****

**Department of Computer Science**

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Computer Science.

|  |  |  |
| --- | --- | --- |
| IOT security using Machine learning | | |
| Project Title (type) | | |
| Purva Surve |
| Student Name (type) |
| Dr. Abdul Motin Howlader |
| Advisor's Name (type) |
|  | |
| Advisor's signature | Date |
|  |
| Reviewer's name |
|  | |
| Reviewer's signature | Date |

**Department of Computer Science**

**IOT Security using Machine learning**

**Purva Surve**

**Table of Contents**

[**1.0**  **Introduction** 5](#_Toc165681890)

[**1.1**  **Description of the Problem** 5](#_Toc165681891)

[**1.2**  **Project Objectives** 5](#_Toc165681892)

[**1.3**  **Environment (software and hardware)** 6](#_Toc165681893)

[1.3.1 Hardware in the Environment 6](#_Toc165681894)

[1.3.2 Software in the Environment 6](#_Toc165681895)

[1.3.3 Dependencies: 7](#_Toc165681896)

[**2.**  **Literature Review** 8](#_Toc165681897)

[**3.**  **Design and Architecture** 10](#_Toc165681898)

[**3.1**  **System Overview** 10](#_Toc165681899)

[3.1.1 Functional requirements 10](#_Toc165681900)

[3.1.2 Non-functional requirement 10](#_Toc165681901)

[3.1.3 Data requirements 10](#_Toc165681902)

[**3.2**  **Architecture Overview** 11](#_Toc165681903)

[**4.**  **Implementation and Testing** 13](#_Toc165681904)

[**4.1**  **Data Visualization** 13](#_Toc165681905)

[**4.2**  **Data Preprocessing** 13](#_Toc165681906)

[**4.3**  **Correlation Analysis** 15](#_Toc165681907)

[**4.4**  **Outliers Handling** 16](#_Toc165681908)

[**4.5**  **ML model selection** 16](#_Toc165681909)

[**4.6**  **Model validation** 17](#_Toc165681910)

[**4.7**  **Flask web application** 18](#_Toc165681911)

[**4.8**  **Reporting** 21](#_Toc165681912)

[**5.**  **Evaluation and Optimization or Future Works** 23](#_Toc165681913)

[**5.1**  **Model Evaluation and Analysis** 23](#_Toc165681914)

[**5.2**  **Model optimization** 24](#_Toc165681915)

[**5.3**  **Future work** 24](#_Toc165681916)

[**6.**  **Conclusion** 25](#_Toc165681917)

**List of Figures**

[**Figure 1 Machine learning model process 11**](#_Toc165684562)

[**Figure 2: System application block diagram 12**](#_Toc165684563)

[**Figure 3: Histogram for data 13**](#_Toc165684564)

[**Figure 4: Data preprocessing process 14**](#_Toc165684565)

[**Figure 5: Correlation matrix 16**](#_Toc165684566)

[**Figure 6: K-fold cross validation process 17**](#_Toc165684567)

[**Figure 7: Sequence diagram 18**](#_Toc165684568)

[**Figure 8: Website welcome page 19**](#_Toc165684569)

[**Figure 9: Preprocessing form page 19**](#_Toc165684570)

[**Figure 10: Results page – Unsafe request 20**](#_Toc165684571)

[**Figure 11: Results page –Safe request 20**](#_Toc165684572)

[**Figure 12: About Us page 21**](#_Toc165684573)

[**Figure 13: Attack Performance analysis graph 22**](#_Toc165684574)

[**Figure 14: Sub Attack Performance analysis graph 22**](#_Toc165684575)

**List of Tables**

**Table 1: Performance metrics for Attack models 15**

**Table 2: Performance metrics for Sub Attack models 15**

# **1.0 Introduction**

## **1.1 Description of the Problem**

Internet of Things (abbreviated as IoT) is basically t3he network of various physical devices, sensors and sophisticated software or hardware technologies connected together to communicate and exchange computational data over the internet. IOT is one of the most emerging and rapidly used technologies for building smart solutions such as connecting smart cities, upgrading supply chain management, innovating the healthcare sector and so on.

The IoT devices mainly operate on three-layer architecture, physical or perception layer, network layer and application layer. In the physical or perception layer, the devices are actually connected to the system or environment through which they perceive the surroundings to collect useful information. Later, at the network layer the devices are connected to each other, a network device and servers which transmits the sensor data. Finally, the application layer is responsible for delivering the frontend application services to the user. Each of these layers are prone to various security attacks such as Distributed Denial of service, replay attack, Man in the middle and so on.

There have been many instances in the past where IOT devices have been hacked and hackers have identified vulnerabilities in IOT devices to gain unauthorised access. St. Jude medical pacemakers is one such incident where hackers got full access to the pacemaker. The only criteria where the user has been in close proximity to the patient to interfere with the RFID waves and get access. Another mass attack was Mirai botnet of 2016 where a malware was introduced on the server which scans the internet for the IOT devices that run on ARC processor and take access to the devices whose default password was not changed. Later attackers used these zombie bots forming a net called a botnet to launch a DOS attack. Apart from this, there have been many attempts to exploit the IOT security and various attacks have been launched.

IoT security vulnerabilities are slowly discovered as more and more machinery and smart devices get linked to the network. Compared to computers and mobile phones, IoT devices are more prone to attack not only because IoT device usage has increased, but also due to the complexity, diversity, and intrinsic mobility of these devices. Due to its large-scale connectivity, low power usage restrictions and lack of human supervision hence maintaining its security is a complex and challenging task.

## **1.2 Project Objectives**

The main objective of this project was to develop and implement a machine learning-based solution to improve the security of IoT systems. Specifically, the objectives were:

* Identifying Security Challenges: Analyse and identify the major security challenges faced by existing IoT solutions, including threats such as impersonation, Denial of Service (DoS), malware, and intrusion attempts.
* Developing a Computing Model: Develop a computing model deployed at the application layer of the IoT infrastructure to predict potential security threats in real-time.
* Training and Evaluation: Train the model using historical data related to security attacks and evaluate its performance using machine learning classification techniques. The aim was to achieve a model accuracy of greater than 90%.
* Model Optimization: Employ model optimization techniques to refine and improve the performance of the developed machine learning model.
* Real-time Alerts and Visualization: Implement a frontend web application to provide real-time alerts to end-users and visualise the results generated by the machine learning model.

## **1.3 Environment (software and hardware)**

The following section contains a detailed description of hardware and software requirements needed for implementing the proposed solution.

### **1.3.1 Hardware in the Environment**

The following bulleted list provides a breakdown of the hardware anticipated for use in this project.

* Minimum 350MB Hard Disk space for installation
* 4GB HD space required for a typical live system with 1000-2000 events
* Recommended minimum CPU - Pentium 4, 3.2GHz
* Recommended 1GB RAM for a Central Server with 3 Nodes
* Network card

### 1.3.2 Software in the Environment

* Python – It is used for backend ML model development as it contains various ML supported libraries for training and visualising the model.
* Python Libraries:

1. Pandas: it will allow us to analyse large dataset and draw inferences.
2. Numpy: the functions available with the library will be used to perform mathematical calculation
3. Scikit-learn: it will be used to implement machine learning models
4. Seaborn/ matplotlib: To visualise and evaluate model results
5. Matplotlib - To visualise the data

* Flask - To host the test server and render frontend
* Plotly - The library extension is used to plot the graphs on the frontend
* Hypertext Markup Language (HTML) 5 & Cascading Style Sheets (CSS) 3– It will be used to develop frontend web interfaces to display visualisation results to the user.
* Bootstrap – Used for enhancing UI

### 1.3.3 Dependencies:

To run the ML model notebook, following libraries needs to preinstalled:

* pandas
* numpy
* matplotlib
* seaborn
* Scikit-learn

To launch the frontend application smoothly, Flask and proper Python environment variable setup are essential.

# **2. Literature Review**

The IoT technological advancements and its applications has increased exponentially over the last few years so as the security concerns related to it. Through research in this problem domain, it seems security and privacy issues related to IoT applications are quite evident. A lot of research has been done in the field of developing real time security systems that can perceive and provide constant feedback by monitoring large numbers of IOT nodes.

Bhabhendu kumar Mohanta, Debasish Jena, Utkalika Satapathy and Srikanta Patnaik in their survey paper “Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology” have discussed about various security challenges faced during IoT implementation [2]. Also briefly summarised about various security attacks anticipated by the existing IoT applications. The survey paper depicts various machine learning, artificial intelligence and blockchain techniques used to address these security concerns.

