International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

An Optimized Deep Learning Framework for Malware Classification Using Integrated LSTM and CNN Approaches

Michael O. Lawanson[1,4], Nehemiah Ibitoye[2], Olugbebi Muyiwa[3], Oluwatomiwa Ajiferuke[1]Oluwakemi Temitope Olayinka[4] , Divine Iloh[4]Emmanuella Patrick Egbon[4]

[1] *Department of Information Science, University of Arkansas at Little Rock, United State of America(USA)*
[2] *Department of Industrial and Production Engineering, University of Ibadan, Nigeria*
[3] *Department of Mechanical Engineering, Ladoke Akintola University of Technology, Nigeria*
[4] *Business Information and System Analytics, University of Arkansas at Little Rock, United State of America (USA)*

Date of Submission: 15-07-2025 Date of Acceptance: 25-07-2025

ABSTRACT
The swift expansion of computer networks has resulted in security challenges, particularly with the abuse of encryption technologies. Malicious individuals utilize encryption and Tor to evade detection systems, facilitating unlawful actions. Conventional approaches such as signature-based scanning and heuristic analysis are insufficient against advanced obfuscation strategies. This paper presents a deep learning framework that integrates Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM) networks to enhance the accuracy and reliability of virus detection. The framework additionally integrates sophisticated preprocessing methods such as normalization and feature selection.

Drawing on earlier successes, this method expands on previous work where biLSTM models achieved high accuracy in cloud-based malware detection and where LSTM-CNN approaches proved effective for insider threat detection. The proposed model advances these techniques by incorporating a filter-based feature selection mechanism (Kbest) into an LSTM-CNN architecture, optimizing it for the evolving complexities of modern malware detection.The rapid proliferation of computer networks has led to security challenges, especially with the misuse of encryption technologies. Malicious actors employ encryption and Tor to circumvent detection measures, enabling illicit activities. Traditional methods like signature-based scanning and heuristic analysis are inadequate against sophisticated obfuscation techniques. This research proposes an intelligent detection system that leverages the combined strengths of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to boost the accuracy and robustness of virus identification. The system is further enhanced by sophisticated preprocessing steps, including data normalization and feature selection.The study highlights the need for innovative cybersecurity solutions, focusing on improving malware detection capabilities to create more resilient network security infrastructures. Its findings have significant implications for future security system design and ongoing cybercrime battle.

Keywords: Malware classification, Deep learning, LSTM, CNN, Feature selection, Chi-Square, Adversarial attacks, Cybersecurity, IoT security, Obfuscation detection.

I. INTRODUCTION

In recent years, computer networks have become essential for growth, and network security concerns have garnered significant attention. Encryption technology safeguards privacy, liberty, and anonymity, it also enables adversaries to circumvent anomaly detection systems[1].An assailant encrypts malicious communication and use technologies such as Tor to access the darknet, facilitating illicit commerce and system intrusions by criminals[2].The improper utilization of encryption technology poses numerous challenges to network anomaly detection and security

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

management. As a result, the identification of malware-encrypted communication has elicited considerable concern in both academic and industrial domains. In the contemporary era of computing, artificial intelligence (AI) stands as a compelling and significant topic for the advancement of intelligent systems that can autonomously make real-time judgments[3].

Malware is software that infiltrates a secured system without the user's consent or knowledge[4]. Record infectors or independent software are software applications or packages that threaten system security. With the increasing use of computers in society, the detection of malicious software becomes a cybersecurity challenge. Currently, a single malware incident may incur costs up to millions, highlighting the inadequacy of existing prevention measures against sophisticated, infection-resistant threats.Contemporary antivirus software utilizes flags, which are tangible characteristics that identify certain categories of infections. These criteria are often explicit and fail to effectively identify novel malware, even if it utilizes a similar functionality[5].

With millions of new malware tests revealed daily, this technique is insufficient since most conditions include unique combinations that have never been spotted before. Conventional malware detection techniques, such as signature-based surveillance and heuristic analysis, often struggle to keep up with the rapidly evolving malware landscape[6]. To address this, a method was developed to classify packed executables, enhancing the accuracy of computer virus detection.

The main limitation of this research was its exclusive emphasis on packed executables, which represent just one facet of malware obfuscation strategies. Other strategies, like polymorphism and metamorphism, were not discussed.

Variants of Mirai virus, including Satori and Miori, continue to inundate networks, setting new records for traffic volume directed at victims, including business systems.

In reality, the attacker often manages several botnets composed of compromised Internet of Things (IoT) devices to execute such an assault. Consequently, early detection of malware traffic during the dissemination of harmful code may substantially aid in preventing the proliferation of malware and reducing the severity of the assault[7].

Moreover, contemporary virus assaults are often enabled via the Internet. As the number of Internet-connected devices increases, it is imperative to protect our devices in order to prevent the theft of personal or private information[8]. Over the past decade, malware has expanded at an alarming rate, and there is currently no reliable method for identifying all types of malwares. In order to circumvent detection systems, a novel form of malware implements sophisticated obfuscation and packaging techniques. This renders the detection of intricate malware using conventional methods exceedingly challenging[9].

Advanced detection algorithms that are capable of identifying both known and unforeseen hazards are urgently required. Convolutional Neural Networks (CNN), Neural Networks (NN), and hybrid models are among the deep learning methodologies that have emerged as viable solutions. The growing number of cyber incidents highlights the urgent need for malware detection systems that are flexible, effective, and robust, given the potentially severe impact these attacks can have on both individuals and organizations[9].

In response to these concerns, the objective of this project is to create and assess a deep learning framework that integrates LSTM and CNN methodologies for the classification of malevolent malware. This research aims to improve the reliability and accuracy of malware detection systems by employing advanced preprocessing techniques, such as feature selection and normalization.

II. LITERATURE REVIEW

Mishra et al. [10]employed a biLSTM model to classify cloud-based malware, utilizing a CNN layer trained on sequences of system calls. Their approach achieved an accuracy of approximately 90%.The scientists discovered that substituting the LSTM layer with a biLSTM layer consistently resulted in decreased accuracy. Renjie [11]employed opcodes from disassembled executables and word embedding for feature engineering. The resultant word embedding is input into an LSTM layer, attaining an average AUC of 0.99 for malware detection and 0.987 for classification.

Using the CERT insider threat v4.2 datasets, Yuan et.al.,[12] used LSTM and CNN techniques to identify insider hazards. The model analyzes user activity, isolates temporal characteristics, generates feature vectors, converts them into fixed-size matrices, and classifies them as anomalous or normal and the model attains a 94.49% AUC.Agarap[13]suggested CNN hybrid networks that work with SVM and other SVM hybrid designs that use deep learning models. The

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

CNN-SVM model is 77.22% accurate, the GRU-SVM model is 84.92% accurate, and the MLP-SVM mixed model is 80.46% accurate. A unified model was proposed that integrates CNN, LSTM, and image processing techniques into a single framework[4].

