**Design of an Intrusion Detection Model using Machine Learning Algorithms for IoT-Enabled Smart Home**

**Abstract**

Machine learning (ML) provides effective solutions to develop efficient intrusion detection system (IDS) for various environments. In the present paper, a diversified study of various ensemble machine learning (ML) algorithms has been carried out to propose design of an effective and time-efficient IDS for Internet of Things (IoT) enabled environment. In this paper, data captured from network traffic and real-time sensors of the IoT-enabled smart environment has been analyzed to classify and predict various types of network attacks. The performance of Logistic Regression, Random Forest, Extreme Gradient Boosting, and Light Gradient Boosting Machine classifiers have been benchmarked using an open-source largely imbalanced dataset ‘DS2OS’ that consists of ‘normal’ and ‘anomalous’ network traffic. An intrusion detection model “LGB-IDS” has been proposed using the LGBM library of ML after validating its superiority over other algorithms using ensemble techniques and on the basis of majority voting. The performance of the proposed intrusion detection system is suitably validated using certain performance metrics of machine learning such as train and test accuracy, time efficiency, error-rate, true-positive rate (TPR), and false-negative rate (FNR). The experimental results reveal that XGB and LGBM have almost equal accuracy, but the time efficiency of LGBM is much better than RF, and XGB classifiers. The main objective of the present paper is to propose a design of an efficient intrusion detection model with high accuracy, better time efficiency, and reduced false alarm rate. The experimental results show that the proposed model achieves an accuracy of 99.92% and the time efficiency comes to be much higher than other prevalent algorithms-based models. The threat detection rate is greater than 90% and less than 100%. Time complexity of LGBM is also very much low as compared to other ML algorithms.

**CHAPTER 1**

**INTODUCTION**

**1.1 Aim of the Study**

The study aims to design and implement an intrusion detection model using machine learning algorithms specifically tailored for IoT-enabled smart homes. The model will focus on enhancing security and privacy by detecting unauthorized access or abnormal activities within a smart home environment, thereby mitigating potential threats to residents and their data.

**1.2 Objectives**

1. **To analyze security vulnerabilities** in IoT-enabled smart home environments, with a particular focus on intrusion vectors that exploit IoT devices and network architecture.
2. **To evaluate and select appropriate machine learning algorithms** for intrusion detection, based on performance metrics like accuracy, precision, and response time.
3. **To design and implement a prototype intrusion detection model** capable of detecting various types of network anomalies and threats in real-time.
4. **To validate the model’s effectiveness** by testing it on a simulated smart home IoT network, measuring its detection rate, false positive rate, and response efficiency.
5. **To optimize the model** for compatibility with limited-resource IoT devices, ensuring that the solution remains lightweight and efficient in real-world applications.

**1.3 Scope**

* **System Design and Implementation**: The study will cover the design of the intrusion detection model and the implementation of selected machine learning algorithms on an IoT network simulation.
* **Smart Home Use Case**: The model will specifically focus on IoT-enabled smart homes, where multiple connected devices increase security risks.
* **Performance Evaluation**: Testing and evaluation will be conducted to ensure the model can accurately detect intrusions with minimal false positives.
* **Scalability and Flexibility**: The project will explore ways to adapt the model for future smart home expansions, enabling it to handle additional devices and evolving attack patterns.

**1.4 Introduction**

IOT devices are small sensors which runs on battery to sense its environment data and then used internet connection to transmit sense data to centralized server for further processing. Smart home electricity sensors, CCTV cameras and other surveillance devices are the examples of IOT sensors. This sensors runs on battery and small memory chip and cannot afford to install antivirus software to detect and prevent any malicious activities so it can be easily hack to alter sense data with false information and this false data send to centralized server which may take incorrect action based on false data.

To detect and avoid such intrude attacks author of this paper employing Machine learning algorithms which can analyse packet data to predict weather packet is benign or malicious. In propose paper author using WIRESHARK tool to capture all network packets data and then input to ML algorithms which will predict weather packet is benign or malicious.

In past few years, the world has witnessed a rapid increase in cybercrimes in the IoT ecosystem [1]. IoT-enabled smart home environments are major targets of several types of intrusions and malicious activities which put the security and privacy of IoT consumers at risk. Thus, network security and user privacy are the major concerns in IoT environment which motivate researchers to develop an effective IDS. The proposed work is motivated by several challenges. The increasing size, autonomous nature, and attractive features of IoT network draws the attention of cybercriminals.

The dramatic increase of crime rate in IoT ecosystem motivates researchers to develop more effective and intelligent solutions to prevent and detect such crimes. Cyber-attacks can be reduced by recognizing any suspicious activity occurring in the network to catch the malicious actors before they breach a particular network or a system. Hence, to monitor network and system assets for unexpected activities is also the motivation behind proposing an intrusion detection system.

An intrusion detection system can raise an alert if any component of a system is under attack [2]. However, it is hard to identify unknown attacks using conventional investigation processes and they can be more harmful than known attacks. Currently, the detection rate of threats is extremely lower than the threat increment rate. Most of the connected IoT devices around the world generally exhibit no symptoms of attacks.

Hence, people don’t realize the presence of anomalous events being launched in their smart network. IoT systems can be victimized due to certain prevalent vulnerabilities. An intrusion detection system (IDS) is one of the crucial components of the cybercrime investigation process [3]. An IDS model scrutinizes the system activities to predict and detect attack patterns and to monitor every user’s behavior in order to prevent any kind of security violations.

A well-developed intrusion detection system can efficiently deal with various forms of cyber-attacks triggered in a smart home environment. The present paper introduces a fast and efficient intrusion detection model that has been evaluated on an IoT dataset ‘DS2OS’ [4]. This dataset has been created in a smart home environment. The dataset accommodates traces captured from various smart home devices namely light controllers, batteries, washing machines, thermometers, smartphones, smart doors, and movement sensors.

The dataset also presents the communication between various IoT devices. Traced patterns of network traffic are able to detect various attack (‘DoS attack’, ‘data probing’, ‘malitious control’, ‘malitious operation, ‘scan’ ‘spying’, ‘wrong Setup’) behaviors as well as ‘normal’ behavior.

Machine learning is an extensively used technique to harmonize IDS with intelligent information systems, to detect various types of malicious activities in a smart network [5]. Network traffic passing through the nodes connected in IoT infrastructure needs to be discriminated between benign and malicious traffic. In most cases, the majority of network traffic showcases normal (benign) behavior. If malicious traffic also shows normal behavior, it could be more dangerous and may lead to the problem of high attack detection rate with low false alarm rate (FAR) [6].

The performance of ML algorithms can be strengthened through various problem optimization techniques. The ensemble technique of ML is an advanced convincing tool that can upgrade the performance of existing models [7] designed for application forecasting in different application areas. In ML, the ensemble-based learning methods perform classification by creating and integrating multiple models to solve a problem [8]. Ensemble-based ML models combine multiple base models and provide better prediction performance than the conventional classification models.

Besides accuracy, latency and true positive rate must also be considered to be essential metrics for evaluating any proposed predictive model. The authors in the present paper aim to develop a model for intrusion detection for IoT systems using light gradient boosting method (LGBM), which is a fast gradient boosting ensemble library of supervised machine learning [9]. This model aims to identify and classify various types of attacks existing in an IoT network. The following are the major research contributions of the present paper:

* To study various machine learning classifiers and to identify the best classifier using the ‘Voting’ method of ensemble technique.
* To design a time-efficient realistic intrusion detection system using an ML-based light gradient boosting machine (LGBM) ensemble classifier by predicting network traffic behavior in an IoT-enabled smart home environment using benchmark dataset ‘DS2OS’.
* To remove irrelevant and repetitive features using dimensionality reduction and feature reducing approaches.
* To evaluate and compare the performance of the proposed intrusion detection model with related approaches of state-of-art in terms of train and test accuracy, TPR, FPR, error-rate, and above all time efficiency.

Gradient boosting-based machine learning (ML) ensemble algorithm is a good approach to perform classification for predicting the behavior of network traffic. The information captured from connected nodes, real-time sensors, and network traffic can become a source of evidence that may help cyber forensic investigators to identify the sources of threats. In the present paper, the proposed intrusion detection model has been trained and tested using the ‘DS2OS’ dataset which is also known as the ‘mainSimulationAccessTraces’ IoT dataset [4].

The metrics utilized to assess the performance of the proposed model are training and testing accuracy, prediction error, time efficiency, true positive rate, false positive rate, etc. The remaining paper is organized as follows. Section II presents several research and state-of-art performed by different researchers. Section III presents a detailed discussion of certain machine learning multi-class classifiers used to achieve the objective. Section IV presents the research methodology of the proposed model that promises to conduct intrusion detection in an IoT-based smart home environment. Section V presents the construction methodology of ensemble-based IDS. Section VI gives the result discussion of various operations performed for the proposed intrusion detection model. Finally, Section VII presents the conclusion of the study and the research work.

**CHATER 2**

**LITERATURE SURVEY**

1. **Atlam, H. F., Alenezi, A., Alassafi, M. O., & Alshdadi, A. A. (2020)**: This chapter explores IoT security concerns, cybercrime risks, and digital forensics. It discusses the vulnerabilities IoT devices face, including potential exploitation by cybercriminals, and presents digital forensic techniques essential for investigating IoT-related crimes. The authors emphasize the need for robust security measures to protect data integrity, confidentiality, and availability across IoT ecosystems.
2. **Alkasassbeh, M., & Al-Haj Baddar, S. (2022)**: This paper surveys and categorizes intrusion detection systems (IDS) applicable to modern networks, focusing on the latest advances in IDS taxonomy and techniques. It discusses the unique challenges posed by IoT environments and provides insights into state-of-the-art approaches, highlighting strengths, limitations, and application-specific considerations in the evolving cybersecurity landscape. *[Source: Arab Journal of Science and Engineering]*
3. **Sun, Z., Lv, Z., Wang, H., Li, Z., Jia, F., & Lai, C. (2020)**: This study introduces a novel data scheduling optimization algorithm designed to enhance the efficiency of IoT data transmission in cloud computing environments. It leverages resource allocation and prioritizes data processing to reduce latency and improve IoT performance, making it suitable for real-time applications across various IoT networks. *[Source: IEEE Access]*
4. **DS2OS Traffic Traces (2018)**: The DS2OS dataset consists of traffic traces collected in a decentralized IoT environment. These traces provide valuable insights into network behavior, enabling researchers to analyze and develop IoT-focused intrusion detection models. The dataset includes diverse IoT interactions that are useful for machine learning and security model testing. *[Source: Kaggle]*
5. security threats and cyber attacks across different layers of the IoT stack. It provides a layered approach to IoT security, detailing vulnerabilities at each level, from physical devices to application interfaces, and proposes best practices to address these issues. *[Source: Journal of Theoretical and Applied Information Technology]*
6. **Sagu, A., Gill, N. S., Gulia, P., Chatterjee, J. M., & Priyadarshini, I. (2022)**: The authors present a hybrid deep learning model enhanced with a self-improving optimization algorithm to detect IoT security attacks. This model adapts to changing IoT threat environments and effectively identifies malicious activity, addressing both network efficiency and detection accuracy. *[Source: Future Internet]*
7. **Chen, T., & Guestrin, C. (2016)**: This paper introduces XGBoost, a scalable machine learning system for tree boosting that significantly improves the computational efficiency and accuracy of classification and regression tasks. XGBoost has been widely adopted for its performance, scalability, and ease of integration into data-driven applications, including intrusion detection systems. *[Source: ACM SIGKDD Conference on Knowledge Discovery and Data Mining]*
8. **Friedman, J. H. (2001)**: This foundational work on gradient boosting introduces an iterative, optimization-based approach to improve prediction accuracy. Gradient Boosting Machines (GBM) are renowned for their capability to handle complex datasets, making them ideal for high-dimensional problems, including cyber intrusion detection. *[Source: The Annals of Statistics]*
9. **Zhao, X., & Zhao, Q. (2021)**: This study explores optimized LightGBM algorithms for stock prediction, focusing on cost-awareness in decision-making processes. The optimized LightGBM model demonstrates significant predictive capabilities, suggesting potential applications in other domains requiring accurate, cost-effective predictions, such as anomaly detection in IoT. *[Source: IEEE International Conference on Cybernetics]*
10. **Rehman, E., Haseeb-ud-Din, M., Malik, A. J., et al. (2020)**: This research addresses intrusion detection in IoT, analyzing attack vectors and proposing machine learning-based countermeasures. By examining the effectiveness of different ML models, the paper provides insights into securing IoT networks against common threats, balancing detection accuracy with resource efficiency. *[Source: The Journal of Supercomputing]*
11. **Htwe, C. S., Thant, Y. M., & Thwin, M. M. S. (2020)**: This study investigates botnet detection in IoT networks using machine learning. By analyzing botnet behavior, the authors develop a model that successfully identifies botnet activity, offering a framework that enhances IoT security through proactive botnet detection. *[Source: Journal of Physics: Conference Series]*
12. **Zhou, Y., Cheng, G., Jiang, S., & Dai, M. (2020)**: This paper proposes an efficient intrusion detection system combining feature selection and an ensemble classifier. The authors demonstrate the system’s capability to achieve high detection accuracy while minimizing computational requirements, making it suitable for real-time IoT applications.

**CHAPTER 3**

**EXISTING METHOD**

So far, several pieces of research have been conducted to resolve the problems related to intrusion detection in various application areas [10]. The models built using machine learning can efficiently deal with large and complex data. Several techniques of machine learning are used for achieving remarkable results. The literature survey can be further classified on the basis of the nature of selection methods and single-class and multiclass classification methods.

In 2020, Htwe et al. [11] applied the classification and regression tree (CART) method for their proposed intrusion detection architecture which was implied on an IDS dataset ‘N-BaIoT’. In this paper, the authors state that the results obtained by their proposed classifier are better than Naïve Bayes classifiers. But the main shortcoming of their proposed system is that the authors evaluated the model only on one metric i.e., accuracy.

In 2020, Zhou et al. [12] proposed an intelligent IDS framework using feature-selection and ensemble learning mechanisms. They also proposed the CFS-BA heuristic algorithm for dimensionality reduction in order to select the most relevant and distinct subsets and show correlations between features. The proposed ensemble approach was based on c4.5 and RF by Penalizing Attributes algorithms with an average of probability rule. The probability distributions of base learners were incorporated using voting techniques for better performance of attack recognition. The results of the proposed system were evaluated using NSL-KDD, AWID, and CIC-IDS2017 datasets. But in this proposed work, the authors’ concern about time efficiency was missing.

In 2020, a comprehensive study was carried out by Verma et al. [13] on ML classifiers for the development of anomaly-based IDS in order to secure IoT systems against DoS attacks. The researchers evaluated the reliability of the anomaly-based intrusion detection model by evaluating on several existing IoT datasets including CIDDS-001, NSL-KDD, and UNSW-NB15. Raspberry Pi was utilized to perform a statistical significance test in order to examine the response time of classifiers as per the requirement of applications. In this study, authors compared several ML algorithms of single classifiers CART and MLP and ensemble classifiers RF, Adaboost, Gradient Boosted Machine, Extreme Gradient Boosting and Extremely Randomized Trees to identify an optimal model for IDS.

In 2021, Hadem et al. [14] proposed an IDS using SVM for software-defined networking. The main focus of this approach is maximizing the detection accuracy with less overhead of computation and improving memory saving. The accuracy performance of the proposed scheme has been recorded at 95.98% using the ‘NSL-KDD’ full set dataset while on the selected features dataset it has been recorded at 87.74%.