According to Temechu G. Zwedie and Anteneh Girma Artificial Intelligence and Machine learning techniques have a major role in combating emerging IoT cyber security threats in the Cloud computing domain. They emphasise on using Artificial Intelligence and Machine learning techniques as Security tools to predict any cyber attack on a cloud environment as a lot of IOT computational data is stored on cloud space [6]. As every instance has two sides, they have also stated the drawbacks and threats of using ML/AI for cloud security.

Murat Kuzlu, Corinne Fair and Ozgur Guler in their research paper have mentioned using AI technologies, like decision trees, K-nearest neighbours, support vector machines, artificial neural networks and ML techniques to detect anomalous behaviour that may go unnoticed by some algorithms to detect cyber security in IoT applications [5].

“Machine Learning-Enabled IoT security: Open Issues and Challenges Under Advanced Persistent Threats” focuses on persistent network attacks in IoT and IIoT systems. The Advanced Persistent threats tend to bypass the control mechanism, deployed to detect the malware, in turn taking full control of the IoT system [5]. The survey paper compares the efficiency of various ML techniques used for detecting both APT and non-APT attacks and further discusses opportunities, various challenges and open issues.

Hui Wu, Haiting Han, Xiao Wang and Shengli Sun in “Research on Artificial Intelligence Enhancing Internet of Things Security” paper discussed various potential threats in the three layer architecture of IoT [3]. The paper details the various IoT security threats at the perception layer, network layer and application layer.

However, due to its complex nature and special requirements, the existing security system does not provide full security coverage. The existing security. The present methods do in fact partially address the issue raised before, but they are still unable to completely address the problem due to following reasons:

* The network protocols such as TCP/IP used for data transmission are already too prone to various security concerns such as intrusion, replay attacks, and identity theft. Also, the networking infrastructure has challenges such as low scalability, underutilization and complexity.
* The resource constraints of IOT devices are another factor as due to small size and low power consumption, these devices are capable of simple computations. Hence implementing a complex model at their end to prevent an attack becomes difficult.
* The IOT devices connect various devices, servers and applications to form a complex device. The basic architecture of IOT is a layered one and problems with one layer of the infrastructure can affect other layers. If an attacker can get access to any of the terminals of the layer, it can bring the whole IOT system down.
* The openness and large-scale connectivity of the IOT system. The IOT systems are connected to various other nodes and can obtain data from various sources integrating with various protocols and standards.

# **3. Design and Architecture**

## **3.1 System Overview**

### 3.1.1 Functional requirements

FR\_01: The system should be able to classify between a safe and infected request

FR\_02: The system should be able to display results on the frontend

FR\_03: Th system should display the graphical analysis of model performance using various performance metrics factors

FR\_04: The model should be trained using various classification techniques to achieve an efficiency of greater than or equal to 90%

FR\_05: The system should be able to preprocess the new input request and feed it to model for prediction

FR\_06: The system should have an informative section with detailed descriptions of techniques and technologies used for building the system

### 3.1.2 Non-functional requirement

NFR\_01: The system UI should be readable and simple to navigate

NFR\_02: The system should not take more than 5 mins to preprocess the new request

NFR\_03: The system should not take more than 5 mins to provide prediction results on the screen

NFR\_04: The system should be reliable and secure

NFR\_05: The system should provide consistent results for test data

### 3.1.3 Data requirements

DR\_01: The data should be correct, accurate and reliable

DR\_02: Data should be diverse, covering various scenarios and attack types to ensure the robustness of the machine learning model

DR\_03: The data should have data points at least 10 times the number of feature elements

## **3.2 Architecture Overview**

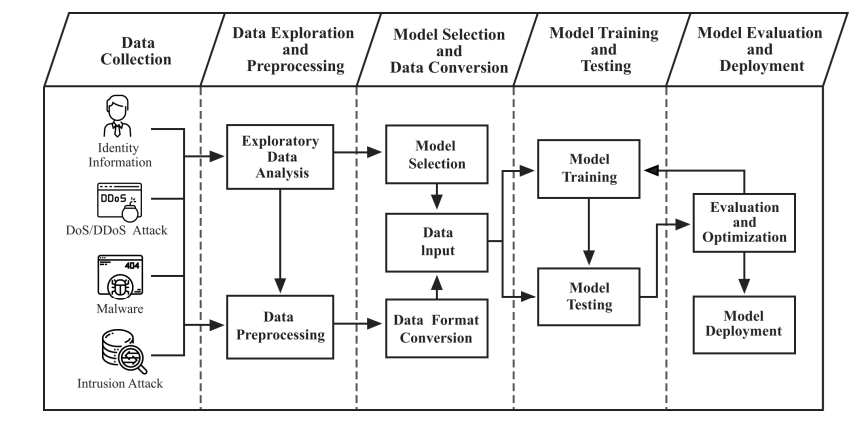


Figure 1 Machine learning model process

* Data Collection

In this phase, the data related to IoT security and potential attacks is collected from various sources and is merged into a single dataset. For our model I have used a dataset from Kaggle which consists of 26 feature variables both numerical and categorical [1].

* Data Exploration and Preprocessing

The next phase is to visualise and explore the dataset to identify relevant patterns and get insights. Data preprocessing is the most crucial and time-consuming step of the ML model. About 80% of the time should be invested in data processing as whatever we are going to feed to the systems it is going to learn it. Hence, it's very important to clean the data before ML modelling. Preprocessing steps mainly include data cleaning, data transformation, data labelling, data scaling, outlier handling and so on.

* Model selection and data conversion

Model selection involves choosing suitable machine learning algorithms and architectures based on the nature of the data and the problem requirement. The selected models are then trained on the pre-processed datasets. For our ML model I have used classification techniques such as Random Forest, Decision Trees and Naive Bayes.

* Model Training and Testing

To train the model I divide the data into train and test datasets, where models are trained using training data around 80% and then tested with the remaining test data. I have additionally validated the model using K-fold cross validation techniques to ensure that model is not overfitted to the training data and better generalises to new data.

* Model Evaluation and Deployment

The final step is to evaluate the model using various performance metrics such as accuracy, recall, F1 score to determine whether the model satisfies the performance requirement and provide good results on deployment. Once the model is deployed on an IoT server it can then be used to monitor the request traffic and identify potential threats.

For frontend I have developed a flask application which displays the results on the webpage and has below modules:

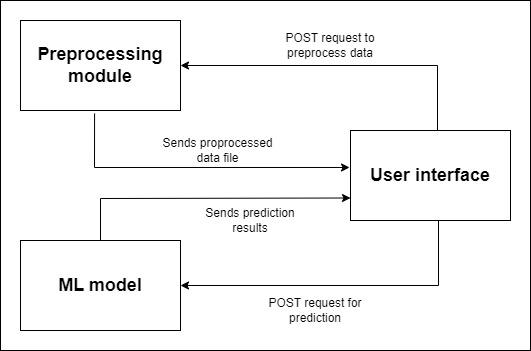


Figure 2: System application block diagram

* ***User Interface*:** It is the graphical interface that helps users to interact with the system. It enables users to submit a request, initiate preprocessing and view results generated by the model.
* ***Preprocess data*:** This module focuses on preprocessing the input request ensuring the compatibility with machine learning model and then feed it to predict modules for results
* ***Renders prediction results from ML model***: This feature displays the outcomes of the machine learning model's predictions in a user-friendly format. It presents the results of the analysis, such as class labels or probabilities, allowing users to interpret and utilise the model's insights effectively

# **4. Implementation and Testing**

## **4.1 Data Visualization**

Firstly, before starting any data pre-processing step, I first visualised the data. To better understand and comprehend our dataset I have used histogram which provides the graphical distribution of numerical data across all the feature set