The CNN is made up of two layers: an LSTM layer with 70 memory blocks and a fully connected network layer with 25 units. It uses softmax with category cross-entropy and the accuracy of the final split ranged from 96.64% to 96.68%. In their model, Akarsh et al., [14]implemented a cost-sensitive approach incorporating 70 LSTM memory blocks and a 0.1% dropout rate, using LSTM for feature extraction. This configuration achieved a reported accuracy of 95.5%.A non-domination-based genetic algorithm for multi-objective optimization (NSGA) was introduced with the objective of achieving data equilibrium[15]. The single-objective method achieved 96.1% accuracy, the genetic algorithm without equalization achieved 92.1% accuracy, and the multi-objective algorithm achieved 97.1% accuracy. An ensemble model is developed by utilizing convolutional neural networks (CNNs) in conjunction with extreme learning machines (ELMs)[16]. Their accuracy was 96.30% with a single CNN layer and 95.7% with two CNN layers.The dataset was partitioned through static analysis, and a support vector machine (SVM) classifier was constructed based on the extracted features. The classification method demonstrates strong detection capabilities for viruses and worms but exhibits reduced accuracy in identifying trojans and indirect access attempts[17].

III. METHODOLODY

3.1 Research Approach

A comprehensive framework that specifies the methodologies and strategies employed to collect and analyze data for a study is known as the research design. It incorporates economic methodologies with significance to the study objective. The objective of this study was to employ deep learning techniques to classify adversarial malware attacks. In order to accomplish the designated objectives, the experimental frameworks implemented the CRISP-DM methodology. The framework used a systematic methodology for organizing a data mining project aimed at developing a deep learning model for identifying assaults in hostile datasets. The stages in processing the deep learning model include subject matter comprehension, data collecting, data preparation, data transformation, modeling, performance assessment, and deployment. Upon achieving a high-accuracy model, it proceeds to the deployment phase. The model undergoes periodic reviews for optimization to enhance outcomes. The research design functions as a strategic framework for executing and overseeing research activities.

3.2 Utilizing the CRISP-DM methodology for model development

For the Deep Learning Algorithm employing the Long Short-Term Memory (LSTM) and Convolution Neural Network (CNN) the following is a summary of the stepwise approach to accomplishing the stated objectives of this research:

3.2.1 Domain Understanding

The A meticulous examination of adversarial malware, its features, and the domain of deep learning was conducted to enhance comprehension of the prerequisites necessary to achieve the goals.

3.2.2 Dataset Acquisition and Understanding

The dataset utilized was sourced from the Kaggle data repository through the following universal resource locator: https://www.kaggle.com/datasets/cnrieiit/adversarial-machine-learning-dataset. This section provides a description and other details of the dataset.

I. Description

The following three applications were examined: DNS tunneling, platooning, and remaining useful life (RUL). These applications pertain to network data that is captured during assaults, vehicle platooning scenarios, and predictive maintenance of aircraft engines.

Data structure

Two categories are used to organize each application scenario (DNS tunneling, platooning, RUL):

i. legitimate, including original data malevolent, encompassing data compromised by Adversarial Machine Learning techniques; in this instance, both training and test data are shown, together with a consolidated dataset.

The target variable for detecting hostile attacks is consistently referred to as "attack," while the target variables in the original scenarios (legitimate data) are as follows:

i. g for the DNS tunneling application
ii. collision for the platooning application

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

RUL_binary for the RUL application

a. Tunneling data structure

Table 3.1 lists the features of the DNS tunnelling application.

Table 3.1 Features of DNS tunneling applications.

Attribute	Description
mDt	Mean inter-packet time interval
mA	Mean answer packet size
mQ	Mean query packet size
vDt	Variance of inter-packet time interval
vA	Variance of answer packet size
vQ	Variance of query packet size
sDt	Skewness of inter-packet time interval
sA	Skewness of answer packet size
sQ	Skewness of query packet size
kDt	Kurtosis of inter-packet time interval
kA	Kurtosis of answer packet size
kQ	Kurtosis of query packet size

b. Platooning data structure

Regarding the platooning application, the subsequent features are enumerated in table 3.2 below:

Table 3.2 platooning application features.

Attribute	Description
N	Number of vehicles in the platoon
F0	Braking force applied by the leader
PER	Packet error rate (probability of packet loss)
d0	Initial distance between vehicles
v0	Initial speed between vehicles

c. RUL data structure

Regarding the RUL application, the subsequent attributes are taken into account:

Table 3.3 RUL application features.

Attribute	Description
s_os2	Skewness of the operational settings indicator 2
m_Nc	Mean physical core speed
v_Nc	Variance of physical core speed
v_phi	Variance of the ratio of fuel flow to Ps30 (static pressure at HPC outlet)
m_htBleed	Mean bleed enthalpy
s_htBleed	Skewness of bleed enthalpy
m_W31	Mean High-Pressure Turbine coolant bleed

Data generated with adversarial-robustness-toolbox - Framework to generate Adversarial ML attacks.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

3.2.3 Dataset Preparation

Data preparation involves several pre-processing phases, including cleansing, filtering, transformation, feature extraction, and feature selection.

3.2.3.1 Data Preprocessing

This project aims to optimize results by pre-processing data, eliminating mistakes and outliers, and filtering out inconsistent sets to enhance the dataset's efficacy and maximize outcomes, thereby readying it for modeling.

3.2.4 Normalization/Scaling

Information data scaling refers to the process of diminishing the magnitude of particular attributes through normalization and standardization methods. The Standard Scaler standardization approach was utilized to improve the dataset's performance when applied to the Recurrent Neural Network model.

$x' = x - \mu/\sigma$ (3.1)

3.2.5 Feature Selection

The Chi Square Statistical approach was employed to choose the most important features for predicting a target variable to reduce over-fitting, improve accuracy, and save training time.

3.2.6 Chi Square

The chi-square test is a statistical technique employed to ascertain the independence of two variables. It resembles the R2 coefficient of determination but is restricted to nominal or categorical data. It computes the chi-square statistics for each feature variable and the target variable to ascertain the relationship between the variables and the target. If the feature variable is independent, it is selected to ascertain dependency.

Employing the calculation for the Chi-Square test:

$$\chi^2 = \frac{1}{d}\sum_{k=1}^{n}\frac{(O_k - E_k)^2}{E_k} \quad (3.2)$$

We have

$$\chi^2 = \frac{(A-E_A)^2}{E_A} + \frac{(B-E_B)^2}{E_B} + \frac{(C-E_C)^2}{E_C} + \frac{(D-E_D)^2}{E_D} \quad (3.3)$$

After Simple calculation we have

$$\chi^2 = \frac{N(AD-BC)^2}{(A+C)(B+D)(A+B)(C+D)} \quad (3.4)$$

Given

B = M – A

C = P – A

D = N – M – (P – A)

We have:

$$\chi^2 = \frac{N(AN-MP)^2}{PM(N-P)(N-M))} \quad (3.5)$$

From a dataset, we may readily acquire:

The overall count of positive instances of feature X, represented by A, equals M, whereby P signifies the total positive instances, and N denotes the total number of occurrences, which remains invariant across all features.

3.2.7 Division of Dataset

The train-test split approach is utilized to assess the efficacy of a deep learning algorithm. The training set comprises the data utilized to instruct the model, from which it subsequently observes and learns. The testing set comprises the data evaluated in the final model, juxtaposed with the original data sets. The dataset is partitioned with a 75% to 25% ratio.

