or false negatives. The support values accompanying these metrics provide insights into the dataset's class distribution, indicating the number of instances for each class.

In the below figure displaying all accuracy scores, a comprehensive overview of the model's performance across various metrics is presented. This includes accuracy, precision, recall, and F1-score for each class or category. Such a comprehensive view enables a nuanced assessment of the model's strengths and potential areas for improvement, offering a detailed understanding of its classification performance across different aspects.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2281768
1	1.00	1.00	1.00	2281768
accuracy			1.00	4563536
macro avg	1.00	1.00	1.00	4563536
weighted avg	1.00	1.00	1.00	4563536

Fig. 20. Evaluation Metrics for KNN

2) Random Forest Classifier: The Random Forest classifier model yields impeccable results with precision, recall, and F1-score all registering a perfect score of 1.0. This underscores the model's ability to achieve flawless classification on the utilized dataset. The accompanying support values complement these scores by revealing the distribution of instances for each class, shedding light on the dataset's composition.

The figure below showcases the accuracy scores and provides a holistic perspective on the Random Forest Classifier model's performance. It encapsulates accuracy, precision, recall, and F1-score for each class, allowing for a thorough evaluation of the model's effectiveness in diverse aspects of classification. This comprehensive overview aids in gauging the model's strengths and potential areas for refinement, enhancing the understanding of its classification capabilities across various metrics.

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2492
1	1.00	1.00	1.00	4985
accuracy			1.00	7477
macro avg	1.00	1.00	1.00	7477
weighted avg	1.00	1.00	1.00	7477

Fig. 21. Evaluation Metrics for RFC

3) Support Vector Machine: The Support Vector Machine model also shows remarkable performance, attaining perfect scores of 1.0 for precision, recall, and F1-score. These results signify the model's exceptional ability to achieve flawless classification on the specific dataset it was trained on. The support values accompanying these metrics provide valuable insights into the dataset's class distribution, conveying the number of instances associated with each class.

In the below figure, illustrating accuracy scores, a comprehensive depiction of the SVM model's performance is shown. This visual representation encapsulates key metrics such as accuracy, precision, recall, and F1-score for each class. This holistic view enables a thorough assessment of the model's proficiency in different facets of classification, facilitating a

nuanced understanding of its strengths and potential areas for improvement.

Accuracy: 1.0
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2492
1	1.00	1.00	1.00	4985
accuracy			1.00	7477
macro avg	1.00	1.00	1.00	7477
weighted avg	1.00	1.00	1.00	7477

Fig. 22. Evaluation Metrics for SVM

An accuracy of 1.0 is achieved for KNN, RF, and SVM as individual models, as seen above. This is attributed to the over-sampling of the minority class DDoS, which is undertaken to address the poor data spread between DDoS and Non-DDoS, resulting in significant duplication in the dataset. The number of records has increased from 39,584 to 11,408,839, leading to a substantial redundancy of data. Despite the implementation of L1 regularization, it is evident that the model is prone to overfitting.

B. Model Evaluation Using Ensemble Techniques

1) SVM and KNN with Voting Classifier: The evaluation results for SVM and KNN along with the voting classifier incurred a low performance for SVM in comparison with KNN with precision, recall, and F1-score at 0.402, 0.0505, 0.448 while KNN displayed an optimal performance with 1.0 for the same, as shown in the below figure 23. The voting classifier exhibits a performance with a precision of 0.891. The findings suggest that KNN stands out in comparison with SVM and the voting classifier exhibits a reasonable overall precision of 89.1%

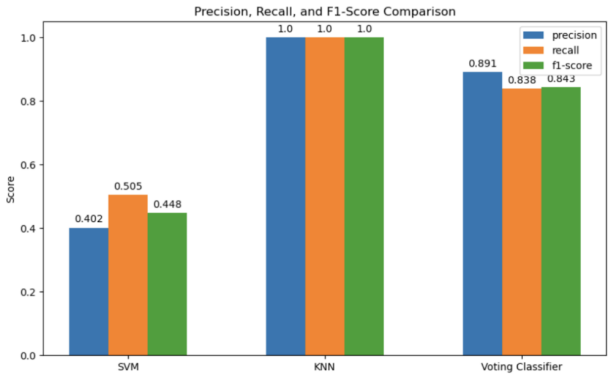

Fig. 23. Grouped bar graph showing the precision, recall, and F1 for SVM, KNN, with Voting Classifier.

