Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Intelligent Detection of IIOT Cyber Threats: A Hyper tuned Ensemble Machine Learning Approach

S Gouri Kiran Kumar[1], Dr. Raavi Satya Prasad[2]

[1]Department of Information Technology, College of Computing and Information Sciences, University of Technology and Applied Sciences, Muscat, Oman, kiran43427@gmail.com
[2]Professor and Dean R & D, Department of Computer Science & Engineering, Dhanekula Institute of Engineering & Technology, India, deanresearch@diet.ac.in

The rapid adoption of Industrial Internet of Things (IIoT) technologies has transformed industrial operations, enhancing efficiency, productivity, and scalability through the interconnection of devices, sensors, and systems. However, this digital integration also introduces significant security vulnerabilities, as attackers increasingly target these complex networks with advanced and evolving cyber threats. Traditional detection systems, such as signature-based methods, are insufficient to address the dynamic nature of IIoT environments, often failing to detect zero-day attacks and producing high false positive rates. This research addresses these challenges by developing a hypertuned machine learning (M L) framework capable of detecting IIoT-specific attacks with high accuracy and speed. The proposed model utilizes feature selection techniques like Principal Component Analysis (PCA) to reduce dimensionality, alongside ensemble learning techniques, including stacking methods, to optimize performance. Key algorithms such as XGBoost, Random Forest, and Logistic Regression form the foundation of the detection system, with LightGBM serving as the meta-learner. The model is trained and evaluated using the WUSTL IIoT 2021 dataset, encompassing various attack scenarios including Denial of Service (DoS), Man-in-the-Middle (MITM), and unauthorized access. Results show that the hypertuned model achieves near-perfect accuracy and significantly reduces false positives compared conventional methods, offering robust, scalable, and real-time detection of IIoT attacks. This research contributes to the advancement of secure IIoT architectures, demonstrating the potential of intelligent machine learning solutions to safeguard critical infrastructure.

Keywords: Ensemble Learning, Anomaly Detection, Industrial Internet of Things, Stacking Method, Feature Selection.

1. Introduction

IIoT attacks are malicious actions targeting IIoT(Industrial Internet of Things) systems. The IIoT systems consist of interconnected devices and sensors used in industrial processes. The IIoT attacks can disrupt operations, compromise data integrity, or damage critical infrastructure by exploiting vulnerabilities in the IIoT network. Most of the IIoT systems were made up of numerous interconnected devices and sensors, creating a complex attack surface.

Attackers can exploit vulnerabilities in various ways, making detecting and responding to evolving threats essential. The complexity of IIoT systems makes it challenging to detect anomalies and potential attacks using traditional methods. Attackers are using advanced techniques. These techniques can evade basic security measures. Hypertuned machine learning models can adapt to and identify subtle patterns indicative of such advanced threats. IIoT systems often require real-time or near real-time detection to prevent damage or disruption as they will be used in most real scenarios. Hypertuned ML models can process large volumes of datasets quickly and accurately. This advantage of Hypertuned ML enables timely responses to potential threats and attacks. Hypertuned ML Models are highly adaptable as they learn from new data and adapt to evolving attack methods in agile way by maintaining their effectiveness over time. Apart from the above-mentioned issues scalability and automated response are also the issues where eve need to look for hyper-tuned ML models for detecting IIoT attacks.

1.1 Different Types of IIoT Attacks Detection: Techniques for detecting IIoT attacks include hybrid techniques, machine learning (ML), network-based, device- based, data-based, physicalinspection, signature-based, anomaly-based, and machine learning (ML). Benefits from these techniques include heightened security, instantaneous threat identification, better incident handling, and decreased downtime. They are able to recognize many different types of assaults, such as supply chain attacks, malware, malware-in-the-middle, Denial of Service (DoS), data manipulation, and illegal access. Nonetheless, several constraints are present, such as elevated false positive rates, the necessity for constant observation, reliance on high-quality data, susceptibility to intricate attacks, and heightened intricacy. Furthermore, signature-based detection is less resilient to zero-day assaults than machine learning (ML)-based detection, which demands a large amount of processing power and training data. A multi-layered strategy that balances detection techniques to minimize drawbacks and maximize benefits is necessary for effective IIoT security. This ensures complete defense against constantly changing threats. Figure 1 presents the popular ways of detecting the IIoT Attacks

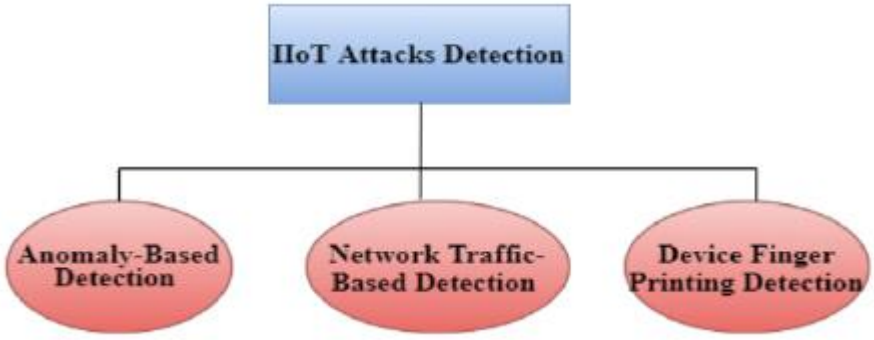


Figure 1: Types of IIoT Attacks Detection

1.1.1. Anomaly-Based Detection: Anomaly-based detection approach is used to identify attacks in Industrial Internet of Things(IIoT) environments by recognizing unusual behavior or deviations from established norms. This technique is particularly used in traditional IIoT systems where traditional security methods like signature-based detection struggle due to the complexity, diversity, and evolving nature of connected devices. The core concept of anomaly-

based detection is that it monitors the typical behavior of IIoT systems and flags any deviations as potential threats. The underlying assumption is that IIoT devices and networks usually follow predictable patterns in terms of data flows, device behaviors, and interactions. Any significant deviation from these patterns might indicate an attack or malfunction. The system learns the "normal" behavior of IIoT devices and networks by observing historical data that might include data transmission rates, device response times, communication patterns, sensor readings, etc.. The system continuously monitors real-time data and compares it to the learned baseline. Any deviations from the baseline are flagged as anomalies, which could be indicative of potential attacks. If the deviation is significant or sustained, the system triggers an alert for further investigation. Various machine learning (M L) and statistical techniques are used to implement anomaly- based detection in IIoT systems like Autoencoders, K-Means Clustering, Isolation Forests, Support Vector Machines, and alsoTime Series Analysis.

1.1.2 Network Traffic-Based Detection: Network Traffic-Based Detection is an important approach for identifying attacks in Industrial Internet of Things(IIoT) systems which analyses the communication patterns, data flows, and packet characteristics within the network. IIoT environments often involve a large number of interconnected devices that generate significant network traffic. Monitoring this traffic for anomalies, unusual patterns, or known attacksignatures can help detect malicious activities. Network Traffic-Based detection focuses on analyzing the communication between IIoT devices and other systems to identify attacks. This method inspects the packets flowing through the network and looks for anomalies in packet content, traffic volume, or unexpected device interactions. Attacks such as Denial-of-Service (DoS), Man-in-the-Middle(MITM), packet injection, and data exfiltration attempts can be effectively detected by this method. Network Traffic-Based Detection models analyze features like packet size, traffic volume, source/destination IPs, protocol usage, and connection frequency. Unusual traffic spikes, unexpected communication with external servers or large and data transfers can be considered as anomalous behavior that will signal potential attacks. Machine Learning models suns Random Forest, K-Means clustering, LSTM networks, and Autoencoders are often used to classify the traffic as normal traffic or malicious traffic and enhance detection accuracy for both known and unknown attacks. Despite these advantages, Network Traffic-Based Detection also faces a few challenges. They are managing high-traffic volumes, handling encrypted traffic, mitigating false positives, and the scalability of large IIoT networks. To address these, a combination of anomaly-based and signature-based methods. along with edge analytics and threat intelligence integration, is often used.