3.2.8 Data Modelling using the Deep Learning

The Long Short-Term Memory (LSTM) unit of the Recurrent Neural Network was utilized in the modelling of an adversarial malware dataset, which typically involves training and memorizing patterns from the training dataset.

i. The Long short-term memory (LSTM) network

The Long short-term memory (LSTM) network architectures, a specialized form of Recurrent Neural Networks (RNN), are utilized in designing IDS due to their ability to persist information or cell states for future network use.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

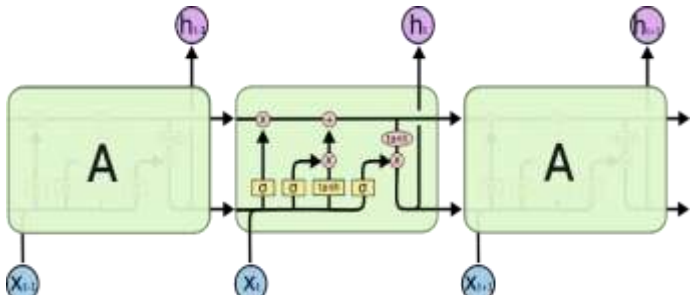

Fig 3.1 Illustration of LSTM algorithm.

ii. The Logic Behind LSTM

The LSTM model utilizes a memory cell known as "cell state" that preserves its state across time, akin to a conveyor belt that transmits data unchanged.

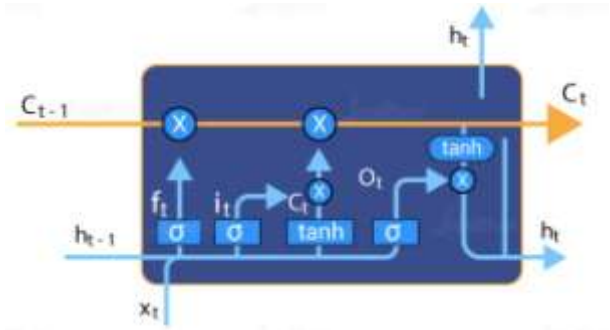

Figure 3.2 LSTM Logic (https://intellipaat.com/)

LSTMs employ gates to manage the incorporation and elimination of input, leveraging a sigmoid neural network layer and pointwise multiplication for effective cell state regulation.

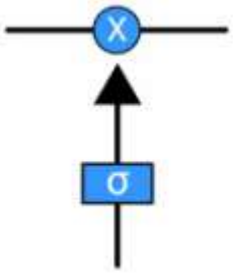

Figure 3.3 Sigmoid Layer (https://intellipaat.com/)

The sigmoid layer produces a value ranging from zero to one, where zero signifies no data transmission and one signifies complete data transmission.

3.2.9 Performance Evaluation

The study assessed the efficacy of a deep learning model utilizing machine learning statistical metrics, including true positive rate, false negative rate, false positive rate, classification accuracy, precision, recall, ROC, and error rate, while comparing the model with and without a feature selection algorithm.

i. Modelling Case Scenario

Case 1: Normalization + Reduced Adversarial Malware Dataset (obtained with Chi Square) + LSTM Technique: The first case scenario passes through data cleaning, standardization, feature engineering selection with chi-square, and LSTM deep learning for classification.

Case 2: Normalized Adversarial Malware Dataset + LSTM Technique: Data cleansing, data standardization, and LSTM-based deep learning categorization are all part of the second case scenario.

Case 3: Normalization + Reduced Adversarial Malware Dataset (obtained with Chi Square) + CNN Technique: The third scenario encompasses data cleaning, normalization, feature engineering utilizing chi-square, and classification via CNN deep learning.

Case 4: Normalized Adversarial Malware Dataset + CNN Technique: The fourth case scenario employs data cleansing, standardization, and CNN deep learning categorization.

ii. Classification Accuracy

Equation 3.6 illustrates the classification accuracy.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

Classification Accuracy (%)

$$= \frac{TP+TN}{TP+TN+FP+FN} \times 100 \quad (3.6)$$

iii. Error Rate (EER)

Error rate can be calculated as the total number of incorrect predictions made on the test set divided by all predictions made on the test set.

$$\text{Error Rate} = \frac{\text{Incorrect Predictions}}{\text{Total Predictions}} \quad (3.7)$$

iv. Sensitivity

In the data, sensitivity is defined as the ratio of true positives to total (actual) positives.

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (3.8)$$

v. **Specificity**

Specificity is the Ratio of true negatives to total negatives in the data.

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (3.9)$$

3.3 Architectural Framework

A framework for architectural artifacts contains the very minimum norms and requirements. See figure 3.4 for the architectural structure.

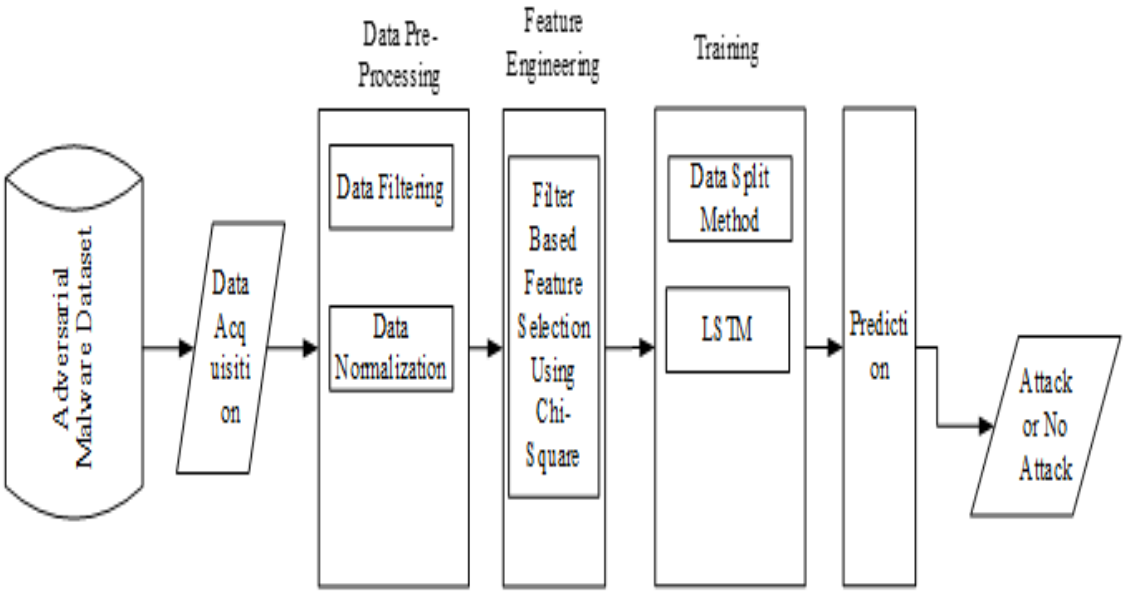

Figure 3.4 Architectural Framework

3.4 System Flowchart

The system algorithm flowchart is depicted below in figure 3.4.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