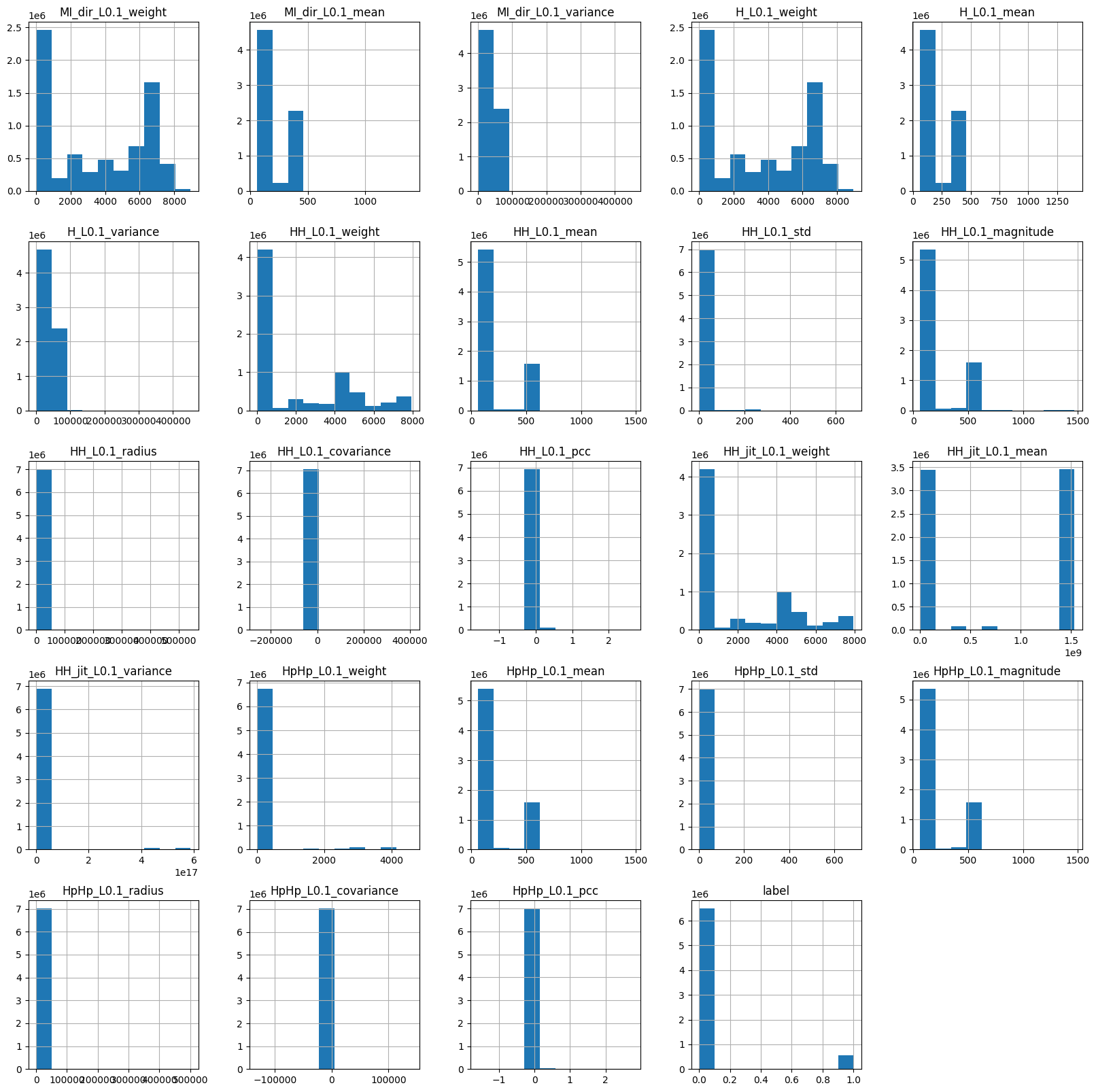


Figure 3: Histogram for data

## **4.2 Data Preprocessing**

The next step is to preprocess the data which mainly consists of various steps such as data cleaning, data integration, data transformation and data reduction. Data preparation is transforming raw data into a format that can be interpreted. Since I cannot work with raw data, this is important for data mining. Before implementing data mining or machine learning techniques, the data quality should be assessed.