1.1.3 Device Fingerprinting Detection: Device Fingerprinting Detection detects attacks in the IIoT environments by identifying the unique characteristics of devices connected to theenvironment. This method creates a unique "finger print" for each device by profiling each IIoT device based on its hardware, software, and network behavior. It will continuously monitor these devices' behavior and compare them with normal device behavior and any deviation mayindicate a potential security threat. The fingerprints for the devices will be created using unique features of any device such as MAC addresses, CPU types, firmware versions, operating systems, and device configurations. These help in identifying the device and distinguishing and detecting any unauthorized modifications. Devices in IIoT networks often have predictable behavior like communication patterns, data transmission rates, protocols used, and communication with specific servers as they are designed for specific tasks

and their scope is limited. Any deviation fromthis behavior indicates a compromised device or an external attack. Fingerprinting helps prevent attacks where malicious entities attempt to impersonate legitimate devices by ensuring that only recognized fingerprints can interact with the network. Machine Learning models can enhance fingerprinting by learning device behaviors over time and detecting even subtle anomalies, improving the detection of advanced attacks. The challenges face by Device Fingerprinting Detection models are IIoT devices may change behavior due to legitimate updates or configurations, which can complicate fingerprinting. Scalability and false positives are the other challenges.

2. Literature Survey

V. Piiya et al [1] an effective IIoT attack detection model utilising an ensemble classifier methodology. The approach uses a two-step categorisation procedure. To remove duplicate and null data, the first step involves normalising the data of three IoT datasets: Bot_IoT, N_BaloT, and WUSTL_IIOT-2018. Next, using various cross-validation ratios, the data is divided into testing and training sets: the best accuracy is obtained with an 80:20 ratio. Using a blending ensemble approach, the first classifying level integrates the SVM, NB, & DT classifiers. After combining the output of both models, a RF classifier receives the new training set, which it uses to make more predictions. Concurrently, the same data are used to deploy an ANN classifier that has been optimisedusing the Adam optimizer. The second level compares theresults of the RF and ANN models and chooses the most accurate result as the final guess. On the test datasets, this technique performed better than others.

Mohammed Amine Ferrag et al [2] The Edge- IIoTset project aims to create a complete and accurate safety collection for both IoT and IIoT apps. The goal of this dataset is to help intruder detection systems learn in both centralised and shared ways. The method has several important steps, the first of which is setting up and configuring a complex seven-layer testbed that looks and works like an IoT or IIoT environment in the real world. This testbed has many layers. some of which are bitcoin, cloud computing, and edge computing. The second phase is threat& attack modelling, which identifies and groups fourteen distinct attack scenarios about IoT and IIoT communication protocols into five main threats, such as malware and DoS/DDoS assaults. After that, tools like Wireshark are used to create and record both regular and attack data. Features are then taken from the gathered packet data. Also included in the methodology arc data pre-processing stages, including the labelling of features for binary as well as multiclass classification, the removal of duplicates, and the standardisation of features. The dataset is then tested using different ML models. The outcomes indicate that the dataset is useful for making attack detection better in IoT and IIoT settings.

Abdullah Alsaedi et al [3] Develops and evaluates the new and comprehensive TON_IoT Telemetry Dataset for IDSs in IoT and IIoT contexts. Create a realistic & representative medium-scale testbed to replicate IoT/IIoT networks with Edge, Fog, and Cloud layers. The collection includes telemetry data from sensors and devices, operating system logs, and network traffic from the testbed. The dataset simulates nine cyber-attacks, including DDoS, ransomware, & injection assaults, to offer varied & realistic data for IDS training and evaluation. The technique also labels and normalises data and analyses different ML models, notably SVM, decision trees, and DL models like LSTM. This all-inclusive strategy seeks to

close the existing gap in IoT/IIoT dataset availability and provide a strong basis for creating and evaluating IDSs in these settings.

Imad Tareq AL-Halboosi et al [4] focuses on applying machine learning methods to identify IoT network cyber-attacks. The WUSTL-IIOT dataset's Traffic data was analysed by using several ML methods, such as SVM, LDA, and QDA. Their method's primary component is the utilisation of computing in parallel, which splits the information into smaller pieces and analyses each one concurrently. Master prediction and training are made possible by this parallelising of computing power. In addition, the methodology employs a parallel voting predictor system, which involves the collaboration of multiple classifiers to determine the ultimate classification through weighted ballots. This helps the model to have great accuracy in spotting assaults like DoS & backdoor intrusions. The highest accuracy among the parallel SVM+LDA models turned out to be shown here. This emphasises the need to scale the information to maximise speed and eliminate pointless elements through pre-processing.

Mohammed S. Alshehri et al [5] IoT networks need an SA-DCNN model to find intuitions. The model uses a DCNN to analyse significance values assigned by self-attention processes and to detect suspicious network behaviors. Data cleansing, feature encoding, & filtering among other important phases comprise the approach. Using a label encoder, info is pre-processed by eliminating duplicate and unknown values and numerically transforming category characteristics. Feature filtering uses the mutual information approach to rank qualities depending on their effect and remove those having a negative impact. Later, layers optimizedfor multi-class classification are added to the SA-DCNN model and trained on the loTlD20 and Edge-IIoTset datasets. Models are improved by tuning hyperparameters like batch size, layers of convolution, & AF. Validations are used to assess results, therefore proving the model's efficacy in highly accurate and efficient intrusion detection.

Lahcen Idouglid et al [6] modem ML methods are utilised to create an IDS to support the IIoT. The process starts with preparing the information, which includes cleaning, transforming, and normalising the data to make sure it is consistent and useful for finding anomalies. Subsets of the dataset are then separated for testing and training to assess the performance of the model. Intruders are accurately found using four ML algorithms: XGBoost, SVM, MLP, and k-NN. Gradient-boosting ensemble approach makes XGBoost efficient and accurate. Finding the best hyperplane is how SVM improves classification, and k-NN finds local trends that are useful for classification tasks. The MLP model can handle complicated nonlinear data structures thanks to its multiple neural network design. Using public datasets like CIDDS and Bot-IoT to try and fine-tune these algorithms and judge their performance based on validation techniques. By combining these methods, the objective is to make monitoring more accurate and provide a flexible, scalable way to protect IIoT networks.

Hong-Yu Chuang et al [7] the method used ML-based detection of harmful activity in IIoT systems. The TON_IoT dataset includes many network traffic characteristics. By use of a feature selection procedure, the initial 45 features were limited down to 10 core characteristics. hence lowering computing complexity. This procedure helps to pinpoint the most important features by using the PCC to evaluate feature correlations and create frequency tables utilisingJamovi tools. Features not numerical were encoded for ML compatibility. These ten key characteristics were used to evaluate four machine learning models: eXtreme Gradient

Boosting, KNN, RF, NB, and NB. Training (70%) & testing (30%) sets comprised the dataset; each model's performance was assessed using validation criteria. Using these lowered characteristics worked well to maintain great precision.

S. Gopalakrishnan et al [8] implements a hybrid ML predictive maintenance solution for the car sector leveraging IIoT. Smart sensors in industrial equipment provide operational data to the IIoT platform for continuous machine performance & product quality monitoring. Using both controlled and untrained learning models on the collected data helps find problems in the production process and machine errors early on. The approach uses feature selection to extract crucial data points for anomaly detection and cloud storage for historical data. This makes it possible to analyse data in real-time and compare it with historical data to increase the prediction system's accuracy. The hybrid model makes machines work better by constantly improving datasets using both unsupervised learning for raw data and controlled learning for trends in known data. Successful implementation of the system results in improved machine efficiency, production precision, proactive maintenance interventions, reduced downtime and improved product quality.