In 2021, Kumar et al. [15] proposed a cyber-attack detection system using random forest, KNN, XGBoost algorithms. The proposed work has been evaluated on BoT-IoT and DS2OS datasets. The proposed work claims to achieve 90% to 100% detection rate. But it is not possible to detect every threat. There might be some hidden threats which can’t be recognized. In other words, there is always the possibility of false alarms. XGBoost is a leading method that is widely used for designing IDS model due to high accuracy. But in this paper too authors ignored the latency factor. Random forest, KNN and XGB are highly time consuming.

In 2021, Huˇc et al. [16] proposed an anomaly detection model for edge devices. In the present paper, authors have analyzed different machine learning algorithms and evaluated the performance of the proposed intrusion detection model on a largely imbalanced dataset ‘DS2OS’. In this study, the entire dataset was divided into training (80%) and test (20%) datasets, and then a new smaller dataset was created by selecting the samples randomly for balancing the dataset. Models based on different ML algorithms (LR, DT, SVM, RF) have been compared with ANN algorithm-based models. The performance of the models has been evaluated only in terms of accuracy. Classification results have been presented using a confusion matrix. Besides this, smaller balanced training datasets have been determined with clustering and performance has been compared with bigger imbalanced datasets.

In 2022, Devprasad et al. [17] proposed a context-adaptive classification mechanism utilizing hierarchy-based chi-square and bat algorithms. The algorithm was deployed and tested on ‘NSL-KDD’ and ‘UNSW-NB15’ datasets. For experimentation, authors used Decision tree (DT) and SVM algorithms which were implemented with NSL-KDD and UNSW-NB15 datasets. The ensemble classifier returns 89.43% prediction accuracy and 3.215% FPR. Accuracy rate is not so good and they did not concern about execution time.

In 2022, Gupta et al. [18] proposed cost-sensitive network-based IDS for handling class imbalance using deep learning and ensemble algorithms. This model was evaluated on CIDDS-001, NSL-KDD, and CICIDS2017. The proposed system focused on the identification of new and infrequent attacks in computer networks with a high detection rate. The entire work is divided into three stages. The first stage utilizes a deep neural network to separate normal and suspicious network traffic. In the second stage, XGB was deployed in the second stage for the classification of major attacks and in the third stage, Random Forest was deployed for the classification of minor attacks. The proposed model returned 99% accuracy for NSL-KDD, 96 % accuracy for CIDDS-001, and 92% accuracy for CIDIDS2017.

In 2022, Çetin [19] proposed a model for imbalanced network attack traces. The performance of the proposed model has been evaluated on some imbalanced datasets namely, DARPA98, NSL-KDD, KDD99, USWN-NB15, and Caida DDoS. Besides being imbalanced, these datasets are also highly dimensional. The proposed model was targeted to overcome high dimensionality and obtain high accuracy. The model was also tested at the CICIDS2017 dataset using a genetic algorithm. The study made performance evaluations on F1-score and G-mean metrics in order to select the most effective classifiers. Authors in this paper also did not concern about time.

In 2022, Xu and Fan [20] proposed an IDS that was based on XGB and logarithmic auto-encoder. The evaluation of the proposed model has been performed on the CICIDS2017 and UNSW-NB15 datasets. The accuracy score recorded on CICIDS2017 has been recorded as 99.92% and on UNSW-NB15 it has been recorded as 95.11%. In this study, the researchers also evaluated the run-time performance of different classifiers.

In 2022, Saheed et al. [21] also proposed an IDS using a machine learning algorithm for detecting network attacks in an IoT environment. The min-max scheme has been used in the first step for normalization on the UNSW-NB15 dataset and dimensionality reduction has been used in the next step using PCA. The dataset has been trained using various machine learning classifiers including XGBoost, CatBoost, KNN, SVM, and naïve Bayes algorithms. The experimental analysis has been done using the BoT-IoT dataset and compared the results obtained on the UNSW-NB15 dataset. The accuracy score of boosting algorithms (XGB and CatBoost) was higher than other ML classifiers. Due to the substantial growth of Industrial IoT (IIoT), this area creates the highest opportunities for cybercriminals to perform easy attacks.

In 2022, Le et al. [22] proposed multiclass classification-based IDS for an imbalanced dataset of IIoT. ML-based XGBoost classifier was used to detect abnormal behavior of network traffic in order to protect the network from cyber-attacks of similar behavior. Two modern IIoT datasets, TON\_IoT, and X-IIoTDS have been used in this study, reflecting the signs of modern network traffic. The proposed model outperforms with a good attack detection rate that has been recorded as 99.9% and 99.87%.

In 2023, Mohamed et al. [23] also proposed an IDS using fog and cloud computing. Authors used Gated Recurrent Unit and Bidirectional LSTM to recognize the existence of network attacks. Until the emergence of IoT, many expert digital forensic approaches were developed for botnet detection in computers and similar systems. Authors calculated the run-time but the accuracy calculated by model was only 96%.

In 2023, Awajan [24] proposed an IDS for IoT devices using deep learning. Five classes of intrusions have been addressed in the paper. The model based on deep neural network is highly dependent on considered dataset. However, the IDS model proposed in the present paper addresses 7 attack classes with one benign class. The average accuracy is 93.74% while in the present paper it is more than 96%. One more drawback of the paper is that it requires retraining for every new IoT network. However, the accuracy rate of the model was not so good.

In 2023, Sharma et al. [25] also proposed a network IDS for IoT attacks. This model was also designed using deep learning. Authors used filter-based feature selection which has been implanted by dropping highly correlated features. However, the accuracy achieved by this model is only 84%. By resolving data unbalancing issues, the accuracy achieved was 91%. However, the accuracy rate of the model was also not so good.

**CHAPTER 4**

**PROPOSED METHOD**

In propose paper author utilizing following modules to detect IOT intrusion

1. Data Collection: data will be collected from the server using WIRESHARK tool which will capture all packets data. we don’t have any server to capture WIRESHARK packets so we are utilizing IOT WIRESHARK dataset from KAGGLE repository which can be download from this URL ‘https://www.kaggle.com/datasets/malqarni/iotdataset’
2. Features Selection: all captured data will be input to ‘Maximal Information Coefficient’ (MIC) features selection algorithm to select features which are capable of yield high prediction accuracy
3. PCA (principal component analysis): PCA will be applied on selected features to reduce dimension and then select only those features which are relevant and reduce all irrelevant features
4. Features Normalization: will be applying MINMAX features normalization algorithm to normalize selected features
5. ML and DL (deep learning) algorithms: in propose paper author has employed various machine and deep learning algorithm such as Random Forest, SVM, LSTM and CNN and then evaluate each algorithm performance in terms of Accuracy, Precision, Recall and FSCORE. Among all algorithms CNN2D is giving best accuracy.

IOT dataset contains 75 training features and then contains two class labels such as 0 (benign) and 1 (malicious) and by using this dataset features will train and test each algorithm performance

In the present paper, the proposed intrusion detection approach deeply focuses on the time efficiency of related work. The present research work has been conducted as a part of a cybercrime investigation in an IoT environment. Reducing the run time can lead to an increase in the efficiency of the rest of the utilized resources. The comparative analysis of the models reviewed in this paper have been compared with the proposed model that can be seen in table 10 given at the bottom of this paper.

This section outlines the design, analysis, and implementation of the proposed model. Initially, all the project dependencies need to be downloaded and the required libraries need to be installed on the system. The entire research has been divided into the following phases:

1. **MODEL DESIGNING**

This section presents the design of an intrusion detection model that characterizes the normal (benign) and intrusive behavior of network traffic data obtained from real-time IoT sensors, IoT devices, and networks. This phase is composed of three sub-phases. Data used in this paper is openly available and can be extracted from Kaggle [4]. The algorithm of the entire intrusion detection modeling process is explained in Algorithm 1.

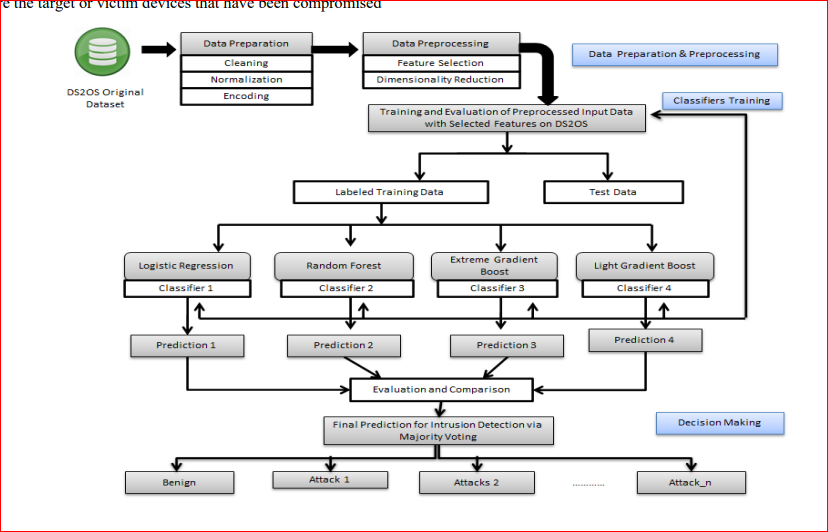
The present paper aims to build an intrusion detection model to identify attacks in IoT-enabled smart environments. This model has been divided into three phases: Data preparation and preprocessing, classifier training, and decision-making. The entire process is executed on a feasible dataset. In the present research work, the ‘DS2OS’ dataset has been taken for the implementation of the proposed work. This dataset contains the traces collected from different sensor-enabled smart devices which have been configured in a smart home environment. The detailed construction process of the proposed intrusion detection model is shown in figure 6.

**A.1. Data Collection (Dataset Exploration and Extraction)** The first phase is the designing phase which includes dataset exploration and extraction. This phase is followed by data preprocessing and preparation. Initially, during dataset exploration, appropriate datasets are considered and explored according to the interest of researchers. Many related datasets are publicly available, out of which some are either outdated or unfeasible for current research environments.

The required dataset needs to be made available on the system where the whole process is to be accomplished. Till now, many datasets have been generated for training and testing the data of IoT network traffic. The classifiers have been trained and tested using the ‘DS2OS’ dataset that contains network traffic data with normal and anomalous behavior [4]. Traces contained in the ‘DS2OS’ dataset have been captured from various smart devices installed in different locations in IoT home environments. The data has been captured using four simulators, designed for smart home environments with different types of services.

**DATASET DESCRIPTION**

‘DS2OS’ dataset was generated by capturing the traces in the IoT environment using four simulated IoT sites with certain types of services. This dataset is a fusion of malicious and benign (normal) traffic. ‘DS2OS’ dataset catalogs a new range of attacks generated in IoT environments from real-world traffic. The features of attacks are inspired by conventional network traffic attacks. The sets of captured network traffic samples are labeled based on their behavior.

****

**FIGURE 6: LGB-IDS (Ensemble-based) Intrusion Detection Model**

Feasible samples (features) are selected out of the total number of captured records and each identical sample record is assigned a class label (malicious or normal). Table I presents the fundamental details of the ‘DS2OS’ dataset. Here, two types of behaviors have been notified of connected nodes: ‘normal’ and ‘malicious. Nodes showing malicious behaviors are affected by various types of attacks. These attacks have been classified according to the properties of the set of records passed to the dataset. Some well-known attacks have been identified using the training process which have been labeled as ‘DoS attack’, ‘dataProbing’, ‘malitious Control’, ‘malitiousOperation’, ‘scan’, ‘spying’, ‘wrongSetUp’. Numbers of occurrences of nodes showing normal and malicious behavior are shown in Figure 7.

1. **DATA PREPARATION**

In the second phase, the raw dataset is transformed into a suitable format and prepared for preprocessing and analysis. This dataset is largely imbalanced that can be prepared using data cleaning.

**C.1. Data Cleaning and Transformation** Initially, the available dataset might not be in an appropriate format. Data cleaning and transformation can make it more constructive. It can be expensive, time-consuming, and tedious if there are incomplete, superfluous, missing, noisy, and repeated data entries in a dataset. Such entries are identified and can either be removed or can be replaced with new values. There are many ways to fill and replace these values.

Inconsistent data values are transformed using some transformation techniques. Transformation can handle missing data, categorical data, skewed data, and data in the form of string or non-alphanumeric. Label-encoder and onehot encoder are the two most useful categorical encoding techniques. Filling the null or missing values and cleaning noisy data are some important operations of data cleaning.

• Filling the Null or missing values: Null or missing values could be replaced either with ‘zero’ or the most suitable values in order to prevent any error. In figure 8, line no. 53 shows the replacement of irrelevant values with most relevant values in different places in the dataset. In line 5 data.value.astype (float) method converts specific numeric data values to float type.

• Data cleaning: Noisy data can be removed or replaced by relevant and suitably fitted data. The replacement of noisy data can be selected from the data existing in the dataset as per suitability. In figure 8, in line no. 6, data.drop() function has been used to remove the unnecessary columns from the dataset. Some examples of the implementation of data preparation techniques using python code are shown in figure 8. It shows how a missing or noisy value can be repaired or replaced and the irrelevant feature can be removed explicitly.

1. **DATA PREPROCESSING**

Data preprocessing is the most necessary phase to prepare a raw dataset for experiments. It is an essential phase for any project to improve the performance [40]. Generally, adequate time and effort are spent during data preprocessing. This process is also essential for ensuring a fitted labeled dataset that is to be created for reducing the complexity and ensuring the premium quality results of analysis [41]. Models developed using ML techniques can produce highly accurate and quick results. During the data preprocessing phase, the following operations are performed.

**i) D.1. Feature Engineering:** High-dimensional datasets can increase time and space complexity. In order to overcome the complexity issues, the feature selection approach is highly recommended. Feature engineering is a process to create a new set of features based on current features according to the requirements of the project [42]. The main activities of feature engineering are: feature extraction, feature scaling, and feature relationship capturing. It becomes too time-consuming to process the whole collected data.

Therefore, the most significant features or attributes are selected from the total number of features of a dataset to perform analysis. Features, which are not useful for the model, can be avoided during processing [43]. Entries containing irrelevant or null values can be dropped. This phase reduces the complexity of the analysis. Most promising features are selected dynamically from the entire dataset. Dimensionality Reduction is significant in ML and predictive modeling for data compression in order to reduce storage space and computation time. It refers to the process of reducing the number of features by obtaining a set of principal instances.

• Feature selection: The main principle of feature selection is to select the most important features (attributes) from the dataset and remove the least important features on which the performance of the model doesn’t depend. The performance of the model can be suffered if irrelevant or less useful features are selected. Best feature selection may lead to improved model accuracy (improves the result by reducing misleading data), reduced overfitting (less redundancy also reduces noise), and reduced training time (reduced algorithm complexity). There are several methods to perform feature selection which are: Univariate selection, feature importance, and correlation matrix with a heat map.

• Feature scaling: There are a total of 13 features, out of which 12 have been used for operations with target variables. Unexpected values of variables (such as ‘none’, ‘true’, ‘false’, and string type values) need to be replaced with the most suitable float values. The ‘OneHotEncoder’ is applied to transform the categorical column of ‘X’ and LabelEncoding using the ‘labelEncoder()’ method has been applied to assign an integer value of ‘Y’. The ‘iloc()’ function of the Pandas module enables it to handle the entire column of the dataset. It enables selecting values belonging to a specific row and column of the dataset. There are 8 classes where 7 classes are types of attacks and one is benign or normal class. Numbers of subparameters need to be tuned for optimal classification.