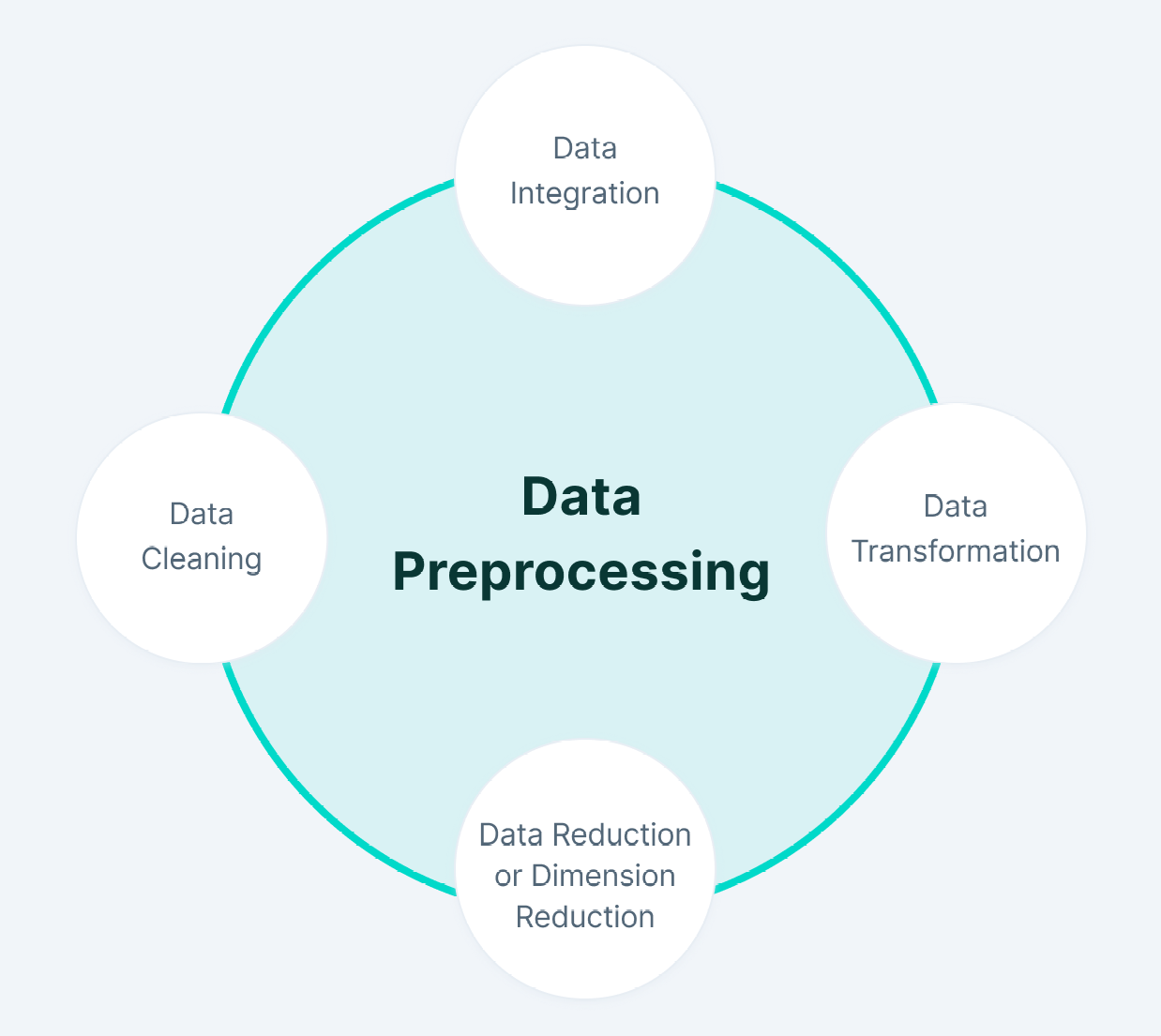


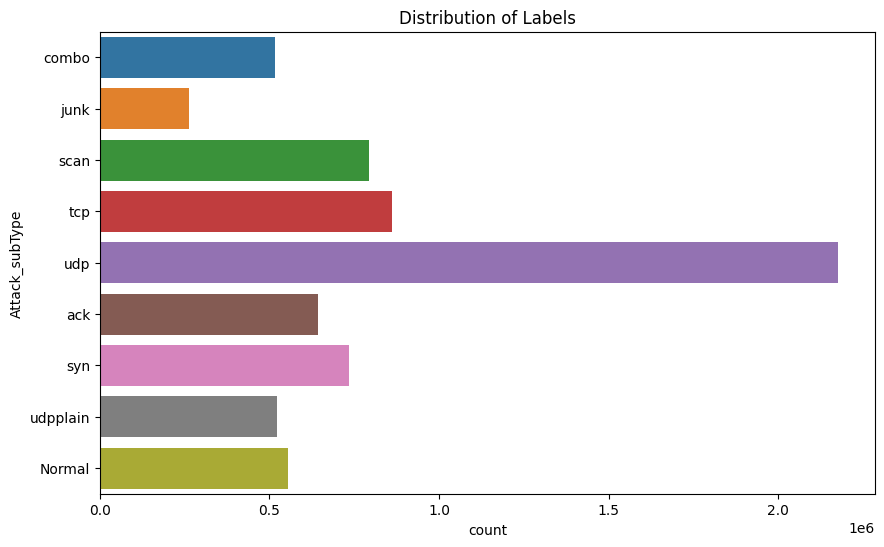
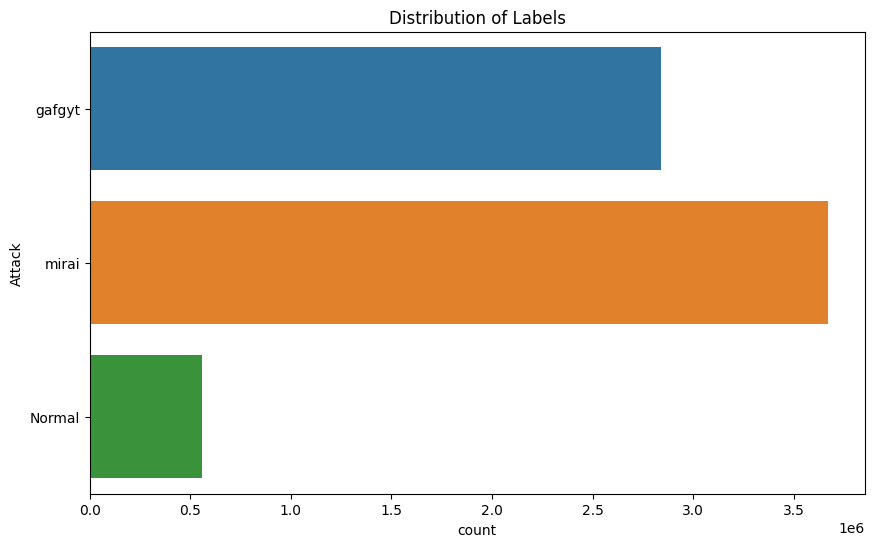
Figure 4: Data preprocessing process

* ***Data collection:*** The data used for building the model is taken from Kaggle which comprises features extracted from network traffic, indicative of potential IoT threats. The dataset consists of numerical data readings from various IOT devices with three categorical data namely Device Name, Attack Type and sub Attack Type which are then further used for training the model
* ***Data cleaning:*** Once the data is collected the next step is to clean it by removing any duplicates, outliers or noise. The aim is to ensure the data is as accurate and reliable as possible. To find the null values I have used isna().sum() function and then dropped the null values. I then found the duplicate entries and removed them from the dataset.

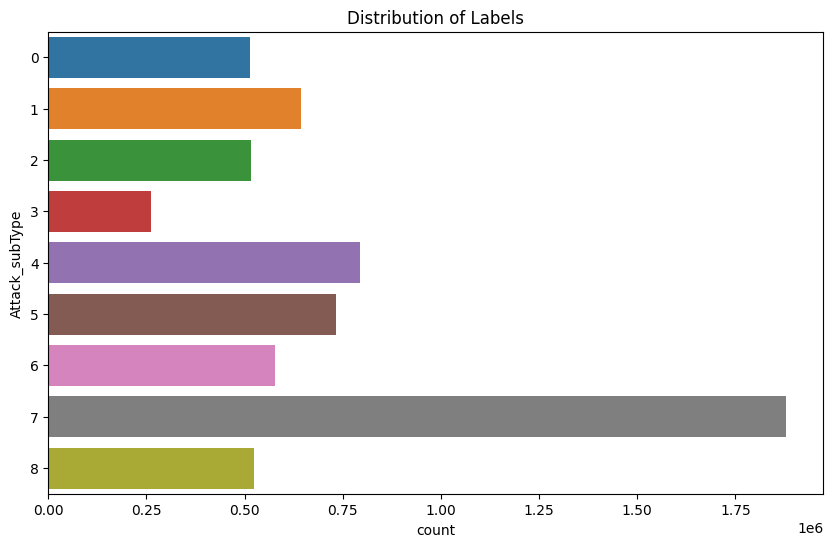
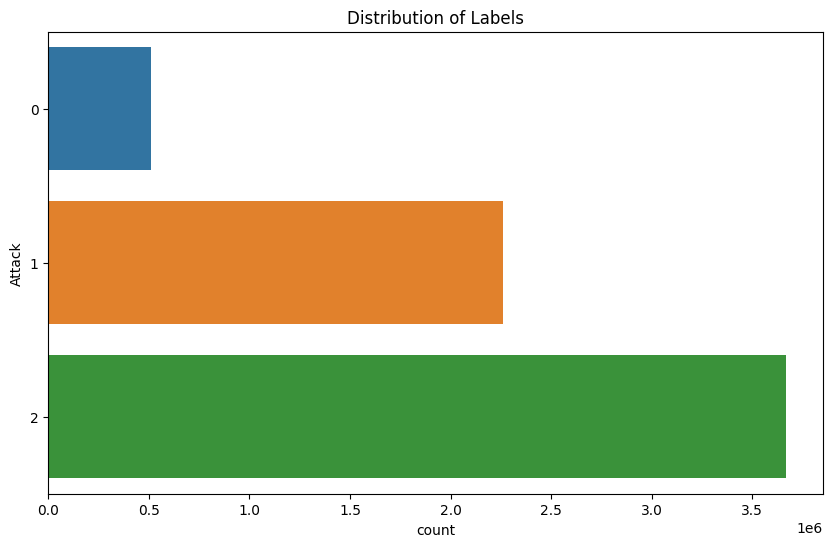
The next step of data cleaning is to fill in the missing values and filler data in our case since all the data was of numeric type I used ‘0’ to fill in the missing values. I then used StandardScalar to normalise the data set with each feature having a mean of 0 and variance of 1.

* ***Data Encoding:*** The data collected can likely contain information about various Attack and sub attack categories. This data needs to be labelled with corresponding attack values for analysis. I have used LabelEncoder to convert the categorical data of Device name, Attack Type and Attack sub Type feature variables into more manageable numeric values.

**Before Label Encoding:**



**After Label Encoding:**



* ***Feature scaling:*** Once the data is labelled, the data is scaled to ensure that all features are on a similar scale. This can be done using techniques such as standardisation, where the data is transformed to have a mean of zero and a standard deviation of one. For project data, I have used MinMaxScaler to scale the numerical value on a uniform scale.

## **4.3 Correlation Analysis**

Once the data has been pre-processed and cleaned, the next step that I did is to analyse the relation between input features and target feature i.e. Attack type and attack subtype. And to analyse the relationship I used the correlation matrix. I have selected the features that had an impact on the output of target features and then created and trained models using those features. To plot the correlation matrix, I have used the correlation function corr() on the training data from seaborn library.