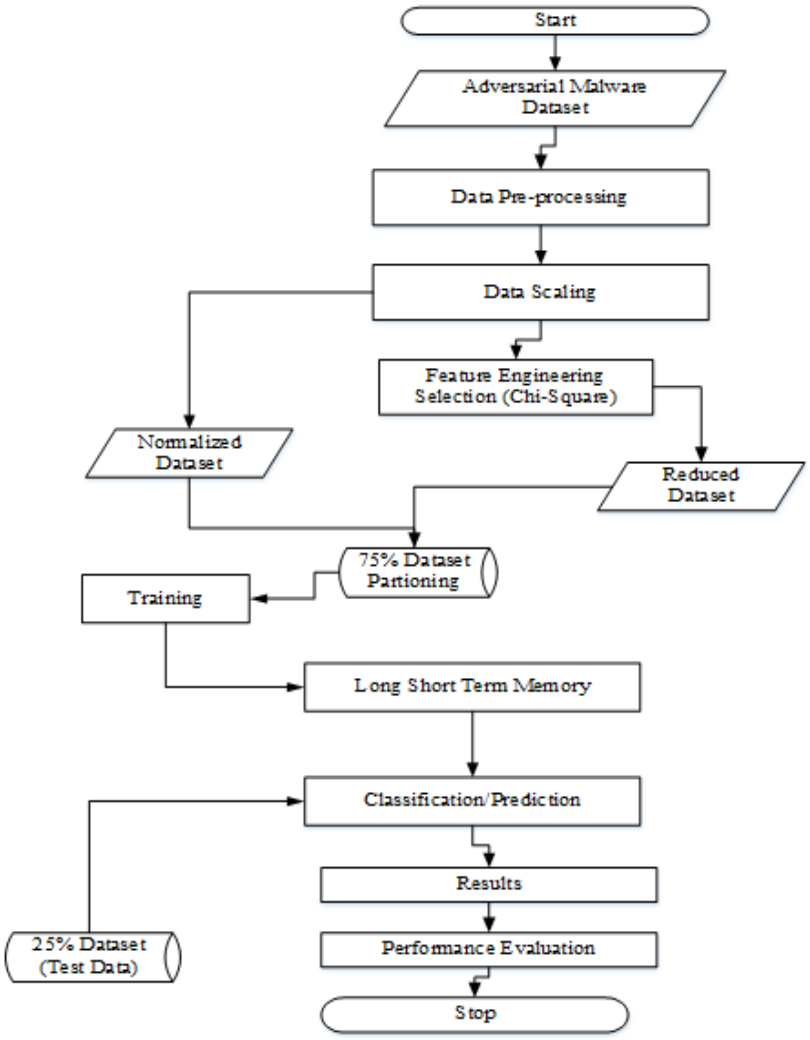

Figure 3.5 System Flowchart

IV. RESULT AND DISCUSSIONS

The investigation evaluates the application of a deep learning algorithm that integrates Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) to identify adversarial malware attacks. The experiment investigates four case scenarios by integrating data mining techniques such as normalization, feature selection, and deep learning classification. The system implemented a progressive data mining methodology, commencing with the acquisition of adversarial malware datasets from the Network Security Group CNR-IEIIT. Following this, the data was filtered, analyzed, and normalized using the Standard Scaler method and Chi-Square filter techniques for feature selection. The LSTM and CNN deep learning classifiers subsequently employed the reduced dataset. The dataset was partitioned into training and testing sets at a 75% to 25% ratio. The results were assessed using statistical machine learning metrics, including classification accuracy, true positive rate, false negative rate, error rate, specificity, sensitivity, and training time. The Python Guide was employed to create experimental configuration on the Google Colab Notebook platform.

4.1 Results Analysis

The results of each segment are presented in this section.

4.1.1 Loading of Dataset

The implementation of the data mining life cycle necessitates the importing of a dataset. The dataset, which has been uploaded to Google Drive and installed onto the Google Colab

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

Platform, comprises 12 predicting factors and 1 response factor, both of which represent legitimate and malicious classes. The dataset comprises 17,997 records.

4.1.2 Explorative Data Analysis

The exploratory data analysis enhances the dataset's performance during classification by performing necessary normalization and filtering techniques, thereby providing more insight into the data.

i. Examine rows for null values

The is null function identifies null observations in rows and columns for predictive factors, revealing no null values in any column observations. The class group size is established, with valid records totaling 13,405 and malicious records amounting to 4,502.

Figure 4.1Checking for null values

The distribution of data among the response variables is more clearly illustrated in the pie chart below.

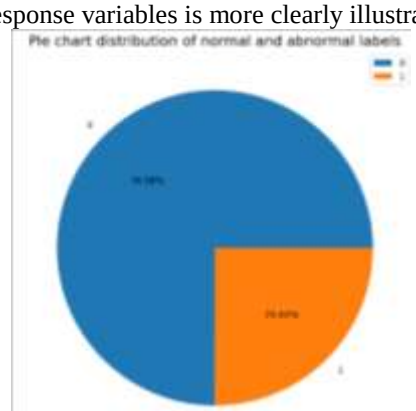

Figure 4.2: Data distribution of the response variable

ii. Feature Visualization

The histogram diagram in figure4.3 below illustrates the distribution of the predicting features in a sample.

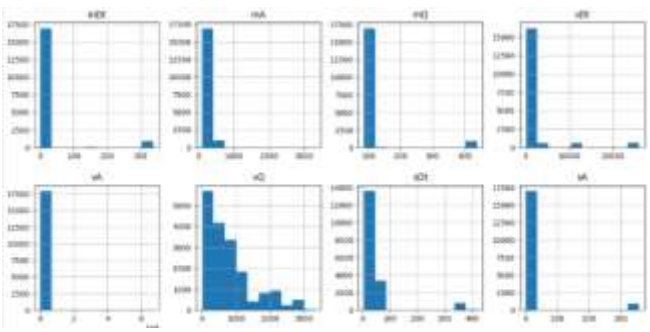

Figure 4.3: Feature distribution across attributes

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

4.1.3 Normalization with minmax scaler

An estimator that scales and transforms each feature individually within a specific range, typically set between 0 and 1, on the training set is known as the minmax scaler.

	mDt	mA	mQ	vDt	vA	vQ	sDt	sA	sQ	kDt	kA	kQ
0	0.001552	0.044467	0.001536	0.003266	0.000039	0.022415	0.141428	0.003773	0.001764	0.814298	0.018832	8.376006e-06
1	0.001895	0.048874	0.003713	0.000539	0.000043	0.278832	0.004707	0.002648	0.106383	0.009335	0.011410	4.570140e-10
2	0.000957	0.052071	0.002928	0.000263	0.000050	0.018127	0.013194	0.002265	0.000865	0.020824	0.007733	1.175457e-09
3	0.002039	0.053839	0.006154	0.000569	0.000057	0.019608	0.003921	0.002848	0.001556	0.008510	0.011581	2.850332e-09
4	0.001249	0.044378	0.006433	0.000379	0.000038	0.270040	0.009447	0.003072	0.109401	0.015320	0.011217	4.536273e-10

Figure 4.4: Dataset after normalization process

4.1.4 Correlation Matrix

The correlation matrix illustrates the relationships among dataset features, demonstrating a strong significance level for each component in connection to the response variable.