Table 1: Analysis of the Existing Approaches

Author	Algorithm	Merits	De-merits	Accuracy
V. Priya el al	Stackedensemble methods, ANN	Has developed two phase anomaly identification model.	Time-complexity	99%
MohamedAmine Ferrag et al	ML	Data analysis was accurate.	Global modelprediction has little performancedifferences compared to remaining.	93%
AbdullahAlsaedi et al	LSTM, & KNN	Overcomes the IDSs enabled systems issues.	Baseline methodshas to perform more accurately.	77%
Imad Tareq AL-Halboosi et al	SVM+LDA	By partitioning the data and training it parallelly the model has acquired good performance.	Reliability has tobe improved.	I00%
Mohammed S.Alshehri et al	SA-DCNN	Every stage has in depth detection which increases the performance.	High cost.	96.5%
Lahcen Idouglidet al	XGBoost	Worked on real world datasets.	The model has tobe updated all the time which is complex.	99%
Hong-YuChuang et al	KNN, RF, NB,eXtremeGradient	Any kind of attacksare determined with good efficiency and effectivity.	Based on featuresprediction time complexity is determined.	96%
S.Gopalakrishnan et al	HML	The model isefficient in speed and productivity.	The process maynot fitforall models.	92%

2.1.1 Research Gaps Identified:

1. The study is mainly concerned with categorization tasks and conventional attack scenarios. Not enough research has been done on the possibilities of anomaly detection, especially unsupervised or semi-supervised methods that can detect unknown or zero-day

assaults. Enhancing detection accuracy in unexpected contexts may benefit greatly from this field of study.

2. Predictive maintenance models often operate as "black boxes," especially when using complex ML algorithms. The study does not focus on the explainability of the predictions, which is crucial for decision-makers in industrial settings. Future research could focus on explainable Al (XAI) techniques to provide insights into how the model detects faults or anomalies.

3. Proposed Methodology:

The proposed model initial analyses the relation between the features using the two-way ANOVA test using degrees of freedom approach because it helps in analysis of network traffic, sensor data and security evaluation. The model then selects the best features using the PCA and applies intelligent stacking approach as shown in the figure 2

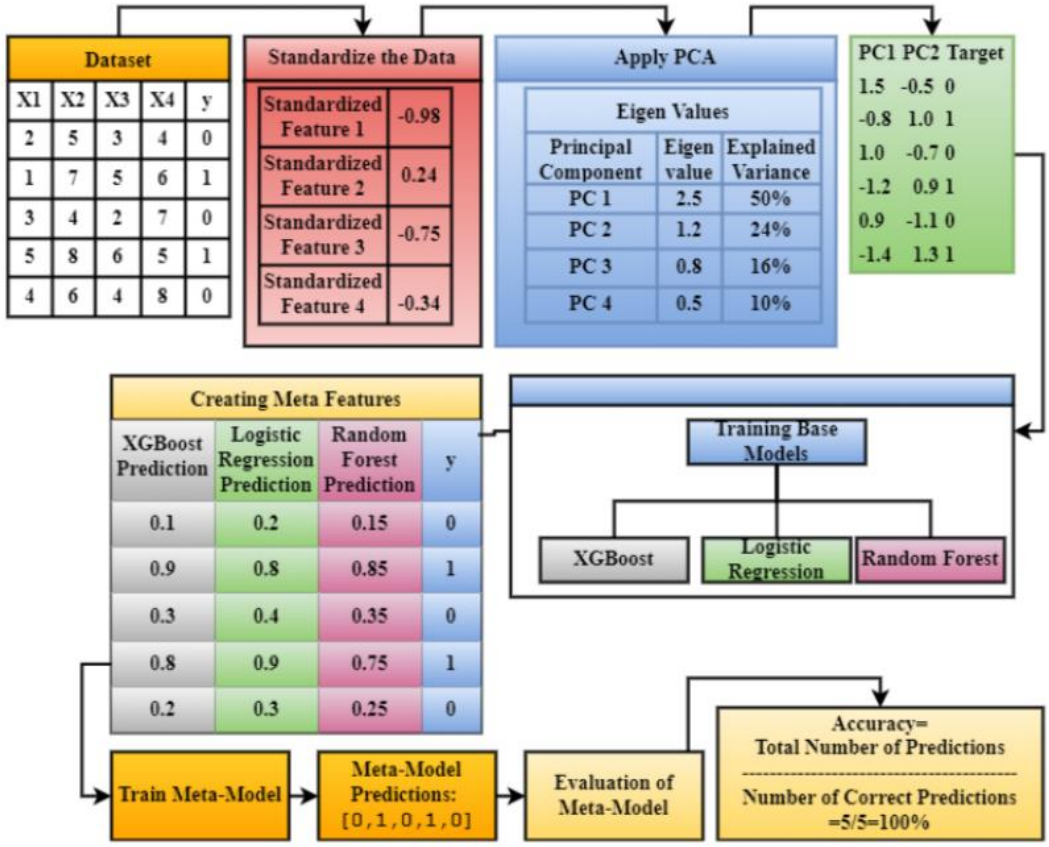


Figure 2: Block Diagram for the Proposed Model

3.1 Working of PCA for Best Feature Selection: Large datasets can be made less dimensional by using a statistical approach called principal component analysis, or PCA. It creates a new

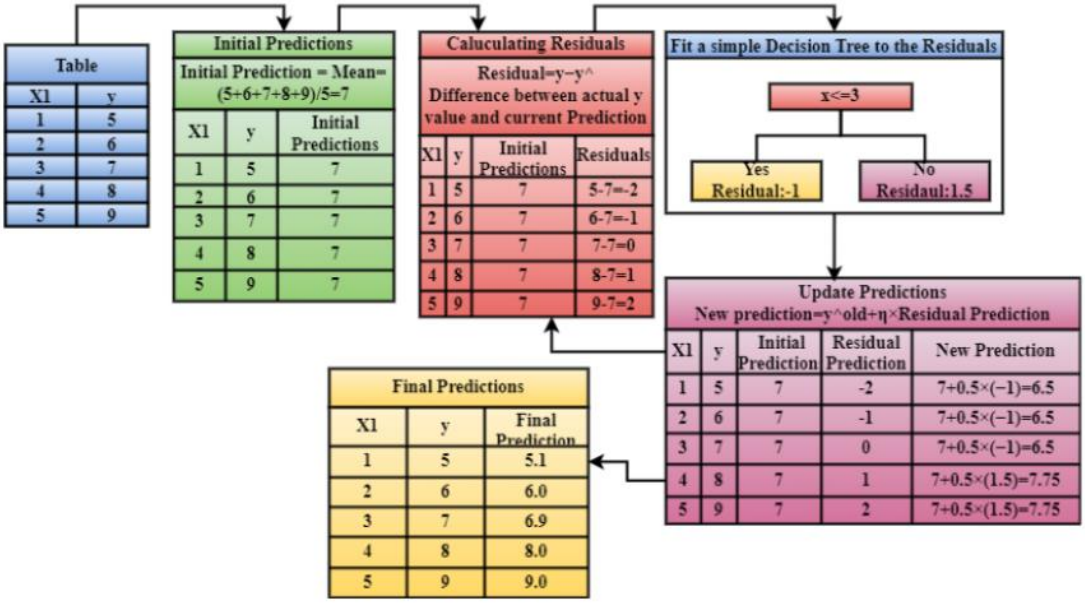
set of attributes known as principle components from the features (columns) of a dataset. The basic features are combined linearly to form these fundamental components. In machine learning, features that are redundant or multicollinear can lead to problems like overfitting and extended training durations, particularly when working with high-dimensional datasets. To overcome these difficulties, dimensionality reduction methods like Principal Component Analysis (PCA) are widely utilized. It transforms the data reduces it into a lower-dimensional space and preserves as much variance as possible. By transforming correlated features into uncorrelated principal components, PCA removes multicollinearity, which helpsimprove model performance. Reducing the number of dimensions (features) improves training speed reduces memory consumption and discards irrelevant or noisy features thus helping to create a more generalizable model. Columns that are not relevant to the analysis were removed (StartTime, LastTime, SrcAddr, and DstAddr), simplifying the dataset. The Traffic column (categorical) was label encoded to convert it into a numeric format, which is necessary for models that cannot process categorical variables directly. The features (X) were separated from the target columns, (y) where X contained all the predictor variables, and y held the target class.