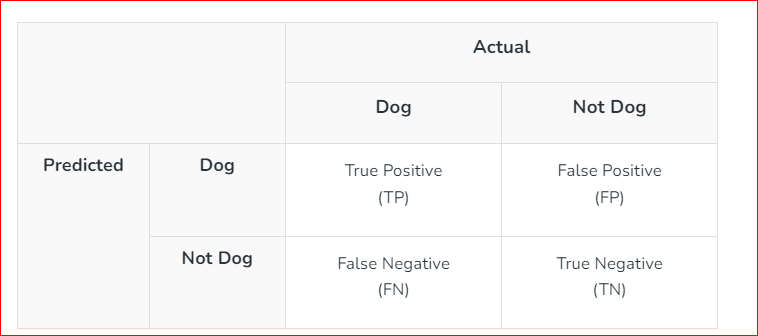
**CONFUSION MATRIX:**

A **confusion matrix** is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model’s predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

The matrix displays the number of instances produced by the model on the test data.

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

When assessing a classification model’s performance, a confusion matrix is essential. It offers a thorough analysis of true positive, true negative, false positive, and false negative predictions, facilitating a more profound comprehension of a model’s **recall, accuracy, precision,**and overall effectivenessin class distinction. When there is an uneven class distribution in a dataset, this matrix is especially helpful in evaluating a model’s performance beyond basic accuracy metrics.



In LGBM, the tree grows leaf-wise, following the bestfirst technique. The leaf is chosen with maximum delta loss to grow. Leaf-wise tree growth is not suitable for the small numbers of Evaluating the performance of a machine learning model is crucial for understanding its effectiveness and reliability. Here, we will discuss four key performance metrics: Accuracy, Precision, Recall, and F1 Score, along with their respective formulas.

We look into the machine learning model's accuracy. We employ a confusion matrix since accuracy and appropriate classification are both required. An NxN matrix called a confusion matrix is used to assess how well the machine learning model handles the categorization problem. There are four variations of the confusion matrix True positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP).

We talk about the accuracy of the machine learning method for categorization using the confusion matrix.

* TP is the number of true positives
* TN is the number of true negative
* FP is the number of false positives
* FN is the number of false negatives

**•Accuracy:** In the context of a Confusion Matrix for evaluating the classification of early-stage detection of psoriasis using deep learning, accuracy is a measure of overall correctness. Using confusion matrix accuracy is calculated as follows,

**Accuracy =**

is calculated as the total number of right predictions divided by the total number of valid predictions in the dataset. The highest level of accuracy is 1.0, while the lowest level is 0.0.

**• Precision:** A metric of the model's optimistic forecast accuracy is called precision. A high precision number means that the model is frequently accurate in predicting the first stages of psoriasis. The ratio of TP classes to all FP classes plus TP classes is known as precision.

**Precision =**

* True Positives (TP) are instances where the model correctly identifies early-stage psoriasis.
* False Positives (FP) are instances where the model incorrectly predicts psoriasis when it is not present.

or Confidence (as called in Data Mining) shows how many Predicted Positive situations correspond to correct Real Positives.

**• Recall:** It also known as sensitivity or actual positive rate is a metric in the context of a Confusion Matrix. The recall is significant in psoriasis, where missing positive (false negatives) cases can have serious consequences. A high recall value indicates that the model effectively captures a substantial portion of the positive cases, minimizing the instances where psoriasis is present but goes undetected.

**Recall =**

* TP are instances where the model correctly identifies early-stage psoriasis.
* FN is instances where the model fails to identify early-stage psoriasis when it is present.

(Sensitivity) is the ratio of correct anomalous measurement detections to the total number of abnormal measures.

**• F1- Score:** A binary classification model's performance is determined using a metric called the F1-score, which is often referred to as the F1 measure or F1-value. This statistic achieves a compromise between Recall and precision. Precision and Recall are balanced by the F1-Score metric. It is beneficial when datasets are unbalanced. The Confusion Matrix values are used to calculate the F1-Score, which thoroughly evaluates a model's performance. It has a 0–1 range, denoting flawless Recall and precision. It is beneficial when considering both false positives and negatives, which is necessary.

***F-score =***

**CONSTRUSTION PROCEDURE OF PROPOSED INTRUSION DETECTION MODEL** After preprocessing and classification of samples, the considered dataset needs to be trained and tested with a selected classifier. The dataset is split into a training set and a test set to calculate the accuracy and error scores of the proposed model. Features or independent variables ‘X’ and ‘target variable ‘y’ need to be initialized.

The LGBM classifier has been implemented using the ‘LGBMClassifier()’ method which is an ensemble learning method shown in figure 9. The usage of some important parameters of the LGBM classifier is given below:

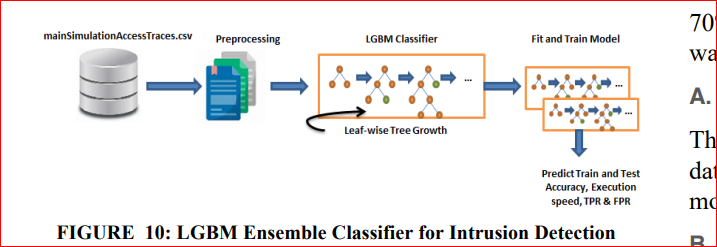
1. boosting\_type: The value ‘gbdt’ i.e. Gradient Boosting Decision Tree. This is used for exclusive feature bundling. This feature overcomes the limitation of the histogram-based approach.

2. max\_depth: A value is assigned to this parameter to limit the depth of the tree. This hyperparameter is effective to control the over fitting. In present scenario, the value is set to 1.

3. n\_estimators: The number of estimators can be varied to optimize the results. Here, the value of n\_estimators is assigned as 400.

4. num\_leaves: It refers to the number of leaves to construct a tree. The value of num\_leaves should be less than the square of max\_depth.

5. learning\_rate: The value of learning\_rate can be in the range of 0.1 to 1.0. In the current scenario, 0.1 gives the best result. 6. max\_depth: Its value should be set to avoid overfitting. The method of the LGBM classifier with necessary hyperparameters that need to be tuned with relevant values in order to obtain optimal results has been given in Figure 8. The classifier predicts the results of the test set. The model is fit and the X and Y data of the ‘DS2OS’ or ‘mainSimulationAccessTraces’ dataset is trained with the LGBM classifier. With default hyperparameters, the LGBMbased intrusion detection model achieves 93.4 % classification accuracy. The LGB-IDS model is an ensemble-based intrusion detection system constructed using an LGBM ensemble classifier with leaf-wise tree expansion is illustrated in Figure 10.

****

**FIGURE: LGBM Ensemble Classifier for Intrusion Detection Model**

The parameter ‘max-depth’ is used to limit the depth of the tree. However, the tree grows leaf-wise; still ‘max-depth’ parameter is specified to control the tree depth. Here, train and test accuracy, run-time, TPR, and FPR are calculated for final prediction. LGBM optimizes the prediction model in the following ways:

• Speed and memory optimization: LGBM uses histogrambased algorithms, which store continuous feature values in discrete bins. If the numbers of bins are small, a small data type can be used for storing the training data. This approach speeds up the training time and reduces memory usage.

• Reduced gain computation cost of every split: Other bagging and boosting algorithms including the histogram have time complexity problems because most of these are pre-sorted. LGBM can sum up the operations faster. Histogram has O(#data) time complexity. But once the histogram is built, the time complexity becomes O(#bins). (#bins) are much smaller than (#data) that makes the reduction in computing costs.

• Optimized accuracy: Due to fixed #leaf, the leaf-wise tree provides optimized accuracy as compared to level-wise.

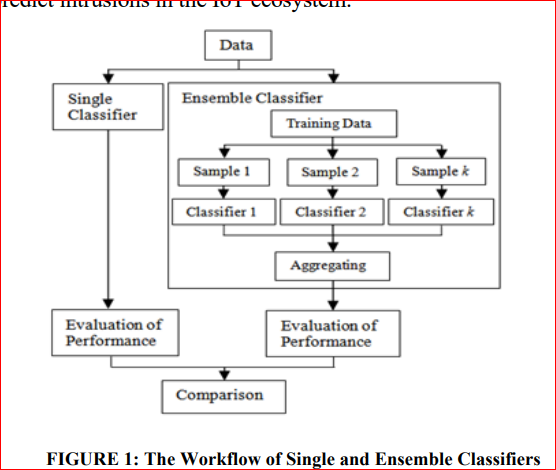
• Optimal categorical feature split: One-hot encoding is used commonly to represent categorical features. But this approach may lead to unbalancing and the tree needs to grow very deep for achieving good accuracy. Instead of one-hot encoding, the better alternative is to perform a split by partitioning features according to their categories into two subsets. LGBM sorts the histogram according to its categorical features and stored values.

• Network optimization: LGBM uses collective communication algorithms instead of point-to-point communication that reduces the scatter

**CLASSIFICATION ALGORITHMS**

There exist many decision-tree-based classification and predictive machine learning algorithms which can be classified as single and ensemble classifiers. In the present paper, a simulation will be performed on a single class classifier and three linearly separable ensemble classifiers. Logistic regression (LR) [26] is a type of single classifier and Random Forest (RF), gradient boosting (GB), XGboost (XGB), and light gradient boosting machine (LGBM) are ensemble classifiers.

An ensemble learning classifier refers to a classification technique that combines multiple base models and provides results as a single optimum classification model. Utami et al. [27] performed a systematic comparison of single and ensemble classifiers in order to characterize their performance. Sub-classifiers are derived from randomly selected samples (Sub-features) which are further aggregated in order to make better decisions. Figure 1 illustrates the workflow of single and ensemble classifiers. In this paper, the performance of various ML classifiers has been analyzed to predict intrusions in the IoT ecosystem.

****

**FIGURE 1: The Workflow of Single and Ensemble Classifiers [27]**

1. **Single Classifier** This section briefly discusses a single classifier in which decision-making is conducted on a single set of features. The parameters of the single classifier-based resulting model are estimated using linear optimization. The performance of the single classifier can be evaluated using statistical algorithms.

**A.1. Logistic Regression** Logistic regression (LR) is also called the sigmoid function. It is a statistical function that was initially developed to determine the properties of the growth of the population in an environment. It is also called as logit() function. It is a supervised ML classification algorithm that is commonly used to calculate the probability of a malicious event. It is used to predict the categorical or discrete value. By default, this is called binary logistic regression. Another type of logistic regression is multinomial logistic regression, which is used for solving the problems of multi-class classification [28]. However, multinomial logistic regression is an extension of traditional logistic regression [26], but this method can handle more than two feasible discrete outcomes. LR is a precise model that anticipates the quality of the dataset. Suppose X is a given set of distinct features. Using logistic regression, the probability estimation can be done using the mathematical formula given in equation 1.

1. **Ensemble Classifier**

Ensemble classifiers are expected to provide better prediction outcomes as compared to single classifiers like logistic regression. Rather than depending on a single decision tree, ensemble classifiers perform predictions by aggregating the outcomes of multiple contributing trees which are constructed using given data points. In the ensemble learning approach, the prediction depends upon the combination of important features of two or more contributing models. The ensemble learning approach attempts to extract the harmonizing information from its contributing models. There are mainly two types of ensemble methods that are bagging and boosting [29].

Bagging algorithms reduce the variance while boosting algorithms reduce the bias. Using bagging methods, all the models are constructed simultaneously while using boosting methods once the first model is built then only the second model is built after knowing the errors of the first model. The aim of an ensemble model is to reduce errors. The proposed model in the present paper performs the classification and prediction using different machine learning ensemble algorithms which will be further analyzed on the traffic patterns collected using a simulator based on different sensorenabled devices used in the IoT home environment. The model has been designed on the basis of optimal outcomes obtained from different classification algorithms which have been evaluated on the ‘DS2OS’ training dataset. The network has been classified on the basis of behavioral patterns of network traffic.

**B.1. Random Forest** Random Forest (RF) is a bagging ensemble technique of ML in which classification relies on multiple decision trees which collectively construct a forest. RF contains multiple decision trees constructed using different subsets of a specified dataset in order to improve predictive accuracy. Instead of depending on a single decision tree, RF captures the outcomes of prediction of every tree. The highest accuracy depends on the largest number of trees which means the accuracy increases with the increase in the number of trees. Unlike many other models of machine learning, it can test and train on the network traffic of real-time datasets. The following conditions need to be satisfied before using this process:

• There must be actual values in the dataset against features to obtain accurate results instead of guessed results.

• There must be a low correlation among the predictions of each tree. The main properties of random forest are independent fast learning over the datasets of distinct nature. It is an ensemble classifier that is composed of multiple decision trees which are generated using two distinct randomization sources [30]. At the split of every node, a randomly selected subset is chosen as the input variable to find the best split. But due to multiple trees, the complexity of the RF may become very high. It may cause more power utilization. Random Forest Algorithm:

1. Select the K number of random data points.

2. Construct a decision tree using the chosen data points.

3. Recognize the behavior of a tree and give it a class label.

4. Repeat steps 1, 2 & 3 until n number of trees.

5. Find the prediction of every decision tree and the winner decided using the ‘majority voting’ is considered as the final output class.

The basic structure of a random forest classifier with a majority voting’ concept is shown in Figure 2. This structure simulates the process to predict the behavior of network traffic. Consider a given dataset ‘D’ that contains the traffic logs which are stored in the ‘.CSV’ file. The individual record in the dataset can be considered as a data point or a subset using which multiple decision trees have been constructed (which collectively make a structure like a forest). The arrows and green nodes present the direction of the level-wise tree growth. Different decision trees show different types of output depending on their network traffic behavior.

Random forest operates two types of decision trees. Decision trees having malicious traffic nodes are labeled as the ‘anomaly class’, while others are labeled as the ‘normal class’. Applying the majority voting operation to decision trees, the best result can be obtained. Suppose, a random forest is operating an ‘n’ number of decision trees, where the value of n >= 2. There are n-1 decision trees with anomalous behavior and one class with normal behavior. Hence, the majority of voting goes to the anomaly class. In the present scenario, the authors are working with multiple decision trees. There may be a possibility of less over-fitting in prediction through voting.

The random forest can efficiently run with higher accuracy on larger datasets, with high dimensionality. But it may not return good results for small and low-dimensional datasets. Random forest classifier works like a black box that cannot be fully controlled by users as well as it does not provide complete visibility. Its computations may also become far more complex; hence it is not easily interpretable. Training time complexity of RF is O(n\*log(n)\*d\*k), where k is no. of decision trees. Run-time complexity is O(depth of tree\*k).

**B.2. Gradient Boosting** Gradient Boosting (GB) is also a widely used decision tree classifier algorithm that is also known as Gradient Boosting Decision Tree (GBDT). Boosting is a technique to integrate multiple weak learners (base classifiers) to construct a strong learner using certain machine learning algorithms [31]. GB classifiers are used to analyze the abnormal behavior of devices in different technological scenarios [26]. It is supported by strong hypothesis results which describe how the powerful predictors can be constructed by integrating multiple base models or through a greedy approach that correlates to gradient descent in a function space. GB-based feature selection enhances the detection rate as well as execution speed [32]. In the GB approach, the classification and prediction are performed on the basis of residuals obtained from previous iterations. The performance can be improved by reducing the over-fitting. The model is computed against classification using the residuals obtained from previous iterations.

**B.3. Extreme Gradient Boosting** Extreme gradient boosting (XGB) is an advanced implementation of the gradient boosting (GB) that is created for improving performance such as better accuracy and reduced false alarm rate [33]. XGB helps in designing a stronger classification model that enables the classification of the data more accurately during entering into a network. XGB is one of the best promising boosting ensemble approaches which come with competitive outcomes. XGB can effectively deal with over-fitting challenges when the system is flooded with data. In this case, the classifier must be faster to adapt to such a large number of data entries.