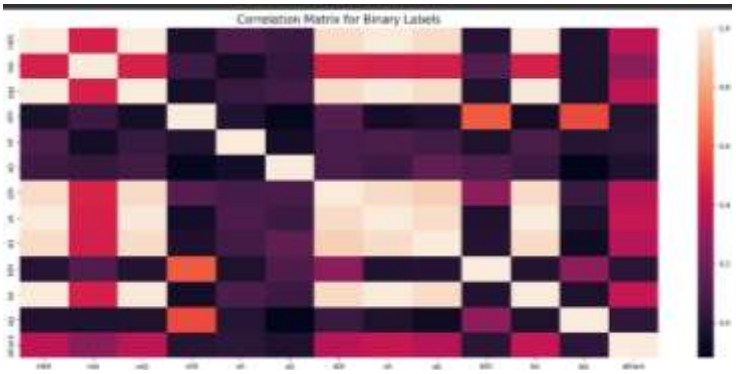

Figure 4.5: Correlation between the features column.

4.1.5 Feature Selection

The Chi-Square Selector identified the top nine traits, which were chosen from a total of twelve criteria. The correlation assesses the significance degree of each predictor variable in relation to the responder variable (attack type). A total of 9 features were picked based on their relevance degree, establishing the rank order.

Perform Feature Selection using Chi=Square

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
# Split the dataset into features and target variable
X = dataN.iloc[:, :-1] # Features
y = data['attack'] # Target variable
# Apply the chi-square test to select the top k features
k = 9 # Number of top features to select
selector = SelectKBest(chi2, k=k)
X_new = selector.fit_transform(X, y)

# Print the names of the top k features
mask = selector.get_support()
selected_features = []
for bool, feature in zip(mask, X.columns):
    if bool:
        selected_features.append(feature)
print('Selected features:', selected_features)
```

```
Selected features: ['mDt', 'mA', 'mQ', 'vA', 'vQ', 'sDt', 'sA', 'sQ', 'kA']
```

Figure 4.6 Selected Features

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

The selected features are further highlighted in the below with its correlation score as obtained by chi-score.

Table 4.1 Selected features

	Features	Pearson Correlation Score
1	mDt	0.334562
2	mA	0.34391
3	mQ	0.344535
4	vA	0.368486
5	vQ	0.387358
6	sDT	0.444874
7	sA	0.459984
8	sQ	0.463153
9	kQ	0.463735

4.1.6 Data Partitioning

The dataset was partitioned with a 75% allocation for training and 25% for testing, with the objective of generating experimental insights for Long Short-Term Memory or CNN and forecasting future data occurrences. The 25% subset was utilized to assess the performance of the trained LSTM/CNN model, as depicted in Figure 4.7.

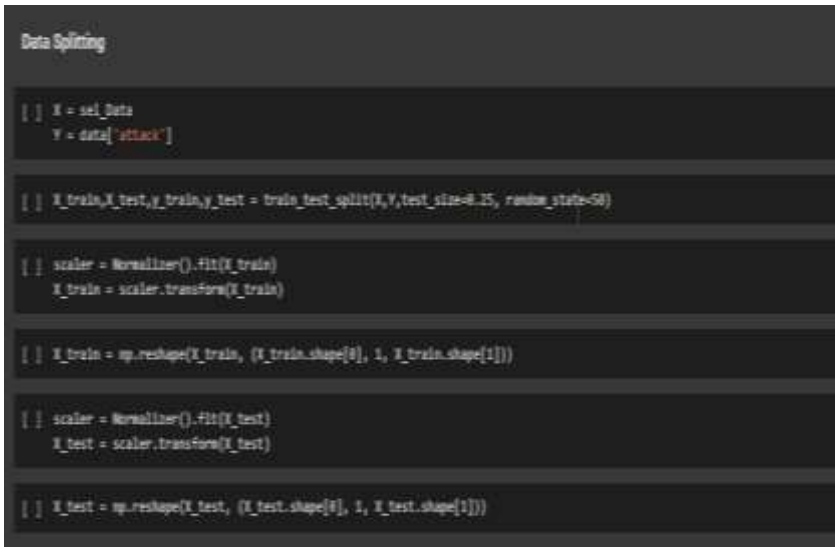

Figure 4.7 Data Partitioning

4.2 Experimental Results Evaluation.

The experimental results derive from a classification algorithm, assessed through True Positive rate, False Positive rate, True Negative rate, False Negative rate, accuracy, and error rate, alongside classification accuracy, sensitivity, specificity, and error rate. Figure 4.8 below illustrates an example setup and parametric configurations for the deep learning algorithm (LSTM); the model was subsequently trained after establishing all necessary parameters for both the LSTM and CNN models.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

```
LSTM MODEL

[ ] id=X_train.shape[2]
    batch_size = 32

[ ] model_lstm = Sequential()
    #model_dnn3.add(Flatten())
    model_lstm.add(LSTM(8,input_dim=id, return_sequences=True))
    model_lstm.add(Dropout(0.1))
    model_lstm.add(LSTM(8, return_sequences=False))
    model_lstm.add(Dropout(0.1))
    model_lstm.add(Dense(1))
    model_lstm.add(Activation('sigmoid'))

[ ] model_lstm.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
    es = EarlyStopping(monitor='val_accuracy', mode='max', min_delta=0.0001,patience=5) ## early stoppoing
    model_lstm.fit(X_train, y_train, batch_size=batch_size, epochs=1000, validation_data=(X_test, y_test),callbacks=[es])
    model_lstm.save("lstm_model_lstm.hdf5")
```

Figure 4.8 LSTM model

Case 1: Normalization + Reduced Adversarial Malware Dataset (obtained with Chi Square) + LSTM Technique: The initial phase encompasses data cleansing, normalization, chi-square feature selection, and LSTM deep learning classification.

a. Confusion Matrix (LSTM)

Technique	F-score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error rate	Training Time
LSTM Model (Case 1)	0.87397	0.82123	0.93395	0.97027	0.821234	94.1999	0.058	60.1040

A confusion matrix encapsulates prediction outcomes for a classification task, delineating accurate and inaccurate predictions by class based on test data. The diagonal elements signify correct classifications, whereas the off-diagonal elements represent incorrect classifications. For instance, class 0 exhibits 3,334 correct classifications and 64 incorrect ones, while class 1 shows 905 correct classifications and 197 incorrect classifications.

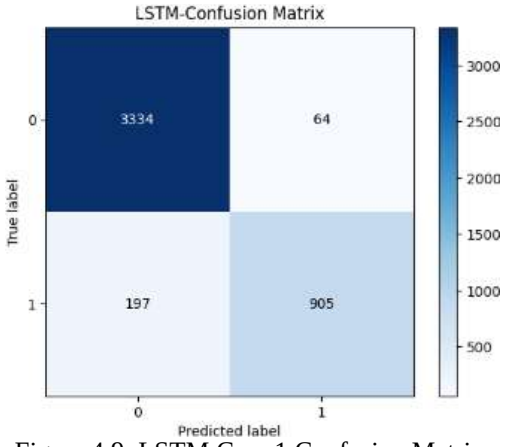

Figure 4.9: LSTM Case 1 Confusion Matrix

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

b. Evaluation Parameters for Classification Phase Case 1

Techniques	F-Score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error Rate	Training Time
LSTM Model (Case 1)	0.87397	0.32123	0.93395	0.97027	0.821234	94.1999	0.058	60.1040

Case 2: Normalized Adversarial Malware Dataset + LSTM Technique: The second scenario integrates data cleansing, data standardization, and deep learning categorization via LSTM.

a. Case 2 Confusion Matrix

A classification problem's prediction results are summarized in a confusion matrix, which delineates correct and incorrect predictions by class based on data testing. Correct classification is indicated by diagonals, whereas incorrect classification is indicated by non-diamond cells. For instance, class 0 has 3326 correct classifications and 72 inaccurate ones, whereas class 1 has 890 correct classifications and 212 incorrect classifications.

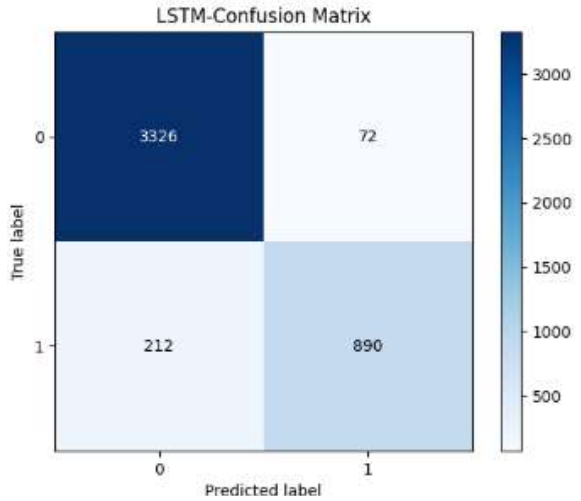

Figure 4.10 LSTM Case 2 Confusion Matrix

b. Evaluation Parameters for Classification Phase Case 2

Technique	F-score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error rate	Training Time
LSTM Model (Case 2)	0.86240	0.80762	0.92515	0.97321	0.80762	93.6888	0.06311	67.4791

Case 3:Normalization + Reduced Adversarial Malware Dataset (obtained with Chi Square) + CNN Technique: The third scenario involves data purification, normalization, feature engineering using chi-square, and CNN deep learning classification.

a. Case 3 Confusion Matrix

Figure 4.11 illustrates that class 0 contains 3,346 accurate classifications and 13 erroneous classifications, whereas class 1 has 748 correct classifications and 393 incorrect classifications, with the diagonal cells denoting the correct classifications.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

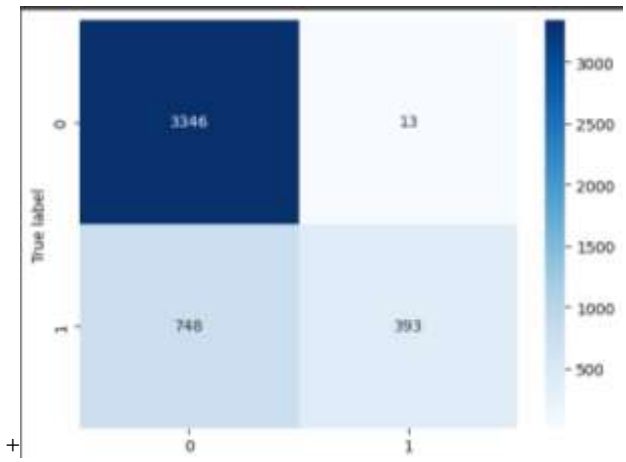

Figure 4.11 LSTM Case 3 Confusion Matrix

b. Evaluation Parameters for Classification Phase Case 3

Technique	F-score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error rate	Training Time
LSTM Model (Case 3)	0.79905	0.83089	0.85550	0.97321	0.83089	83.0889	0.16911	41.0247

Case 4:Normalized Adversarial Malware Dataset + CNN Technique: The fourth case scenario employs data cleansing, standardization, and CNN deep learning categorization.

a. Case 4 Confusion Matrix