The data was split into training (80%) and testing (20%) subsets to evaluate model performance on unseen data. Since the dataset was imbalanced, RandomOverSampler was used to balance the training data by increasing the representation of the minority class, ensuring that the model is not biased towards the majority class. To begin with, the covariance matrix of the pre-processed data is computed rising PCA. The covariance matrix captures how features are correlated with each other. Covariance quantifies how two variablechange together. If two features are positively correlated, PCA aims to combine them in a way that reduces the redundancy (similarity) between them. After computing the covariance matrix, PCA calculates the eigenvalues and eigenvectors. Eigenvectors determine the direction of the new feature axes(called principal components), and eigenvalues correspond to the amount of variance each principal component captures from the data. The first principle component absorbs the largest variation, while each successive component captures less, with the principal components arranged in decreasing order of eigenvalue. Following the identification of the primary components, the original data is projected onto this new axis system. The pre-processed data was converted into a Twenty-dimensional space for features (instead of the original higher- dimensional space) by means of a PCA model that chose 20 primary components. The initial features are combined linearly to create the new features (principal components). A more condensed version of the data that keeps the Majority of the variance is produced by this transformation.

3.2 Intelligent Stacking Approach: The proposed model in the base level tunes and combines the three algorithms namely logistic regression, random forest and XGBOOST. The process of tuning the base models is explained in the below section

3.2.1 Tuning and Working of XGBoost: Extreme Gradient Boosting or in simple terms XGBoost, is an optimized and simplified implementation of the Gradient Boosting algorithm, designed to reduce the complexity and increase the performance, accuracy, and efficiency. In order to build a powerful model, it employs ensemble learning, which combines several novice learners (decision trees). Every new tree in XGBoost minimizes a differentiable loss function (such as log loss for classification) in order to fix the faults of the preceding ones. The

algorithm builds trees sequentially to reduce the error step-by-step. XGBoost is highly efficient because it parallelizes the tree construction process, making it much faster than other implementations of gradient boosting. In both the SelectKBest and PCA cases, XGBoost was one of the base models. Atler applying SelectKBest, the top 20 features were selected based on their correlation with the target variable. These features were passed to the XGBoost model. The selected features were scaled using StandardScaler to standardize the feature values, which is important for models sensitive to the scale of input data. XGBoost was trained on the scaled featuresX_train_kbest_scaled to learn the patterns in the data. During training, it used its gradient boosting mechanism to iteratively minimize the classification error(logarithmic loss). The XGBoost model's predictions(probabilities for class 1) were used as meta-features for thestacking model. These were combined with predictions from other base models(Logistic Regression and Random Forest) to train the LightGBM classifier as the meta-model. The working process is shown in figure 3.



X1	y
1	5
2	6
3	7
4	8
5	9

X1	y	Initial Predictions
1	5	7
2	6	7
3	7	7
4	8	7
5	9	7

X1	y	Initial Predictions	Residuals
1	5	7	5-7=-2
2	6	7	6-7=-1
3	7	7	7-7=0
4	8	7	8-7=1
5	9	7	9-7=2

X1	y	Initial Prediction	Residual Prediction	New Prediction
1	5	7	-2	7+0.5×(−1)=6.5
2	6	7	-1	7+0.5×(−1)=6.5
3	7	7	0	7+0.5×(−1)=6.5
4	8	7	1	7+0.5×(1.5)=7.75
5	9	7	2	7+0.5×(1.5)=7.75

X1	y	Final Prediction
1	5	5.1
2	6	6.0
3	7	6.9
4	8	8.0
5	9	9.0

Figure 3: Working of Tuned XGBoost

In PCA case the dimensionality of the data was reduced to 20 components. These elements, which capture the mostuncertainty in the data, are the linear amalgamation of the original features. The PCA-transformed features were passed to the XGBoost model. While these features are not the original raw features. XGBoost still worked by building decision trees that partition the transformed feature space. Similar to the SelectKBest case, XGBoost was trained on the PCA-transformed data and the predicted probabilities we used as meta-features for the stacking model, contributing to the final prediction through the LightGBM meta-model. XGBoost builds decision trees in a sequential order, where the results of the previous tree will be passed to the next tree and the current processing tree will focus on minimizing and correcting the mistakes made by the previous tree. The trees are constructed by partitioning the feature space into regions that best separate the target classes. Each tree is a weak learner(a

decision tree with limited depth) that is built to minimize a differentiable loss function(log loss- mgloss), which is commonly used for classification problems. Instead of updating the model weights as in classical gradient descent, XGBoost fits the new tree to the residuals(errors) of the previous trees. Each new tree aims to predict the gradient(or direction of error) to improve the model's predictions. XGBoost uses the gradients to update the model iteratively. During training, XGBoost transformed the data by learning the splits in the decision trees that best separated the target classes. These splits were based on the gradients computed from the loss function. For eachinput sample, XGBoost assigns it to a region of the feature space defined by the tree structure. The final prediction is a combination of the predictions from all trees, adjusted based on the learning rate and gradient information. After training, when new data is passed to the model, XGBoost applies the learned trees to make predictions, combining the outputs of all trees to generate a probability score for each class. Table 2 presents the tuning parameters and their best values of the XGBoost.

Table 2: Parameters of the XGBoost

S.No	Name	Description	PossibleandBest Values
1	Evaluation Metric	Inordertodeterminethe performance measure that will be used to assess the model both during training and validation, the eval metric in XGBoost (XGB) classifiers is important. This measure makes ensuring the model optimizes for the appropriate goal given the current situation and aids intracking its learning process.	Possible: logloss, auc, mlogloss, merror Best: mlogloss
2	Sub Sample	Itrepresentsthefractionamount toconstructthetreefromthetraining data	Possible: any value between 0 to 1 Best: 0.6
3	Colsample_bytree	It represents the fraction of features fortree construction based on sub samples	Possible: any value between0to1 Best: 0.6

3.2.2 Tuning & Working of Random Forest: As part of an ensemble learning technique, the Random Forest (RF) classifier constructs several decision trees during training and then combines the output to increase precision and avoid overfitting. A random subset of the data, including samples and characteristics, is used to train each tree in the forest. Predictions are then formed by averaging or voting on the outputs of these trees. As part of an ensemble learning technique, the Random Forest (RF) classifier constructs several decision trees during training and then combines the output to increase precision and avoid overfitting. A random subset of the data, including samples and characteristics, is used to train each tree in the forest. Predictions are then formed by averaging or voting on the outputs of these trees. Random Forest is used as one of the base models in the stacking process. It operates by aggregating decisions from multiple trees to generate its final predictions. The SeleckKBest method selects the top 20 features based on their relationship with the target variable. These 20 features are chosen using statistical tests which help identify which features are most informative. After scaling the best featurethe data is standardized using StandardScaler. The pre-processed data is then passed into the Random Forest model. The model creates several decision trees, each of which receives a portion of the samples and data. The final prediction is calculated by averaging the outcomes of all the trees. Instead of feature selection, PCA (Principal

Component Analysis) reduces the dimensionality of the data by transforming it into 20 uncorrelated components. These components are linear combinations of the original features, capturing as much variance as possible with fewer dimensions. Once PCA transforms the data, these 20 components (rather than the original features) are passed into the Random Forest model. Unlike SelectKBest, PCA provides components that may not directly correspond to the original features but represent a combination of them, aimed at maximizing variance. After applying PCA, the Random Forest model is trained on the transformed data. Since PCA components are orthogonal, the model gets input features that are statistically uncorrelated, which might enhance its performance in certain scenarios. Figure 4 presents the working of the tuned random forest.

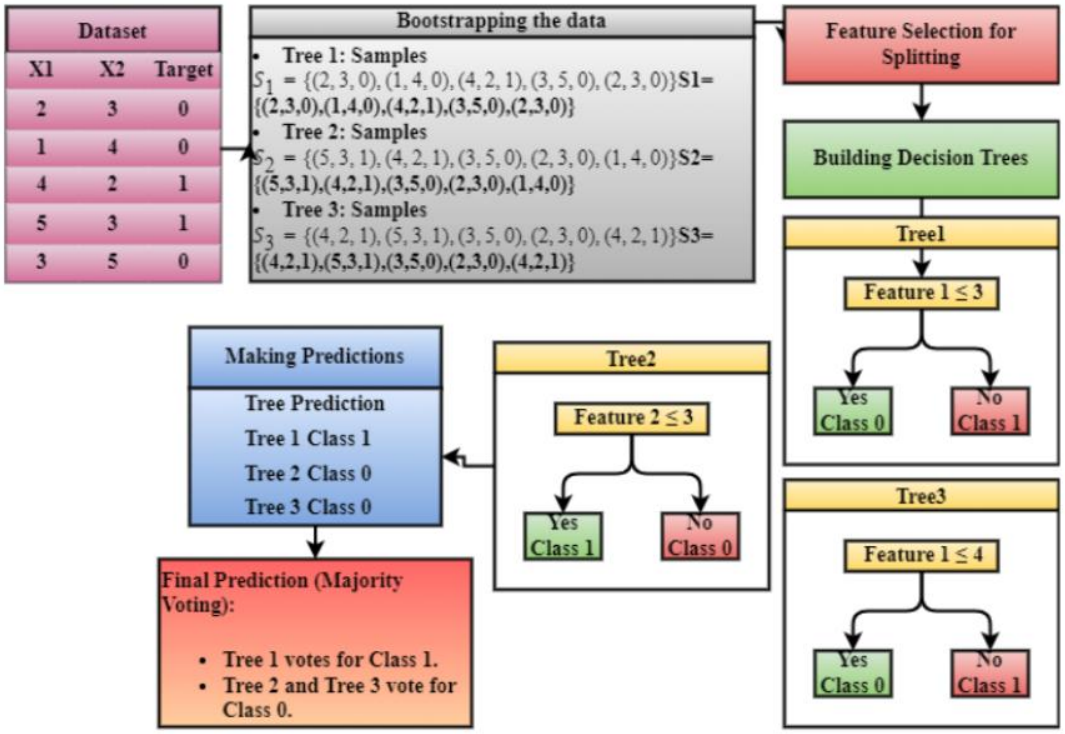


Figure 4: Working of Tuned Random Forest

Random Forest doesn't apply an explicit transformation to the data itself but learns patterns from the data during the training phase. In the SelectKBest approach, Random Forest receives the top 20 original features after selection. It splits the data at different points based on those features and their importance, constructing multiple decision trees. Each tree's decision-making process varies slightly because they receive different subsets of the data. The ensemble of trees works together to classify or predict based on the majority vote or average outcome of all trees. In the PCA approach, Random Forest works with the 20 components generated by PCA. The components generated are derived from the linear combinations of thedata we are passing. Even though this input data is in the new structure, Random Forest still follows the same tree-building mechanism. It splits the data along the component axes rather than the original features, aggregating decisions from multiple trees to output a final prediction.

Random Forest is trained on either the scaled original features (SelectKBest) or PCA components, learning from the training data to classify the target variable. After training, Random Forest is used to generate meta-features for the stacking model. Specifically, itoutputs class probabilities for each instance in the training set. These probabilities are then used as input to the meta-model (LightGBM), forming one-third part of the feature set used to train the meta-model and predict the final results. The dataset is randomly divided into subsets and the decision trees will be trained on each subset. This randomness helps avoid overfitting by introducing variation between trees. Each tree makes splits based on feature importance, using criteria like Gini impurity or entropy to decide where to split the data. for classification, each tree outputs a class label, and the final prediction is determined by majority voting (classification) or averaging (regression). Table 3 presents the tuning parameters and their best values of the Random Forest

Table 3: Parameters of the Random Forest

S.No	Name	Description	PossibleandBest Values
1	Criterion	It determines the best way to split thefree	Best: Entropy
2	n_estimators	Itdeterminesthenumberoftrees tocombine	Anyinteger value Best:t

3.2.3 Working & Tuning logistic Regression: Although it sounds like a regression procedure, logistic regression is a supervised method of machine learning that is used for binary or multiclass classification applicationsModeling the likelihood that a given input belongs to a specific class is the main concept. Logistic Regression uses a logistic function (sigmoid function) to model the probability of a binary outcome (0 or 1). Any real number can be transformed by this function to a value between 0 and 1, which is the probability of the positive class. Py=1X=11+e-(0+1x1 +2x2+…..+nxn). The formula uses a linear combination of input features (the X's) and their corresponding weights(β's) to compute the probability that the instance belongs to the positive class. Once the probability is computed, a threshold (usually 0.5) is applied to decide the final classification. If Py= 1X>0.5 the prediction is class 1: otherwise, it is class 0. Logistic Regression is sensitive to feature scaling because it relies on gradient-based optimization algorithms(such as gradient descent). In the code, for the PCA case, the data didn't get scaled but SelectKBest-selected are scaled using StandardScaler before being passed into the Logistic Regression model. This ensures that all features contribute equally to the model and improves convergence during training. In the SelectKBest approach, the dimensionality of the data was reduced by selecting the 20 most significant features. Logistic Regression will use these top features to compute the linear combination used in the logistic function. This process simplifies the data and improves interpretability and computational efficiency. Figure 5 presents the working of logistic regression

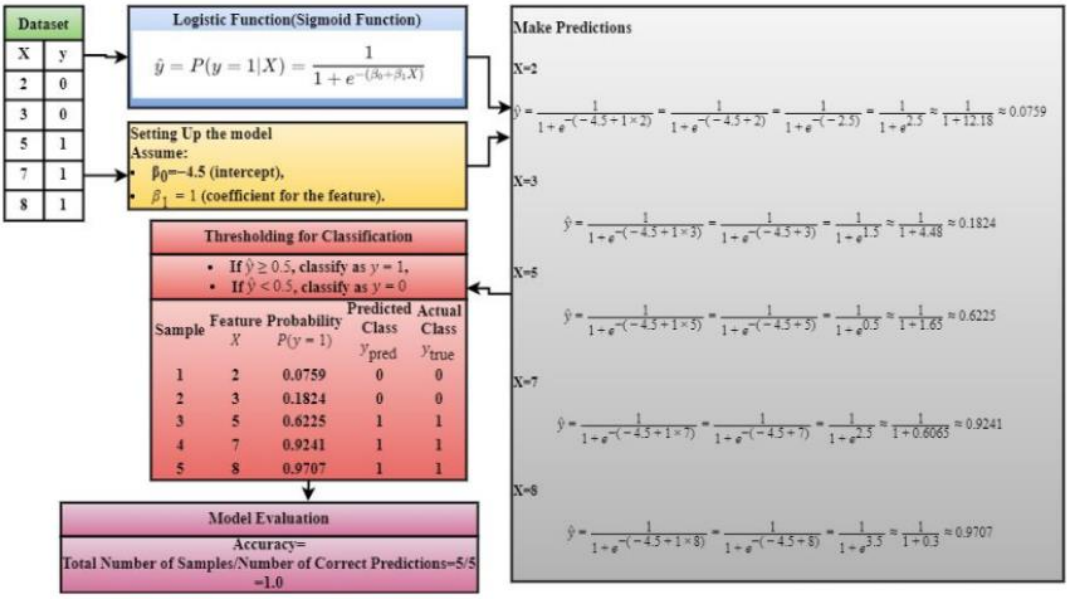


Figure 5: Working of Logistic Regression

In the PCA approach, the input data is transformed into 20 principal components. These components represent a linear combination of the original features, capturing the most variance in the data. Logistic Regression works on these transformed features, learning the relationships between the principal components and the target variable. Logistic Regression is used as one of the base models in stacking ensembles. It learns to predict the target variable(the class labels) based on either the top 20 features selected by SelectKBest in the first approach or the 20 principal components generated by PCA in the second approach. In the SelectKBest approach, the features are raw, domain-specific features chosen for their strong individual predictive power. Twenty principal components, which are linearly transformed versions of the initial attributes and indicate the major trends in the data but may not have an obvious meaning in terms of the original characteristics, were fed into the PCA method model. Logistic regression is a relatively simple model that performs well when the data is linearly separable. It provided a fast and reliable baseline for making predictions. The probabilities predicted by the logistic regression model (for both SelectKBest and PCA cases) are used as meta-features. These probabilities are combined with the outputs from the XGBoost and Random Forest models to form the input to the final LightGBM meta-model. Logistic regression's output helps LightGBM differentiate between classes by providing an additional viewpoint on the data. Table 4 presents the parameters that are tuned in logistic regression.

Table 4: Tuned Components of Logistic Regression

S.No	Name	Description	possible and Best Values
1	Solver	It finds the best co-efficients for eachfeature	Best:liblinear
2	Max Iterations	It controls the convergence speed	Any integer value Best:1000

3.2.4 Working of LightGBM: In this research, the LightGBM classifier serves as the meta-model in the stacking ensemble. It is trained on the data which is a combination of predictions from base models (XGBoost, Logistic Regression, and Random Forest) as new input features, rather than getting trained on the original features directly. The base models are trained on the original dataset, and their predictions are used as training and test sets. For each data point, the predictions of base models become the features for the training of the meta-model i.e. LightGBM. These are referred to as meta-features. So there are three new features in the meta-data which are the predicted probabilities of the base models stacked together as a feature vector. This feature vector is then fed into the LightGBM classifier, the meta-model. LightGBM finds the optimal combination of these meta-features to make the final prediction. LightGBM learns how to combine these best to make the final classification decision. It will build an ensemble of decision trees to minimize the prediction error(classification error) as it is a gradient-boosting algorithm. It learns the relationship between meta-features and true labels. After training it made final predictions on the test set. These predictions are more accurate than any of the individual base models as LightGBM can capture patterns or dependencies in the predictions that individual models might have missed. LightGBM is efficient in handling the large datasets. The meta-features are larger as they were generated from a large IIoT dataset. The dimensionality of meta-features in high and LightGBM helped in handling that and achieved good accuracy in both cases. The working of LightGBM is shown in figure 6

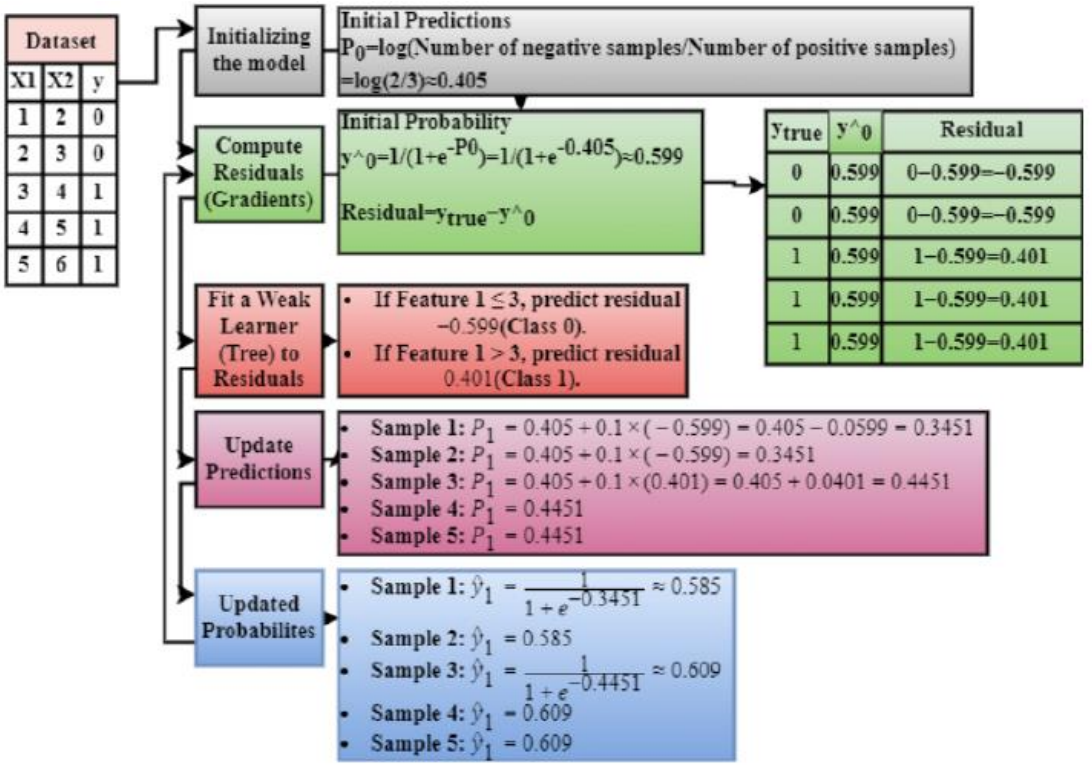


Figure 6: Working of Tuned LightGBM

The meta-features passed to the LightGBM model in the two scenarios vary not in terms of shape and size but in terms of values. The meta-model, LightGBM, receives the predictions from these base models, and these predictions are based on the transformed features from either PCA or SelectKBest. In the SelectKBest scenario, the base models were trained on a part of the data consisting of the most important original features. In the PCA scenario, the base models are trained on transformed features known as principal components. These principal components are derived from original features and their linear combinations. So, the meta-features for LightGBM are different in both cases because the base models are trained on different representations of the data, leading to other predictions. In case 1 where we used SelectKBest for selecting the 20 best features, the cross-validation scores remain perfect, it's crucial to be cautious when interpreting such results, as they might not hold up in real-work scenarios with slightly different data distributions. The PCA approach introduced a smaller number of errors, resulting in slightly better generalization compared to the previous approach. This resulted in a slight accuracy fall in the LightGBM classifier from the PCA approach.

Pseudocode for Classification with SeIectKBest, PCA, and Model Stacking:

1. Data Preprocessing:

- Load the dataset.
- Drop unnecessary columns: 'StartTime', 'LastTime', 'SrcAddr', and 'DstAddr'.
- Encode the 'Traffic' column using LabelEncoder.
- Separate the features X and target y.
- Split the dataset into training and testing sets:

$$X_{train,}X_{test,}Y_{train,}Y_{test} = \text{train\_test\_split}(X, y, \text{test}_{size} = 0.2)$$

2. Balancing the Data:

- Handle class imbalance using RandomoverSampler:

$$X_{resampled,}X_{resampled,} = \text{Random\_Over\_Sampler.fit\_resample}(X_{train,}Y_{train,})$$

3. Feature Selection and Dimensionality Reduction:

- SeIectKBest:
  - Apply SelectKBest to select the top 20 features.
  - Formula used in SelectKBest

$$X^2 = \Sigma\left(\frac{(O_i - Ei)^2}{E_i}\right)$$

Where O_i is the observed frequency, and E_i is the expected frequency.

- StandardScaler: Standardize features by removing the mean and scaling to unit variance:

$$z = \frac{x - \mu}{\sigma}$$

Where x is a feature value, μ is the mean, and σ is the standard deviation.

- PCA (Principal Component Analysis}:

- PCA reduces the dimensionality by identifying the directions (principal components) that maximize variance in the data.

- Covariance Matrix:

$$C = \frac{X^T \cdot X}{n - 1}$$

- Eigen Decomposition

Where v are the eigenvectors (principal components), and λ are the eigenvalues.

$$C \cdot \vartheta = \lambda \cdot \vartheta$$

- Transform data into the new coordinates (principal components):

$$X_{PCA} = X.V^T$$

V^T Where is the matrix of eigenvectors, and X is the original datamatrix.

4. Model Building:

- Initialize models: XGBoost, LogisticRegression, RandoinForestClassifier.

- Train base models on SelectKBest features

model.fit($X_{train\_selectKBest,}$ $y_{resampled}$)

- Train base models on PCA-reduced features

model.fit(X_{train_PCA}, $y_{resampled}$)

5. Model for Stacking:

- Initialize a meta-model (e.g., Logistic Regression) to combine the predictions of the base models.

- Gather predictions from the base models as new features for the meta-model X_{meta_train}=[predictions of base models]

- Train the meta-model on the gathered predictions

meta model. fit(X_{meta_train}, $y_{resampled}$)

6. Model Evaluation:

- Predict on the test set using the trained base models and meta-model Y_{pred}=meta_model.predict(X_{test})

- Evaluate model performance using accuracy, precision, recall, and F1-score.

4. Results & Discussions:

4.1 Dataset Overview

The WUSTL IIoT 2021 dataset is a comprehensive collection of traffic data from simulated industrial control systems, designed for evaluating intrusion detection systems (IDS) in IIoT environments. It includes both normal operations and various attack scenarios like Man-in-the- Middle (MitM), Denial of Service (DoS), and unauthorized access. Key features include Timestamp, Source IP, Destination IP, Protocol Type, and Packet Size. The dataset supports training machine learning models for IDS development by providing diverse classes of benign and malicious activities. It serves as a valuable resource for building effective detection algorithms for complex IIoT systems. Available on Kaggle.

4.2 Numerical Features Summary

The WUSTL IIoT 2021 dataset provides rich numerical features crucial for detecting anomalies in IIoT traffic. Key elements include source (Sport) and destination (Dport) ports, with a wide range of port utilization reflected by a mean source port value of ~54,452 and a standard deviation of 12,008. Packet transmission features show a higher average of source packets (166.56) compared to destination packets (16.88), typical of device-to-cloud communication. Source bytes (19,380) outweigh destination bytes (7,602), highlighting data asymmetry. Features like source load, application bytes, and jitter reveal significant variability, suggesting diverse traffic patterns. Connection durations and jitter values provide insight into device communication behavior, indicating possible network congestion or irregular activities

Table 5: Statistical Approach Analysis on the Dataset

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
Sport	1.194464e+06	5.445253e+04	1.200834e+04	0	5.221800e+04	5.663500e+04	6.104300e+04	2.765721e+06
Dport	1.194464e+06	7.907604e+02	3.299492e+03	0	502	502	502	6.552200e+04
SrePkts	1.194464e+06	1.665579e+02	5.266192e+04	0	10	10	10	2.773967e+07
DstPkts	1.194464e+06	1.688389e+01	1.137763e+03	0	8	8	8	3.092160e+05
TotPkts	1.194464e+06	1.756631e+02	5.266221e+04	0	18	18	18	2.773967e+07
SrcBytes	1.194464e+06	1.938043e+04	4.730229e+06	0	644	644	644	2.108646e+09
DstBytes	1.194464e+06	7.601579e+03	7.508699e+05	0	508	508	508	2.108646e+09
TotByes	1.194464e+06	2.779692e+05	1.918939e+07	0	1152	1152	1152	2.143725e+09
SrcLoad	1.194464e+06	1.571207e+07	8.339078e+07	0	8.514543e+04	8.818777e+04	8.968784e+04	1.156000e+09

SAppBytes	1.194464e+06	2.192527e+02	2.852114e+03	0	24	24	24	9.979300e+04
DAppBytes	1.194464e+06	7.051507e+03	7.445876e+05	0	20	20	20	8.182314e+07
TotAppBytes	1.194464e+06	6.581117e+05	4.167380e+07	0	44	44	44	4.293700e+09
RunTime	1.194464e+06	1.994698e-01	7.966454e-01	0	5.134900e-02	5.206700e-02	5.324900e-02	5.103213e+00
SrcJitAct	1.194464e+06	6.189383e+01	4.143742e+02	0	0	0	0	4.999440e+03
DstJitAct	1.194464e+06	2.653724e-01	5.001929e+00	0	0	0	0	7.695150e+02

4.3 Statistical Analysis (ANOVA Tests)

In Figure 7, ANOVA z as used to analyze the relationship between numerical features and the target variable (benign vs. malicious traffic). Features like total packets (TotPkts), total bytes (TotBytes). source load (SrcLoad), and connection duration (RunTime) showed significant p-values, indicating their discriminatory power in distinguishing between normal and malicious traffic. Malicious traffic often exhibits higher packet volumes and prolonged connection durations, potentially pointing to attack patterns like DDoS. Features like source jitter (SrcJitAct) and destination jitter (DstJitAct) also showed significant differences, suggesting that irregular packet timing can help detect network anomalies or congestion.

Fig 7: Hypothesis Testing for important features

4.4 Machine Learning Models: Stacking Ensemble Approach

Figure 8 represents an ensemble stacking strategy was used with K-Nearest Neighbors (KNN), Random Forest (RF), and Extreme Gradient Boosting (XGBoost) as foundational models and LightGBM as the meta-learner to improve anomaly identification in the WUSTL IIoT 2021 dataset. When SelectKBest was first employed for feature selection, it achieved 100% accuracy, which suggested that overfitting might have occurred. Principal Component

Analysis (PCA), which reduces dimensionality while maintaining important data features, took the place of SelectKBest to address this. Following retraining, the model demonstrated enhanced generalization and decreased overfitting, reaching a more realistic accuracy of 99%. This demonstrates how well PCA handles IIoT datasets in conjunction with ensemble stacking, improving anomaly identification without compromising accuracy.

```
# Base models in dictionary
base_models_kbest = {
    "XGBoost": xgb_kbest,
    "Logistic Regression": log_reg_kbest,
    "Random Forest": random_forest_kbest
}

base_models_pca = {
    "XGBoost": xgb_pca,
    "Logistic Regression": log_reg_pca,
    "Random Forest": random_forest_pca
}
```

Figure 8: Stacking Base Models

4.5 Base Models Performance (SelectKBest V/S PCA)

Principal Component Analysis (PCA) and SelectKBest were applied assess machine learning models like XGBoost, Logistic Regression, and Random Forest on the WUSTL IIoT 2021 dataset. Using SelectKBest, all models achieved perfect accuracy (1.0000) in both training and cross-validation, indicating significant overfitting. To mitigate this, PCA was introduced to reduce dimensionality while preserving key features. XGBoost and Random Forest with PCA showed slight decreases in accuracy (0.9999) but maintained strong cross-validation scores (1.0000), suggesting improved generalizability. Logistic Regression with PCA saw a more notable accuracy drop to 0.9700. While SelectKBest led to overfitting, PCA provided a more balanced approach, reducing overfitting and maintaining high performance, though with slightly lower accuracy.

```
SelectKBest - XGBoost Accuracy: 1.0000
SelectKBest - XGBoost Cross-Validation Accuracy: 1.0000

SelectKBest - Logistic Regression Accuracy: 1.0000
SelectKBest - Logistic Regression Cross-Validation Accuracy: 1.0000

SelectKBest - Random Forest Accuracy: 1.0000
SelectKBest - Random Forest Cross-Validation Accuracy: 1.0000

PCA - XGBoost Accuracy: 0.9999
PCA - XGBoost Cross-Validation Accuracy: 1.0000

PCA - Logistic Regression Accuracy: 0.9700
PCA - Logistic Regression Cross-Validation Accuracy: 0.9428

PCA - Random Forest Accuracy: 0.9999
PCA - Random Forest Cross-Validation Accuracy: 1.0000
```

Figure 9: Base Model Accuracies

4. 6 Meta-Model Performance (LightGBM)

The stacking ensemble's meta-model, LightGBM, merged predictions from XGBoost, Random forest, and Logistic Regression. It obtained perfect accuracy (1.0000) and AUC-ROC (1.0000) using SelectKBest; however, this suggested overfitting. PCA was included to increase generalizability; as a result. LightGBM's accuracy was somewhat decreased to 0.9999, but its cross-validation accuracy remained at 1.0000 and its AUC-ROC was nearly perfect at 0.9925. PCA maintained strong performance while reducing overfitting in spite of small misclassifications. LightGBM validated the efficacy of stacking ensembles by showing overall strong prediction power and generalizabiliiy across both approaches.

```
SelectKBest Accuracy Score: 1.0000

SelectKBest Confusion Matrix:
[[221556      1]
 [     0  17336]]

SelectKBest AUC-ROC Score: 1.0000
```

Fig 10a: LightGBM Scores for SelectKBest Approach

```
PCA Accuracy Score: 0.9999

PCA Confusion Matrix:
[[221548      9]
 [    18  17318]]

PCA AUC-ROC Score: 0.9995
```

Fig 10b: LightGBM Scores for PCA Approach

5. Conclusion

This research successfully demonstrates the effectiveness of a hypertuned machine learning model in addressing the growing security concerns within IIoT systems. By combining advanced machine learning techniques such as PCA for feature selection and an ensemble-based stacking approach, the proposed model significantly enhances detection accuracy while minimizing false positives and false negatives. The model‘s ability to process large-scale IIoT data in real-time makes it well-suited for industrial applications where timely threat detectionis critical to maintaining operational integrity and minimizing downtime. Additionally, the integration of algorithms such as XGBoost, Random Forest, and Logistic Regression, followed by meta-learning with LightGBM, provides a layered defense mechanism capable of detecting both known and unknown attack vectors. Despite the model's outstanding performance, particularly when tested on the WUSTL IIoT 2021 dataset, some limitations persist. for instance, while PCA and other dimensionality reduction techniques improved computational efficiency and addressed overfitting, the model's reliance on high-quality, labeled datasets still poses a challenge in real-world scenarios where data may be incomplete or noisy.

Furthermore, the issue of explainability in machine learning models for critical industrial systems remains an open area for further exploration. Decision-makers require transparency and insight into model predictions to ensure trust and regulatory compliance, which could be addressed through the integration of Explainable AI (XAI) techniques. In conclusion, the proposed hypertuned machine learning framework provides a scalable, adaptive, and highly accurate solution for securing IIoT environments against evolving cyber threats. Future research should focus on extending this approach to unsupervised or semi-supervised models to improve detection of zero-day attacks and incorporating XAI methodologies to enhance the transparency and trustworthiness of the system. By doing so, this research paves the way for more secure and resilient IIoT infrastructures, which are crucial for the continued digital transformation of industrial processes.

References

[1] Priya, V. &Thaseen, Sumaiya &Gadekallu, Thippa&Aboudaif, Mohamed &Abouel Nasr, Emad. (2021). Robust Attack Detection Approach for IIoT Using Ensemble Classifier. 10.48550/arXiv.2102.01515.

[2] Ferrag, Mohamed Amine &Friha, Othmane & Hamouda, Djallel& Maglaras, Leandros & Janicke, Helge.

(2022). Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. 10.36227/techrxiv.18857336.

[3] Alsaedi, Abdullah & Moustafa, Nour & Tari, Zahir & Mahmood, Abdun & Anwar, Adnan. (2020). TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. IEEE Access. 8. 10.1109/ACCESS.2020.3022862.

[4] Lala, Hany & Ahmed, Hoda. (2024). An impact on convolutional neural networks amelioration for early detection of skin cancer. International Journal of Intelligent Computing and Information Sciences. 24. 1-17. 10.21608/ijicis.2024.273450.1325.

[5] Alshehri, Mohammed & Saidani, Oumaima&Alrayes, Fatma & Abbasi, Saadullah & Ahmad, Jawad. (2024). A Self-Attention-Based Deep Convolutional Neural Networks for IIoT Networks Intrusion Detection. IEEE Access. PP. 1-1. 10.1109/ACCESS.2024.3380816.

[6] Idouglid, Lahcen &Tkatek, Said & El Fayq, Khalid &Guezzaz, Azidine. (2024). Next-gen security in IIoT: integrating intrusion detection systems with machine learning for industry 4.0 resilience. International Journal of Electrical and Computer Engineering (IJECE). ijece.v14i3.. 3512-3521. 10.11591/ijece.v14i3.pp3512-3521.

[7] Chuang, Hong-Yu & Chen, Ruey-Maw. (2024). Feature Selection for Malicious Detection on Industrial IoT Using Machine Learning. Sensors and Materials. 36. 1035. 10.18494/SAM4666.

[8] Gopalakrishnan, S. & Kumaran, M.. (2022). IIoT Framework Based ML Model to Improve Automobile Industry Product. Intelligent Automation & Soft Computing. 31. 1435-1449. 10.32604/iasc.2022.020660.

[9] Ellappan, Vijayan & Mahendran, Anand & Subramanian, Murali & Jotheeswaran, Jeevanandam&Khadidos, Adil &Khadidos, Alaa &Selvarajan, Shitharth. (2023). Sliding principal component and dynamic reward reinforcement learning based IIoT attack detection. Scientific Reports. 13. 10.1038/s41598-023-46746-0.

[10] Abdulkareem, Sulyman&Foh, Chuan &Carrez, Francois & Moessner, Klaus. (2024). A lightweight SEL for attack detection in IoT/IIoT networks. Journal of Network and Computer Applications. 230. 103980. 10.1016/j.jnca.2024.103980.

[11] Tiwari, Ravi & Lakshmi, Deepa & Das, Tapan & Tripathy, Asis & Li, Kuan-Ching. (2024). A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security. Telecommunication Systems. 1-20. 10.1007/s11235-024-01200-y.

[12] Golchha, Roopa & Joshi, Apoorv & Gupta, Govind. (2023). Voting-based Ensemble Learning approach for Cyber Attacks Detection in Industrial Internet of Things. Procedia Computer Science. 218. 1752-1759. 10.1016/j.procs.2023.01.153.

[13] Alwasel, Bader &Aldribi, Abdulaziz &Alreshoodi, Mohammed &Alsukayti, Ibrahim &Alsuhaibani, Mohammed. (2023). Leveraging Graph-Based Representations to Enhance Machine Learning Performance in IIoT Network Security and Attack Detection. Applied Sciences. 13. 7774. 10.3390/app13137774.

[14] Koppula, Manasa & I, Leo. (2023). LNKDSEA: Machine Learning Based IoT/IIoT Attack Detection Method. 655-662. 10.1109/ICAECIS58353.2023.10170095.

[15] Gaber, Tarek & Bamidele, Awotunde& Folorunso, Sakinat & Ajagbe, Sunday & Eldesouky, Esraa. (2023). Industrial Internet of Things Intrusion Detection Method Using Machine Learning and Optimization Techniques. Wireless Communications and Mobile Computing. 2023. 1-15. 10.1155/2023/3939895.

[16] Kumar, Pradeep & Banerjee, Indrajit. (2023). Attack and Anomaly Detection in IIoT Networks Using Machine Learning Techniques. 1-7. 10.1109/ICCCNT56998.2023.10308014.

[17] Ni, Chunchun& Li, Shan. (2024). Machine learning enabled Industrial IoT Security: Challenges, Trends and Solutions. Journal of Industrial Information Integration. 38. 100549. 10.1016/j.jii.2023.100549.

[18] Soliman, Sahar &Oudah, Wed &Aljuhani, Ahamed. (2023). Deep learning-based intrusion detection approach for securing industrial Internet of Things. Alexandria Engineering Journal. 81. 371-383. 10.1016/j.aej.2023.09.023.