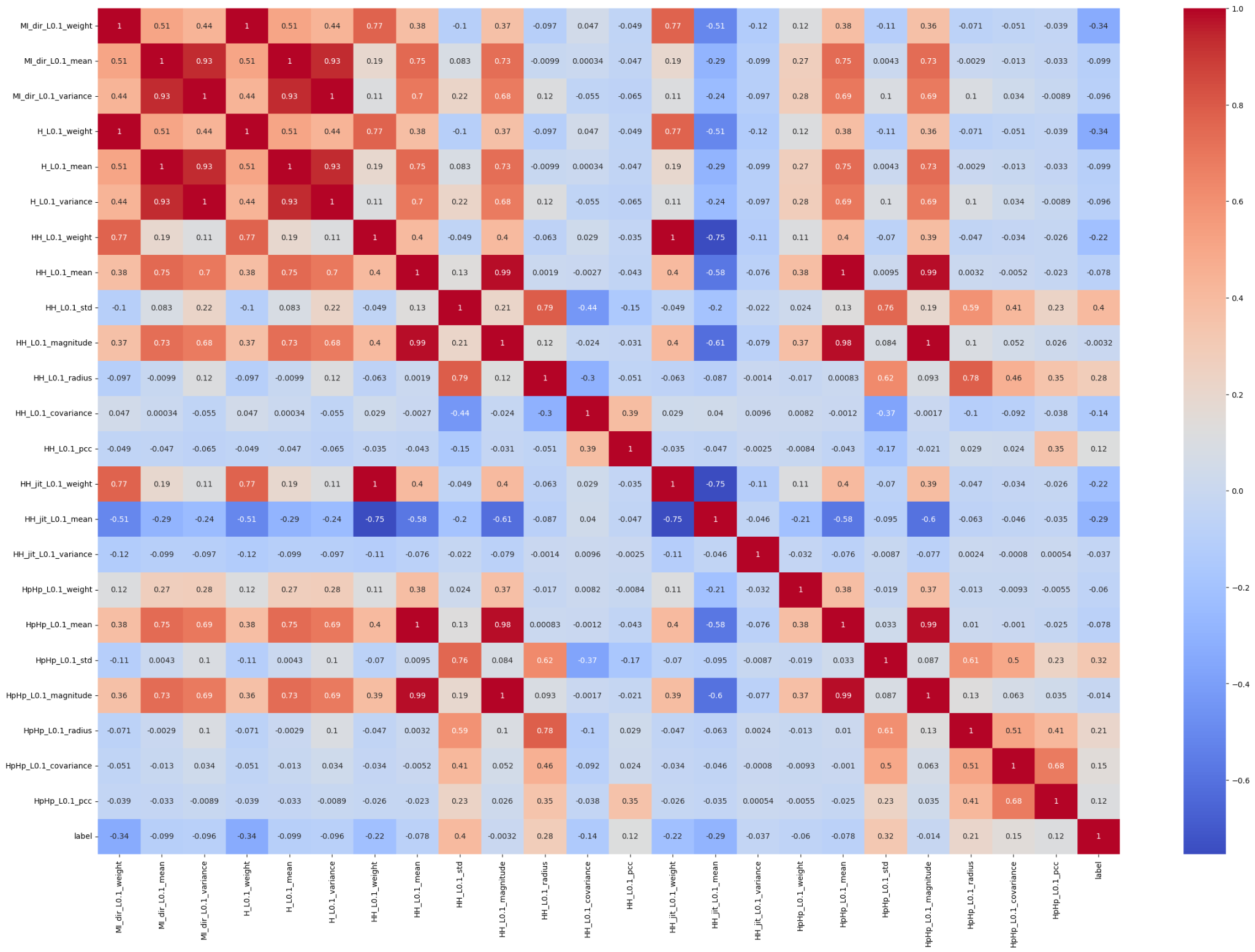


Figure 5: Correlation matrix

## **4.4 Outliers Handling**

Once the data is pre-processed and cleaned, I used the InterQuartile Range (IQR) method to identify the outliers in the dataset. Outliers are the extreme values that vary from the general observations of the dataset and it is important to identify and remove the outliers in order to reduce the errors. IQR is a statistical method that measures the spread of the dataset. It is basically calculated using the difference of third quartile (Q3) and first quartile (Q1). The third quartile is about the 75 percentile of the dataset value and the first quartile is the 25 percentiles of the dataset. The IQR is calculated using below “equation (1)”.

IQR = Q3 - Q1 (1)

After calculating IQR, I have removed the outliers from the dataset by removing the data points less than the difference of Q1 and greater than the difference of Q3.

## **4.5 ML model selection**

Once the data is pre-processed and outliers are handled, using the relation matrix I have identified the relevant feature and perform feature selection. For feature selection I have used a built -in library SelectKBest which using the f-classif variable selects the best k features from the dataset. I can explicitly specify the k value in the function. Then I used the train-test split approach to divide the dataset and train the model. In the train and test approach, the entire dataset is split into two subsets, namely training set and testing set. The training dataset includes all the attribute features and target features. The model is trained using a training dataset and then later tested using a test set. The input element is provided to the test set to make the predictions. The predictions are then compared to the expected values in the dataset to get the model performance. For our model, I divided the data into 80% training data and 20% test data.

I selected around 3 classifiers to build, train and test the model. At the end after evaluating the results from individual classifiers, Stacking Classifier is used to create an ensemble model to further improve model performance [12]. The models were then saved on the disk using the pickle function.

1. ***Naïve Bayes:*** This algorithm works best for binary or multi-class classification problems with categorical data. For our model, I had used a gaussian bayes model which assumes the input features follow the normal distribution. For our model, I used Gaussian Naïve Bayes [11] which is a variation of Naive Bayes that assumes the data is normally distributed.
2. ***Decision Trees:*** This algorithm is easy, fast and operates well on large datasets. It requires less computational power and is easy to visualise [12]. The predictions on the target feature are made on the basis of the decision rules inferred from the input variables. I used a random state of 42 and limited the depth of the tree to 10 levels for building our model. Setting a tree depth level ensured that the model is not over fitted.
3. ***Random Forest:*** It is an ensemble learning technique which uses multiple decision trees to make the predictions. The prediction is made by aggregating the results of individual classifiers [12]. Since this method uses a bagging, it increases the accuracy of the model. For our model I have used about 75 estimators, with maximum depth of 5 and random state of 42.

## **4.6 Model validation**

To validate the model’s performance, I have used k-fold cross validation technique on the dataset to help detect overfitting. First, the entire dataset is divided into the k folds and for each unique fold (hold out data), the remaining data is taken as training data. Then the model is fit on the training set and evaluated on the holdout data. I have taken the number of folds as 5 (i.e. cv=5), which mean the data will be split using 5 folds of train and test data. During each fold, one-fold is considered for testing and rest will be training and so on [10].



Figure 6: K-fold cross validation process

To implement the approach, I have used the feature\_importances function of random forest, I got the significant features in the dataset. Then I ran the random forest classifier on these tuned datasets to check if I can get any better results. After executing the model with the selected feature set, I overall got an accuracy of greater than 90% which is good. I also calculated the mean cross validation and standard deviation of cross- validation score to evaluate the model which both came out to be greater than 90%, signifying good model performance.

## **4.7 Flask web application**

In order to accept the real time input from the user and display whether the request is safe or not, I have developed a flask application that accepts an excel file as an input from the user forms and submit it for preprocessing. Once the preprocessing of user input is completed, the user will proceed to predict the results.