XGB is empowered by tuning the maximum number of hyper-parameters regularization. It is a decision-tree-based gradient-boosting algorithm of ML that can upgrade the efficiency, accuracy, and attainability of ensemble-based IDS by tuning the hyperparameters. It can steadily handle the bias-variance trade-offs. High memory usage and slow running speed is the main drawback of this model. Overfitting in boosting methods can be reduced by carefully tuning the hyper-parameters [34].

**B.4. Light Gradient Boosting Machine (LGBM)** Light Gradient Boosting is a histogram-based decision tree algorithm that improves the efficiency of the model as well as reduces the execution time and memory usage of a machine. LGBM is greatly optimized over other boosting ensemble decision tree algorithms [35]. It is a faster, more distributed, more powerful, and highly improved learning algorithm. LGBM can handle large data traffic efficiently [36]. Like many other boosting algorithms, LGBM uses a Histogrambased algorithm and a pre-sorted algorithm for decision tree learning and computing the superlative split [37].

LGBM makes use of two new mechanisms which are Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS down-samples the instances, based on the size of the gradients to segregate the data samples for locating a split value. Small gradient samples are discarded and large gradient samples are targeted for the model. Samples with small gradients are well-trained and samples with large gradients are undertrained. This method provides more accuracy as compared to uniform random sampling. On the other hand, EFB overcomes the limitation of traditional histogram-based algorithms. The LGBM algorithm expands leaf-wise (best-first) while other decision trees grow level

**B.4.1. LGBM Methods** Here, an analysis of LGBM is obtained using variance gain at splitting features with the help of GOSS and EFB techniques. It provides the following advantages:

• It outperforms in terms of time-efficiency i.e., much better that many other classification algorithms.

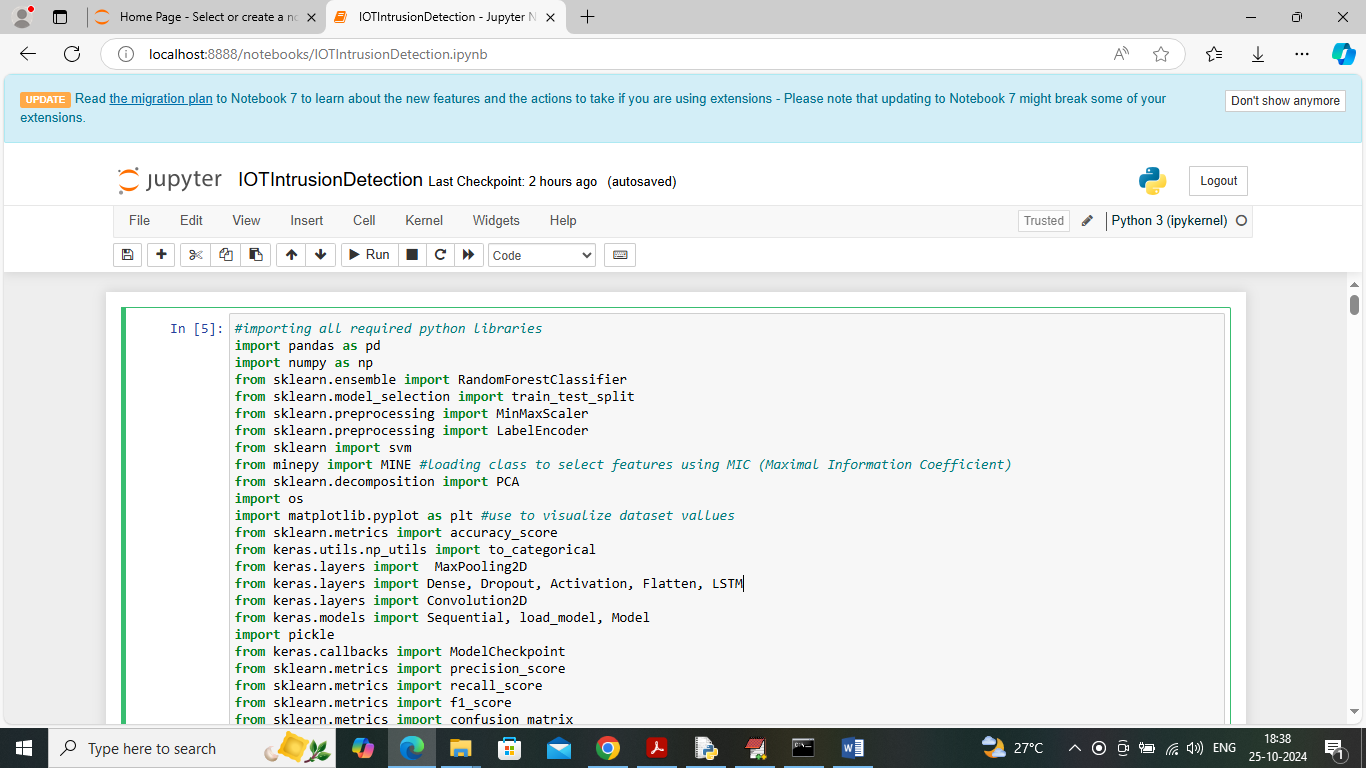
• It provides good training and test accuracy.

• Besides classification, it can also be used for regression.

• The overfitting can be reduced by setting a suitable value for the ‘max\_depth’ hyperparameter. Gradient-based One-Side Sampling (GOSS) for LGBM The GOSS method is developed by modifying the Gradient boosting method that focuses on those training samples which produce a larger gradient. The gradients used in this method speed up the learning by reducing the computational complexity of the learning method. A significant proportion of the samples with smaller gradients is excluded and the samples with larger gradients are used for information gain. GOSS provides better accurate estimation with smaller data sizes.

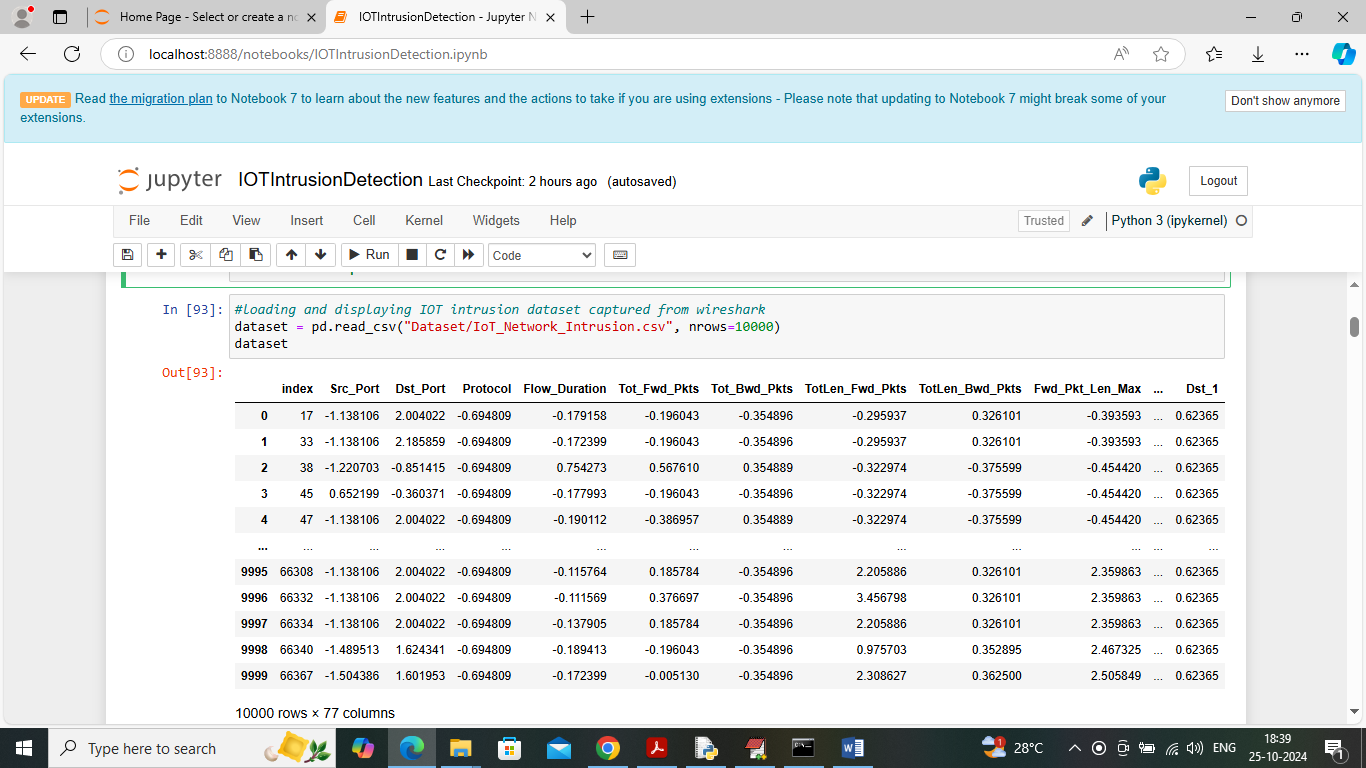
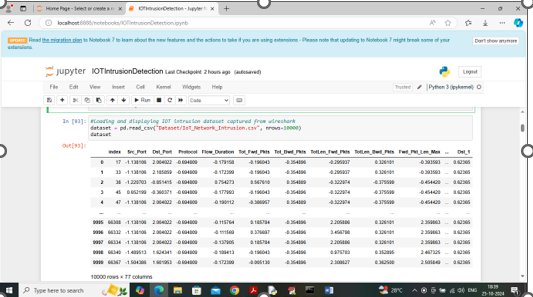
**Exclusive Feature Bundling (EFB)** for LGBM EFB is a mechanism used for bundling sparse mutually exclusive features. High-dimensionality data is generally sparse data that contains the probability to design an approximately lossless mechanism of feature reduction. A sparse space contains numerous features with mutual exclusion. They do not accept nonzero values concurrently. The exclusive features are selected and put into a bundle that is called exclusive bundling. It is a sort of automatic feature selection. The complexity of the histogram also changes from O (#data \* #feature) to O(#data \* #bundle). It improves the framework training speed without reducing the accuracy [37]. Ordinary decision trees work with discrete data. If the given data is in continuous form, that needs to be transformed into a discrete form to fit into a decision tree. The main challenge is to identify the optimal splitting points for selecting features. A small number of split points may lead to a loss of information but can also reduce overfitting. In contrast, a large number of splitting points may be lossless but may increase the training time. This algorithm provides a solution in which the data points are iterated many times until the optimal split point is accessed. It provides more information gain with reduced variance. The LGBM approach divides the data into a fixed number of split points of uniform length.

**CHAPTER 5**

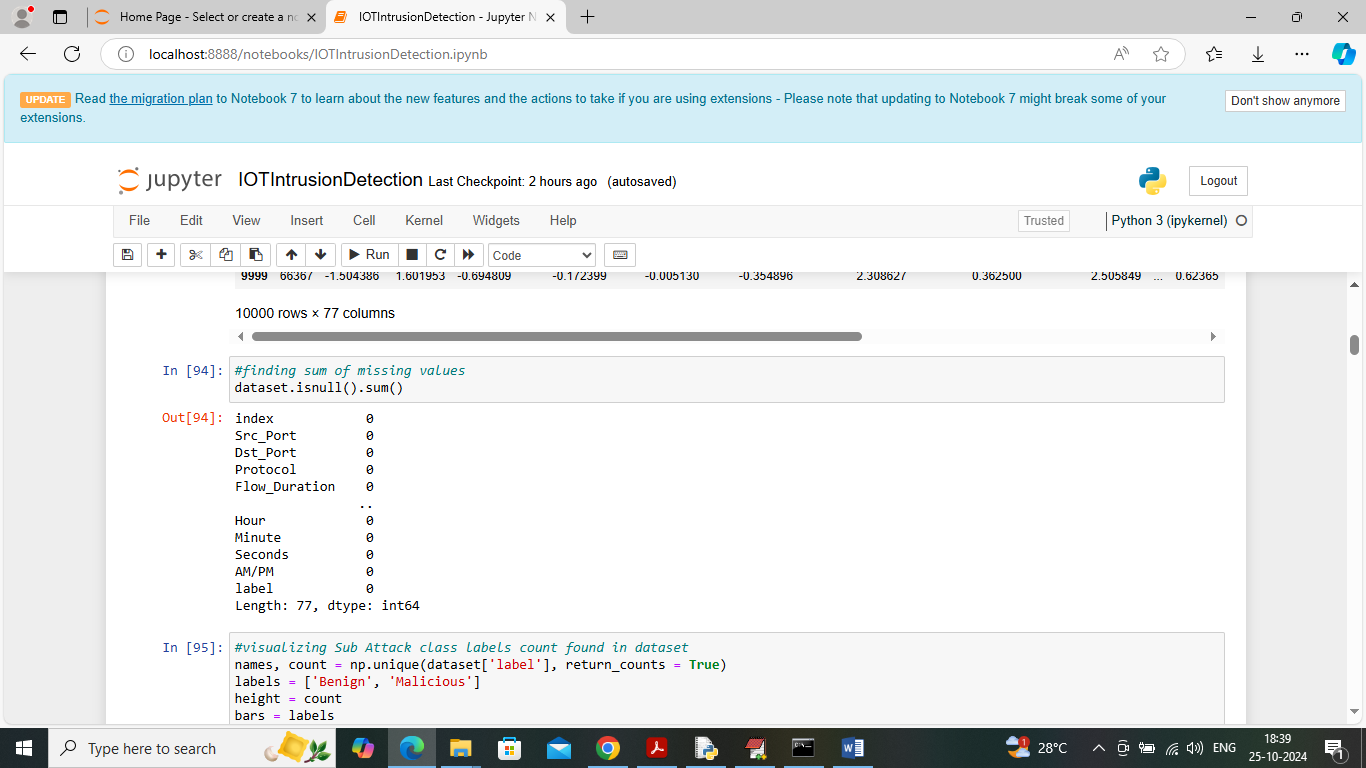
**RESULT**

We have coded this project using JUPYTER notebook to show processing of each algorithm describe above and below are the code and output of each algorithm screens