The diagonals in each class indicate accurate classifications, whereas the non-diagonal cells in each row represent wrong classifications for each class. Figure 4.13 illustrates that class 0 has 3,335 accurate classifications and a total of 24 erroneous classifications, while class 1 displays 819 on the diagonal, indicating correct classifications, alongside a total of 322 incorrect classifications.

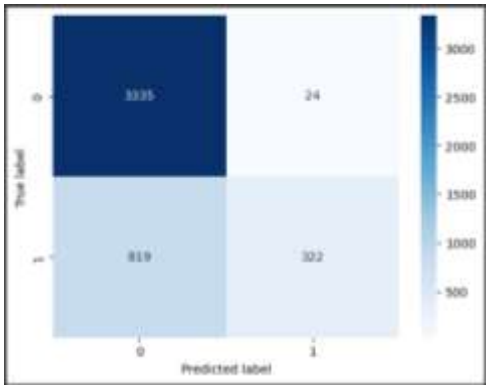

Figure 4.12 LSTM Case 4 Confusion Matrix

b. **Evaluation Parameters for Classification Phase Case 4**

Technique	F-score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error rate	Training Time
LSTM Model (Case 4)	0.77250	0.81267	0.83524	0.99286	0.81267	81.267	0.18733	45.1138

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

4.3 Graphical Analysis for the case Models

The graphical analysis presents a comparative assessment of the training duration, classification accuracy, specificity, sensitivity, and error rates of the two evaluated classifiers.

i. Result Analysis for Training Time

The training duration indicates the time required for the model to establish knowledge retention from the provided dataset for each model.

The LSTM Case 1 surpassed the CNN Case 3, since the diminished dataset mitigated time complexity resulting from dimensionality reduction. Figure 4.13 clearly demonstrates that the optimal time for LSTM Case 1 surpasses that of CNN Case 3. Case 3 yielded the most effective results in terms of timing.

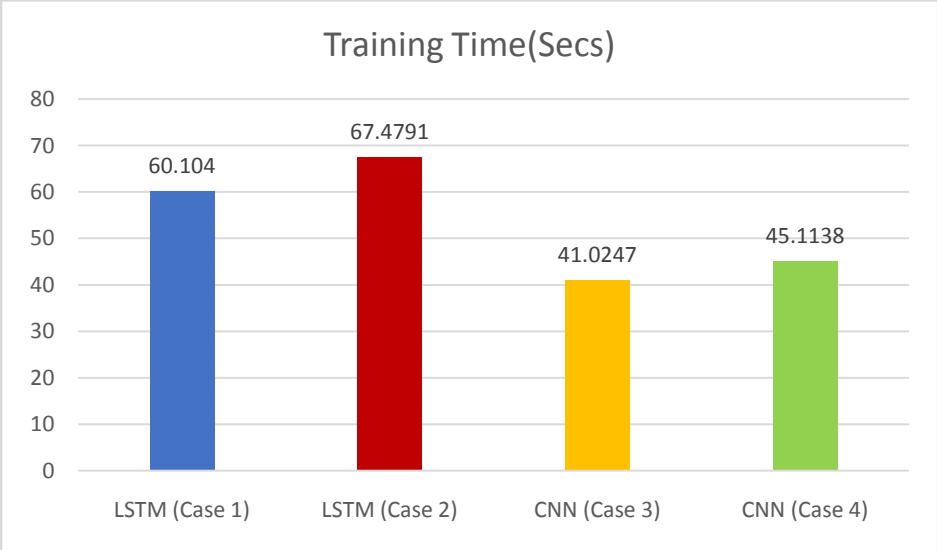

Figure 4.13: Training Time obtained for the 4 cases

ii. Result for Classification Accuracy

The classification accuracy of LSTM Model Case 1 was found to be 94.1999% higher than Case 2 Model, and CNN Model Case 3 achieved 83.0889% higher than Case 4 Model, indicating a higher classification rate.Case 1 however showed the best efficient results for the classification accuracy pointing it as the best model in all of the four.

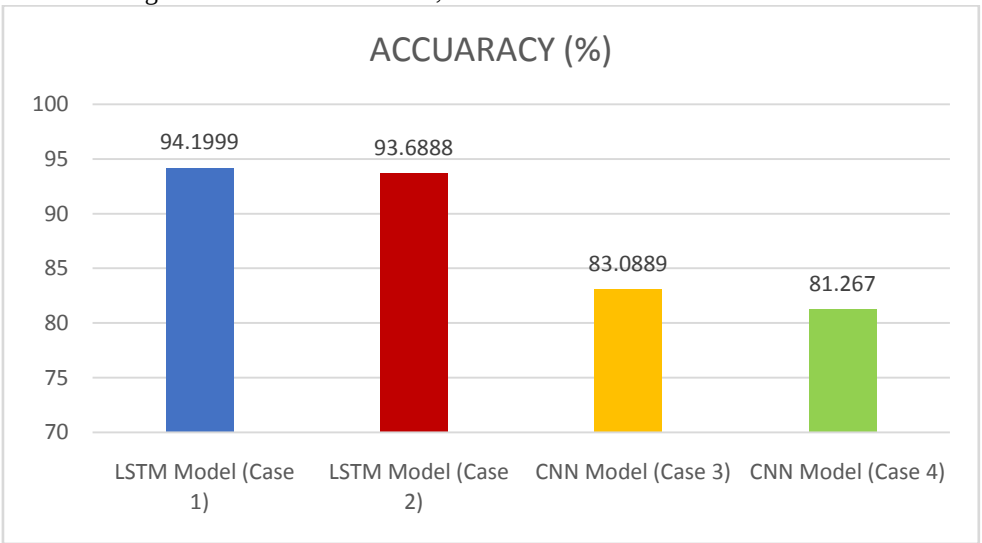

Figure 4.14: Classification Accuracy of the 4 models.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

iii. Sensitivity and Specificity

The research computes sensitivity and specificity by employing a ratio of positive to negative predictions. The optimal sensitivity and specificity rate is 1, signifying an excellent prediction rate. The LSTM Case 1 model attained the highest positive rate, whilst the LSTM Case 2 model reached the optimal specificity rate. In CNN cases, the CNN Case 3 model attained the highest positive rate in sensitivity, whereas the CNN Case 4 model achieved the highest specificity rate with negligible fluctuation.

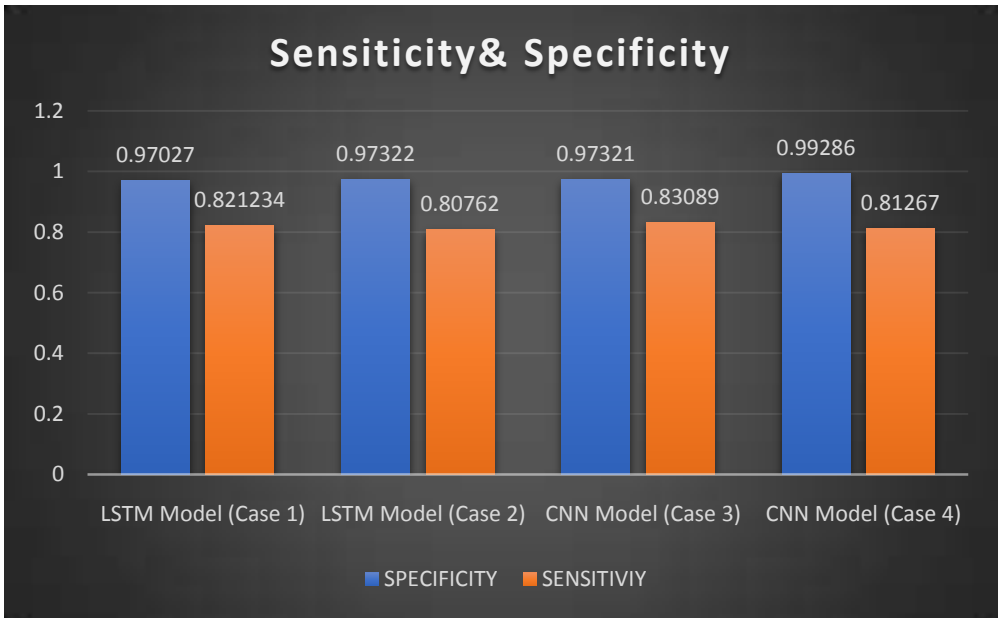

Figure 4.15: Sensitivity and Specificity rate of the 4 models.

iv. Result of Comparative Analysis for Error rate

The LSTM Case 1 model demonstrates a reduced error rate of 0.058, signifying superior positive rate detection and minimal false alarm detection relative to the LSTM model Case 2, whereas the CNN Case 3 model shows a lower error rate of 0.16911, indicating high positive rate detection and low false alarm detection compared to the CNN model Case 4. In all instances, the LSTM scenario demonstrated the minimal error rate.

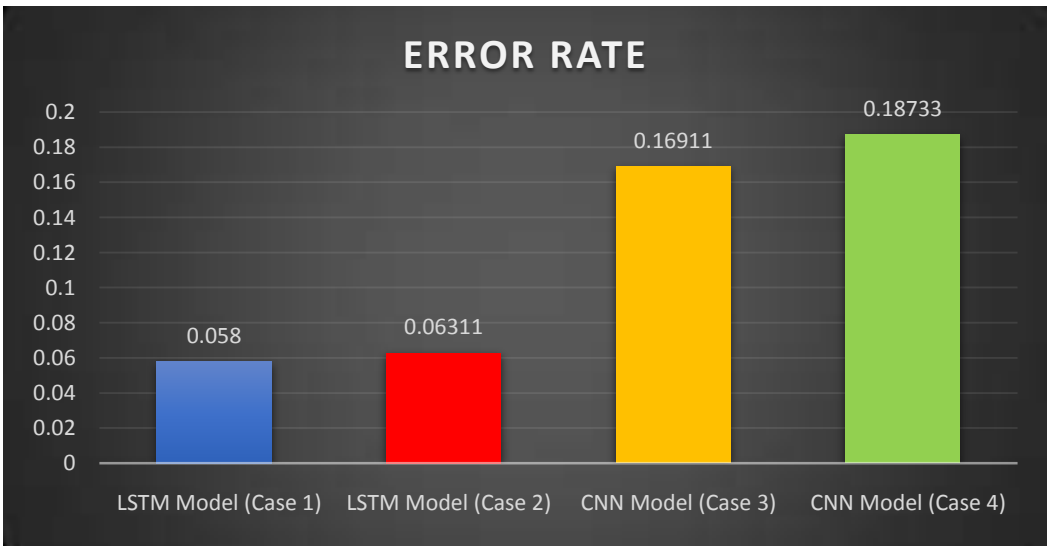

Figure 4.16 Error Rate for the 4 case models

International Journal of Advances in Engineering and Management (IJAEM)
Volume 7, Issue 07 July 2025, pp: 659-676 www.ijaem.net ISSN: 2395-5252

v. F-score and Precision

The F-score, or F-measure, evaluates a test's accuracy by juxtaposing its precision and recall. Precision is the proportion of true positive findings to the total positive results, encompassing those inaccurately recognized, whereas recall is the proportion of actual positive results to the samples that ought to have been categorized as positive. The LSTM Case 1 model has the highest F-score and precision among the four case models.

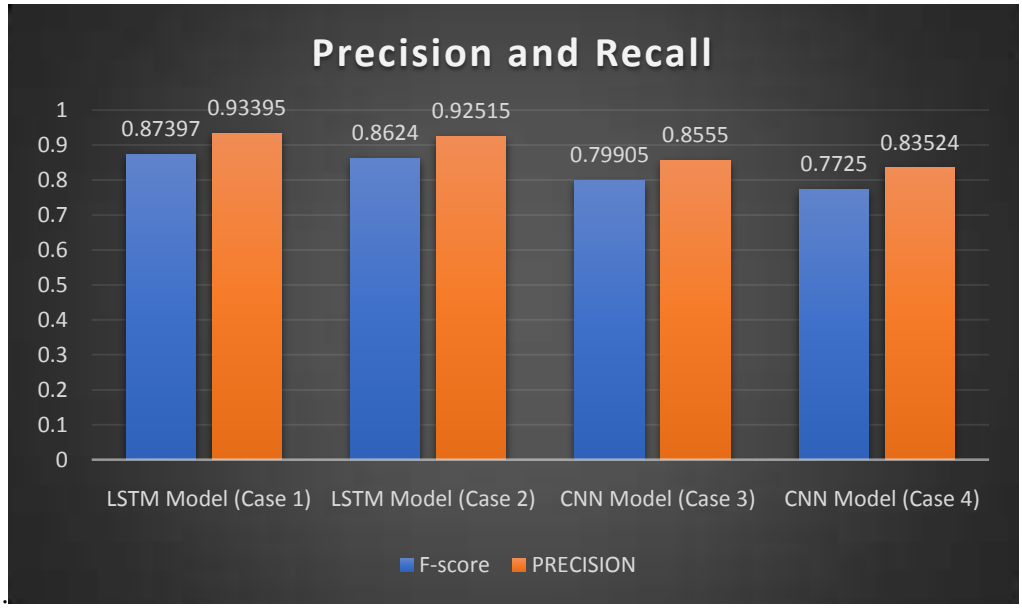

Figure 4.17 F-score and Precision chart for the four case models