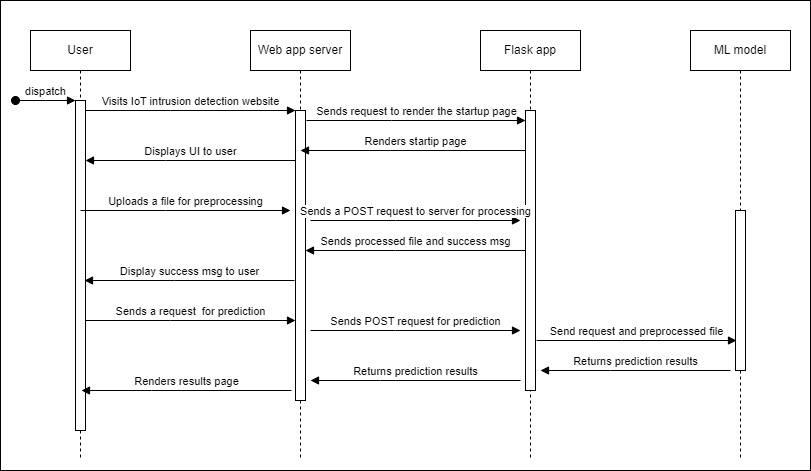


Figure 7: Sequence diagram

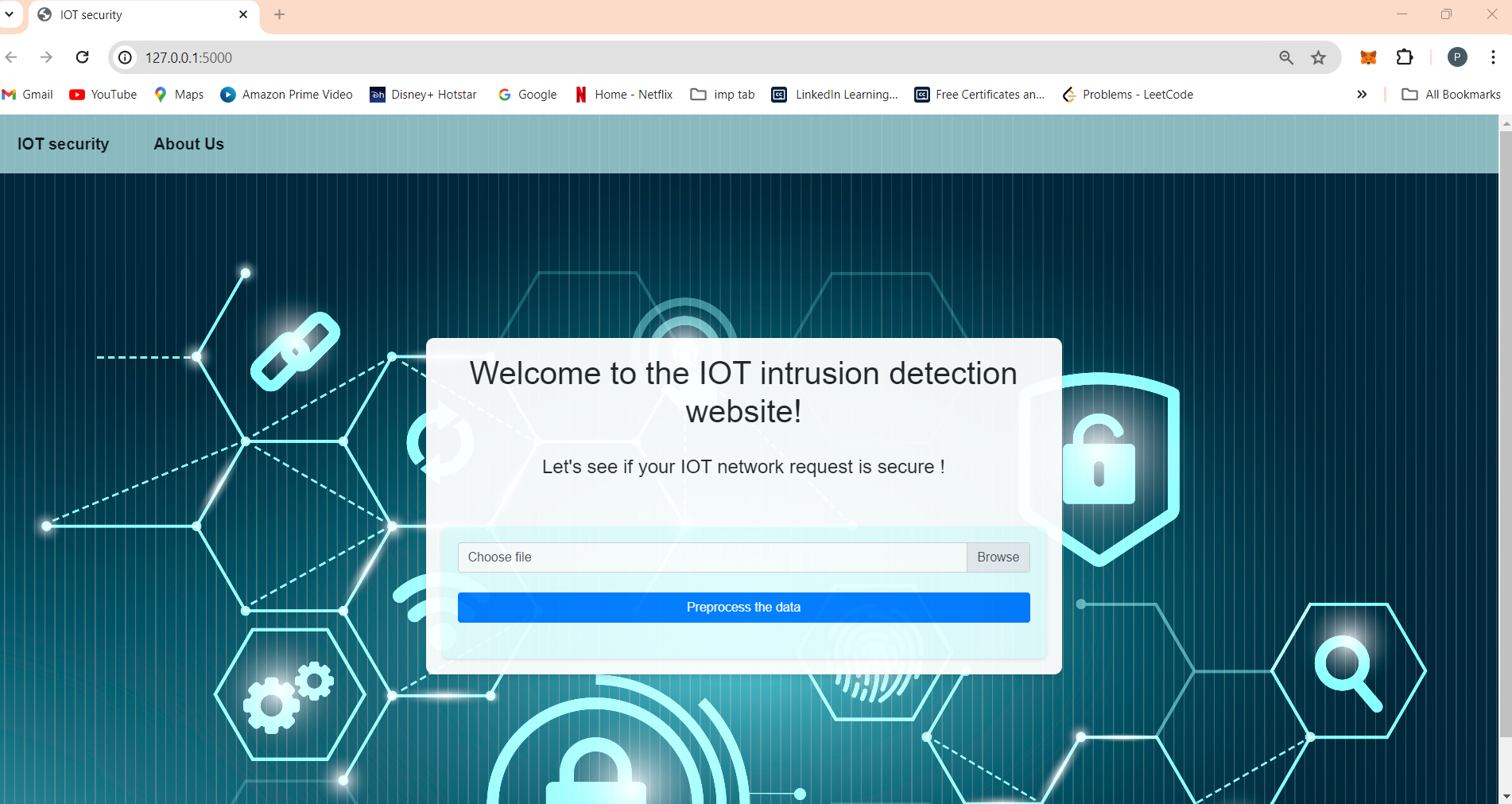


Figure 8: Website welcome page

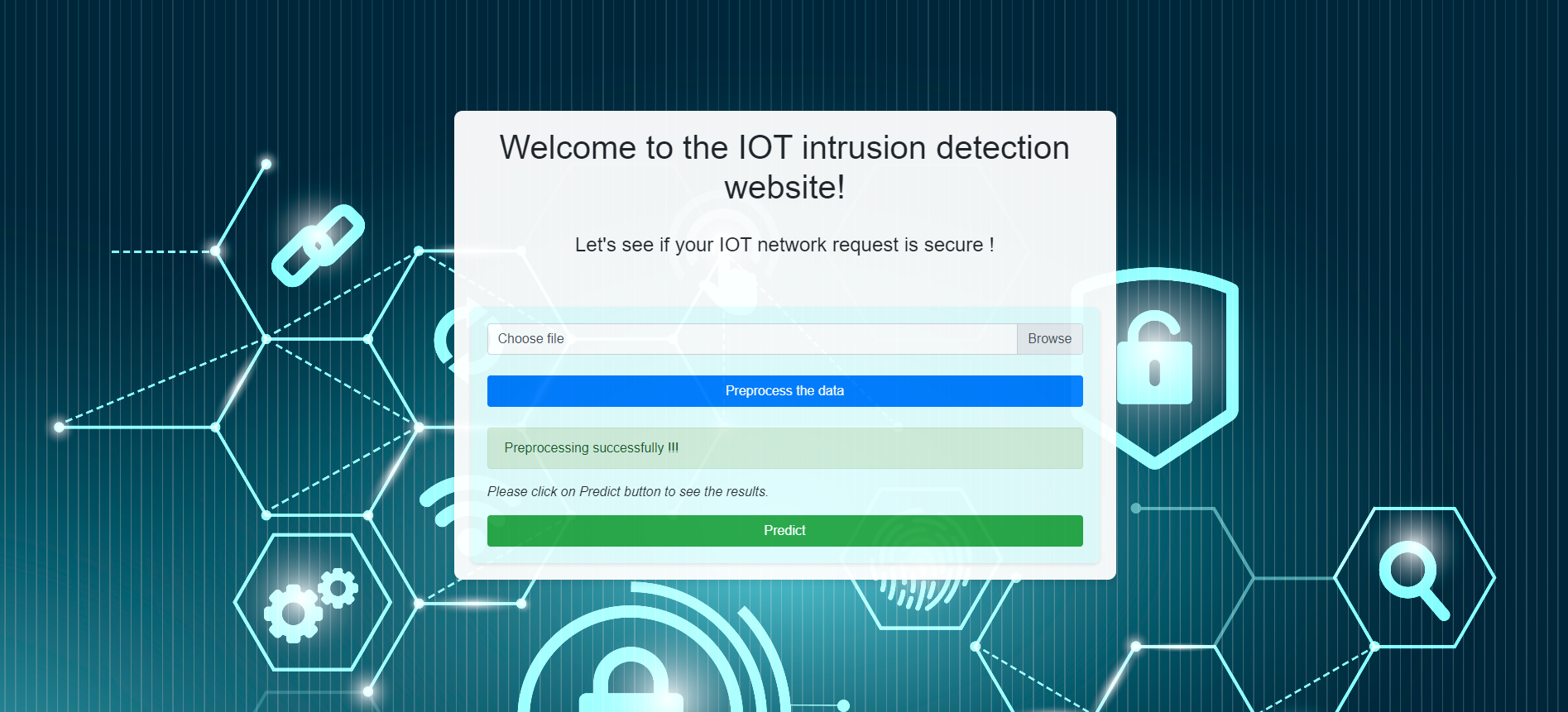


Figure 9: Preprocessing form page



Figure 10: Results page – Unsafe request

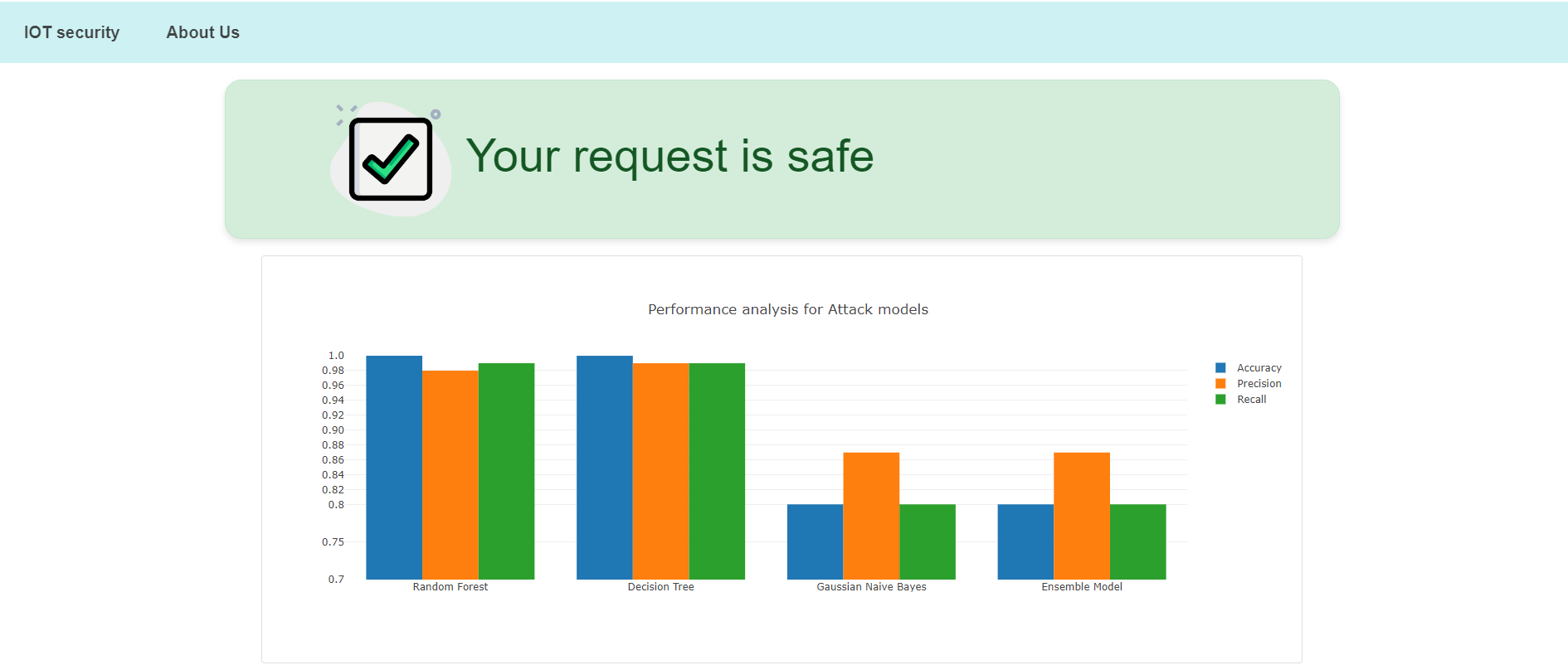


Figure 11: Results page –Safe request

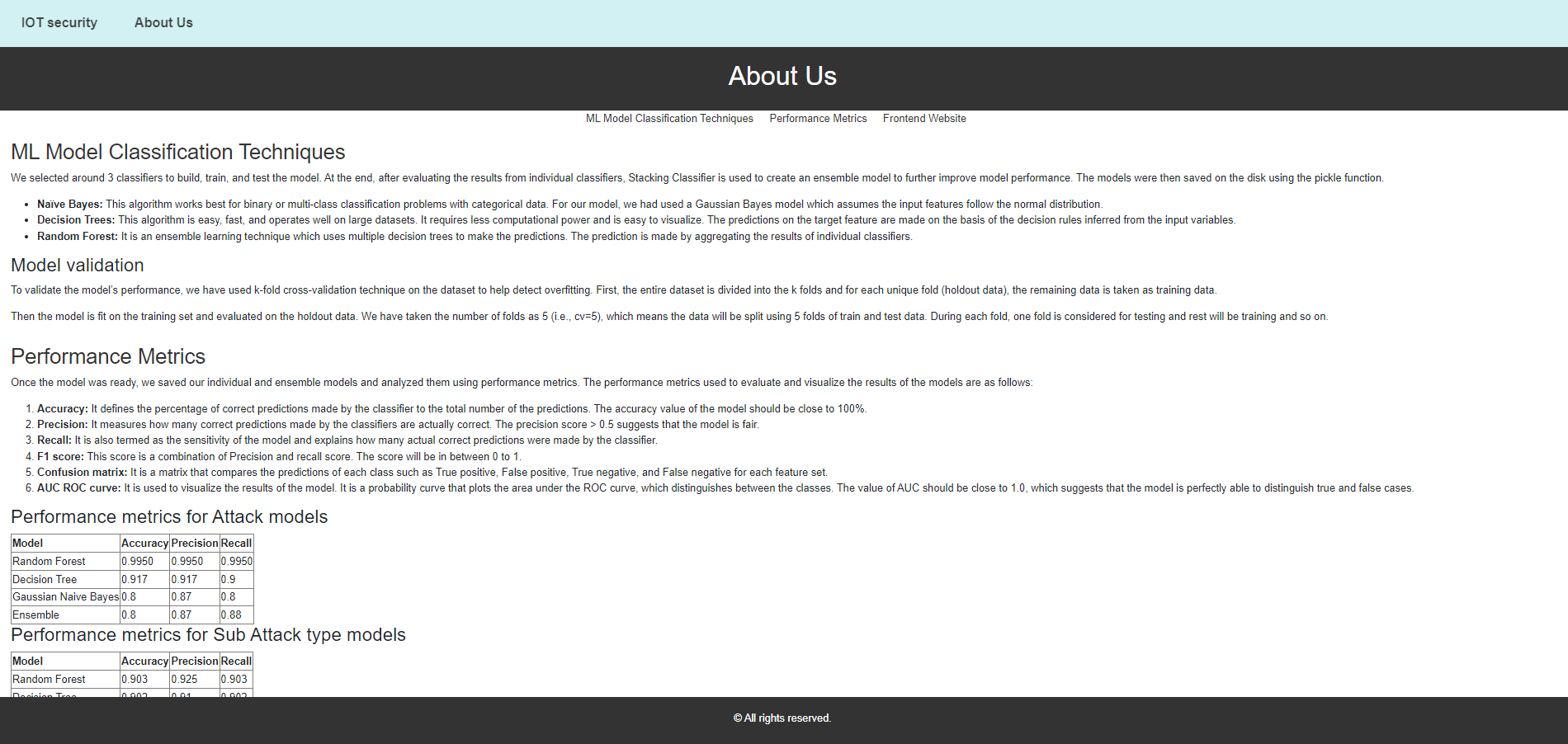


Figure 12: About Us page

The web application module consists of below section to render a simple, user friendly interface for IOT threat prediction:

***Imports:*** The necessary libraries and modules, including Flask, render\_template, request, pandas, pickle, and LabelEncoder, are imported to facilitate web development and machine learning functionalities.

***Flask App Initialization:*** An instance of the Flask application is created using Flask(\_\_name\_\_).

***Model Loading:*** Trained machine learning models for classifying attacks and subattacks are loaded into memory using the pickle.load() function.

***Data Preprocessing:*** A function preprocess\_data() is defined to preprocess the uploaded data and prepare it for prediction. This function replaces the actual preprocessing logic used in the system.

***Label Encoding:*** The label\_data() function is implemented to encode categorical features using LabelEncoder from scikit-learn. This function prepares the label encoders for later use in decoding predictions.

***Route Definitions:***

Index Route: The root route ('/') renders the index.html template, which contains the file upload form.