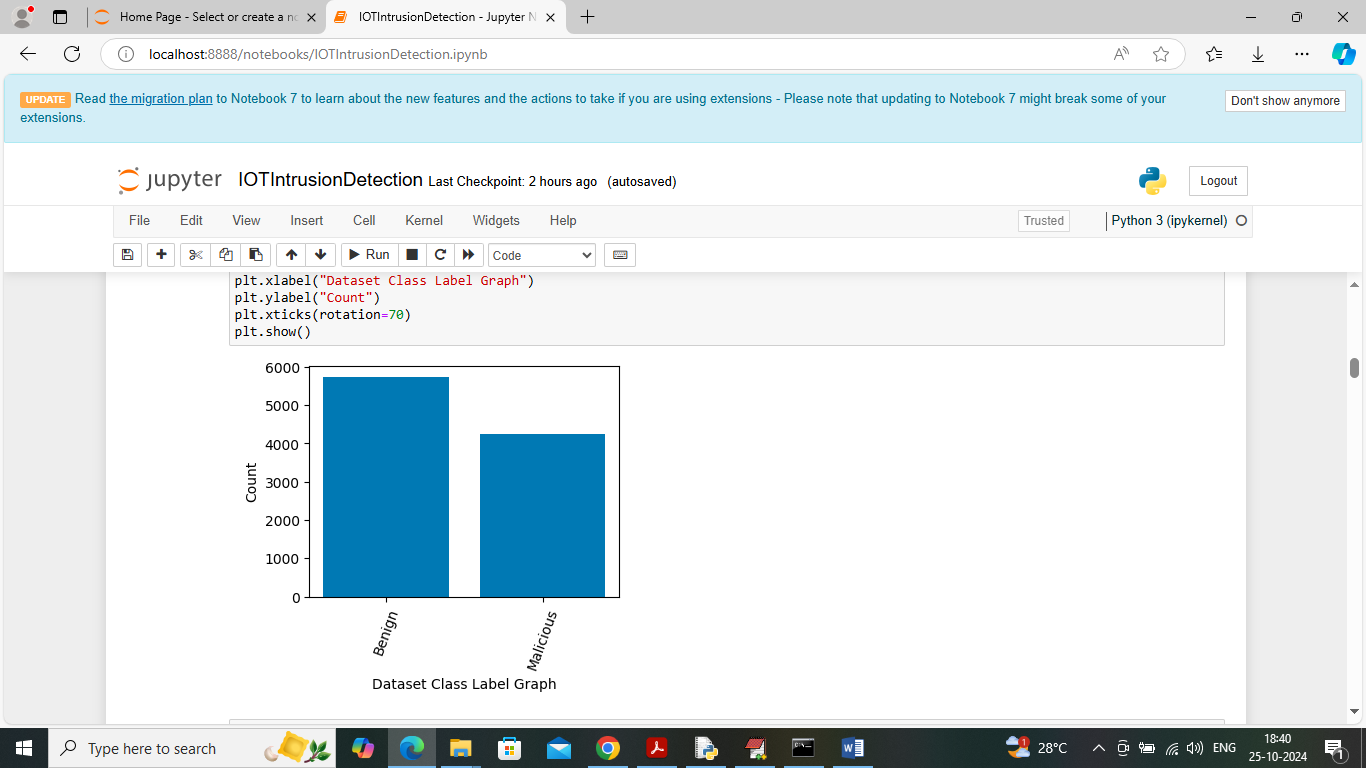
In above screen loading all required python classes and packages



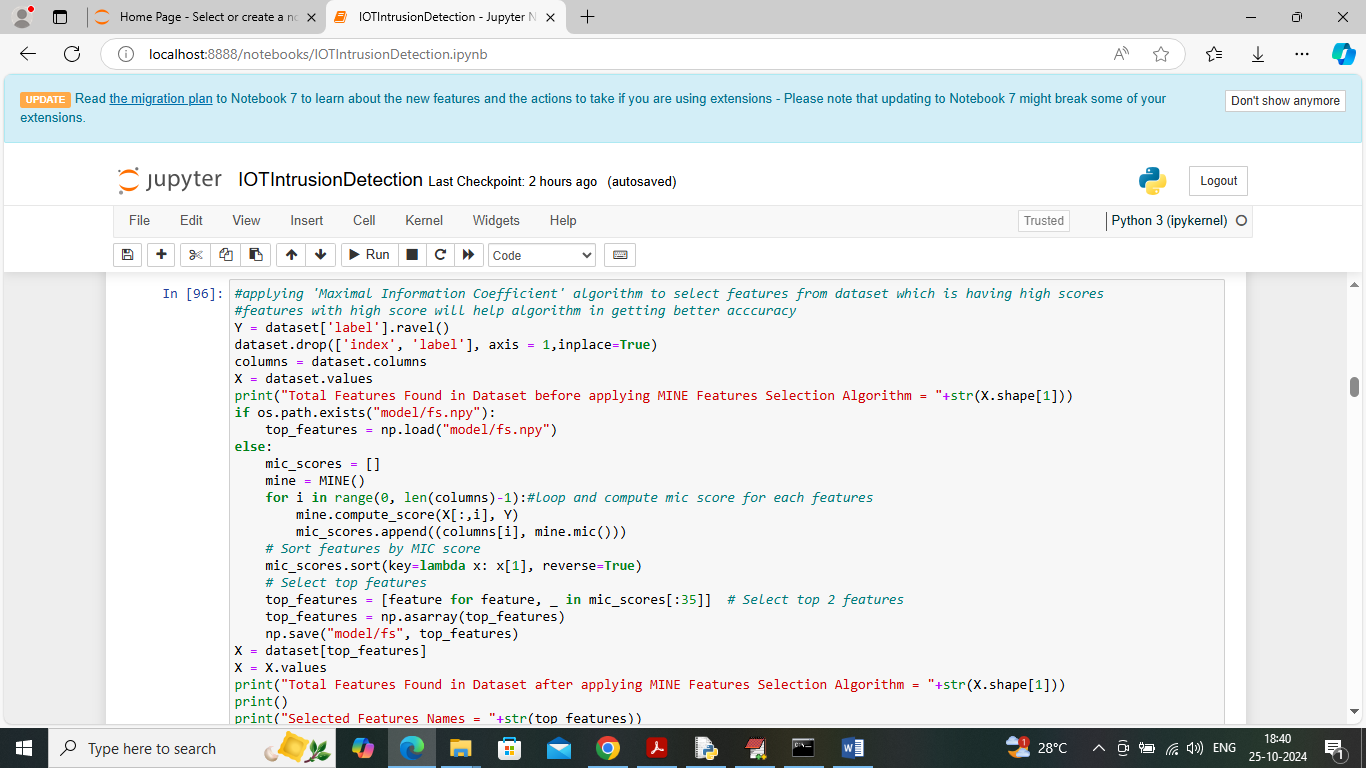
In above screen loading and displaying IOT intrusion dataset



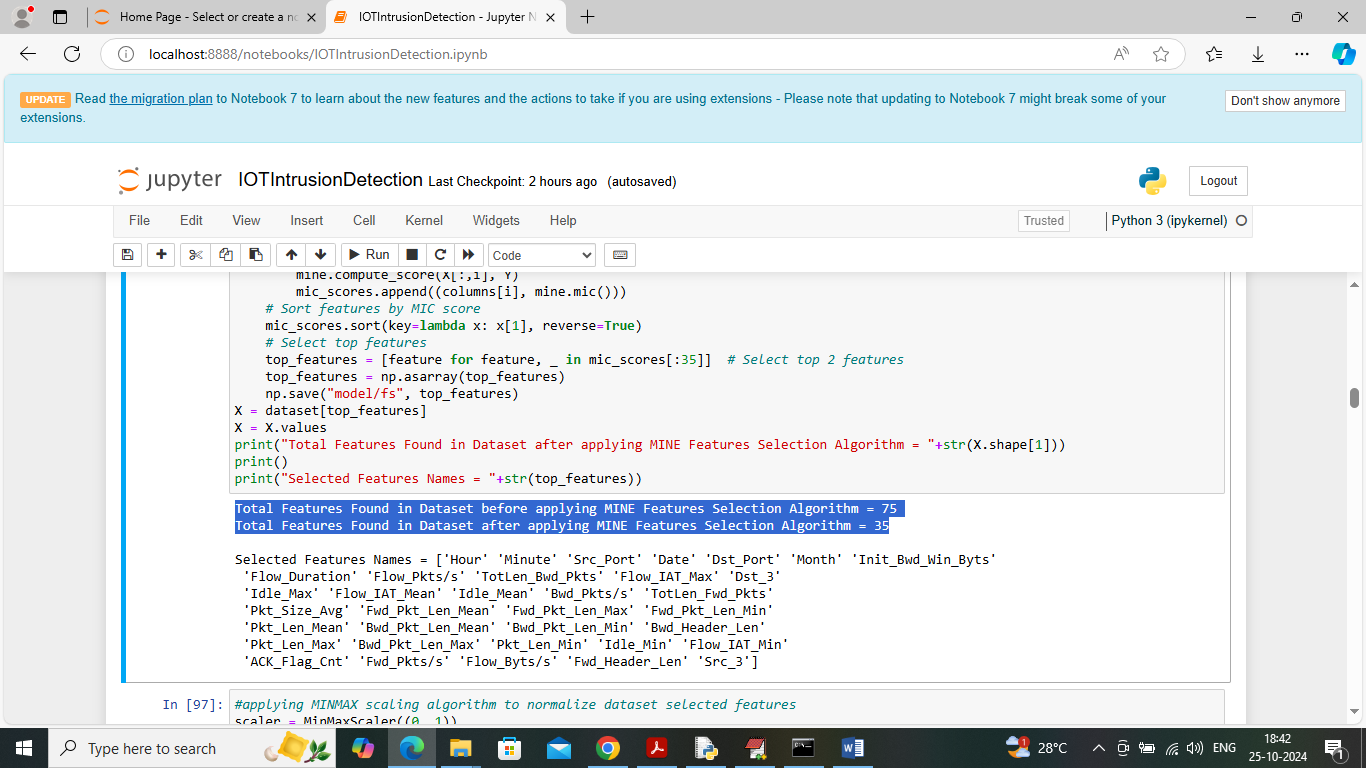
In above screen displaying count of missing values for each features available in dataset



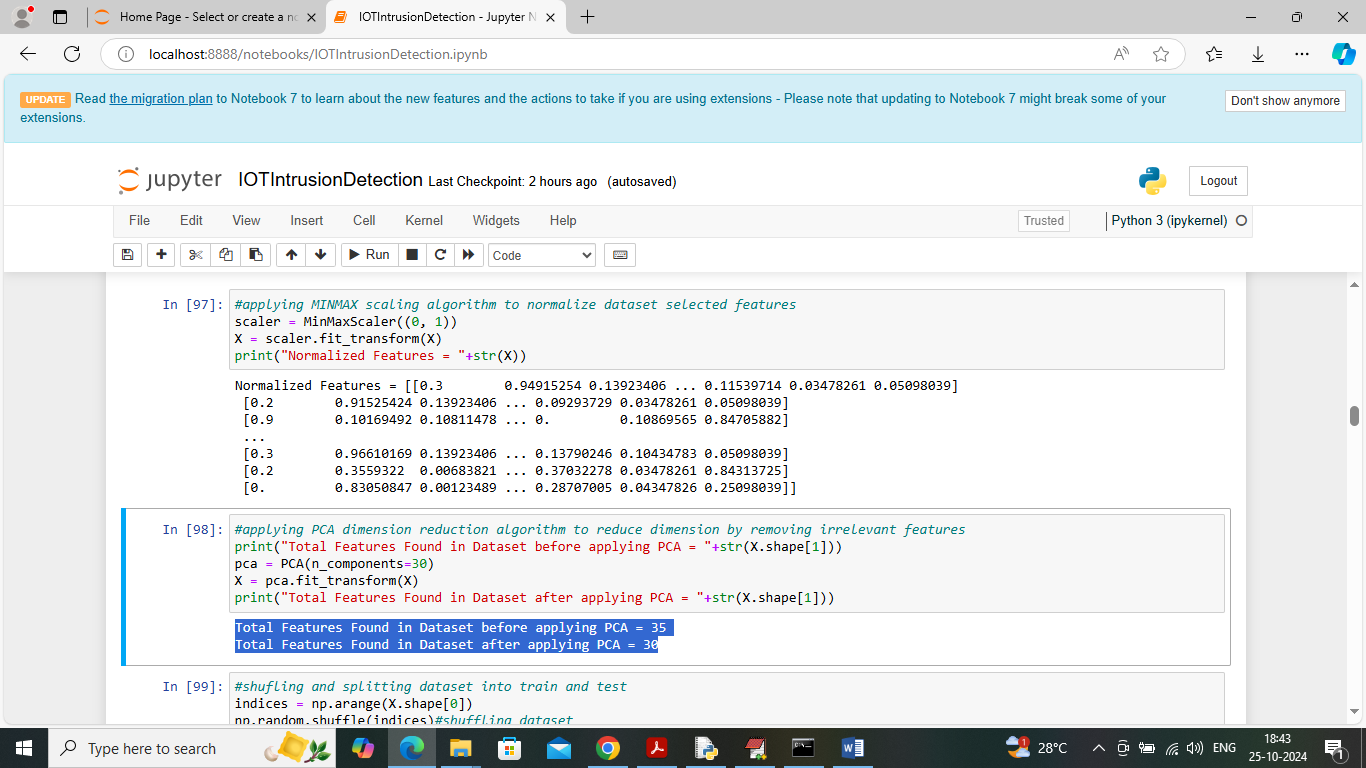
In above screen visualizing graph of different class labels found in dataset where x-axis represents class label and y-axis represents number of records found in that class category



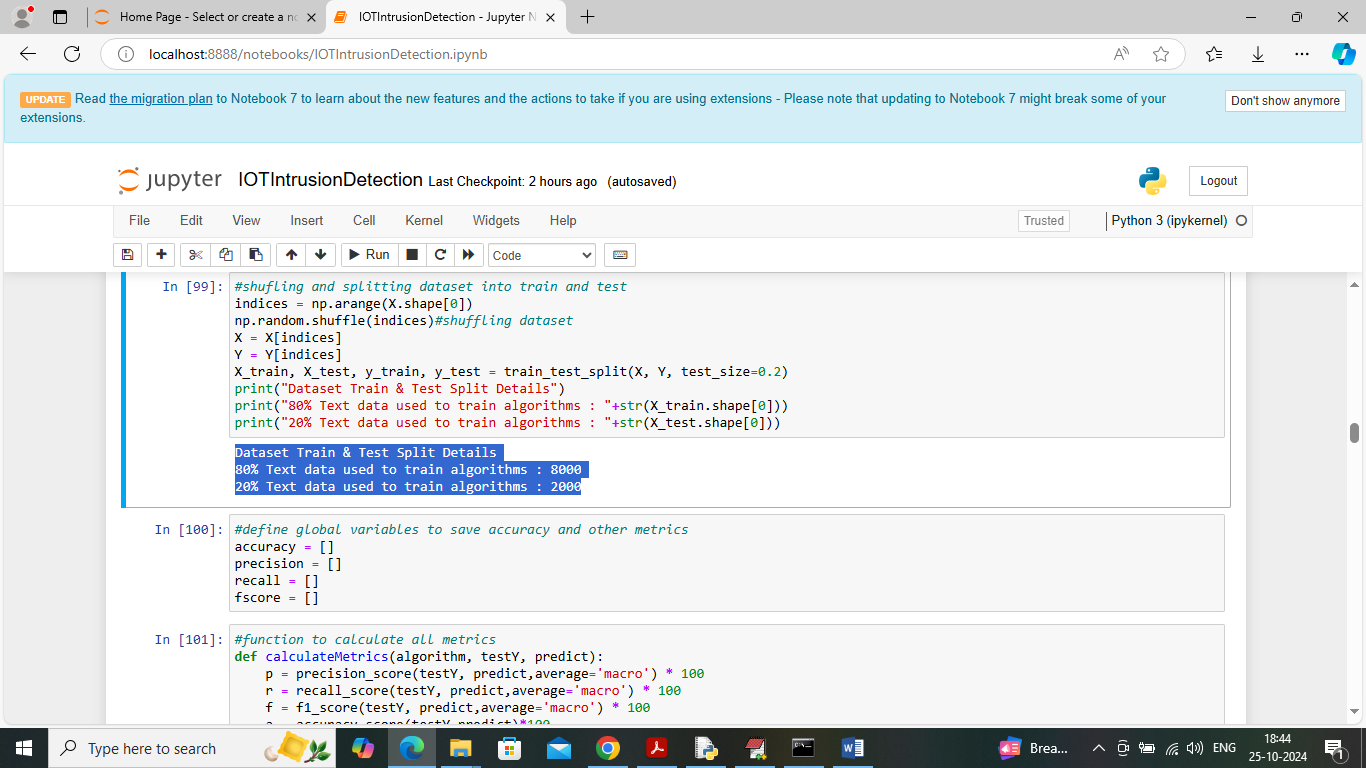
In above screen applying MIC features selection algorithm to select features which are capable of classifying test data as benign or malicious with high accuracy.