2) RF and SVM with Voting Classifier: The evaluation results for RF and SVM along with the voting classifier incurred a low performance for SVM in comparison with RF with precision, recall, and F1-score at 0.445, 0.667, 0.553 while RF displayed an optimal performance with 1.0 for the same, as shown in the below figure. The voting classifier exhibits a performance with a precision of 1.0. The findings suggest that RF stands out in comparison with SVM and the voting classifier exhibits an optimal precision of 100%.

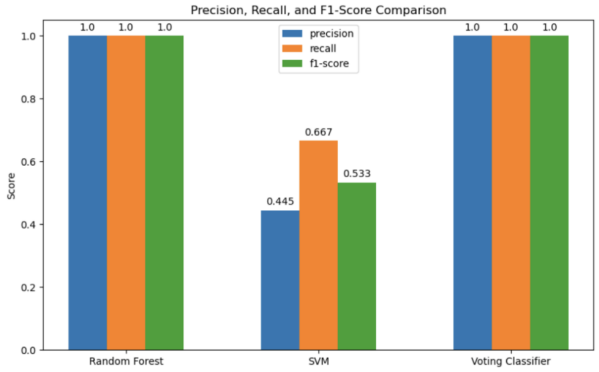

Fig. 24. Grouped bar graph showing the precision, recall, and F1 for RF, SVM, with Voting Classifier.

3) RF and KNN with Voting Classifier: The evaluation results for RF and KNN along with the voting classifier incurred results with precision, recall, and F1-score at 1.0 for displaying an optimal performance. The voting classifier exhibits a performance with a precision of 1.0. The findings suggest that RF, KNN, and the voting classifier exhibit an optimal precision of 100%.

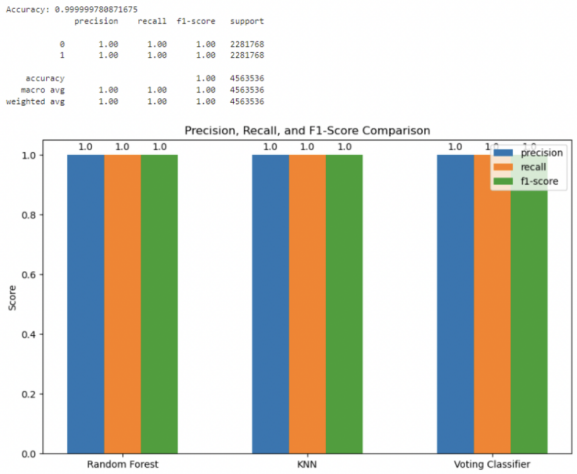

Fig. 25. Grouped bar graph showing the precision, recall, and F1 for RF, KNN, with Voting Classifier

C. Training Models and Results

The results obtained show remarkably high performance for both individual models and ensemble techniques in the project. The incorporation of the concept of overfitting is a strategic move to prevent the models from fitting too closely to the training data, ensuring their generalizability. Through

thorough analysis, it is deduced that overfitting tendencies may arise due to a lack of inconsistency in the data, a factor that is prevalent in the dynamic nature of real-time data flow. Therefore, the primary objective becomes twofold to prevent overfitting and to ensure accurate results. In this context, the Random Forest (RF) and K-Nearest Neighbors (KNN) ensemble model emerges as a robust solution. Although there's a slight drop in accuracy for the ensemble model, each model within it attains perfect accuracy scores. This nuanced trade-off suggests that, despite the marginal decrease in overall accuracy, the RF and KNN ensemble model excels in striking a balance between avoiding overfitting and delivering accurate and reliable results. This performance makes it the preferred choice, earning high regard compared to other models in the project.

VII. Deployment

For the deployment of the model using the 'pickle' package, it needs to be loaded using the saved '.pkl' file on the local machine. The front end of the application is designed using 'Streamlit'. The Spyder interface in the Conda environment serves as the Integrated Development Environment (IDE) for deploying the project. To access the saved model globally, it needs to be placed in cloud storage, and an S3 bucket is chosen for this purpose. In Amazon S3, a new bucket named 'ML_proj' is created, and the saved model is uploaded. To enable global access to this S3 bucket, Identity and Access Management (IAM) roles are created, and the corresponding key is shared in the code. The 'S3fullaccess' IAM role is assigned to the bucket, and the secret key and key ID are generated.

The Spyder interface encompasses the entire codebase, initiating the creation of the page title and relevant content. The input for this application is a comma-separated value (CSV) file containing network parameters, which is imported into the system. Subsequently, the model is loaded from the designated Amazon S3 bucket utilizing secret key access. Following the model loading, it is executed on the imported CSV file, and the resulting output is presented to the user within the front-end environment. Spyder code structure starts with loading of the model using pickle, and then the procuring of the CSV file as input from the user. Processing of the CSV file is initialized and then parsed through the model to secure output in terms of the count of malicious DDoS and non-malicious DDoS. The sample output is shown in the below figure for the user in the front end.

The code for the entire project is committed to a Git repository, establishing a direct link with the front-end application. Accompanying the code is a requirements file encapsulating all version specifications for necessary libraries, ensuring seamless compatibility and satisfying dependencies during code execution. Upon initiation, the front-end environment accepts user input, primarily in the form of an uploaded CSV file containing network parameters. A background process is triggered, leveraging the loaded model to generate results, which are then displayed to the user through the front-end interface. The application's final appearance is depicted in the figure below.

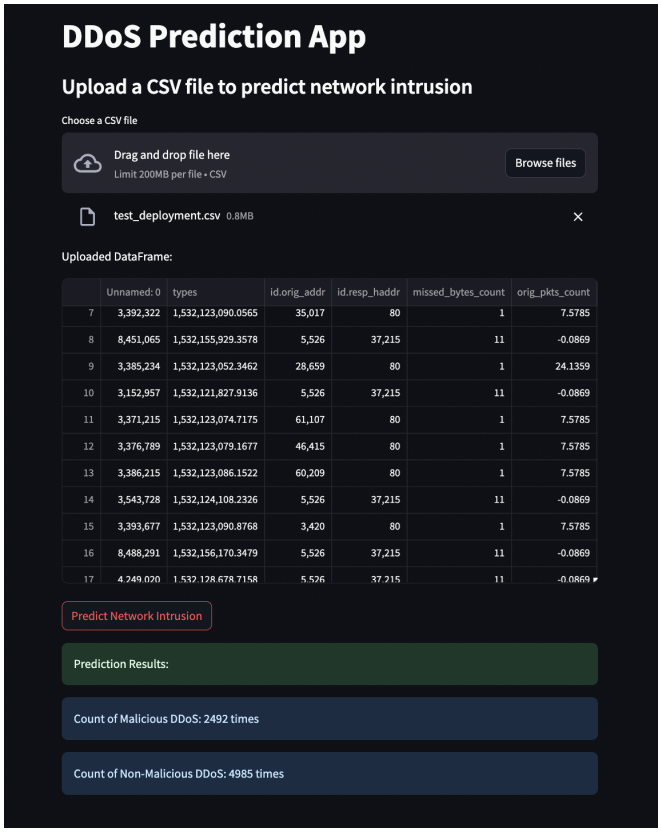

Fig. 26. Sample Look of the Front End of the DDoS Prediction Application

VIII. Conclusion

The project is centered around detecting DDoS attacks, utilizing ensemble machine learning techniques as opposed to basic algorithms. The decision to utilize ensemble techniques stems from the aspiration to leverage the collective strengths of multiple algorithms within a unified framework. Although individual models demonstrate high accuracy, real-world complexities impact the final output. To mitigate this, the project adopts a straightforward yet effective approach using a voting classifier within the ensemble technique. The voting classifier, incorporating Random Forest Classifier (RFC) and K-nearest neighbor (KNN), explores optimal combinations, identifying and preserving models with superior accuracy and precision. Thus, these refined models are integrated into the project's front end. While direct accuracy percentages are not exposed to users, the front end facilitates the assessment of improvements in overall model performance and predictions through user-friendly metrics or visualizations.

IX. Required Links

Source Code Link:
https://github.com/shashikumar1998/ML_Project/
YouTube Presentation Link:
https://youtu.be/XUws3YKKAKU/

X. Limitations

The primary limitations of the project stem from the models tending to overfit due to the handling of class imbalance

through oversampling. This leads to near-perfect accuracy on the training data, but it doesn't comprehensively evaluate performance on new, unseen data, thereby questioning the models' ability to generalize effectively. Also, the study falls short by only measuring a few classification metrics like accuracy without considering other crucial performance metrics for security applications, such as detection delay and false alarm rate.

Moreover, the research is constrained by its reliance on a single publicly available IoT dataset, which may limit its generalizability across different network scenarios. The study exclusively involves static network architectures and attack patterns, neglecting the dynamic evolution inherent in real-world scenarios. Despite the models successfully classifying attacks, the project lacks a proposed mitigation approach. Furthermore, a more exhaustive hyperparameter optimization process could enhance performance, thereby restricting the applicability and generalization of the proposed solution.

XI. Innovation

The project demonstrates innovation in its exploration of ensemble techniques for improved performance in detecting DDoS attacks from IoT network traffic. Ensemble techniques aim to combine multiple machine learning models to leverage their strengths. Specifically, the project experiments with voting classifiers that integrate the predictions from different base models like Random Forest, KNN, and SVM. A voting classifier acts as a meta-estimator that averages the predictions from base models after fitting on portions of the dataset. This allows the classifier to benefit from the accuracy of some models while mitigating errors from others. The project evaluates different combinations of base models in the voting classifier framework, identifying Random Forest and KNN as the best-performing combination. By fusing the predictions from these two algorithms, the ensemble approach can achieve perfect precision, recall, F1-score, and support in classifying DDoS and non-DDoS instances, outperforming the individual models. This demonstrates the potential of ensemble learning techniques to improve network security analytics tasks like attack detection by leveraging the collective strengths of multiple models.

XII. Future Scope

While the project excels in identifying DDoS attacks with proficiency and precision, there is still a notable gap in the critical step of mitigating these identified threats. Mitigation remains a crucial aspect that not only holds the key to improving the overall project outcome but also signifies the completion of the project itself. To bridge this gap, it becomes essential to explore and experiment with complex ensemble techniques. Beyond just assessing the computational abilities of these advanced models, such experimentation offers a valuable opportunity to evaluate their accuracy in effectively mitigating DDoS attacks. Integrating these sophisticated ensemble techniques has the potential to significantly enhance the project's mitigation capabilities, providing a more comprehensive defense against DDoS threats and contributing to the overall success of the project.

References

[1] *IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic. — Stratosphere IPS.* (n.d.). Stratosphere IPS. https://www.stratosphereips.org/datasets-iot23

[2] Liang, X., & Kim, Y. (2021). A Survey on Security Attacks and Solutions in the IoT Network. *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC).* https://doi.org/10.1109/ccwc51732.2021.9376174

[3] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIOT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors, 23(13)*, 5941. https://doi.org/10.3390/s23135941

[4] Patel, S., Gupta, A., Nikhil, Kumari, S., Singh, M., & Sharma, V. (2018). Network traffic classification analysis using machine learning algorithms. *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN).* https://doi.org/10.1109/icacccn.2018.8748290

[5] Shahid, M., Blanc, G., Jiang, X., & Débar, H. (2018). IoT Devices Recognition Through Network Traffic Analysis. *2018 IEEE International Conference on Big Data (Big Data).* https://doi.org/10.1109/bigdata.2018.8622243

XIII. Appendix

Criteria	Comments
Version Control	Git and GitHub are used for version control and the repository is publicly accessible.
Significance to the real world	The project aims to detect DDoS attacks in IoT environments which has real-world applications in strengthening cybersecurity.
Lessons learned	The report and presentation include lessons learned in detecting DDoS attacks using ML models, with both technical and non-technical aspects.
Innovation	The project explores the use of ensemble techniques like voting classifiers to combine different ML models for improved performance in detecting DDoS attacks.
Teamwork	All tasks are divided equally among team members to ensure that everyone gets exposure to using different tools.
Limitations	Project limitations include potential overfitting due to class imbalance handling, a reliance on a single IoT dataset limiting generalization, and a lack of comprehensive metrics like detection delay and false alarm rate. The study focuses on static network architectures, lacks a proposed mitigation approach, and could benefit from more exhaustive hyperparameter optimization for improved performance and generalization.
Pair programming	Google Colab is used by the team to practice pair programming, where team members collaborate to resolve implementation issues. After completing the task, team members get to review all merged changes and push them to the GitHub repository. This ensures that the functionality is tested and working as intended.
Used Grammarly / other tools for language.	Google Docs has a very good built-in feature to point out a suggestion for any wrong usage of words or auto-correcting the wrong wordings.
Slides	Project presentation slides are created using Microsoft PowerPoint with all the essential content

Criteria	Comments
	and are made visually appealing.
Unique tools	Microsoft PowerPoint is used to create the project presentation slides, and Latex is used to create the report. The recorded presentation is on YouTube, ensuring easy access for a broader audience. All slides are included in the video description for reference and further study.
Report Format	The report follows IEEE format and includes all the details about the project