V. CONCLUSION

This research effectively created and evaluated a deep learning system designed to categorize adversarial malware assaults using LSTM and CNN models with various preprocessing methods. The findings demonstrate that the LSTM model including feature selection (Case 1) had superior performance, with the greatest classification accuracy of 94.2% and the lowest error rate of 0.058. In contrast, the CNN models often exhibited reduced accuracy and elevated error rates, despite their strong performance in specificity and sensitivity. The results highlight the efficacy of using LSTM in conjunction with chi-square feature selection to improve classification accuracy and computational efficiency, presenting a potential method for identifying adversarial malware.

REFRENCES

[1]. Guo, J., Sang, Y., Chang, P., Xu, X., & Zhang, Y. (2021). MGEL: A robust malware encrypted traffic detection method based on ensemble learning with multi-grained features. In International Conference on Computational Science (pp. 195-208). Springer, Cham.

[2]. Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. IEEE Communications Magazine, 57(5), 76–81.

[3]. Awujoola, O. J., Ogwueleka, F. N., Irhebhude, M. E., & Misra, S. (2021). Wrapper based approach for network intrusion detection model with combination of dual filtering technique of resample and SMOTE. In Artificial Intelligence for Cyber Security: Methods, Issues and Possible Horizons or Opportunities.

[4]. Raff, Edward, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. (2018). "Malware detection by eating a whole exe." In Workshops at the thirty-second AAAI conference on artificial intelligence. doi:1710.094335v1.

[5]. Proska, K., Hildebrandt, C., Kappellmann Zafra, D., & Brubaker, N. (2021). Portable executable file infecting malware is increasingly found in OT networks. Retrieved from https://www.mandiant.com/resources/pe-file-infecting-malware-ot.

[6]. Perdisci, Roberto, Andrea Lanzi, and Wenke Lee. (2018). "Classification of packed executables for accurate computer virus detection." Pattern recognition letters 29, no. 14: 1941-1946.
[7]. Hwang, J.; Kim, J.; Lee, S.; Kim, K. (2020). Two-stage ransomware detection using dynamic analysis and machine learning techniques. Wirel. Pers. Commun.
[8]. Aslan O. and Samet R., (2021) A comprehensive review on malware detection approaches, IEEE Access. 8, 6249–6271, https://doi.org/10.1109/access.2019.2963724
[9]. Oliveira, J. A., Pacheco, J. P., &D'Abadie, A. (2022). Deep learning approaches for malware detection: A review. University of Washington ETD (Doctoral Dissertation). https://doi.org/10.31979/etd.s3hj-hrk4.
[10]. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S., (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials
[11]. Renjie Lu, (2019) Malware Detection with LSTM using Opcode Language, University of Chinese Academy of Sciences arXiv:1906.04593v1 [cs.CR].
[12]. Yuan, F.; Cao, Y.; Shang, Y.; Liu, Y.; Tan, J.; Fang, B. (2018). Insider Threat Detection with Deep Neural Network. In Proceedings of the ICCS, Wuxi, China, 11–13.
[13]. Agarap, A. F. (2017). Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification.
[14]. Akarsh, S., Poornachandran, P., Menon, V. K., & Soman, K. (2019). A detailed investigation and analysis of deep learning architectures and visualization techniques for malware family identification. In Cybersecurity and Secure Information Systems (pp. 241–286). Springer, New York, NY, USA.
[15]. Cui, Z., Du, L., Wang, P., Cai, X., & Zhang, W. (2019). Malicious code detection based on CNNs and multi-objective algorithm. Journal of Parallel and Distributed Computing, 129, 50–58.
[16]. Jain, M., Andreopoulos, W., & Stamp, M. (2021). CNN vs ELM for image-based malware classification. arXiv preprint, arXiv:2103.13820.
[17]. Kim, Samuel (2018). "PE header analysis for malware detection.".