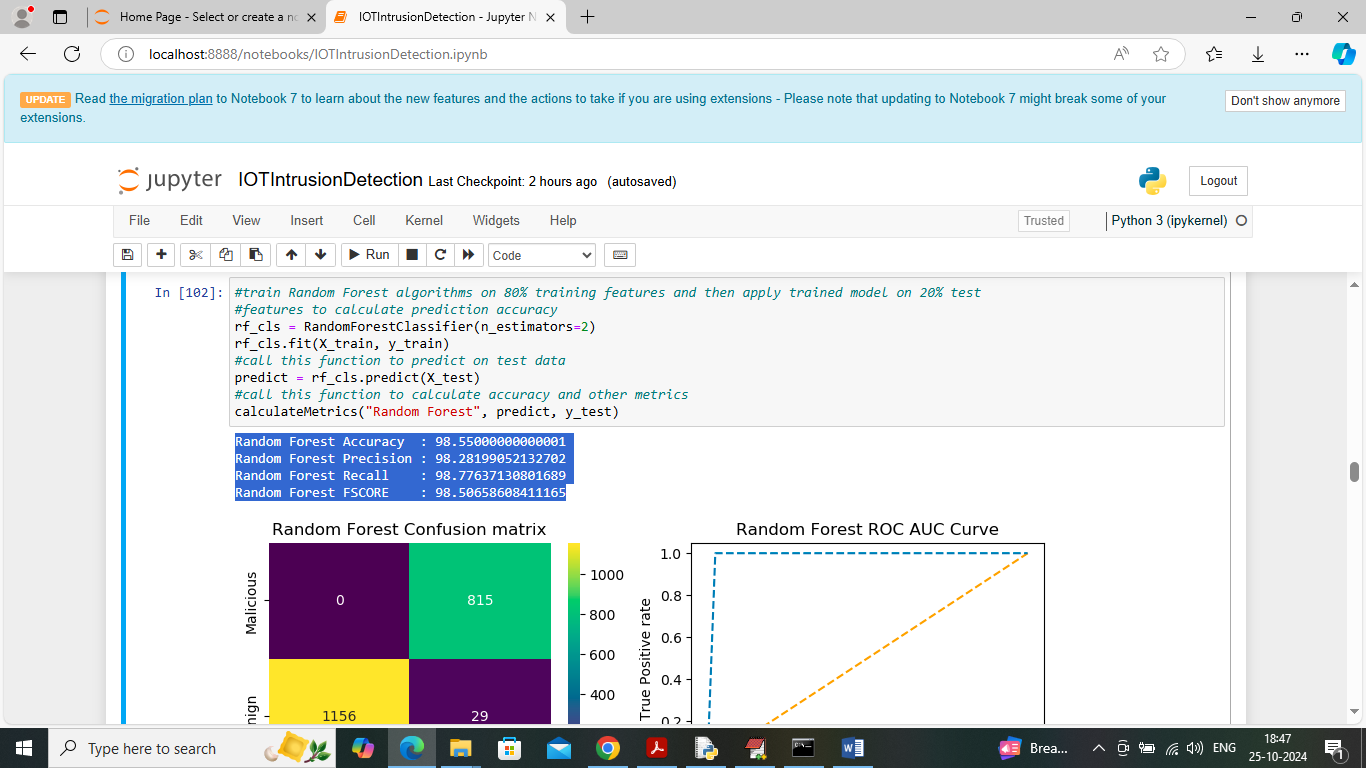
In above screen in first blue line can see dataset having total 75 features and then MIC features selection algorithm selected 35 features out of 75 and then displaying Names of selected features columns from dataset



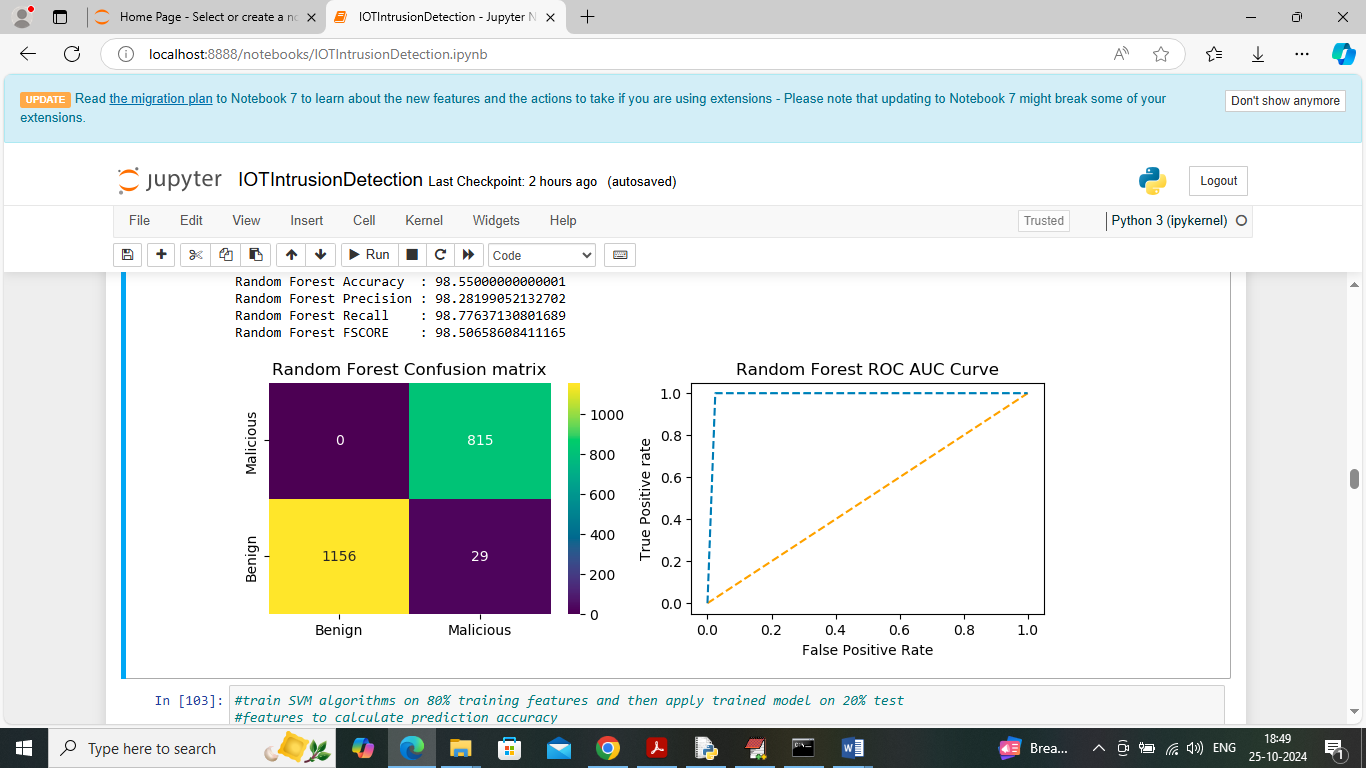
In above screen in first block applying MINMAX normalization algorithm to normalize dataset features and then in second block applying PCA algorithm to reduce features size and before applying PCA dataset where having 35 features and after reduction we got 30 features



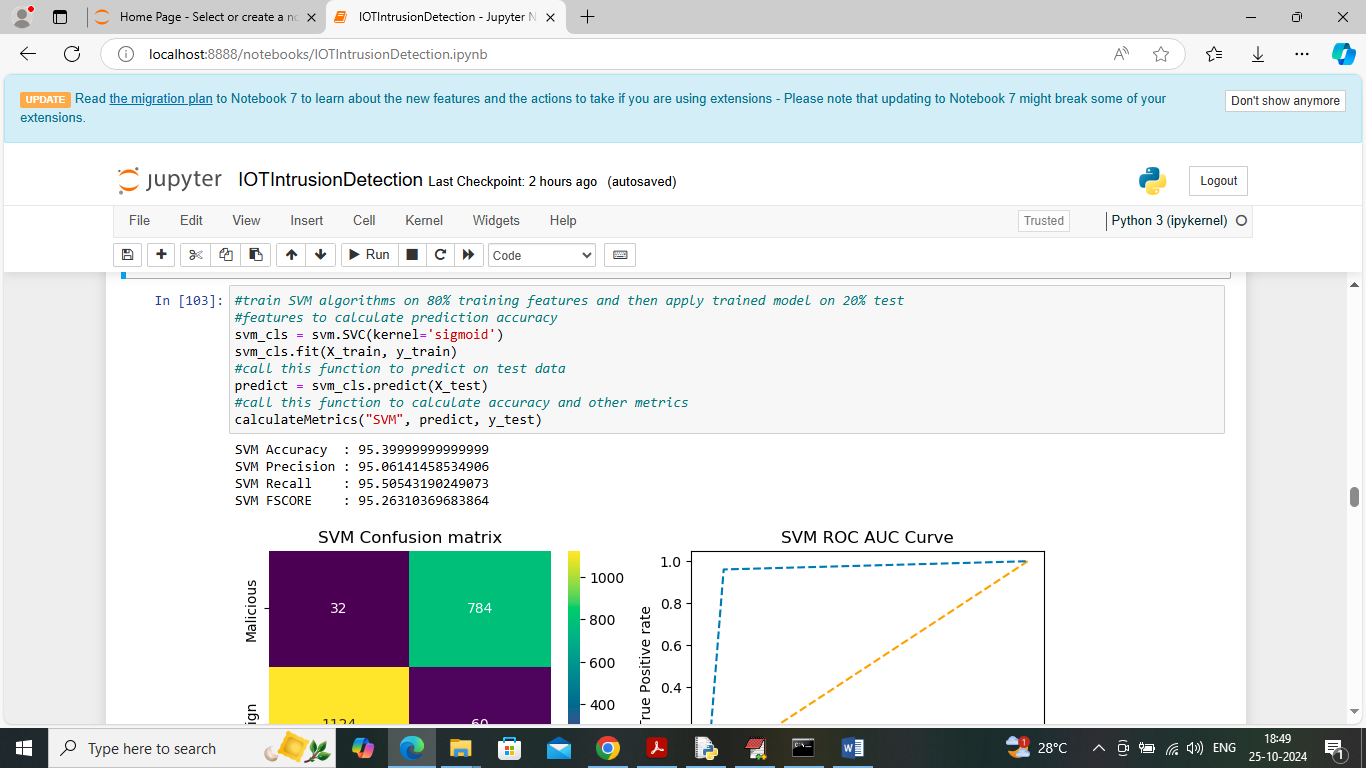
In above screen applying features processing techniques such as shuffling and then splitting dataset into train and test where application using 80% dataset size for training and 20% for testing and then defining function to calculate accuracy and other metrics



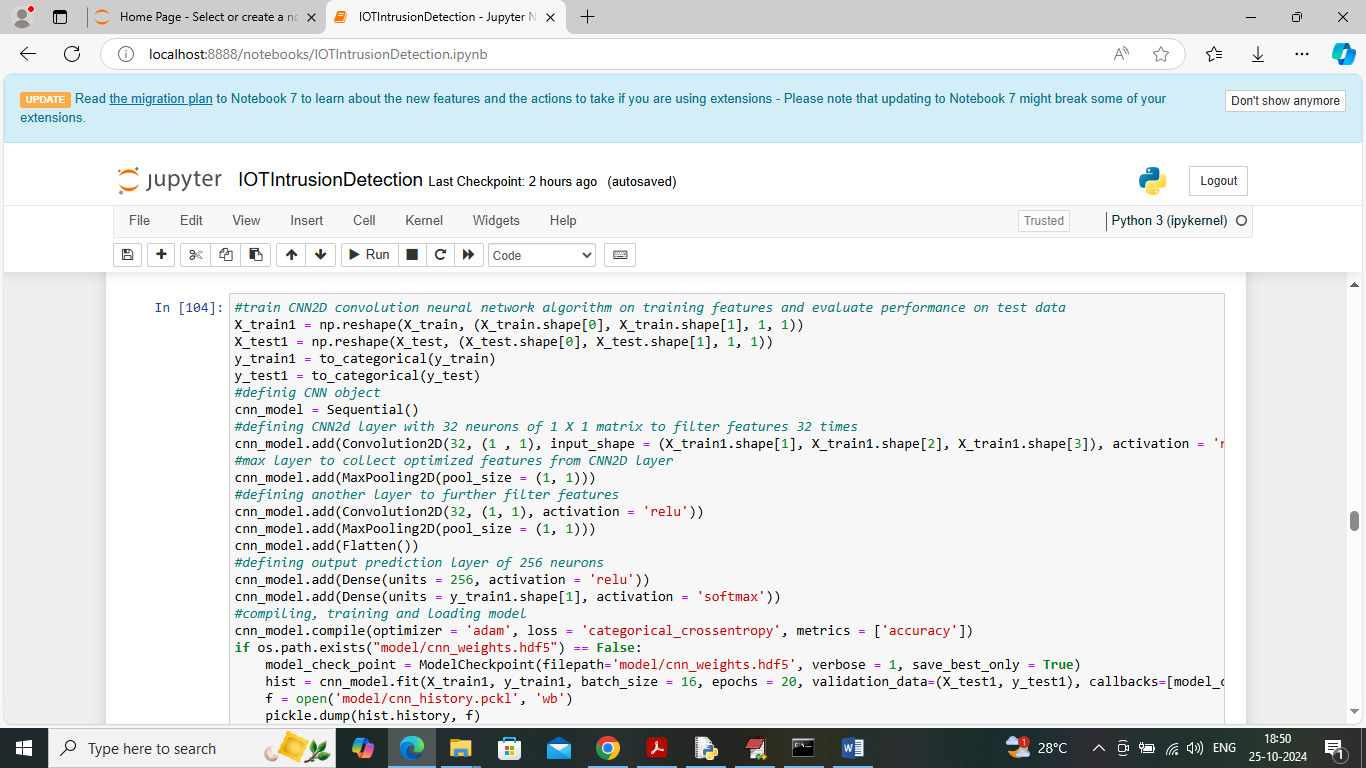
In above screen training Random Forest algorithm on training data and then performing prediction on test data and then Random Forest got 98% prediction accuracy on test data and can see other metrics like precision, recall and FSCORE. In below screen can see confusion matrix and ROC AUU graph