Upload Route: The /upload route handles file uploads, preprocesses the data, and saves the pre-processed data to a CSV file.

Prediction Route: The /predict route processes the preprocessed data, performs predictions using the loaded machine learning models, and renders the results.html template with the prediction outcomes.

***Model Prediction***: Predictions are made using the loaded machine learning models, and the results are displayed in a tabular format.

***Result Rendering:*** Based on the prediction outcomes, appropriate messages are generated to inform users about the security status of their requests.

## **4.8 Reporting**

To display the evaluation analysis of the classification algorithms used for building models, I have used plotly library to display graphs in real time. The graph plots models on the X-axis with performance metrics value on Y axis. The graph plots accuracy, precision and recall for both attack and sub-attack type models.

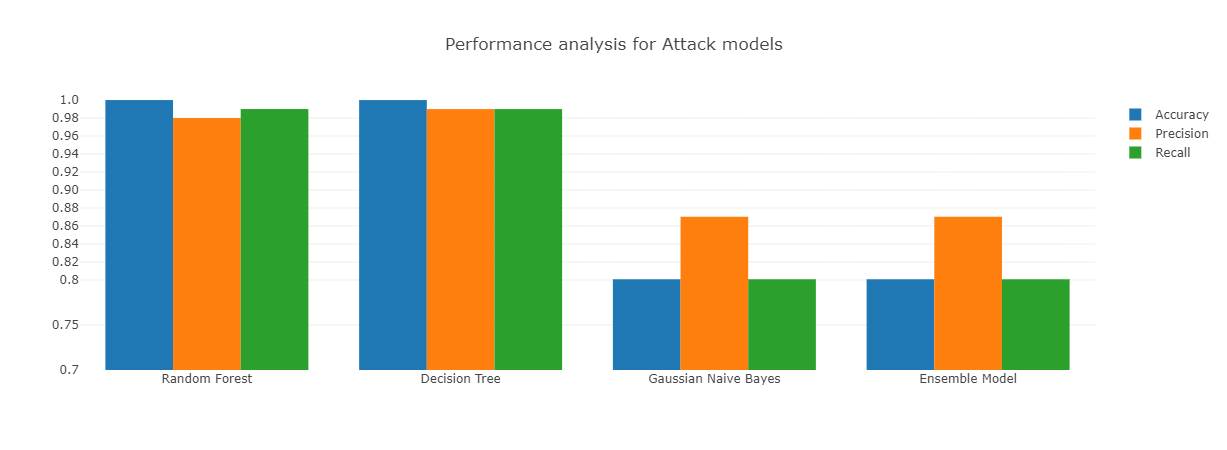


Figure 13: Attack Performance analysis graph

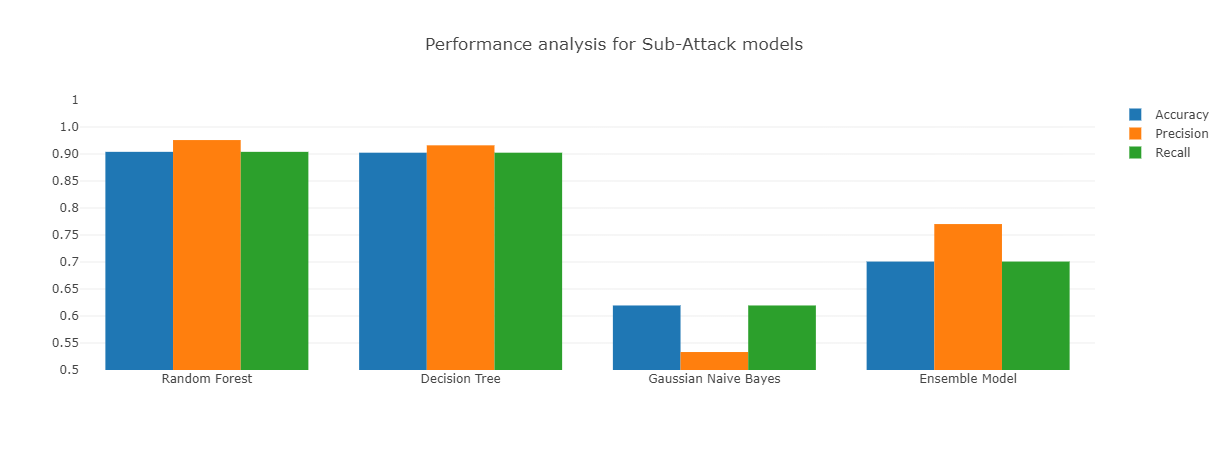


Figure 14: Sub Attack Performance analysis graph

# **5. Evaluation and Optimization or Future Works**

## **5.1 Model Evaluation and Analysis**

Once the model was ready, I saved the individual and ensemble models and analysed them using performance metrics. The performance metrics used to evaluate and visualise the results of the models are as below:

* ***Accuracy:*** It defines the percentage of correct predictions made by the classifier to the total number of the predictions [13]. The accuracy value of the model should be close to 100%.
* ***Precision:*** It measures how many correct predictions made by the classifiers are actually correct. The precision score >0.5 suggests that model is fair [14].
* ***Recall:*** It is also termed as the sensitivity of the model and explains how many actual correct predictions were made by the classifier.
* ***F1 score***: This score is a combination of Precision and recall score. The score will be in between 0 to 1 [13].
* ***Confusion matrix***: It is a matrix that compares the predictions of each class such as True positive, False positive, True negative and False negative for each feature set.
* ***AUC ROC curve:*** It is used to visualise the results of the model. It is a probability curve that plots the area under the ROC curve, which distinguishes between the classes. The value of AUC should be close to 1.0 which suggests that model is perfectly able to distinguish true and false cases [15].

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** |
| Random Forest | 0.9950 | 0.9950 | 0.9950 |
| Decision Tree | 0.917 | 0.917 | 0.9 |
| Gaussian Naive Bayes | 0.8 | 0.87 | 0.8 |
| Ensemble | 0.8 | 0.87 | 0.88 |

Table 1: Performance metrics for Attack models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** |
| Random Forest | 0.903 | 0.925 | 0.903 |
| Decision Tree | 0.902 | 0.91 | 0.902 |
| Gaussian Naive Bayes | 0.61 | 0.5 | 0.61 |
| Ensemble | 0.7 | 0.77 | 0.7 |

Table 2: Performance metrics for Sub Attack type models

## **5.2 Model optimization**

In the model optimization phase, I employed several techniques to enhance the performance and efficiency of our machine learning models. One crucial aspect was feature selection, where based on the correlation matrix I meticulously narrowed down the feature variables to those most relevant for our predictive tasks. This process helped improve the model's accuracy and generalization capabilities by focusing on the most informative features while reducing noise and overfitting risks. Also, the data was imbalance in nature so to make model impartial I tried to use oversampling and under sampling techniques to balance the data for target features.

Additionally, I utilized k-fold cross-validation techniques [11] to assess model performance and detect potential overfitting issues. By splitting the dataset into multiple subsets and iteratively training and testing the model on different partitions, I could effectively evaluate its robustness and ensure it could generalize well to unseen data. Moreover, this approach allowed me to identify the best feature set for building our models, optimizing their predictive power and ensuring their reliability in real-world scenarios.

## **5.3 Future work**

In future scope of work, I would try to refine the existing model and use deep learning for training purposes. Also, I would like to collect more relevant data instances and train algorithms to get accurate output for the model. The existing data consisted of null values and outliers which affected the performance of the model. I would also add more values to the target feature (i.e., attack and sub attack type) to more precisely identify the type of attack and sub attack rather than generalising whether the request is safe or not. Along with it, I can also add the behaviour analysis of the IoT model by using the day-to-day data and can also identify which IoT device is more prone towards an intrusion attack.

# **6. Conclusion**

In this project I have built the models that predict whether the incoming request to IoT server is safe or not based on the user data captured via day-to day activities in a IoT network. The model learns and trains on IoT attack data and provide the output for the target feature like Attack and Attack Subtype using ML techniques such as Naïve Bayes (Gaussian), Random Forest and Decision tree. Since it is a multiple classification problem, I didn’t get good results at the start however by using optimization techniques I was able to achieve expected results. I also developed an ensemble ML model using Stacking Classifier to get the best of all the individual model and was able to achieve the performance of greater than 90%. Taking into consideration the complexity, time and the analysis of ML models based on performance metrics I can conclude Random Forest ML technique is well suited to solve the problem followed by Decision Tree. The Naïve Bayes technique has the lowest performance among all.

I have developed a web application using Flask for user to interact with the system and display the results. Overall, the turnaround time of UI to accept the input and provide output is around 30-40 secs. I have provided results visualization to analyse and examine the efficiency of all the models for further improvement

**Bibliography (citing all references used)**

[1] Kaggle dataset website:

<https://www.kaggle.com/code/sripadkarthik/iot-data-for-intrusion-detection-using-ml-and-dl/input?select=BoTNeTIoT-L01-v2.csv>

[2] Bhabendu Kumar Mohanta, Debasish Jena, Utkalika Satapathy, Srikanta Patnaik,

Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology, Internet of Things, Volume 11, 2020, 100227, ISSN 2542-6605, <https://doi.org/10.1016/j.iot.2020.100227>.

(<https://www.sciencedirect.com/science/article/pii/S2542660520300603>)

[3] H. Wu, H. Han, X. Wang and S. Sun, "Research on Artificial Intelligence Enhancing Internet of Things Security: A Survey," in IEEE Access, vol. 8, pp. 153826-153848, 2020, doi: 10.1109/ACCESS.2020.3018170.

[4] Kuzlu, M., Fair, C. & Guler, O. Role of Artificial Intelligence in the Internet of Things (IoT) cybersecurity. Discov Internet Things 1, 7 (2021).

<https://doi.org/10.1007/s43926-020-00001-4>

[5] ZhiyanChen, JinxinLiu, YuShen, MuratSimsek, BurakKantarci, HusseinT.Mouftah, and PetarDjukic. 2022. Machine Learning-Enabled IoT Security:Open Issues and ChallengesUnder AdvancedPersistentThreats. ACM Computing Surveys (2022), 35pages.

[6] Zewdie, Temechu & Girma, Anteneh. (2020). IoT security and the role of AI/ML to combat emerging Cyber threats in Cloud Computing Environment. Information Systems Journal. 21. 253 - 263. 10.48009/4\_iis\_2020\_253-263.

[7] Girma, Anteneh. (2018). Analysis of Security Vulnerability and Analytics of Internet of Things (IOT) Platform. 10.1007/978-3-319-77028-4\_16.

[8] IOT classification using ensemble methods:

<https://github.com/jorgesandoval/iotthreat-classification-ensemble/blob/main/README.md#-dataset>

[9] Human Activity Recognition using Ambient Sensors

<https://ieeexplore.ieee.org/abstract/document/10323956?casa_token=XM7m71S-Y4sAAAAA:fcachQoJaHv5XtOHWnZs1FFgHNH_hK9E-6VEv7k-t4K9GX-AmvXOPRiAqq1X8hbcpo0KNXaKuZA>

[10] K-Fold Cross Validation Technique and its Essentials

<https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/>

[11] Papalote, “How to implement a Gaussian Naive Bayes Classifier in python from scratch?” Medium, Feb. 13, 2020.

<https://towardsdatascience.com/how-to-impliment-a-gaussian-naive-bayes-classifier-in-python-from-scratch-11e0b80faf5a>

[12] Oracle.com, 2023.

<https://blogs.oracle.com/ai-and-datascience/post/random-forests-decision-trees-and-ensemble-methods-explained#:~:text=The%20random%20forest%2C%20first%20described>

(accessed May 11, 2023)

[13] R. Vidyalaya, “Performance Metrics for Classification Machine Learning Problems,” Medium, Jul. 26, 2020.

<https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007>

[14] Evaluation Metrics For Classification Model | Classification Model Metrics, Analytics Vidhya, Jul. 20, 2021.

<https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/#:~:text=There%20are%20many%20ways%20for>

[15] R. Vickery, “8 Metrics to Measure Classification Performance,” Medium, Dec. 08, 2021. <https://towardsdatascience.com/8-metrics-to-measure-classification-performance-984d9d7fd7aa>