Cloud Security: A Machine Learning Approach to Intrusion Detection

[1]Azmeera Santhosha [2]Dr. M. Dhanalakshmi
[1]*Student, Department of Information Technology, University College of Engineering Science and Technology, JNTUH Hyderabad*
[2]*Professor of IT & Deputy Director, Directorate of Innovative Learning and Teaching (DILT), University College of Engineering Science and Technology, JNTUH Hyderabad*

Abstract: Cloud computing provides on-demand access to a broad range of network and computer resources, encompassing storage, data management services, computing power, applications, and more. Users can easily access and utilize these resources as needed. The project focuses on enhancing cloud security by implementing an intrusion detection model leveraging machine learning techniques. The primary aim is to monitor and analyze resources, services, and networks within the cloud environment to effectively detect and prevent cyber-attacks. The proposed intrusion detection model utilizes machine learning techniques, specifically emphasizing the use of the Random Forest (RF) algorithm. Random Forest is a powerful ensemble learning method that combines multiple decision trees to make more accurate predictions. Feature engineering is a critical aspect of the model development process. It involves selecting and optimizing relevant features from the dataset to feed into the machine learning model. Effective feature engineering contributes to the model's ability to discern patterns and identify potential attacks accurately. The model's implementation is aimed at improving cloud security by continuously monitoring cloud resources, services, and networks. By applying machine learning algorithms, the model identifies unusual activities or patterns associated with cyber-attacks, thereby enhancing the overall security posture of the cloud infrastructure. The model's performance is evaluated and validated using two datasets: Bot-IoT and NSL-KDD. These datasets are common benchmarks in the field of intrusion detection. The model demonstrates high accuracy in detecting intrusions compared to recent related works, indicating its effectiveness and reliability in identifying potential security threats. The project's includes a Voting Classifier combination of RF + ADaBoost and Stacking Classifier with RF + MLP with LightGBM got 99% and 100% of accuracy for Kdd-Cup data respectively for enhanced cloud detection performance.

Index terms - cloud security; anomaly detection; features engineering; random forest.

1. INTRODUCTION

Cloud technologies allow practical access on demand to a shared network, storage, and resources and offer more choices regarding their service models [1]. These models are platform as a service (PaaS), software as a service (SaaS), and infrastructure as a service (IaaS)[2], used in one of the deployment models private, public, and hybrid cloud[3]. The cloud provides services with high performance due to its characteristics [2] according to the National Institute of Standards and Technology [4]: network access, resource pooling, quickelasticity, and measured service.

Recently, the cloud suffers from many security problems like availability, data confidentiality, integrity, and control authorization. In addition, the Internet is used to facilitate access to the services offered by the cloud representing a major source of threats that can infect the cloud systems and resources[2]. Then enhancing cloud security becomes a primary challenge for cloud providers[5]. Therefore, several approaches such as firewall tools, data encryption algorithms, authentication protocols, and others have been developed to better secure cloud environments from various attacks[6]. However, traditional systems are not sufficient to secure cloud services from different limits[7]. Therefore, a set of intrusion detection approaches are proposed and applied to detect and prevent undesirable activities in realtime[8, 9].

In general, the detection methods are divided into misuse detection method which uses known attacks to detect intrusion and anomaly detection method which

detect intrusion using unknown attack. The hybrid method is obtained by combining the advantages of these two methods [10]. Despite of more solutions given to secure cloud environments, the recent intrusion detection systems (IDSs) are affected by various significant limitations [8], for example, huge amounts of analyzed data, real-time detection, data quality, and others that aim to decrease the performance of detection models.

Nowadays, academic researchers show that intelligent learning methods [6, 11] such as machine learning (ML), deep learning (DL), and ensemble learning are useful in various areas [12, 13] and are able to perform network security [14–18]. Our main goal in this research work is to propose an anomaly detection approach based on random forest (RF) binary classifier and feature engineering is carried out based on a data visualization process aiming to reduce the number of used features and perform the proposed anomaly detection model. The evaluation performances of the model are implemented on NSL-KDD and BoT-IoT datasets. Then, the obtained outcomes demonstrate model performances.

2. LITERATURE SURVEY

The cloud computing exhibits, remarkable potential to provide cost effective, easy to manage, elastic, and powerful resources on the fly, over the Internet. The cloud computing, upsurges the capabilities of the hardware resources by optimal and shared utilization. The above mentioned features encourage the organizations and individual users to shift their applications and services to the cloud [1]. Even the critical infrastructure, for example, power generation and distribution plants are being migrated to the cloud computing paradigm. However, the services provided by third-party cloud service providers entail additional security threats. The migration of user's assets (data, applications etc.) outside the administrative control in a shared environment where numerous users are collocated escalates the security concerns. This survey details the security issues that arise due to the very nature of cloud computing. Moreover, the survey presents the recent solutions presented in the literature to counter the security issues. Furthermore, a brief view of security vulnerabilities in the mobile cloud computing are also highlighted [18,30]. In the end, the discussion on the open issues and future research directions is also presented.

The cloud computing provides on demand services over the Internet with the help of a large amount of virtual storage. The main features of cloud computing is that the user does not have any setup of expensive computing infrastructure and the cost of its services is less. In the recent years, cloud computing integrates with the industry and many other areas, which has been encouraging the researcher to research on new related technologies [2]. Due to the availability of its services & scalability for computing processes individual users and organizations transfer their application, data and services to the cloud storage server. Regardless of its advantages, the transformation of local computing to remote computing has brought many security issues and challenges for both consumer and provider. Many cloud services are provided by the trusted third party which arises new security threats. The cloud provider provides its services through the Internet and uses many web technologies that arise new security issues [1,23,5,7,19]. This paper discussed about the basic features of the cloud computing, security issues, threats and their solutions. Additionally, the paper describes several key topics related to the cloud, namely cloud architecture framework, service and deployment model, cloud technologies, cloud security concepts, threats, and attacks. The paper also discusses a lot of open research issues related to the cloud security.

3. METHODOLOGY

i) Proposed Work:

The Random Forest machine learning algorithm, known for its accuracy and robustness, is harnessed alongside strategic feature engineering. This combination is utilized to create a sophisticated intrusion detection system for cloud environments, aiming to substantially enhance security. The approach focuses on accurately identifying potential threats and abnormal patterns, contributing to an efficient and reliable solution that strengthens overall cloud security measures. And also included the combination of a Voting Classifier, incorporating Random Forest (RF) and ADaBoost, achieves an impressive 99% accuracy for the Kdd-Cup dataset. Additionally, the Stacking Classifier, integrating Random Forest (RF), Multi-Layer Perceptron (MLP),

and LightGBM, attains an outstanding 100% accuracy for the Bot-IoT dataset [28,29,39]. These ensemble models showcase the project's commitment to robust and high-performing intrusion detection in cloud environments. The user-friendly Flask framework with SQLite integration ensures practical usability, offering a seamless experience for user testing while maintaining data security in cybersecurity applications.

ii) System Architecture:

It begins with dataset exploration and data preprocessing, followed by the crucial steps of train-test split and model training. The core architecture involves the implementation of ensemble techniques, specifically the Stacking Classifier and the Voting Classifier extensions, designed to enhance the overall intrusion detection performance [24]. These classifiers demonstrate their efficacy through robust model evaluations, achieving notable accuracies of 99% and 100% respectively. The architecture prioritizes the versatility of the models, ensuring effective detection across diverse datasets, and emphasizes practical usability through a user-friendly interface facilitated by the Flask framework and SQLite integration. This unified system architecture positions the project as a sophisticated and adaptable solution for cloud-based intrusion detection using machine learning techniques.

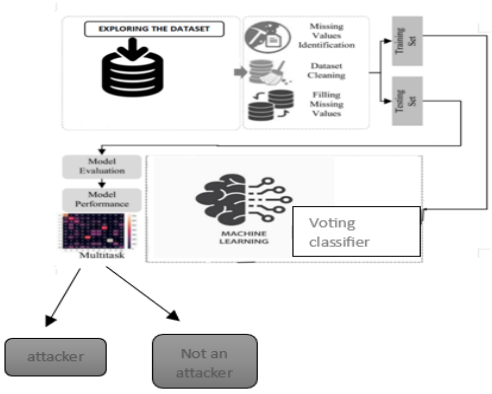


Fig 1 Proposed architecture

iii) Dataset collection:

KDD CUP DATASET

The KDD-CUP (Knowledge Discovery and Data Mining Cup) dataset [35,26] is a widely used dataset for intrusion detection system research. In the context of a cloud-based intrusion detection approach, the KDD-CUP dataset serves as a foundational dataset for training and evaluating machine learning models to detect intrusions and cyber-attacks. It allows the development of models that can analyze network traffic and detect abnormal or malicious patterns, crucial for securing cloud-based environments.

```
data = pd.read_csv("archive/kdd_train.csv")
data.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	ftp_data	SF	491	0	0	0	0	0
1	0	udp	other	SF	146	0	0	0	0	0
2	0	tcp	private	S0	0	0	0	0	0	0
3	0	tcp	http	SF	232	8153	0	0	0	0
4	0	tcp	http	SF	199	420	0	0	0	0

5 rows × 42 columns

Fig 2 KDD-CUP dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature selection:

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The main goal of feature selection is to improve the performance of a predictive model and reduce the computational cost of modeling. Feature selection, one of the main components of feature engineering, is the process of selecting the most important features to input in machine learning algorithms. Feature selection techniques are employed to reduce the number of input variables by eliminating redundant or irrelevant features and narrowing down the set of features to those most relevant to the machine learning model. The main benefits of performing feature selection in advance, rather than

letting the machine learning model figure out which features are most important.

vi) Algorithms:

Random Forest (RF), Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees. It's effective for intrusion detection due to its ability to handle a large number of features, deal with overfitting, and provide high accuracy.

Pseudocode: RandomForest

Input: Training data D = {(x1, y1), (x2, y2), ..., (xm, ym)}
Number of trees N
Number of features F to sample at each split
Maximum tree depth d (optional)

Output: A set of trained decision trees (Forest)

1. Initialize an empty list Forest.
2. For i = 1 to N do:
   a. Draw a bootstrap sample Di of size m from the training data D.
   b. Initialize a decision tree Ti.
   c. While the maximum depth d is not reached and stopping criteria are not met:
      i. Select F random features from the total p features.
      ii. For each feature in the selected F, calculate the best split based on a chosen criterion (e.g., Gini impurity, entropy, mean squared error).
      iii. Choose the best split point and split the node into child nodes.
      iv. Assign child nodes as leaves or continue splitting.
   d. Add the trained decision tree Ti to Forest.
3. End For
4. Return Forest.

Procedure: Predict

Input: A trained Forest, new instance x

1. Initialize an empty list predictions.
2. For each tree Ti in Forest:
   a. Use Ti to predict the output hi(x).
   b. Append hi(x) to predictions.
3. End For
4. If classification:
   a. Return the mode of the predictions as the final output y_hat.
   Else (if regression):
   a. Return the average of the predictions as the final output y_hat.

End Procedure

Decision Tree (DT)Decision Trees are a type of supervised learning model that makes decisions based on asking a series of questions related to the features in the dataset. It splits the data into subsets based on the feature values to create a tree-like structure, aiding in intrusion detection by understanding decision rules [28].

Pseudocode: DecisionTree

Input: Training data D = {(x1, y1), (x2, y2), ..., (xm, ym)}
Maximum tree depth d (optional)
Minimum samples per node n_min (optional)

Output: A trained decision tree

1. Define a function to calculate the best split:
   Function BestSplit(data, features):
   a. Initialize best_gini (or another criterion) to infinity and best_split to None.
   b. For each feature in features:
      i. For each unique value in the feature:

- Split the data into two groups based on the value.
- Calculate the impurity (e.g., Gini, entropy) for the split.
- If this split has a lower impurity than best_gini:
- Update best_gini and best_split with this split.

   c. Return best_split.
2. Define a function to build the tree:
   Function BuildTree(data, depth):
   a. If all data belong to the same class or depth == d or len(data) < n_min:
   - Return a leaf node with the majority class label (for classification) or mean value (for regression).
   b. Else:
   - Determine the best split using BestSplit(data, all features).
   - If no split can be found, return a leaf node as above
   - Split the data into two subsets: left_data and right_data.
   - Create a decision node with the split condition.
   - Recursively build the left and right subtrees:
   - Set the left child to BuildTree(left_data, depth + 1).
   - Set the right child to BuildTree(right_data, depth + 1).

3. Initialize the root of the tree:
Root = BuildTree(D, 0)
4. Return Root.
Procedure: Predict
Input: A trained tree Root, new instance x
1. Start at the Root node.
2. While the current node is not a leaf:
a. If x meets the split condition, move to the left child node.
b. Else, move to the right child node.
3. Return the prediction stored at the leaf node (class label for classification or mean value for regression).
End Procedure

Support Vector Machine (SVM): SVM is a powerful supervised learning algorithm used for classification tasks. It creates a hyperplane or a set of hyperplanes in a high-dimensional space to separate different classes. SVM is effective in intrusion detection for its ability to handle complex data relationships and non-linearity.
Pseudocode:
1. Initialize parameters:
- Choose a kernel function (linear, polynomial, radial basis function (RBF), etc.)
- Set regularization parameter `C` (trade-off between maximizing the margin and minimizing the classification error)
2. Preprocess data:
- Normalize or standardize the data (optional but recommended)
3. Define the decision function:
- For linear kernel: `f(x) = w · x + b`
- For other kernels: `f(x) = sum(alpha_i * y_i * K(x_i, x)) + b`
where `K(x_i, x)` is the kernel function
4. Define the objective function:
- Minimize `(1/2) * ||w||^2 + C * sum(max(0, 1 - y_i * (w · x_i + b)))`
For non-linear kernels, the objective function involves dual variables `alpha`:
- Maximize `sum(alpha_i) - (1/2) * sum(sum(alpha_i * alpha_j * y_i * y_j * K(x_i, x_j)))`
- Subject to `0 <= alpha_i <= C` and `sum(alpha_i * y_i) = 0`
5. Train the SVM model:
- Use an optimization algorithm (e.g., Sequential Minimal Optimization (SMO), gradient descent, etc.) to find the optimal weights `w` and bias `b` (or `alpha` for dual form)
6. Decision boundary:
- The decision boundary is defined by the support vectors (data points for which `y_i * (w · x_i + b) = 1` or `alpha_i > 0`)
7. Prediction:
- For a new data point `x`, predict the class label using the sign of `f(x)`
8. Post-process results:
- Evaluate the model's performance using appropriate metrics (accuracy, precision, recall, etc.)
9. Output: - The model outputs the predicted labels for new data points

Naive Bayes: Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that features are independent of each other, even though this assumption may not always hold. Naive Bayes is commonly used in intrusion detection due to its simplicity and speed, particularly with text-based data [28].
Pseudocode:
1. Initialize: - Let `X` be the set of training examples, where each example `x` consists of features (x1, x2, ..., xn) and a class label `y`. - Let `Y` be the set of all possible class labels. - Initialize class prior probabilities and conditional probabilities. 2. Calculate Class Prior Probabilities: - For each class `c` in `Y`: P(c) = (number of instances in class c) / (total number of instances) 3. Calculate Conditional Probabilities (Likelihood): - For each feature `xi` (for i = 1 to n) and each class `c` in `Y`: P(xi | c) = (number of instances in class c with feature xi) / (number of instances in class c) - For continuous features, calculate the probability density function (e.g., Gaussian distribution) instead. 4. Train the Model: - Store the calculated class prior probabilities and conditional probabilities. 5. Classification (Prediction): - For a new example `x` with features (x1, x2, ..., xn), calculate the posterior probability for each class `c` in `Y`: P(c | x) ∝ P(c) * P(x1 | c) * P(x2 | c) * ... * P(xn | c) - The class label `y` is assigned as: y = argmax_c P(c | x) 6. Output: - The model outputs the predicted class label for the new example.
Deep Learning (DL): Deep Learning involves neural networks with multiple layers (deep neural networks). DL models, like multi-layer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent

neural networks (RNNs), can learn complex patterns in the data, making them effective for intrusion detection where features might be intricate.
Pseudocode:
1. Initialize the Network: - Define the architecture of the neural network, including: - Number of layers - Number of neurons in each layer - Activation functions for each layer - Initialize weights and biases for each layer (commonly with small random values) 2. Forward Propagation: - For each layer in the network: - Compute the weighted sum of inputs plus bias: `z = W * x + b` - Apply the activation function: `a = activation_function(z)` - The output of the final layer produces the network's prediction. 3. Compute Loss: - Calculate the loss using a loss function (e.g., Mean Squared Error for regression, Cross-Entropy Loss for classification). - The loss function measures the difference between the network's prediction and the actual target values. 4. Backward Propagation (Backpropagation): - Compute the gradient of the loss with respect to each weight and bias using the chain rule of calculus. - For each layer in reverse order: - Calculate the error term for the current layer. - Compute the gradients for the weights and biases. 5. Update Weights and Biases: - Use an optimization algorithm (e.g., Gradient Descent, Adam, RMSprop) to update the weights and biases: - `W = W - learning_rate * gradient_W` - `b = b - learning_rate * gradient_b` 6. Iteration: - Repeat steps 2-5 for a predefined number of epochs or until convergence (i.e., until the loss stops decreasing significantly). 7. Model Evaluation: - After training, evaluate the model's performance on a separate validation/test dataset. - Use appropriate metrics (e.g., accuracy, precision, recall, F1 score) depending on the problem. 8. Prediction: - Use the trained model to make predictions on new data.

Long Short-Term Memory (LSTM): LSTM is a specialized type of recurrent neural network (RNN) designed to model sequences and time-dependent data. LSTM is valuable for intrusion detection, especially in handling sequences of events or network activities, allowing the model to capture long-term dependencies effectively [33].

Pseudocode:
1. Initialize the LSTM Network: - Define the architecture, including: - Number of LSTM units (neurons) in each layer - Number of LSTM layers (stacked LSTMs) - Define the activation functions and other hyperparameters - Initialize the weights and biases for each LSTM unit and layer 2. LSTM Cell Structure: - For each LSTM unit, define the following components: - Forget Gate: Controls the extent to which a value from the previous cell state is passed to the next cell state. - Input Gate: Controls the extent to which new information is added to the cell state. - Cell State: Maintains long-term memory. - Output Gate: Controls the extent to which the value in the cell state is used to compute the output. 3. Forward Pass (LSTM Cell): - For each time step `t` and each LSTM unit in the layer: - Forget Gate: `f_t = sigmoid(W_f * [h_{t-1}, x_t] + b_f)` - Input Gate: `i_t = sigmoid(W_i * [h_{t-1}, x_t] + b_i)` - Candidate Memory: `C~_t = tanh(W_C * [h_{t-1}, x_t] + b_C)` - Update Cell State: `C_t = f_t * C_{t-1} + i_t * C~_t` - Output Gate: `o_t = sigmoid(W_o * [h_{t-1}, x_t] + b_o)` - Hidden State: `h_t = o_t * tanh(C_t)` - Where: - `W` are the weight matrices - `b` are the bias vectors - `x_t` is the input at time step `t` - `h_{t-1}` is the previous hidden state - `C_{t-1}` is the previous cell state 4. Sequence Processing: - Repeat the forward pass for all time steps in the input sequence. 5. Compute Loss: - Use an appropriate loss function (e.g., Mean Squared Error for regression, Cross-Entropy Loss for classification) to measure the difference between the predicted output and the actual target. 6. Backward Pass (Backpropagation Through Time - BPTT): - Compute the gradients of the loss with respect to the weights, biases, and cell states. - Backpropagate the errors through the network across time steps. 7. Update Weights and Biases: - Use an optimization algorithm (e.g., Gradient Descent, Adam) to update the weights and biases: - `W = W - learning_rate * gradient_W` - `b = b - learning_rate * gradient_b` 8. Iteration: - Repeat steps 3-7 for all training data across epochs until convergence. 9. Prediction: - Use the trained LSTM network to make predictions on new sequences by feeding the initial states and inputs and iteratively producing outputs.

Stacking Classifier (RF + MLP with Light GBM)
The Stacking Classifier extension combines the predictive power of Random Forest (RF) and Multi-Layer Perceptron (MLP) with Light Gradient Boosting Machine (LightGBM). RF, an ensemble of decision trees, excels at capturing complex patterns, while MLP

with LightGBM introduces diverse learning techniques. The Stacking Classifier intelligently merges their outputs, leveraging the strengths of each base classifier to enhance overall intrusion detection performance, especially in cloud environments with diverse cyber threats.

Pseudocode:
1. Split Data:
- Split the training data into `K` folds for cross-validation.
2. Train Base Models:
- For each fold `k` (k = 1 to K):
a. Split the data into training set and validation set.
b. For each base model `m` in the list of base models:
i. Train model `m` on the training set.
ii. Make predictions on the validation set.
iii. Store the predictions as `P_mk`.
- After all folds:
a. Concatenate predictions from all folds to form a new dataset of predictions for each base model.
b. The new dataset will have features as the predictions from each base model and targets as the original targets.
3. Train Meta-Model:
- Use the new dataset (base model predictions) as input features and original targets as output labels.
- Train the meta-model (also called a blender) on this new dataset.
4. Make Predictions on Test Data:
- For each base model `m`:
a. Train model `m` on the entire training dataset.
b. Make predictions on the test data.
c. Store these predictions.
- Combine the predictions from all base models to form the input features for the meta-model.
- Use the meta-model to make final predictions based on these combined predictions.
5. Output:
- The final predictions from the meta-model are the output of the stacking classifier.

Voting Classifier (RF + AdaBoost)
The Voting Classifier extension integrates the capabilities of Random Forest (RF) and AdaBoost to create a robust intrusion detection model. RF excels in capturing intricate patterns through decision trees, while AdaBoost adapts by adjusting weights to prioritize the correct classification of previously misclassified instances. This combination ensures a strong ensemble model that leverages the strengths of both classifiers, achieving high accuracy and reliability in identifying potential intrusions in cloud-based systems. The versatility of this ensemble makes it well-suited for handling various types of cyber threats, contributing to the effectiveness of the overall intrusion detection approach in the project.

Pseudocode:
1. Initialize Base Models: - Initialize the Random Forest model (RF). - Initialize the AdaBoost model (AB). 2. Train Base Models: - Train the Random Forest model on the training data. - Train the AdaBoost model on the training data. 3. Make Predictions on New Data: - For a new data point, get predictions from both base models: a. `pred_RF = predict (RandomForest, new_data)` b. `pred_AB = predict(AdaBoost, new_data)` 4. Combine Predictions (Voting): - **Hard Voting (Majority Vote)**: a. For classification tasks, count the votes from each model's predicted class label. b. The final class is the one with the majority of votes. ```final_prediction = mode ([pred_RF, pred_AB])``` - **Soft Voting (Average Probabilities)**: a. For each class, average the predicted probabilities from each model. b. The final class is the one with the highest average probability. ```prob_RF = predict_proba (RandomForest, new_data) prob_AB = predict_proba(AdaBoost, new_data) avg_prob = (prob_RF + prob_AB) / 2 final_prediction = class with highest avg_prob ``` 5. Output: - The final prediction from the voting classifier is the output.

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:
Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

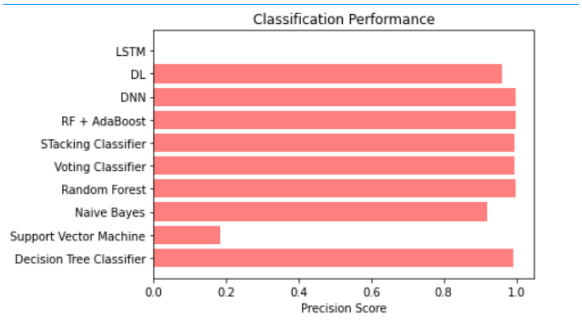


Fig 6 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

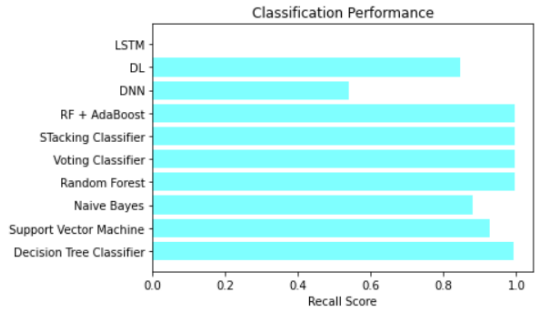


Fig 7 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

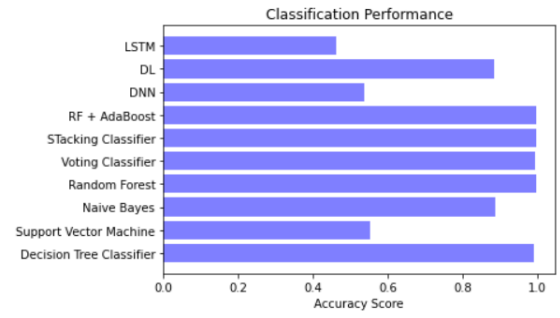


Fig 8 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

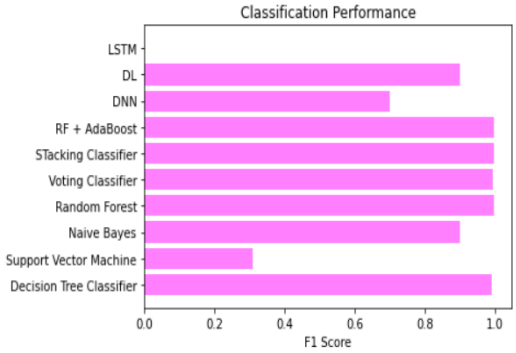
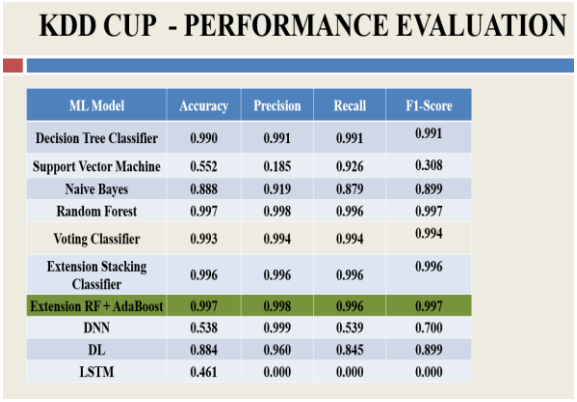


Fig9F1Score

5. User Interface



KDD CUP - PERFORMANCE EVALUATION

ML Model	Accuracy	Precision	Recall	F1-Score
Decision Tree Classifier	0.990	0.991	0.991	0.991
Support Vector Machine	0.552	0.185	0.926	0.308
Naive Bayes	0.888	0.919	0.879	0.899
Random Forest	0.997	0.998	0.996	0.997
Voting Classifier	0.993	0.994	0.994	0.994
Extension Stacking Classifier	0.996	0.996	0.996	0.996
Extension RF + AdaBoost	0.997	0.998	0.996	0.997
DNN	0.538	0.999	0.539	0.700
DL	0.884	0.960	0.845	0.899
LSTM	0.461	0.000	0.000	0.000

Fig 10 Performance Evaluation

Fig 11 Home page

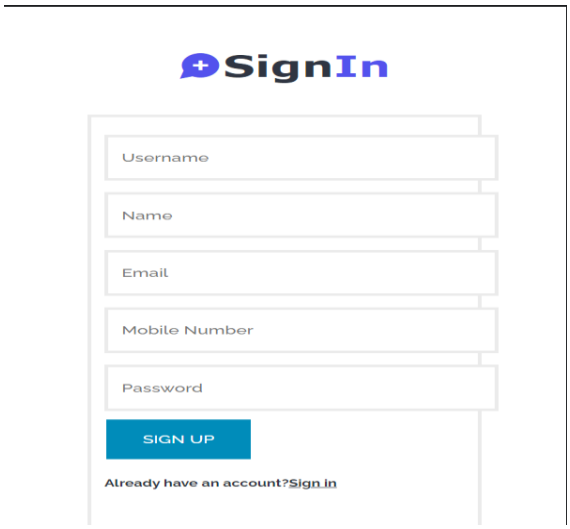


Fig 12 Signin page

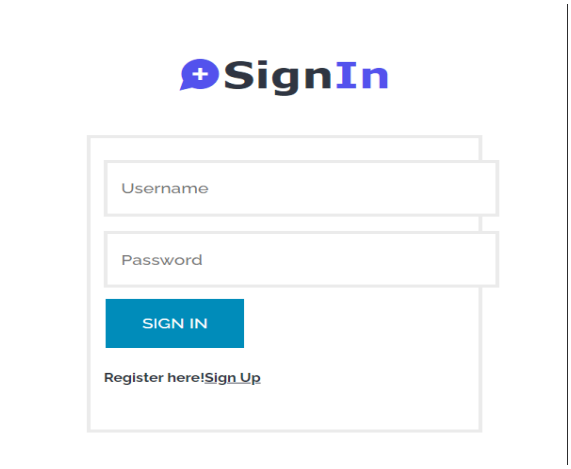


Fig 13 Login page

Fig 14 User input

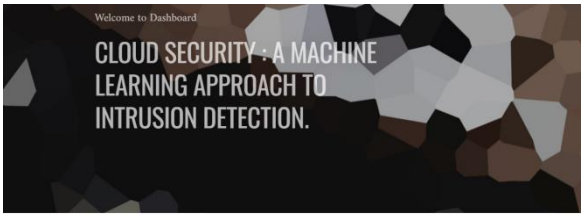


Fig 15 Predict result for given input

5. CONCLUSION

The intrusion detection model built for the cloud, employing voting classifier and feature engineering, excels in accuracy, precision, and recall. It demonstrates superior performance in detecting abnormal activities within the cloud environment compared to recent works. This highlights the effectiveness and reliability of the proposed approach. Random Forest (RF) [26,29] is a pivotal component of the model, contributing to its success. RF is effective in handling outlier data, providing robustness in abnormal activity detection. Its simplicity in parameter establishment and automatic creation of variable importance and accuracy metrics make it an efficient choice, enhancing the overall performance of the intrusion detection model. The project extends accuracy through ensemble techniques like Voting Classifier. Integration of a user-friendly Flask interface with secure authentication improves the testing experience, emphasizing practical usability in cybersecurity applications.

6. FUTURE SCOPE

Future work aims to enhance the recall rate, especially using the NSL-KDD dataset, by integrating deep learning (DL) and ensemble learning techniques [27]. Deep learning models can capture complex patterns, potentially improving the system's ability to detect intrusions. Ensemble techniques, on the other hand, combine multiple models to boost prediction accuracy, further enhancing the overall performance of the intrusion detection system. Future systems will focus on understanding user and system behavior through behavioral analysis. This approach is crucial for accurate anomaly detection, enabling the identification

of abnormal patterns and potential security threats. Analyzing behaviors helps in creating a baseline for normal activities, making it easier to detect deviations that could signify security breaches. The research will strive to develop intrusion detection systems capable of efficiently scaling with the growing complexity and volume of cloud data. Optimizing resources for efficient performance and cost-effectiveness will be a priority, ensuring the system can handle the increased data load and adapt to evolving cloud infrastructures while maintaining cost-efficiency. Ensemble learning techniques will be leveraged to combine multiple models, harnessing their collective strength to make more accurate predictions. By integrating ensemble learning, the intrusion detection system can enhance its overall performance, achieving higher accuracy and reliability in identifying potential security threats in the cloud.

7. REFERENCES

[1] M. Ali, S. U. Khan, and A. V. Vasilakos, Security in cloud computing: Opportunities and challenges, Information Sciences, vol. 35, pp. 357–383, 2015.

[2] A. Singh and K. Chatterjee, Cloud security issues and challenges: A survey, Journal of Network and ComputerApplications, vol. 79, pp. 88–115, 2017.

[3] P. S. Gowr and N. Kumar, Cloud computing security: A survey, International Journal of Engineering and Technology, vol. 7, no. 2, pp. 355–357, 2018.

[4] A. Verma and S. Kaushal, Cloud computing security issues and challenges: A survey, in Proc. First International Conference on Advances in Computing and Communications, Kochi, India, 2011, pp. 445–454.

[5] H. Alloussi, F. Laila, and A. Sekkaki, L'etat de l'art de la ´ securit ´ e dans le cloud computing: Probl ´ emes et solutions ` de la securit ´ e en cloud computing, presented at Workshop ´ on Innovation and New Trends in Information Systems, Mohamadia, Maroc, 2012.

[6] J. Gu, L. Wang, H. Wang, and S. Wang, A novel approach to intrusion detection using SVM ensemble with feature augmentation, Computers and Security, vol. 86, pp. 53–62, 2019.

[7] Z. Chiba, N. Abghour, K. Moussaid, A. E. Omri, and M. Rida, A cooperative and hybrid network intrusion detection framework in cloud computing based snort and optimized back propagation neural network, Procedia Computer Science, vol. 83, pp. 1200–1206, 2016.

[8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, Survey of intrusion detection systems: Techniques, datasets and challenges, Cybersecurity, vol. 2, p. 20, 2019.

[9] A. Guezzaz, A. Asimi, Y. Asimi, Z. Tbatou, and Y. Sadqi, A global intrusion detection system using PcapSockS sniffer and multilayer perceptron classifier, International Journal of Network Security, vol. 21, no. 3, pp. 438–450, 2019.

[10] A. Guezzaz, S. Benkirane, M. Azrour, and S. Khurram, A reliable network intrusion detection approach using decision tree with enhanced data quality, Security and Communication Networks, vol. 2021, p. 1230593, 2021.

[11] B. A. Tama and K. H. Rhee, HFSTE: Hybrid feature selections and tree-based classifiers ensemble for intrusion detection system, IEICE Trans. Inf. Syst., vol. E100.D, no. 8, pp. 1729–1737, 2017.

[12] M. Azrour, J. Mabrouki, G. Fattah, A. Guezzaz, and F. Aziz, Machine learning algorithms for efficient water quality prediction, Modeling Earth Systems and Environment, vol. 8, pp. 2793–2801, 2022.

[13] M. Azrour, Y. Farhaoui, M. Ouanan, and A. Guezzaz, SPIT detection in telephony over IP using K-means algorithm, Procedia Computer Science, vol. 148, pp. 542–551, 2019.

[14] M. Azrour, M. Ouanan, Y. Farhaoui, and A. Guezzaz, Security analysis of Ye et al. authentication protocol for internet of things, in Proc. International Conference on Big Data and Smart Digital Environment, Casablanca, Morocco, 2018, pp. 67–74.