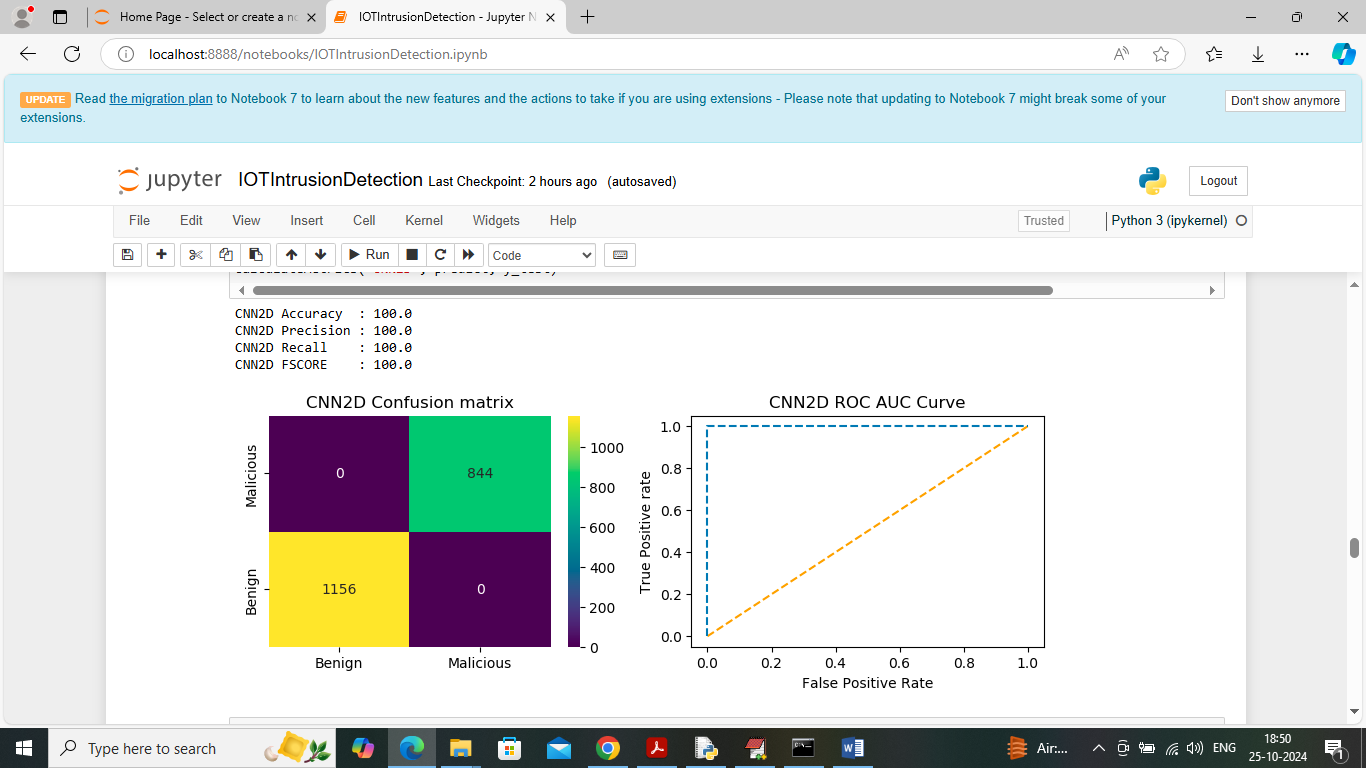
In above confusion matrix graph x-axis represents ‘Predicted Labels’ and y-axis represents ‘True Labels’ and then yellow and green boxes in diagonal represents correct prediction count and blue boxes represents incorrect prediction count which are very few. In above ROC graph x-axis represents ‘False Positive rate’ and y-axis represents True Positive rate and if blue line comes on top of orange line then all predictions are correct and if goes down below orange line then predictions are incorrect.



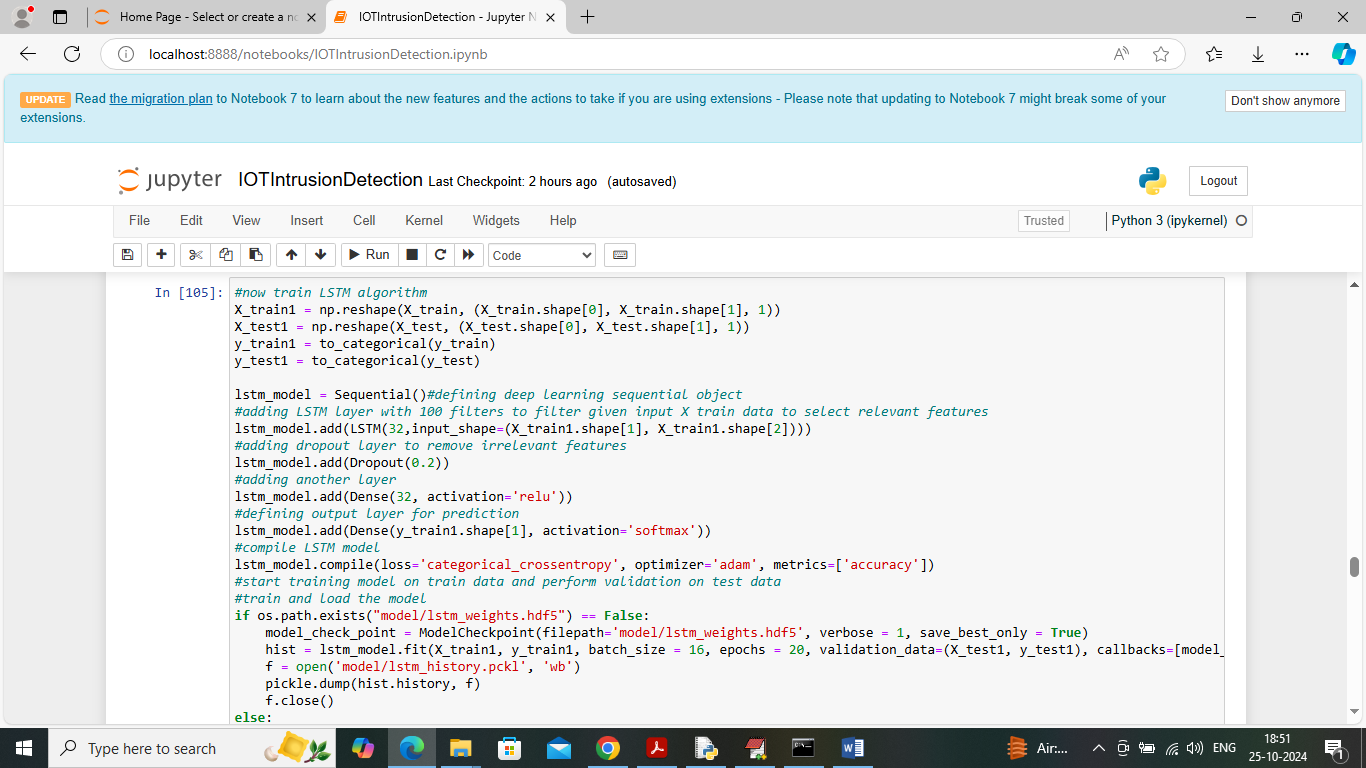
In above screen SVM got 95% accuracy and can see other metrics also



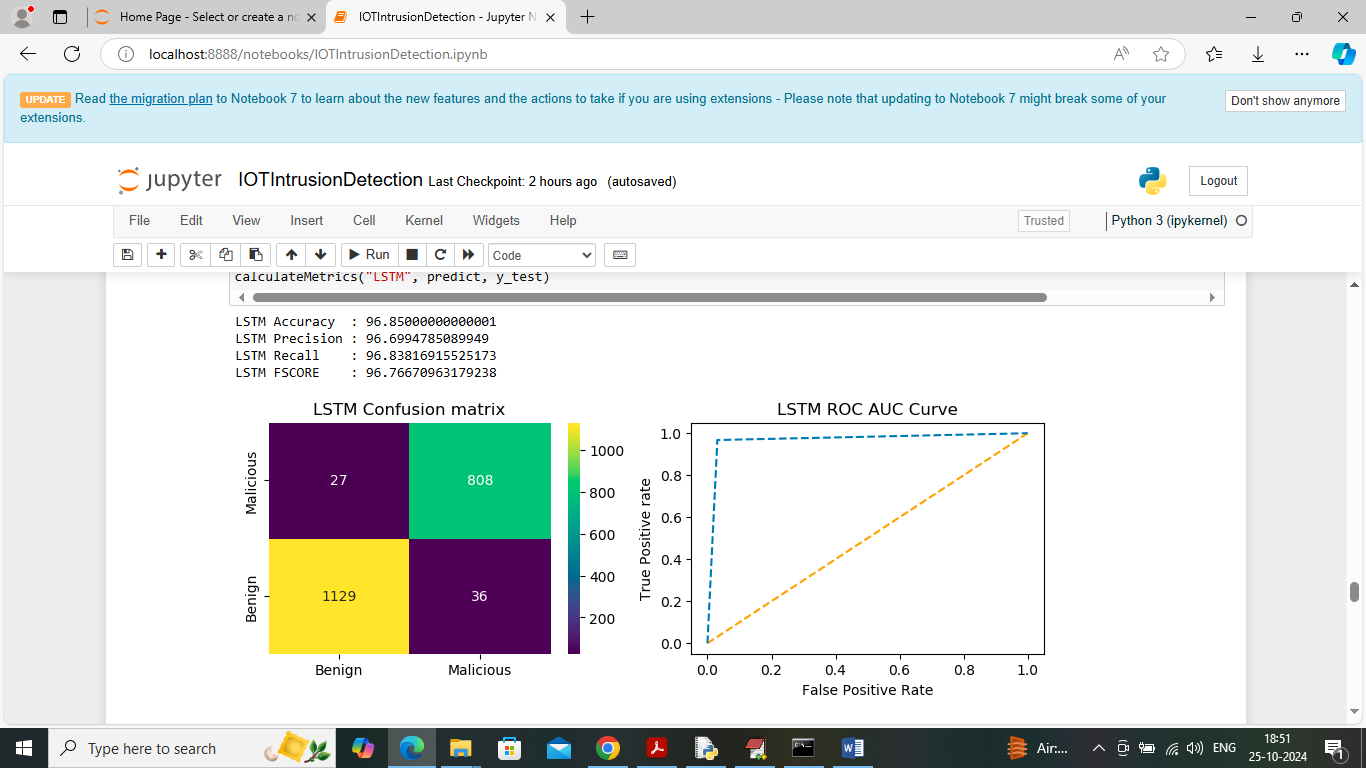
In above screen training CNN2D algorithm and after executing this block will get below output



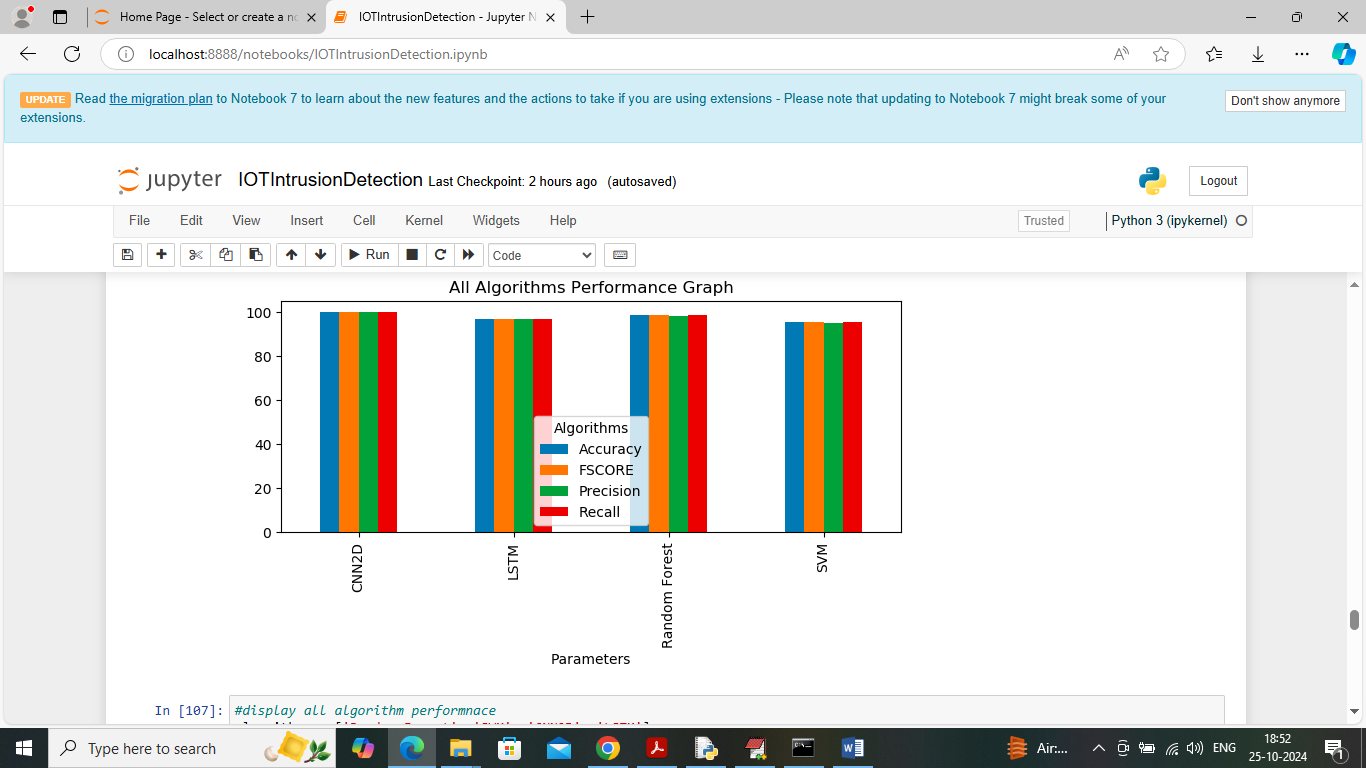
In above screen CNN2D got 100% accuracy and can see other metrics also



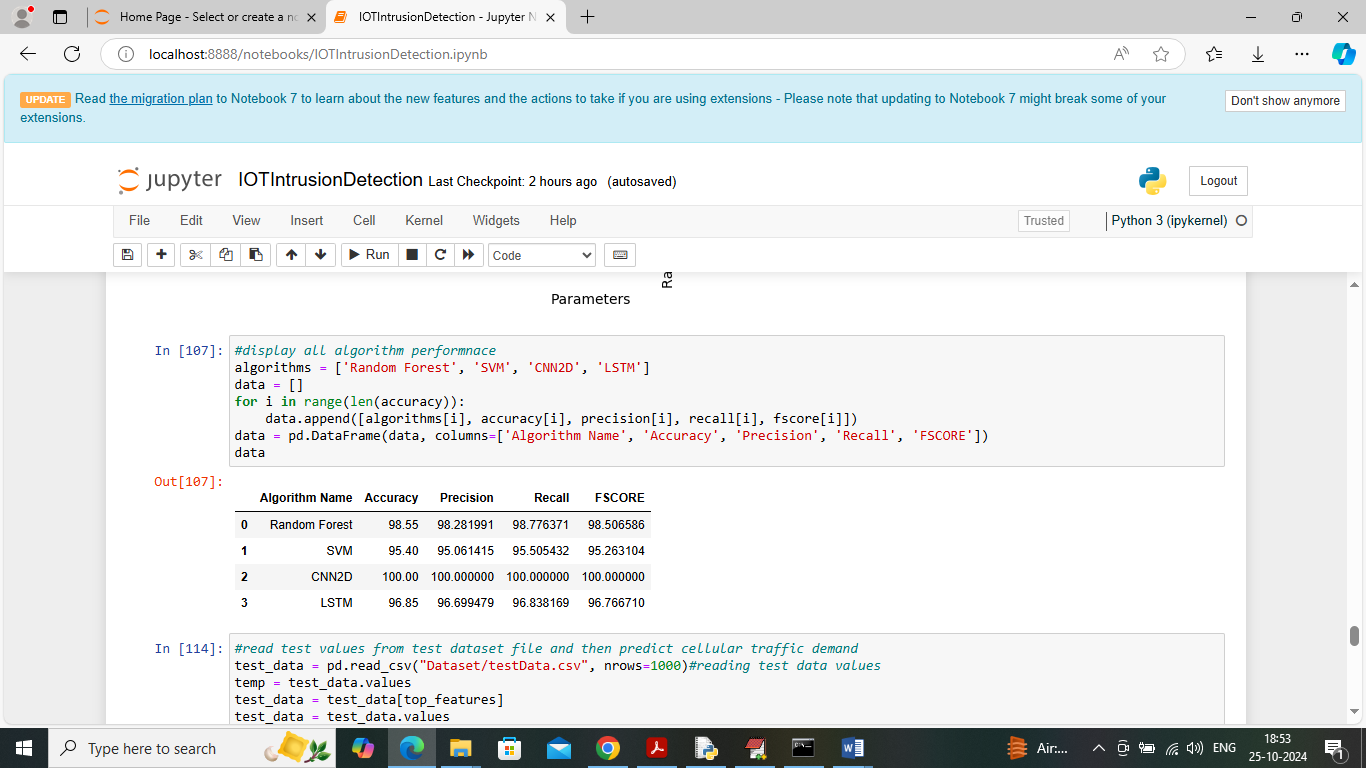
In above screen training LSTM algorithm and can read blue colour comments to know about the algorithm and below is the output of this algorithm



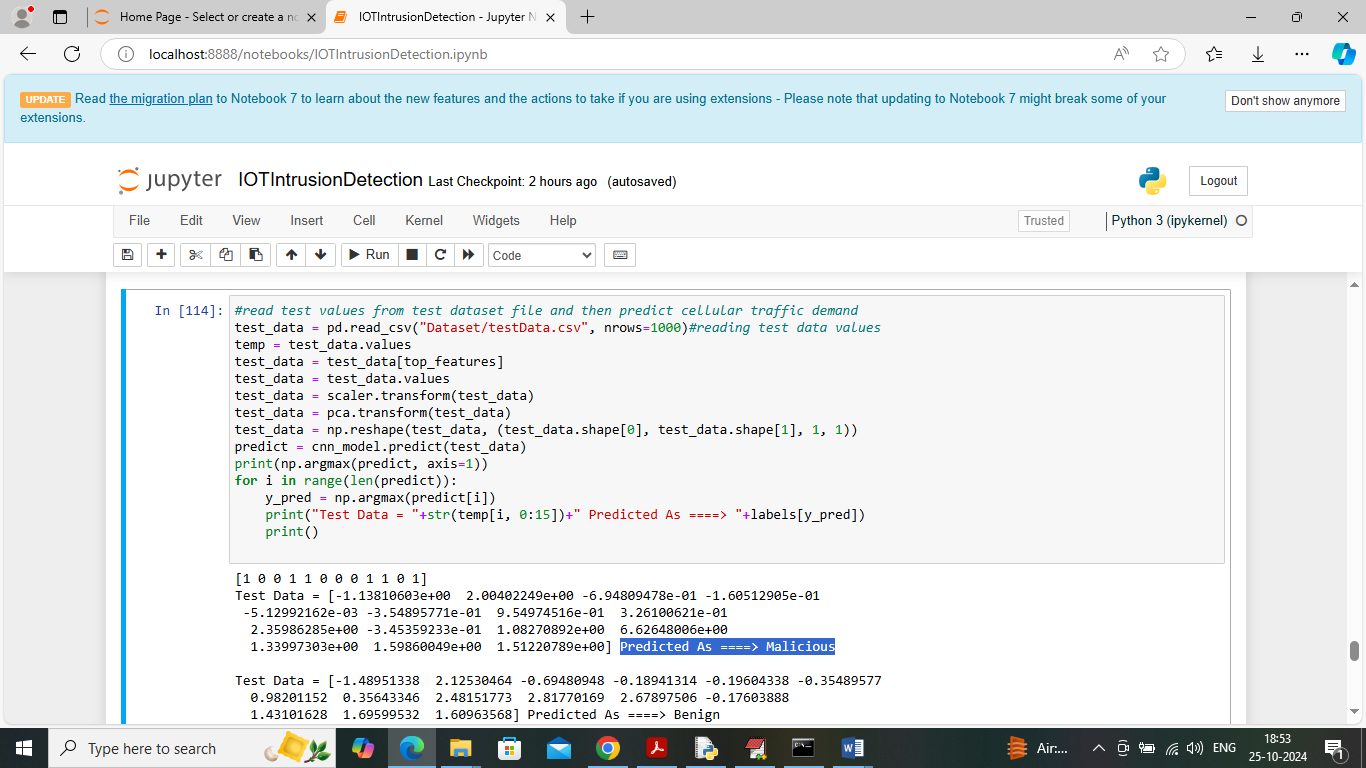
In above screen LSTM got 96% accuracy



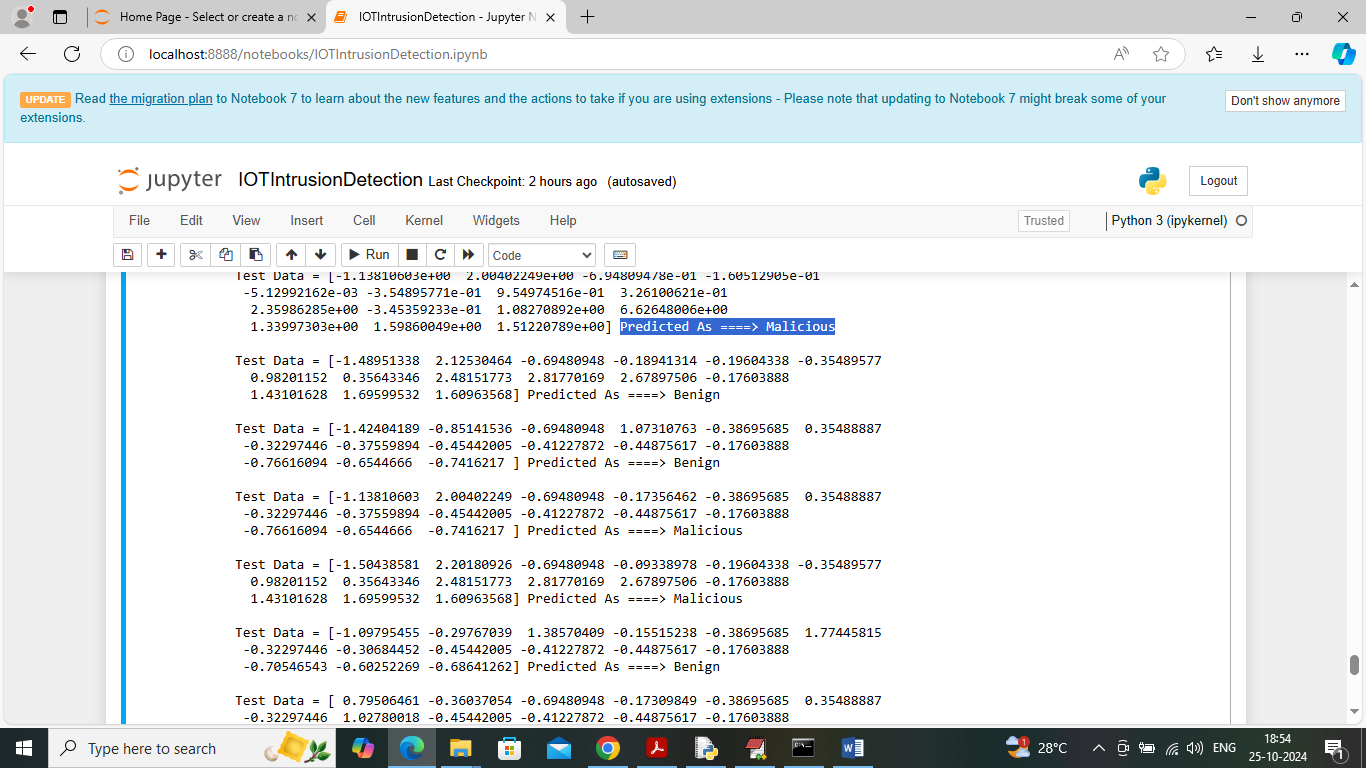
In above screen showing comparison graph between all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms CNN2D and Random Forest got high performance



In above screen displaying all algorithm performance in tabular format



In above screen reading test data values from test file and then apply processing technique and then applying CNN2D algorithm to predict test data as Benign or Maliciois. In above screen in square bracket can see Test Data values and then after arrow symbol =🡺 can see predicted output as ‘Benign’ or ‘Malicious’.



In above screen can see each test data record classification output as Benign or Malicious.

**CHAPTER 6**

**CONCLUSION**

The main objective of the present paper is to propose an effective and time-efficient intrusion detection system (IDS) having efficient attack detection capability. The present work has been evaluated on the smart home dataset ‘DS2OS’ that exhibits the modern traffic in the IoT-enabled environment. The proposed Intrusion Detection Model is a novel approach that has been designed using ML-based ensemble algorithms. Feature selection methods have been applied to the dataset for reducing the model’s prediction latency. Feature selection reduces the size of input data which also reduces time complexity, space complexity, and memory efficiency. The model has been evaluated using certain important metrics such as accuracy, time, errors-rate, TPR, and FPR. The results in the present paper indicate that no single algorithm can be taken into consideration as fiercely superior to the others. However, extreme gradient boosting (XGB) and light gradient boosting (LGBM) based ensemble models outperform in terms of accuracy and error rate. Still, the LGBM-based IDS showcases a lower threat prediction latency as compared to models based on other algorithms. Other benefits of the proposed model include low power consumption, high accuracy, and reduced over fitting. The proposed model efficiently balances the input dataset and detects the behavior of traces for intrusion detection. Its computation speed is very high and its complexity is only O(#data). TPR and TNR of the LGB-IDS model also exhibit higher performance. The ‘DS2OS’ dataset contains more than two classes and therefore, instead of binary classification, the multi-class classification has been performed in the present model to segregate normal (benign) and many types of anomalous classes. These classes have been identified on the basis of decision trees constructed from behavioural patterns of connected nodes. The proposed work may be quite helpful to perform classification, prediction, and decision-making in large, complex, and high-dimensional datasets as well as small and low-dimensional datasets not only in various IoT environments but also in various other real-time scenarios. Time efficiency is the most important feature of the proposed model. In future, the proposed model may prove to be significant to investigate cybercrimes in different IoTenabled environments with more reduced time consumption, low memory usage, low false alarm rate, high accuracy, and low error rate. A faster and efficient intrusion detection system (IDS) will be helpful to combat various security threats in different IoT environments. The results obtained using the proposed model may be quite helpful to target various sources of attacks in much less time and can block further attempts of attacks in various scenarios deployed in government and private sectors.

**REFERENCES**

1. H. F. Atlam, A. Alenezi, M. O. Alassafi, & A. A. Alshdadi, “Security, Cybercrime and Digital Forensics for IoT,” *In book: Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, pp. 551-577, Jan. 2020, <https://doi.org/10.1007/978-3-030-33596-0_22>.
2. M. Alkasassbeh & S. Al-Haj Baddar, “Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey,” *Arab Journal of Science and Engineering*, 2022, <https://doi.org/10.1007/s13369-022-07412-1>.
3. Z. Sun, Z. Lv, H. Wang, Z. Li, F. Jia, & C. Lai, “Sensing Cloud Computing in Internet of Things: A Novel Data Scheduling Optimization Algorithm,” *IEEE Access*, Vol. 8, pp. 42141-42153, 2020, <https://doi.org/10.1109/ACCESS.2020.2977643>.
4. DS2OS Traffic Traces, IoT traffic traces gathered in the DS2OS IoT environment, created in 2018. Available at <https://www.kaggle.com/francoisxa/ds2ostraffictraces>, Last accessed: 28th July, 2022.
5. D. Rani, N. S. Gill, & P. Gulia, “Classification of Security Issues and Cyber Attacks in Layered Internet of Things,” *Journal of Theoretical and Applied Information Technology*, Vol. 100, No. 13, pp. 4895-4913, July 2022.
6. A. Sagu, N. S. Gill, P. Gulia, J. M. Chatterjee, & I. Priyadarshini, “A Hybrid Deep Learning Model with Self-Improved Optimization Algorithm for Detection of Security Attacks in IoT Environment,” *Future Internet*, Vol. 14, No. 10, Article 301, 2022, <https://doi.org/10.3390/fi14100301>.
7. T. Chen & C. Guestrin, “Xgboost: A scalable tree boosting system,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD’16, pp. 785–794, NY, USA, Aug. 2016, <https://doi.org/10.1145/2939672.2939785>.
8. J. H. Friedman, “Greedy function approximation: A Gradient Boosting machine,” *The Annals of Statistics*, Vol. 29, No. 5, pp. 1189-1232, Oct. 2001, <https://doi.org/10.1214/aos/1013203451>.
9. X. Zhao & Q. Zhao, “Stock Prediction Using Optimized LightGBM Based on Cost Awareness,” *2021 5th IEEE International Conference on Cybernetics (CYBCONF)*, Sendai, Japan, pp. 107-113, June 2021, <https://doi.org/10.1109/CYBCONF51991.2021.9464148>.
10. E. Rehman, M. Haseeb-ud-Din, A. J. Malik, et al., “Intrusion Detection Based on Machine Learning in the Internet of Things, Attacks and Countermeasures,” *The Journal of Supercomputing*, Vol. 78, pp. 8890-8924, 2020, <https://doi.org/10.1007/s11227-021-04188-3>.
11. C. S. Htwe, Y. M. Thant, & M. M. S. Thwin, “Botnets Attack Detection Using Machine Learning Approach for IoT Environment,” *Journal of Physics: Conference Series*, IOP Publishing, Vol. 1646, Issue 1, Article ID 012101, 2020.
12. Y. Zhou, G. Cheng, S. Jiang, & M. Dai, “Building an Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier,” *Computer Networks*, Vol. 174, Article No. 107247, 2020, <https://doi.org/10.1016/j.comnet.2020.107247>.
13. A. Verma and V. Ranga, “Machine Learning Based Intrusion Detection Systems for IoT Applications,” *Wireless Personal Communication*, Vol. 111, pp. 2287–2310, Apr. 2020, <https://doi.org/10.1007/s11277-019-06986-8>.
14. P. Hadem, D. K. Saikia, S. Moulik, “An SDN-based intrusion detection system using SVM with selective logging for IP traceback,” *Computer Networks*, Vol. 191, Article 108015, May 2021, https://doi.org/10.1016/j.comnet.2021.108015.
15. P. Kumar, G. P. Gupta, & R. Tripathi, “Toward Design of an Intelligent Cyber Attack Detection System using Hybrid Feature Reduced Approach for IoT Networks,” *Arab Journal of Science and Engineering*, Vol. 46, pp. 3749–3778, 2021, <https://doi.org/10.1007/s13369-020-05181-3>.
16. A. Huč, J. Šalej, & M. Trebar, “Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices,” *Sensors (Basel)*, Vol. 21, No. 14, Article 4946, 2021, <https://doi.org/10.3390/s21144946>.
17. K. D. Devprasad, S. Ramanujam, & S. B. Rajendran, “Context Adaptive Ensemble Classification Mechanism with Multi-Criteria Decision Making for Network Intrusion Detection,” *Concurrency and Computation: Practice and Experience*, Vol. 34, Issue 21, e7110, June 2022, <https://doi.org/10.1002/cpe.7110>.
18. N. Gupta, V. Jindal, & P. Bedi, “CSE-IDS: Using Cost-Sensitive Deep Learning and Ensemble Algorithms to Handle Class Imbalance in Network-Based Intrusion Detection Systems,” *Computers & Security*, Vol. 112, Article 102499, Jan. 2022, <https://doi.org/10.1016/j.cose.2021.102499>.
19. G. Çetin, “An Effective Classifier Model for Imbalanced Network Attack Data,” *Computers, Materials & Continua*, Vol. 73, No. 3, pp. 4519-4539, 2022, <https://doi.org/10.32604/cmc.2022.031734>.
20. W. Xu and Y. Fan, “Intrusion Detection Systems Based on Logarithmic Autoencoder and XGBoost,” *Security and Communication Networks*, Volume 2022, Article ID 9068724, 8 pages, <https://doi.org/10.1155/2022/9068724>.

**APPENDIX**

**PYTHON**

**1.1 Introduction**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**1.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**1.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**1.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI applicatioMakeske the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**1.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**1.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics