ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAiT)
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

USER BEHAVIOR-BASED INSIDER THREAT DETECTION USING FEW SHOT LEARNING

BY
EDEN TEKLEMARIAM

ADVISOR
Dr. FITSUM ASSAMNEW

A thesis submitted to the School of Electrical and Computer Engineering in partial fulfillment of the requirements for the Degree of Master of Science in Telecommunication Engineering

AUGUST, 2023
ADDIS ABABA, ETHIOPIA

ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

USER BEHAVIOR-BASED INSIDER THREAT DETECTION USING FEW SHOT LEARNING

BY
EDEN TEKLEMARIAM

APPROVED BY BOARD OF EXAMINERS

_____

Dean, SECE, AAiT(Name and Signature)

_____

Advisor (Name and Signature)

_____

Internal Examiner (Name and Signature)

_____

External Examiner (Name and Signature)

Declaration

I, Eden Teklemariam Game, declare that this thesis is my original work. All sources of information in this study have been appropriately acknowledged. I further confirm that this thesis has not been submitted either in part or in full for any other requirements to any other learning institution.

Declared By:

_____

Student's Name and Signature

Approved By:

_____

Advisor's Name and Signature

AUGUST, 2023

Acknowledgments

To begin with, I express my sincere gratitude and appreciation to the Divine for the abundant blessings and opportunities that have enabled me to complete this thesis successfully.

I extend thanks to my esteemed advisor, Dr. Fitsum Assamnew, whose valuable insights, constructive feedback, and expert direction played a pivotal role throughout the entirety of this research.

I am also indebted to Ethiotelecom for providing me with this invaluable opportunity, and to AAiT for equipping me with the necessary knowledge to undertake this research endeavor.

My deepest appreciation goes to my family, especially to my husband Abraham Ashenafi for his unwavering love, assistance, and unmatched support, and my two little sons. I am forever grateful to my family for granting me the chances and experiences that have shaped my identity.

Abstract

Insider threats are among the most difficult cyber threats to counteract since they emerge from an Organization's own trusted employee who knows its organizational structure plus system and often leads that organization to a significant loss. The problem of insider threat detection has been researched for a long time in both the security and data mining sectors. The existing studies face challenges due to a lack of labeled datasets, imbalanced classes, and feature representation. The machine learning approaches depend on manual feature engineering that takes time and requires expertise and knowledge. The deep learning approaches depend on a huge amount of labeled and balanced training data. In the case of insider threat detection, the number of malicious users compared to normal ones is significantly imbalanced in both real-world scenarios and insider threat working datasets. Data on insider threats often includes a lot of features, which can make the data high-dimensional and difficult to represent the relevant features. In this paper, we propose a novel approach that includes CNN (Convolutional Neural Network) and LSTM (Long short-term memory) approach to act independently over the publicly available insider threat dataset, CERT (Computer Emergency Response Team) release 4.2 for feature extraction and few-shot learning based detector for insider threat detection. We concatenate features extracted from the dataset using both deep learning models and do feature selection to have relevant and best features. We use the Siamese neural network (SNN), called the 'twin' network of few-shot learning, to detect malicious insiders. We do an experiment using three datasets which are CNN-extracted features datasets (datasets found from a process of feature extraction using CNN), LSTM-extracted features datasets (datasets found from a process of feature extraction using LSTM), and Selected-features datasets (datasets found after applying feature concatenation and selection techniques). We do compare our model with other baseline models such as RNN, isolation forest, and XGBoost. The experimental result shows that with the experiment done using CNN-extracted features datasets, the proposed model best performs with an F1 score of 68% which is 18% better than the isolation forest which performs the worst, and an FNR (False Negative Rate) value of 0.006. The second experiment is done using LSTM-extracted features datasets, and the results of SNN in terms of an F1 score is 69% which is 11% better than the isolation forest which performs the worst with an F1 score of 58% and an FNR value of 0.2.

The last experiment is done using the Selected-features dataset, the proposed model out-performs in terms of F1 score by having the highest value of 87% which is 10% greater than compared to the least performer baseline model (XGBoost) that has 77% of F1 score. In terms of FNR (False Negative Rate) the lowest value of 0.11 FNR with the CNN-extracted features datasets.

Keywords: Insider threat, detection, Training, CNN, LSTM, Few-shot learning, dataset, SNN, features, dataset

Table of Contents

List of Figures

List of Tables

List of Acronyms

AUC = Area Under the ROC Curve

AE = Autoencoder

BCE = Binary-Cross Entropy

CMU = Carnegie Mellon University

CERT = Computer Emergency Response Teams

CNN = Convolutional Neural Network

csv = Comma-Separated Values

CISA = Cyber and Infrastructure Security Agency

DNN = Deep Neural Network

FPR = False Positive Rate

FC = Fully Connected

FSL = Few Shot Learning

CGANs = Generative Adversarial Networks

GPUs = Graphics Processing Units

HMM = Hidden Markov Model

iForest = Isolation Forest

IDE = Integrated Development Environment

KMC = K-means Clustering

KNN = K-nearest Neighbor

LDA = Latent Dirichlet Allocation

LODA = Lightweight On-Line Anomaly Detection

LOF = Local Outlier Factor

LSTM = Long Short-Term Memory

LR = Linear Regression

MSE = Mean Squared Error

NMF = Non-negative Matrix Factorization

OCSVM = One-Class Support Vector Machine

PCA = Principal Component Analysis

ReLU = Rectified Linear Unit

RNN = Recurrent Neural Network

ROC = Receiver Operating Characteristic Curve

SVM = Support Vector Machine

SUTD = Singapore University of Technology and Design

TFIDF = Term Frequency-Inverse Document Frequency

TWOS = The wolf of SUTD TNR = True Negative Rate

TPR = True Positive Rate

XGBoost = Extreme Gradient Boosting

Introduction

1.1 Background

Telecommunications and computer networks are important for the exchange of information. Threats have increased as a result of an increase in valuable information and enabling technological advancements. Threats come from both external and internal sources within the organization. Both threats pose a serious security concern but internal threats appear to be challenging to identify.

Any individual who has gained acceptance into an organization and, as a result, possesses special or confidential knowledge or who has or has had permitted access to an organization's resources, including its staff, facilities, information, equipment, networks, and systems, is considered to be an insider.

Insider threat is the risk when an insider uses their access to an organization or their familiarity with it to cause harm to that organization. This damage may include deliberate, negligent, or unintentional actions that compromise the organization's integrity, confidentiality, and availability.

Different international security agencies tried to define threats caused by insiders on their own: Insider threat is defined by the Cyber and Infrastructure Security Agency (CISA) in [1] as "The threat that an insider will use their authorized access, wittingly or unwittingly, to do harm to the department's mission, resources, personnel, facilities, information, equipment, networks, and systems." Computer Emergency Response Teams (CERT) defines insider threat as "A current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and who intentionally or unintentionally exceeds or misuses that access to negatively affect the confidentiality, integrity, or availability of the organization's information or information systems" in [2].

1.1.1 Types of Insider Threats

According to the Cyber and Infrastructure Security Agency (CISA) in [1], insider threats can be divided into two categories: intentional insider threats (malicious insider threats) and unintentional insider threats (inadvertent insider threats), depending on the insider's motivation. A threat caused by a current or former employee, contractor, or business partner who has or had authorized access to an organization's network, system, or data and purposefully overstepped or abused that access in a way that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems is considered as a malicious insider threat or intentional insider threat. An unintentional insider threat, on the other hand, is a threat caused by a current or former employee, contractor, or business partner who has or had authorized access to a company's network, system, or data and who, through action or inaction without malicious intent, harms or significantly raises the possibility of future serious harm to the confidentiality, integrity, or availability of the company's information or information systems.

1.1.2 Motivations Behind Intentional/Malicious Insider Threats

The motivations behind insider threats can vary, R. Nasir et al in [3] list some common motivations as described below which are also shown in Figure 1.1.

Emotion-Based

There is a strong possibility that an insider will act maliciously if they are bored, unhappy, dissatisfied, or outraged about something at work or in their company.

Financial-Based

Many people are highly motivated by money, there is a chance for an employee to profit from their insider position if they are struggling financially or want to improve their circumstances.

Political-Based

While less common, state-sponsored insider threat attacks and corporate espionage have been documented in many cases. National pride, political motivations, or even a combination of the other two categories of malicious insider threat—emotional retaliation and financial gain—might serve as these people's primary motivators.

1.1.3 Causes for Unintentional Insider Threats

Lack of knowledge or understanding

An insider may run the danger of unintentionally causing a threat if they aren't computer-knowledgeable or used to thinking about the security consequences of their actions. This is particularly true if the cyber security policies are comprehensive and unnecessarily complex. It might involve exchanging private information and data over unsecured networks, using insecure public Wi-Fi to access vital systems, asking users to metadata-tag frequently accessed content, etc.

Convenience

In the present world, convenience sadly dominates almost everything else. Insiders will probably try to go around the existing systems if the cyber security rules, tools, etc. are not well suited.

Misplaced technology

Placing equipment like work laptops or cell phones in open areas may increase the accessibility of secured data. Insiders have a significant danger of mistakenly misplacing their equipment in addition to the increased possibility of outsiders seeking to access these secured devices.

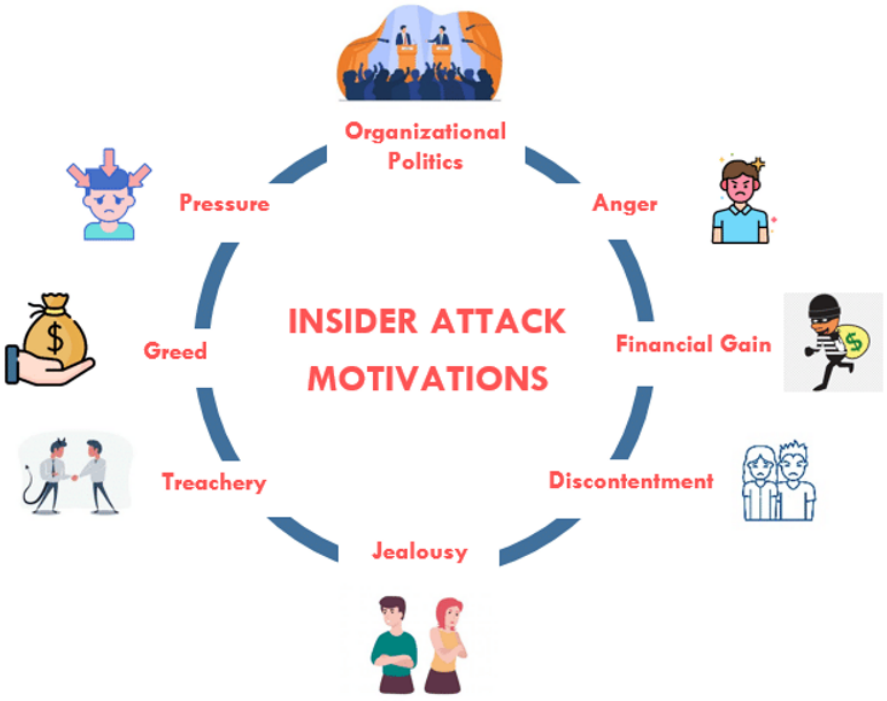

Figure 1.1: Motivations behind intentional Insider threats

Insider threat detection

Insider threat detection refers to the process of identifying and mitigating potential risks posed by malicious insiders. Detecting and addressing insider threats is crucial for protecting an organization's data, intellectual property, reputation, and overall security. Insider threat detection is a tedious process that requires constant user activity monitoring, behavior analytics, threat management, and attention to unusual actions.

Insiders naturally hold a position of trust; in order for them to carry out their assigned duties, they must be given specific privileges, authority, and access. Even when these insiders are internal users or hired workers or partners, organizations still need to keep an eye on every action to spot and stop abnormal behavior [4].

The need for insider threat detection arises from the recognition that traditional cybersecurity measures primarily concentrate on external threats, such as hackers or malware attacks. However, insider threats can be equally or even more damaging as they often have legitimate access to critical assets, making it challenging to differentiate between authorized and unauthorized activities

1.2 Problem Statement

Insider threat is one of the worldwide key security issues in telecom and any other business companies. Enterprises' reputations, financial resources, and intellectual property are all seriously at risk from insider threats. The 2022 global insider threats cost report in [5] shows that 67 % of companies are experiencing incidents between 21 and more than 40 per year. This is an increase from organizations with the same range of 53% in 2018 and 60% in 2020 and a total average yearly cost of $15.4M, which does not include the damage to the organization's reputation, loss of business, and decrease in company value. To reduce the damages from insider threats, several types of research are conducted on how to prevent, mitigate, and detect an insider threat using different mechanisms including a combination of research techniques from various fields, like cybersecurity, data analytics, machine learning, deep learning, and user behavioral analysis.

However, the existing research in the domain of insider threat detection exhibits notable gaps such as:

Data scarcity: The ability to train and test detection algorithms for insider threats depends on having access to sufficient and relevant data. However, because insider attacks are rare acquiring labeled insider threat data is difficult.

Imbalanced datasets: Insider threat datasets often have a class imbalance, where the proportion of instances representing normal users is much higher than that of instances representing malicious users. As a result, algorithms that attempt to detect insider threats may be biased.

Feature selection and representation: Accurate insider threat detection depends on selecting the appropriate features and accurately representing user behavior. Finding the most accurate representation that reflects the delicacy of insider threats is another challenge.

To address the mentioned gaps, we propose an approach that brings Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks together to extract features integrating with the concept of few-shot learning, enabling adept detection of insider threats with limited labeled instances even in imbalanced class scenarios.

1.3 Research Questions

The research gap has been addressed, using the proposed approach by answering the following question:

What is the effect of using combined CNN and LSTM for feature extraction with a few-shot learning model for insider threat detection?

1.4 Objectives

1.4.1 General Objective

The general objective of this research is to detect malicious insider threats using a model that is capable of dealing with imbalanced data, few-shot learning which can learn from small samples with Convolutional Neural Network (CNN), and Long Short-Term Memory(LSTM) as feature extractors.

1.4.2 Specific Objectives

To achieve the general objective of the research, some specific objectives are needed and listed as follows:

- To build a feature extractor using CNN & LSTM in order to obtain well-represented features which are given later as input for the detectors
- To build a detector using a few-shot learning model
- To evaluate the performance of the proposed method
- To compare the proposed method with existing insider threat detection methods

1.5 Motivations and Contribution of the Research

1.5.1 Motivations

This research has two motivations, the first one is the interchangeable applicability of combined CNN and LSTM in insider threat detection research of [6] and [7]. In [7] the authors used CNN as a feature extractor and LSTM to detect insider threats. Whereas in [6] the authors used in reverse which means they used LSTM to extract features and CNN for detection purposes. So this interchangeable applicability of the models motivates us to explore the effect of using both as feature extractors by enabling them to act separately over the dataset and using the concatenated output of both to detect insider threats. The second one is the limited number of insiders in real-world companies again in the working datasets and the ability of few-shot learning in computer vision to generalize new sets using few samples motivates us to investigate the applicability of the few-shot learning model in insider threat detection tasks.

1.5.2 Contributions

This research will make significant contributions in two key aspects, laying the foundation for advancing the field of insider threat detection:

- A Pre-processed CMU CERT Insider Threat Dataset r4.2:

This study presents a pre-processed version of the CMU CERT Insider Threat Dataset r4.2, optimized to ensure data integrity and facilitate meaningful analysis. The enhanced dataset will serve as a valuable resource for the research community, fostering reproducibility and enabling comprehensive evaluations of insider threat detection models.

- A State-of-the-Art Model: CNN and LSTM as Feature Extractors combined with Few-Shot Learning as Insider Threat Detector:

  Integrating the cutting-edge capabilities of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks as feature extractors, the model demonstrates unparalleled proficiency in capturing intricate behavioral patterns and dynamics from user activity data.

  Moreover, by leveraging the power of Few-Shot Learning techniques as an insider threat detector, the model transcends the limitations of traditional supervised approaches, adeptly handling the scarcity of labeled data and equipping research community with the ability to detect potential threats even in data-scarce environments.

  By providing a comprehensive understanding of the model's architecture, training methodology, and optimization techniques, this research empowers practitioners to implement and adapt this state-of-the-art model within their own contexts, thereby enhancing security posture and safeguarding against internal security risks.

1.6 Scope and Limitation

1.6.1 Scope

The scope of this research is to present experimental findings by harnessing only the power of deep learning models for feature extraction and incorporating the Siamese neural network approach of few-shot learning models as efficient detectors. The research is conducted only with a focus on detecting intentional or malicious types of insider threats, even though there are other types of insider threats that are also important to consider in the experiment, such as unintentional insider threats and insider threats caused by negligence.

1.6.2 Limitation

The proposed approach is done by using only one type of dataset which is CERT insider threat dataset r4.2 but it might be better if more than one dataset is used. The other limitation is that again the used dataset is a synthetic dataset even though it is created considering real-world scenarios, the result obtained in this research paper might not be similar to the experiment conducted using the real-world dataset collected from companies.

1.7 Methodology

To achieve the research objectives the following procedures are followed:

I. **Data Gathering and Pre-processing**

We gather and pre-process data for training a model to identify insiders. The data is from the CERT r4.2 dataset, which includes five activity log files. The first step is to label each dataset and use stratified random sampling to take representatives of the two classes from each dataset. The next step was to pre-process the data, which included cleaning the data, removing irrelevant rows and columns, and aggregating the data. For the datasets with content features, topic modeling was also performed using TF-IDF, NMF, and LDA.

Finally, all of the pre-processed data was combined into a single dataset. This dataset is then used for training the model.

II. **Feature Extraction**

After preprocessing the data, the research extracts features from the preprocessed master dataset using CNN and LSTM. CNN extracts spatial patterns from the activity log files, while LSTM captures temporal and sequential behavior patterns. The research extracts features from the last layers of each model, just before the dense (fully connected) layers. This ensures that the features are representative of the data and can be used to train an accurate model.

III. **Feature Concatenation & Feature Selection**

We concatenate the features obtained from CNN and LSTM to create a unified and comprehensive feature representation of user behavior. This merging process ensures that the resulting feature set captures a comprehensive representation of user behavior, effectively harnessing the insights gleaned from both models. The next step is to drop constant features from the concatenated and unified file using VarianceThreshold, then select the best features using the SelectKBest method.

IV. **Threat Detection**

We use a Siamese Neural Network (SNN) to detect insider threats using three datasets: the CNN-extracted features dataset, the LSTM-extracted features dataset, and the selected-features dataset. A SNN is one of a few-shot learning approach, which means that it can learn to perform a task with only a few examples. This is useful for tasks like insider threat detection, where there may be limited data available.

V. **Model Evaluation and Comparative study research work**

We evaluate the performance of the proposed method using essential measurement metrics, including precision, recall, F1-score, and FNR (False Negative Rate). We also conduct a comparative analysis, wherein the proposed approach is compared against existing insider threat detection methods, such as Recurrent Neural Network (RNN), Xtreme Gradient Boosting (XGBoost), and Isolation Forest (iF).

Details of each procedure are provided in chapter four of this research.

1.8 Organization of the Research

This paper is structured as follows. Chapter two portrays a literature review, which includes; Anomaly detection, Behavioral analysis, Machine learning-based, Deep learning-based, and other methodologies used for detecting insider threats. The models, algorithms, parameters, and platforms used in this research are briefly discussed in Chapter Three. Chapter Four elaborates system model and research methodologies. It contains the experimental setup, data pre-processing, feature extraction, and selection procedures. The experimental findings of the study and discussions are presented in Chapter Five, whereas Chapter Six is about conclusions and future work.

Literature Review and Related works

2.1 Literature Review

In this section, we provide a review of the past literature that addressed the issues with insider threat detection. The papers included in this research are limited to the time range of 2015 to 2023 in order to get the recent state of the art. The review is arranged based on the methods used to do the research.

2.1.1 Insider Threat Detection

Today's digital economy relies heavily on robust cybersecurity. Unlike external risks, the most significant threats to security come from within. A. Yazdinejad et al. in [8], describe that the Key strategies to address this internal risk include identifying and predicting potential insider threats. Insider threat detection refers to the process of identifying and mitigating risks that arise from individuals within an organization who have authorized access to sensitive data, systems, or resources but may misuse or abuse these privileges [9]. Insider threats can be both intentional, such as employees or contractors with malicious intentions, and unintentional, such as employees who accidentally expose sensitive information. The primary goal of insider threat detection is to proactively identify unusual or potentially malicious activities carried out by authorized individuals within the organization [10], this involves monitoring and analyzing various data sources, such as user behaviors, network activities, access logs, and data transfers, to identify patterns that deviate from normal behavior. By doing so, organizations can detect potential security breaches, data leaks, or unauthorized activities before they escalate into major incidents.

2.1.2 Anomaly Detection

Anomaly detection is the process of employing machine learning and statistical models to automatically identify abnormal patterns or outliers in user activities. This can help identify subtle threats that might be hard to detect using rule-based systems alone. Different researchers such as Duc C. Le et al. in [11], Bartoszewski F. et al. in [12] and Gayathri R. G. et al. in [13] have conducted research by using anomaly detection for insider threats.

The main objective in [11] was to propose an effective approach for detecting insider threats using unsupervised ensembles. The authors use two publicly available datasets, CERT R4.2 and CERT R6.2, to evaluate the proposed approach. The methodology involves using various unsupervised learning methods, such as Autoencoder, Isolation Forest, Lightweight On-line detection of anomalies, and Local Outlier Factor, to create anomaly detection ensembles. The authors also explore different data representations and evaluate the performance of the proposed approach under different working conditions. The experimental environment involves using a computer with an Intel Core i7-6700K CPU, 32GB RAM, and an NVIDIA GeForce GTX 1080 Ti GPU. The authors use Python and TensorFlow to implement the proposed approach and evaluate its performance. The limitations of the proposed approach include the need for a large amount of training data and the potential for false positives. The authors justify the use of unsupervised learning methods by highlighting their ability to detect anomalies without the need for labeled data.

The results of the experiments show that the proposed approach outperforms previous works in the literature in terms of detection performance and the ability to generalize. The authors demonstrate that the proposed approach can detect malicious insiders under very low investigation budgets and provide comprehensive anomaly detection results per instance and per user.

Bartoszewski F. et al. in [12] conducted research with the main objective of providing an objective comparison of machine learning models and ensembles for anomaly detection in insider threats. The authors use the CERT dataset, which is a synthetic dataset created to simulate log behaviors of a virtual organization and includes manually developed attacks. The methodology includes feature extraction, machine learning model optimization, ensemble creation, and reporting metrics using Local Outlier Factor (LOF), One-Class Support Vector Machine (OCSVM/SVM), and Isolation Forest (IF). The experimental environment is not explicitly described in the text. The authors justify the use of their selected method by following guidelines of good conduct, including using multiple metrics to describe outcomes, creating models without prior knowledge of answers in the dataset, and informing readers about any shortcomings of the experiment design. The results show that ensembles outperform individual models in detecting insider threats [12].

Gayathri R. G. et al. in [13]The main objective of the paper is to propose a novel approach to detecting insider threats using Conditional Generative Adversarial Networks (CGANs) to enrich under-represented minority class samples. The motivation for this approach is the limited availability of real-world data for insider threat analysis, which hinders the exploration of effective solutions. The paper uses a dataset of email communications from the Enron corporation, which is pre-processed and used to generate synthetic data using the proposed CGAN method. The methodology involves a multi-stage workflow consisting of behavior extraction, CGAN-based data generation, and anomaly detection. The experimental environment includes the use of various performance metrics to evaluate the effectiveness of the proposed method, as well as comparisons to other existing methods. The limitations of the proposed method include the assumptions made and the vagueness of the qualitative and quantitative evaluation for synthetic data. The justification for using the CGAN method is its effectiveness in enriching under-represented minority class samples, and the found result is that the proposed method demonstrates effectiveness and capability for multi-class anomaly detection.

2.1.3 Behavioral Analysis

The challenge of detecting insider threats rises with the variety of insider attacks, intentional or unintentional loss or degradation of an organization's resources or skills. Monitoring user behaviors and activities to establish baselines of normal behavior and identify anomalies that might indicate malicious intent. Several studies have been conducted in the field of detecting insider threats using behavioral analysis [14], [15], [16] and the findings indicated that a thorough understanding of user behavior will aid in the investigation of any insider threat. By using their own to represent texts of arbitrary length, L. Liu et al. in [14] conduct research to improve the prior substantially streamlining the decision-making process. This study analyzed user behavior and determined how similar texts are to one another. Security logs are converted into a corpus, and then the corpus is trained. The predicted outcome in this case was to tend to show the likelihood of how much one user differed from the others. The insider threat database v6.2 dataset from CMU CERT Programs was used to test the model. Three alternative pairs of metrics were used to provide the performance measurement results: date. Corpus, date hour. Corpus, and user date hour. Corpus. A corpus aggregated by user is more adaptable to the temporal metrics, whereas a corpus aggregated by date works relatively better with the spatial metrics, according to experiments. The approach offers better simplicity and flexibility while also requiring less training time than the preceding approach.

J. Kim et al. in [15] also aimed to model user behavior and provide a reliable insider threat detection model that was adaptive for user behavior. For the purpose of identifying insider threats, CERT R6.2 datasets were subjected to the one-class classification techniques Gaussian density estimation (Gauss), Parzen window density estimation (Parzen), principal component analysis (PCA), and K-means clustering (KMC). Using various cut-off values, the true detection rate of each anomaly detection technique based on each role and content-based data is provided for all chosen One-class classifiers, and the suggested method demonstrates efficient detection performance.

X. Wang et al. in [16] proposed and conducted research on characterizing user behavior for insider threat detection with a data-centric approach. This approach is based on characterizing user behavior by extracting features of user interaction behavior, such as keystroke dynamics and consecutive queries to model users' access patterns. The data-centric strategy, which is based on defining user behavior by extracting characteristics, was applied to different types of datasets.

2.1.4 Machine Learning Approach

The machine learning approach, an unsupervised ensemble for insider threat anomaly detection was suggested by [11], [17], [18]. D. C. Le and N. Zincir-Heywood explored various computational intelligence approaches in [11] with the goal of combining these models to construct anomaly detection ensembles for enhancing detection performance. The following algorithms were used for the CMU-CERT insider threat dataset (r4.2) and (r6.2), LANL, and TWOS datasets: Autoencoder (AE), Isolation Forest (IF), Lightweight On-Line Anomaly Detection (LODA), and Local Outlier Factor (LOF). It should be noted that each malicious insider in the CERT data corresponds to one of the five common insider threat scenarios listed by CMU-CERT. The experimental results indicate that some of these scenarios can be identified by the proposed system very quickly and easily, while others are much more difficult to identify.

F. Janjua et al. suggested supervised machine learning in [17] utilizing the wolf of SUTD (TWOS) data set of malicious insider threat to tackle insider threat. This included Adaboost, Naive Bayes, Logistic Regression, KNN, Linear Regression, and Support Vector Machine (SVM). Accuracy, Recall, and Area under the Curve were some of the performance indicators used to gauge the model's effectiveness (AUC). AdaBoost classifiers perform satisfactorily, with KNN coming in second place and Naive Bayes coming in last with the greatest AUC.

Using natural language processing and insider danger personality profiles, M. Naghmeh and H. Adam conducted a study in [18] to proactively identify insider threats using imbalanced data. The research was conducted utilizing the TWOS (A dataset of malicious insider threat behavior), CMU's insider threat test dataset, and Enron Datasets. The final results show that although computing time was improved, the performance of the ML techniques used remained the same.

2.1.5 Deep Learning Approach

Deep learning become the leading machine learning scheme in recent days. F. Yuan et al. in [6] proposed insider threat detection with a Deep Neural Network to address issues with machine-learning techniques to identify insider threat, which necessitates a challenging and time-consuming feature engineering. DNN was implemented in two stages: using an LSTM, the first stage extracts the temporal aspects of user activity that have been abstracted. The feature vectors are then converted into fixed-size feature matrices. These fixed-size feature matrices are provided to the CNN in the subsequent step so that it can determine whether they are normal or anomalous. The experimental result was validated using the CMU-CERT insider threat dataset (r4.2), and the proposed method was assessed using the receiver operating characteristics curves (ROC) and area-under-curve (AUC) measurements. The LSTM2 with CNN3 delivers a better outcome than the other CNNs and obtains the best result when comparing the accuracy of other LSTM- CNNs.

S.Ahmed et al. in [7] also proposed LSTM combined with CNN (LSTM-CNN) by aiming to improve the insider threat detection models performance in terms of time. The research was done by using the publicly available insider threat dataset version 4.2, the experimental result showed that performance was improved with high precision and recall values by using the combination of LSTM-CNN model.

An implementation of an autoencoder for sequence data utilizing using an encoder-decoder LSTM architecture is known as LSTM Autoencoder. An encoder-decoder LSTM is set up to read the input sequence, encode it, decode it, and recreate it for a certain dataset of sequences.B. Sharma et al. in [19] and R. Nasir et al. in [3] used this LSTM autoencoder to detect insider threat based on user behavior. The proposed approach was implemented over CMU CERT insider threat dataset r4.2, the experimental result was compared using different baseline approaches (LSTM-CNN, LSTM-RNN, One Class SVM, Multi-State LSTM & CNN, Hidden Markov Model (HMM) and Isolation Forest) in terms of performance assessment metrics, dataset upon which evaluation is performed, and feature set used. When compared to other methods, it is shown that the suggested strategy generates relatively good accuracy and precision. in both papers, the resulting accuracy was above 90 %.

Deep learning-based attribute categorization insider threat detection for Data Security was proposed by F. Meng et al. in [20]. The research's goal was to solve the problems caused by classic Machin learning techniques and earlier detection systems that relied exclusively on information that was already known. Kernel PCA (Principal Component Analysis) and long short-term memory recurrent neural networks (LSTM-RNNs) method was presented, which consists of data event aggregator, feature extractor, attribute classifier, and anomaly calculator, and was applied to CERT insider threat dataset v6.2 to compare the true positive rate (TPR), false positive rate (FPR), precision, and accuracy of the proposed method with the prior methods such as SVM (Support Vector Machine), Isolation Forest, and PCA. In order to minimize the impact of secondary features on the classification effect, the authors only used the principal component features to train the classifier. This result was also made possible by the anomaly calculator based on multi-classifier fusion decision, which can significantly increase analysis accuracy.

D. Sun et al. proposed Deep Malicious Insider Threat Detection (DeepMIT) Framework based on Recurrent Neural Network in [21] to address the issues with previous insider threat detection approaches, which only cache data in an offline manner, combine different behaviors to model users without taking into account the impact of user attributes (such as user role), and only generate anomaly scores about the users before manually analyzing the alerts without further consideration. DeepMIT: A comprehensive strategy for preventing malicious insider threats that also includes threat detection and alert intelligence. Insider threat dataset (CERT version 6.2) was used to apply Long Short-Term Memory (LSTM and a variation of RNN) to continuous online training to adapt to the changing user behaviour. Results demonstrated that DeepMIT outperformed the baseline solutions of SVM, PCA, Iforest, and replicator neural networks in terms of overall performance.

2.1.6 Other Methods

V. Koutsouvelis et al. in [22] conducted research to detect insider threats using Artificial Intelligence (AI) and Visualization with the aim of answering the question of whether AI can be successfully used to detect malicious activity. The CERT insider threat dataset was used to apply the models and with the methodology used, the malicious activity of the users of the information system was achieved. The forecasting of the results was absolutely successful, with a percentage that reached almost 100 %.

A critical survey and review of insider threat was done, and J. R. Schoenherr and R. Thomson suggested in [23] a multidimensional classification system for insider threat behavior, SIEVE: severity (S), intentionality (I), type of employee norm violation (EV), and ethicality. Examining insider threat, definitions, research techniques, and models provided by existing academic and grey (nonacademic) literature was the main goal of the study. The suggested method was proven to be quite helpful in categorizing organizational actions and identifying particular types of insider threats that develop as a result of different reasons.

An artificial immune system (AIS) algorithm was developed by researchers in an effort to relate the human immune system to the identification of insider threats. The AIS method was used by O. Igbe and T. Saadawi in [24] to categorize the chosen user actions from the CERT dataset r4.2. An anomaly-based insider threat detection system that makes use of an ensemble model made up of a combination of various negative selection algorithms (NSA) is called an artificial immune system (AIS)-based algorithm. The accuracy, false positive rate (FPR), and true positive rate (TPR) of external evaluation measures are used in experiments. The results show that the suggested system was successful in classifying the input data with a lower FPR value but a higher AUC, and TPR.

2.2 Related Works

The research conducted by Yuan S. et al. in [25] tries to propose a novel framework of few-shot insider threat detection that combines self-supervised pre-training and metric-based few-shot learning to improve classifier performance. The motivation for this work was that existing approaches to detecting insider threats are limited by the lack of labeled data and the dynamic nature of user behavior. The paper uses two datasets: the CERT dataset and the UMDWikipedia dataset.

The methodology of the proposed framework involves pre-training a model on a large amount of unlabeled data using a self-supervised learning approach and then fine-tuning the model on a small amount of labeled data using a metric-based few-shot learning approach. The model consists of two transformer layers with a multi-head attention sub-layer and a feed-forward sub-layer.

The experimental environment involves training and testing the proposed framework on the CERT and UMDWikipedia datasets, and comparing its performance to existing anomaly detection approaches such as OCSVM, iForest, and RNN. The justification for using the selected method was that it addresses the limitations of existing approaches by leveraging self-supervised pre-training to learn useful representations from unlabeled data and using metric-based few-shot learning to adapt to new users with limited labeled data.

The results of the paper demonstrate that the proposed framework outperforms existing anomaly detection approaches in terms of precision, recall, F1 score, and false positive rate on both the CERT and UMDWikipedia datasets. The limitation of the proposed framework was that it requires a large amount of unlabeled data for pre-training, the CERT dataset contains activity logs of 4000 employees, and the UMDWikipedia dataset contains around 770000 edits with 17105 malicious and 17105 normal users.

Deep Learning

This chapter will cover a brief explanation of each deep learning algorithm, platform, activation & loss functions, dataset, and other topics that are used in this research.

3.1 Introduction to Deep Learning

Deep learning is a subset of machine learning that revolves around the utilization of artificial neural network architecture. These neural networks are inspired by the structure and functionality of the human brain's neurons, enabling them to process and learn from data in a complex manner.

The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data [26]. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering.

In a fully connected deep neural network, various layers of interconnected nodes, known as neurons, collaborate to process and understand the input data. Each neuron receives input from the previous layer or the input layer, and its output becomes the input for other neurons in the subsequent layer. This interconnected flow of information continues until the final layer produces the output of the network. Throughout this process, the layers of the neural network apply non-linear transformations to the input data, enabling the network to learn intricate representations of the data [27].

3.2 Deep Learning Algorithms

3.2.1 Convolutional Neural Network (CNN)

CNN, a type of feedforward neural network, excels in extracting features from data using convolution structures. Unlike conventional feature extraction methods, CNN eliminates the need for manual feature extraction. Its design draws inspiration from visual perception (how people see things) [28]. In this context, artificial neurons mirror biological neurons, while CNN kernels function as receptors sensitive to diverse features. Activation functions mimic the biological mechanism where only neural electric signals surpassing a specific threshold can be transmitted to the next neuron.

A Convolutional Neural Network (CNN) is comprised of an input layer, an output layer, and several hidden layers as shown in Figure 3.1. The hidden layers in a CNN commonly include convolutional layers, pooling layers, and fully connected layers[29].

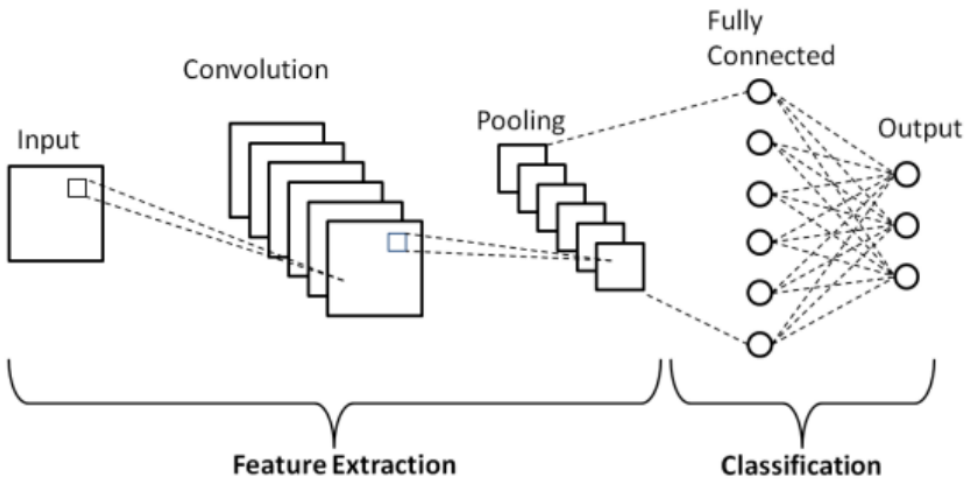

Figure 3.1: Convolutional Neural Network Architecture

Convolution layers:

The convolution layer is a fundamental component of the CNN and plays a central role in its computational process. Convolutional Layers utilize a moving filter, also known as the weight matrix, which slides over the input data through a convolution operation. This process generates a feature map as the output.

Pooling layers: The pooling layer decreases the volume of information in each feature acquired from the convolutional layer while preserving the crucial details, often involving multiple iterations of convolution and pooling. The pooling operation is performed individually on each slice of the representation.

Fully connected (FC) layers:

A fully connected layer is a type of neural network where each neuron performs a linear transformation on the input vector using a weights matrix. In this configuration, all possible connections between layers are established, allowing every input of the input vector to influence every output of the output vector.

Based on the dimensionality of the input space, CNN can be Conv1D, Conv2D, and Conv3D [30]. In a 1-dimensional CNN (Conv1D), the kernel operates in a single direction, allowing for left or right (horizontal) movements. This architecture is commonly employed in processing Time-Series data, where sequential movement is significant.

On the other hand, a 2-dimensional CNN (Conv2D) utilizes a kernel that moves in two directions, enabling movements in both the horizontal (left and right) and vertical (up and down) directions. It finds extensive application in tasks related to images, where spatial features in two dimensions are critical.

In the case of a 3-dimensional CNN, the kernel moves in three directions: horizontal, vertical, and depth-wise. As a result, the input and output data of a 3-dimensional CNN are represented in four dimensions. This architecture is particularly useful in scenarios like medical imaging, where data is captured in 3D slices and analyzed collectively.

3.2.2 Long Short-Term Memory (LSTM)

A Long Short-Term Memory network (LSTM) is a specific type of Recurrent Neural Network (RNN) designed to model sequences or time-related data, such as user activities. Standard RNNs face challenges when dealing with long sequences, leading to issues like vanishing or exploding gradients. However, LSTM addresses these problems by learning long-term dependencies using a new structure called a memory cell [20]. This memory cell consists of three key components: an input gate, a forget gate, and an output gate as shown in Figure 3.2 that enable LSTM to effectively learn long-term dependencies by managing interactions between the memory cell and its environment.

Forget Get: The forget gate plays a role in discarding irrelevant information which is no longer useful in the cell state.

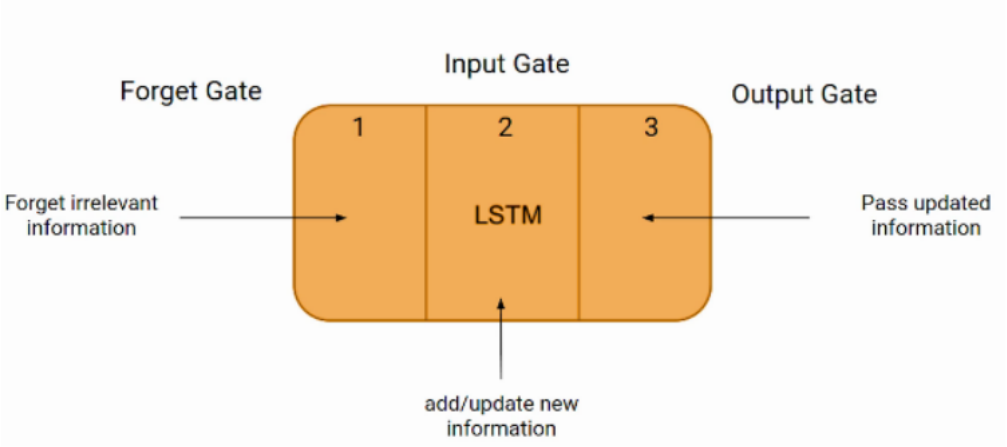

Figure 3.2: LSTM Memory cell

Input Gate: The input gate is responsible for incorporating valuable information into the cell state.

Output Gate: The role of the output gate involves extracting valuable information from the current cell state to be presented as output.

3.2.3 Few-shot Learning (FSL)

Few-shot learning is a machine learning paradigm that aims to train models to recognize new classes or concepts with only a small number of examples, or ”shots,” available for each class. Traditional machine learning models usually require a large amount of labeled data for each class to perform well. However, in real-world scenarios, collecting a substantial amount of labeled data for every new class might be impractical or time-consuming.

Few-shot learning addresses this limitation by enabling models to generalize from a small number of examples. It falls under the broader umbrella of transfer learning, where knowledge gained from one task or domain is leveraged to perform better on a related task or domain with limited data.

Few-shot learning is often inspired by the remarkable ability of humans to quickly learn new concepts with only a few examples. Human learning is characterized by the capacity to generalize knowledge from prior experiences to new situations, even when the available data is sparse. This ability to learn from limited examples is a fundamental aspect of human cognition and is closely related to the concept of few-shot learning in machine learning [31] [32].

Some important terms related to few-shot learning:

- Shot: The number of examples or instances available per class for training in a few-shot learning task.
- Support Set: The subset of the training data used to provide information about a few-shot learning task. It typically includes examples from different classes with limited instances.
- Query Set: The set of examples used to evaluate the model's performance after training on the support set. The model is tested on these examples to assess its ability to generalize from few-shot training.
- Episode: In few-shot learning, an episode refers to a single iteration of training and evaluation. It involves selecting a support set and a query set, training the model on the support set, and evaluating its performance on the query set.
- N-way K-shot learning scheme: is a frequently used term within the realm of Few-Shot Learning (FSL). It outlines the specific scenario a model addresses in FSL. In this context, "N-way" signifies the presence of "N" distinct new categories that a pre-trained model must extend its knowledge to encompass. A greater value of "N" implies a more challenging undertaking. On the other hand, the "K"-shot specifies the number of labeled instances accessible in the support set for each of the "N" novel classes [33].

There are various approaches to few-shot learning, including:

1. Matching Networks: These networks are designed to learn a similarity metric between examples in the support set and the query set, allowing the model to make predictions for new classes.
2. Meta-Learning or Learning to Learn: This approach trains models on a variety of tasks with the goal of enabling them to quickly adapt to new tasks with minimal data.
3. Siamese Networks: Siamese networks use a twin neural network architecture to learn the similarity between pairs of examples, often used for one-shot or few-shot image recognition.

4. Prototypical Networks: These networks compute a prototype for each class based on the support set examples and classify query set examples based on their similarity to these prototypes.

5. Transfer Learning and Pre-trained Models: By training models on large datasets for related tasks, they can be fine-tuned on smaller datasets for specific few-shot tasks.

Overall, few-shot learning has gained significant attention due to its potential to make machine learning and deep learning more applicable to situations where data is limited or rapidly changing.

1. **Prototypical Network (ProtoNet)**

   Prototypical networks are a type of neural network that can be used for few-shot learning. They learn a metric space in which classification can be performed by computing distances to prototype representations of each class. Compared to recent approaches for few-shot learning, they reflect a simpler inductive bias that is beneficial in this limited-data regime and achieves excellent results [34]. The concept behind prototypical networks is that there exists an embedding in which several points cluster around a single prototype representation for each class. It aims to learn per-class prototypes based on sample averaging in the feature space. By leveraging the distances between data points and these prototype representations, the prototypical network provides a robust and intuitive mechanism for categorization [35].

2. **Siamese Neural Network (SNN)**

   A Siamese network is a special type of neural network that uses identical sub-networks to compare input data for similarity. These sub-networks are like twins, having the same settings and sharing their learning. They work together to measure how similar or different two inputs are [36].

   Imagine the Siamese network as having two identical workers (the sub-networks) who look at different inputs. These workers then communicate and decide how similar the inputs are. Importantly, they both learn and think in the same way, so they don't give very different answers for very similar inputs. It's like having a mirror image of the same process. Most often Siamese networks are used when a machine is expected to learn from just a few examples and then apply that learning to new, unseen data. The structure of the network is replicated across the top and bottom sections to form twin networks with shared weight matrices at each layer as shown in Figure 3.3.

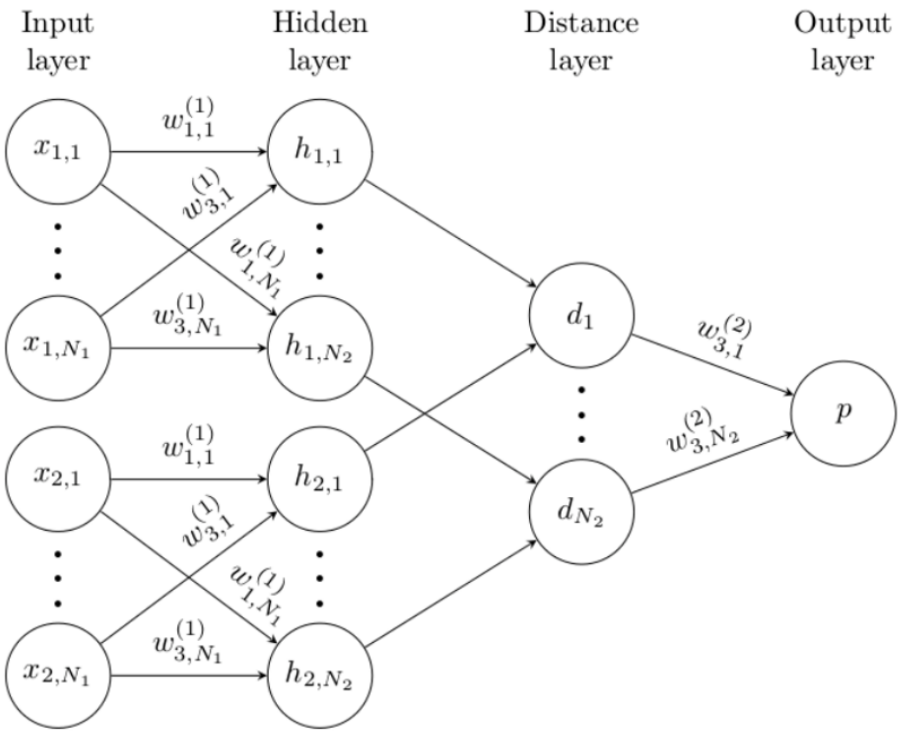

Figure 3.3: Siamese Neural Network Architecture

Contrastive Loss: A loss function used in Siamese networks to encourage similar inputs to be close in the feature space and dissimilar inputs to be far apart.

Embedding: The feature representation learned by the Siamese network for each input. Embeddings are vectors that capture the characteristics of the input data.

Distance Metric: A mathematical function used to measure the distance (similarity or dissimilarity) between two embeddings. Common distance metrics include Euclidean distance and cosine similarity.

3.3 Deep Learning Platforms

A deep learning platform serves as an essential toolkit and resource hub for the creation and deployment of deep learning solutions. These platforms empower users to swiftly construct, train, and implement powerful neural networks for a diverse range of applications, eliminating the need to start entirely from scratch. Fundamentally, a deep learning platform comprises several integral components. These include an artificial intelligence (AI) engine, tools for data management, development frameworks for crafting or refining models, and deployment options for pre-trained models in operational environments, offering readily deployable neural networks that require no additional training or customization. This accelerates the development process for developers and researchers, enabling them to quickly initiate projects without the necessity of building models from scratch [37] [38].

3.3.1 Tensorflow

This framework is a free, open-source, end-to-end machine learning platform. It is a symbolic math toolkit designed to handle a variety of deep neural network training and inference-related tasks by integrating data flow with differentiable programming. It allows developers to create machine learning software using a variety of tools, frameworks, and open-source materials [39] [40].

3.3.2 Keras

Keras, a rapidly expanding high-level Neural Network API, has been created in the Python language to facilitate the swift and straightforward implementation of Deep Learning applications. This versatile framework can operate with TensorFlow, Theano, or CNTK as its backend [40] [41].

3.4 Activation and Loss Functions

3.4.1 Activation Functions

In a neural network, activation functions are functions that compute a weighted sum of inputs and biases, which are then used to determine whether or not a neuron can be activated.

Since the neuron is unable to determine the activation pattern, it cannot effectively bind to value, as a result, an artificial neural network's activation function is crucial because it determines whether a neuron should be activated or not. In this way, it limits the value of the net input. Before transmitting the input to the next layer of neurons or completing it as output, it will be changed nonlinearly using the activation function [42] [43] [44] [45].

ReLU: Rectified linear unit (ReLU) is one of the linear activation functions, that, if the input is positive, will output the input directly; else, it will output zero. The ReLU function's primary advantage over other activation functions is that it does not simultaneously activate all of the neurons, rather if the input is negative, the ReLU function will convert it to zero and the neuron will not be activated [42] [43]. It can be represented mathematically as:

$$f(x) = max(0, x) \tag{3.1}$$

where x represents the input.

Sigmoid: is a mathematical function that can be used to solve binary classification and logistic regression tasks since it converts any input value to a value between 0 and 1 [43] [46]. It gives the model non-linearity, enabling the neural network to learn complex decision boundaries. Mathematically it can be represented as

$$\sigma(x) = \frac{1}{1 + e^{-}x} \tag{3.2}$$

where x represents the input value and e is the mathematical constant of 2.718.

tanh: is a hyperbolic function that is pronounced as "tansh". It is the ratio of the hyperbolic sine and hyperbolic cosine functions, which are related to the exponential function [45] [47]. The tanh function can also be written as

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{(e^{x}) - (e^{-}x)}{(e^{x}) + (e^{-}x)} \tag{3.3}$$

where x represents the input value and e is the mathematical constant of 2.718.

3.4.2 Loss Functions

Loss functions in deep learning play a critical role in training neural networks by optimizing the model's parameters. They serve as the measure of how well the model's predictions match the actual target values, allowing the network to adjust its parameters to minimize the error during training [48].

Binary Cross-Entropy Loss (BCE): used for binary classification problems. It measures the difference between predicted probabilities and binary target labels [48]. The BCE loss is defined mathematically as:

$$BCE(y_{\text{true}}, y_{\text{pred}}) = -\left[y_{\text{true}} * \log(y_{\text{pred}}) + (1 - y_{\text{true}}) * \log(1 - y_{\text{pred}})\right] \tag{3.4}$$

where y_{true} is the true binary labels (ground truth) for each data point which takes a value of 0 or 1 and y_{pred} is the predicted probabilities of the positive class by the model for each data point.

Mean Squared Error (MSE): measures the average squared difference between predicted values and true target values. MSE is commonly used as a loss function for regression problems, where the goal is to predict continuous numerical values. A lower MSE indicates a better fit of the model to the data. [49]. Mathematically MSE is calculated as:

$$MSE = \left[\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true}} - y_{\text{pred}})^2\right] \tag{3.5}$$

where, n is the number of data points in the dataset, y_{true} is the true value (ground truth) for the i-th data point and y_{pred} is the predicted value for the i-th data point

Contrastive Loss: is a method used in deep learning to teach neural networks how to understand the similarity between pairs of things. if we have a data point from different classes, and we want the computer to know which data point belongs to which class, Contrastive loss helps the computer figure this out. Mathematical contrastive loss can be expressed as:

$$L = (1 - Y) * D^2 + Y * max(0, m - D)^2 \tag{3.6}$$

Where, L is the contrastive loss, Y is the binary label (0 for similar pairs and 1 for dissimilar pairs), D is the Euclidean distance between the two representations and m is the margin, which is a hyperparameter that defines the lower bound distance between dissimilar samples.

3.5 Optimization and Normalization

3.5.1 Optimization and Optimizers

Optimization is a fundamental concept in various fields, including mathematics, engineering, and machine learning. In the context of machine learning and deep learning, optimization refers to the process of finding the best possible solution to a problem, typically characterized by minimizing or maximizing a certain objective function usually by using mathematical methods or techniques [50]. The choice of optimizer can significantly impact the convergence speed, stability, and final performance of the trained model [51].

Adam: Adaptive Moment Estimation (Adam) is a popular optimization algorithm used in training deep learning models that individually adapts learning rates for each parameter while maintaining a consistent learning rate for weight updates that remain constant during training. It has gained widespread adoption due to its efficiency, robustness, and ability to handle a variety of loss surfaces. [52].

Learning rate: serves as a crucial tuning hyperparameter within optimization algorithms for deep neural networks (DNNs). It plays a significant role in determining the step size during each iteration, guiding the algorithm toward the minimum of a given loss function. This hyperparameter effectively controls the balance between new and existing information, representing the speed at which a machine-learning model gains knowledge (”learns”) [53].

Batch-size: It refers to the number of training examples utilized in a single iteration during gradient descent or any optimization process[54].

Epochs: An epoch is a hyperparameter that refers to one cycle through the full training dataset, and dictates how many complete passes the learning algorithm will make over the training dataset.

3.5.2 Normalization

Normalization is a fundamental technique used in deep learning to preprocess input data before feeding it into neural networks. Its primary goal is to scale the input features to a standardized range, thereby improving the convergence and performance of the model during training [55].

Normalization addresses the issue of varying scales and ranges in input features, which can lead to slow convergence and make it challenging for the model to learn effectively. By bringing all features to a common scale, normalization ensures that no single feature dominates the learning process, allowing the model to focus on learning the underlying patterns in the data [56].

Min-Max scaling: Also known as feature scaling or normalization, is a popular technique used in data preprocessing for machine learning and deep learning. Its purpose is to transform the features of the dataset to a specific range, typically between 0 and 1, while preserving the relative relationships between the data points [56] [57]. Mathematically it can be expressed as:

$$m = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.7}$$

Where m is the new value, x is the original cell value, x_{min} is the minimum value of the column and x_{max} is the maximum value of the column.

3.5.3 Regularization

Regularization is a technique used in machine learning and statistics to prevent overfitting and improve the generalization of a model. It involves adding a penalty term to the loss function during training, discouraging the model from fitting the training data too closely and instead encouraging it to learn simpler and more general patterns.

Dropout: is one of the regularization techniques in neural networks to prevent overfitting. It works by randomly deactivating a fraction of neurons during each training iteration, which encourages the network to learn more robust and generalized features.

3.6 Integrated Development Environments (IDEs)

An integrated development environment (IDE) is a software application that provides a comprehensive set of tools and features to facilitate the development, testing, and debugging of computer programs. IDEs are designed to enhance the efficiency and productivity of software developers by providing a centralized environment where they can write, edit, compile, and debug code. These environments typically offer a range of features tailored to specific programming languages or domains, making them indispensable for various software development tasks [58] [59].

Anaconda: It is a widely used and comprehensive open-source platform designed for data science, machine learning, and scientific computing. It provides a powerful and flexible environment that simplifies the process of managing software packages, creating virtual environments, and conducting data analysis [60].

Google Colab: Google has introduced an exciting initiative by offering a free cloud service named Colab, which is based on Jupyter Notebooks and includes access to free GPUs. Colab is a versatile platform suitable for a wide range of tasks, from enhancing Python programming skills to working with popular deep-learning libraries such as PyTorch, Keras, TensorFlow, and OpenCV [61].

Jupyter Notebook: is an interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. Originally developed for the Python programming language, Jupyter Notebook supports various programming languages, making it a versatile tool for data analysis, scientific computing, machine learning, and more [62].

3.7 Dataset

To evaluate the performance of the proposed method, a suitable dataset is essential for profiling user behaviors based on log data. In this study, we employed the CMU CERT Insider Threat Datasets [63], which were collaboratively created by the CERT Division in partnership with Exact Data. These datasets consist of synthetic, unlabeled data, including both synthetic base data and synthetic malicious user data, making them highly versatile [7]. The dataset encompasses data from 1000 users, generated across five distinct activity types. These activities span a time period of 18 months, starting in December 2009 and ending in May 2011. The CERT dataset covers a wide range of domains and encompasses various computer-based log events/activities as shown in Table 3.1 such as logon/logoff, file open/closed events, records of visited websites, thumb drive Connect/Disconnect events, and email message logs [64].

For our research, we utilized the r4.2 dataset, which contains diverse event logs from a multitude of users and 70 are malicious insiders. Comparatively, version/release 4.2 has a larger number of instances when compared to other versions, and its size is 12GB. Despite the availability of several releases of the data sets, we chose r4.2 due to its substantial representation of insider users across multiple scenarios, making it a suitable choice for adaptation in this study [65].

Table 3.1: CERT insider threat test dataset r4.2

File name	Column Names	No. of Activities	No. of malicious and normal activities
device.csv	id, date, user, pc, activity	405381	Malicious: 44134, Normal: 361247
logon.csv	id, date, user, pc, activity	854860	Malicious: 49527, Normal: 805334
file.csv	id, date, user, pc, filename, content	445582	Malicious: 28683, Normal: 416899
http.csv	id, date, user, pc, url, content	1048576	Malicious: 71916, Normal: 976660
email.csv	id, date, user, pc, to, cc, bcc, from, size, attachments, content	1048576	Malicious: 72226, Normal: 976350

3.7.1 Insider Threat Scenarios

The CERT Insider Threat Dataset offers valuable insights into real-world instances of insider threats, enabling the identification of patterns and behaviors that may otherwise go unnoticed. This dataset encapsulates a variety of scenarios as listed below that shed light on distinct manifestations of insider threats, each presenting unique challenges to the cybersecurity landscape.

1. User who did not previously use removable drives or work after hours begin logging in after hours, using a removable drive, and uploading data to wikileaks.org. Leaves the organization shortly thereafter.
2. User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.

3. System administrator becomes disgruntled. Downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

4. A user logs into another user's machine and searches for interesting files, emailing to their home email. This behavior occurs more and more frequently over a 3-month period.

5. A member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain.

Through the exploration of these scenarios, we gain an understanding of the complex interplay between human behavior and digital security, providing a foundation for the development of proactive measures to mitigate and counter insider threats [66] [67]. The first three of the lists are scenarios used in this research which are found in r4.2 of the dataset.

3.7.2 Data Preprocessing

Directly from the source obtained data is likely to have errors, inconsistencies, or—most importantly it will not be ready for a data mining process. In addition, the growing volume of data used in modern scientific, industrial, and business applications requires the use of more sophisticated analytical tools. Data preprocessing makes the seemingly impossible possible by transforming the data to fulfill the input requirements of each data mining technique [68]. Data preprocessing is a crucial phase in the data mining and analysis process that entails preparing raw data for analysis by cleaning, converting, and integrating it [69].

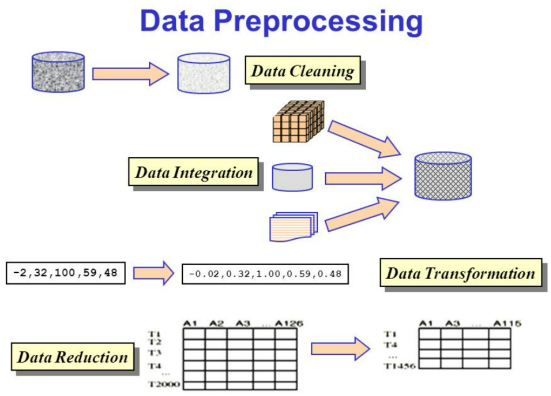

Figure 3.4: Steps included in data preprocessing

The purpose of data preprocessing is to enhance the data's quality and suitability for the particular data mining operation. Data cleaning, data transformation, data integration, and data reduction are some of the specific phases in data preprocessing. Finding and fixing mistakes or inconsistencies in the data, such as missing numbers, outliers, and duplicates, is known as data cleaning. Transforming data entails putting it in the appropriate format(s) for analysis and other operations that come after. Data integration is the process of merging information from many datasets. Data reduction involves reducing the dataset's size while keeping its crucial elements. Data preprocessing can be done using a variety of tools and techniques, such as feature engineering, feature selection, and sampling. The capacity of data preparation to assure the quality of data and the accuracy of the analysis results is what makes it so important [70].

Data Sampling: sampling encompasses the process of choosing a subset of data instances from a larger dataset. Its primary purpose is to curtail the dataset's size while retaining vital information. Various methodologies, including random sampling, stratified sampling, and systematic sampling, can be employed to achieve this objective [8].

Stratified sampling: Is employed when the source population, from which a sample is to be drawn, lacks homogeneity within its groups. Generally, this technique is used to procure a well-representative sample. In stratified sampling, the overall population is divided into distinct subgroups or strata, each of which is inherently more homogeneous than the entire population. The differences among these subpopulations are referred to as strata. From each stratum, a selection of items is made to form a sample. This approach ensures that each stratum closely mirrors the population, thereby generating more precise estimates. The advantage of this method lies in obtaining accurate estimates for each stratum, which in turn contributes to an enhanced overall estimate. Consequently, stratified sampling furnishes more dependable and detailed insights about the sample [71] [72].

TF-IDF: Stands for Term Frequency Inverse Document Frequency, measures the significance of a word within a text or collection of texts. It quantifies the importance of a word by considering how frequently it appears in the text while also factoring in its frequency across the entire dataset (corpus). This balance ensures that common words are given less weight, while those that are more specific to the text are highlighted [73].

NMF: Non-negative matrix factorization (NMF) is a machine learning technique that can be used to reduce the dimensionality of data and identify topics. It works by decomposing a matrix into two smaller matrices, each of which contains unnormalized probabilities. The first matrix called the document-topic matrix, shows how likely each document is to contain each topic. The second matrix called the topic-term matrix, shows how likely each topic is to contain each term. The resulting decomposed matrices have a smaller number of entries than the original matrix, which makes NMF a dimension-reduction method.

LDA: Latent Dirichlet allocation (LDA) is a statistical model that can be used to identify topics in a collection of text documents. It works by assuming that each document is a mixture of topics and that each topic is a distribution over words. The model is trained by estimating the probability of each word appearing in each topic, and the probability of each document containing each topic [74].

3.8 Feature Engineering

Feature engineering is the process of generating novel features or transforming existing ones to enhance the effectiveness of a model. In other words, it is the process of selecting, extracting, and transforming the most relevant and important features from the available data to build more accurate and efficient models [75] [76].

3.8.1 Feature Extraction

Feature extraction is the process of obtaining valuable information from raw data that can be utilized by machine learning or deep learning algorithms. The first layers of deep neural networks carry out feature extraction in deep learning. A machine learning or deep learning algorithm can more quickly absorb the most discriminant properties in data identified by the first layers of deep networks [77]. Feature extraction can be done manually or automatically, manual feature extraction requires the discovery and clarification of features that are important for a given problem and carrying out a way to extract those features, it also requires having a good understanding of the background or domain to make informed decisions as to which features might be useful. Automated feature extraction employs specialized algorithms or deep networks to autonomously derive features from raw datasets, eliminating the necessity for human involvement in the process [78].

3.8.2 Feature Selection

Feature selection is a process of removing redundant, irrelevant, and noisy features and choosing a subset of significant features to incorporate into model creation. Acts as a filter and refers to the automated procedure of picking attributes from the dataset (like columns in tabular data) that hold the utmost relevance to the task of modeling [79]. Distinguishing itself from dimensionality reduction, the process of feature selection involves the inclusion or exclusion of attributes within the data without altering their intrinsic characteristics.

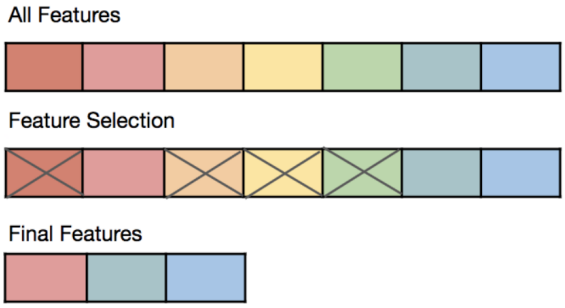

Figure 3.5: Feature selection concept in diagram

Feature selection can be done by removing features with low variance using *VarianceThreshold*, through univariate feature selectors such as *SelectKBest* & *SelectPercentile*, Recursive feature elimination (RFE), Tree-based feature selection and other techniques [80].

3.8.3 Feature Concatenation

Feature concatenation involves merging multiple features into a unified feature vector. While the values of the features are unchanged, their arrangement and data type may be altered. Concatenation is essential as it allows models to receive feature vectors as inputs [81].

3.8.4 Feature Scaling

Feature Scaling refers to the procedure of adjusting feature values to achieve a consistent scale. In the context of machine learning and deep learning, this holds significance as the scale of features can influence the model's effectiveness. Min-Max Scaling, Standard Scaling, Robust Scaling are some types of feature scaling techniques [82].

Experimental Analysis

4.1 Proposed System Model

The proposed system model's architecture is a combination of advanced deep-learning models for feature extraction and Few-Shot Learning for detection. Figure 4.1 shows the general structure of the proposed approach. Where,

- F_{cnn} and F_{Lstm} represent features extracted by CNN and LSTM respectively
- T1 & T2 represent the training using individual feature extraction phase outputs and
- T3 represents training with the concatenated features outputs

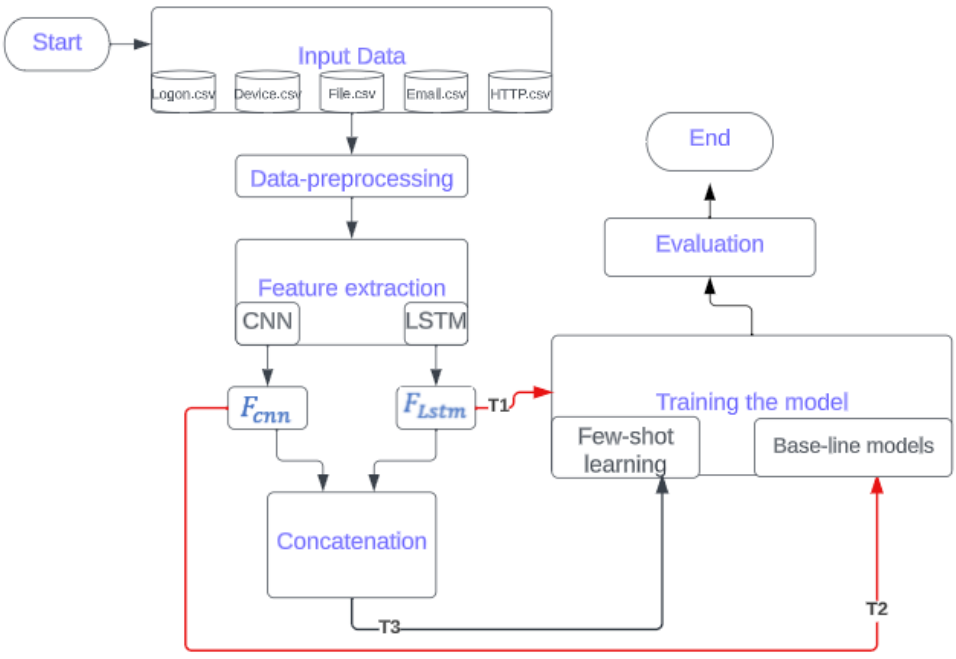

Figure 4.1: Overview of system model

4.2 Experimental Setup

An experimental framework was configured to assess and examine the importance and effectiveness of the proposed method and to answer the research question. The experimental environment was set up on a computer system equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz CPU, and 20 GB of RAM. The experimental process was conducted utilizing Anaconda distribution version 23.5.1, Python 3.10.9 integrated with Jupyter Notebook 5.7.1, and Tensorflow library version 2.10.0.

4.3 Data Gathering and Pre-processing

As shown in Figure 4.2, to preprocess the working dataset we follow some steps that are discussed in detail in this section.

1. **Data collection:** In this initial stage, the research endeavors to gather and preprocess the essential data required for the model's training. From the CERT r4.2 dataset, the number of sample users, datasets, and user activities are carefully identified to construct a representative sample for analysis. Emphasizing user behavior, the study exploits CERT's five distinct activity log files, namely Logon.csv, device.csv, file.csv, email.csv, and http.csv. The CERT dataset is unlabeled data, so the first step was labeling each dataset, and by using a stratified random sampling technique we took 1000 instances from each dataset.

2. **Data cleaning:** Handling the missing values by dropping them, removing irrelevant rows and columns, and data parsing such as converting the 'date' column of each csv file from a string format to a 'datetime' format to facilitate time-based analysis is done in this step.

3. **Topic modeling:** this step is for files with a 'content' column that are file.csv, http.csv, and email.csv which are labeled as content-files in the preprocessing. A series of functions for creating and saving topic models, specifically Term Frequency-Inverse Document Frequency (TF-IDF), Latent Dirichlet Allocation (LDA), and Non-Negative Matrix Factorization (NMF) models, from text data.

TF-IDF: Is used to determine the importance of each term in each document where each row corresponds to a document, and each column corresponds to a unique term (word) in the text corpus. It calculates weights for each term based on its importance within individual documents and across the entire corpus. Terms that are frequent within a document but rare across the corpus receive higher weights, indicating their significance in that document.

NMF: It is used in our preprocessing phase to factorize the TF-IDF matrix into two lower-dimensional matrices, where one matrix represents topics, and the other matrix represents the distribution of topics in documents and for dimensionality reduction purposes.

LDA: Is applied to the textual content associated with user activities in the 'content' column aiming to discover latent topics within a collection of contents. These topics represent underlying concepts present in the text. LDA assigns topics to each user's activities based on the distribution of words within the content.

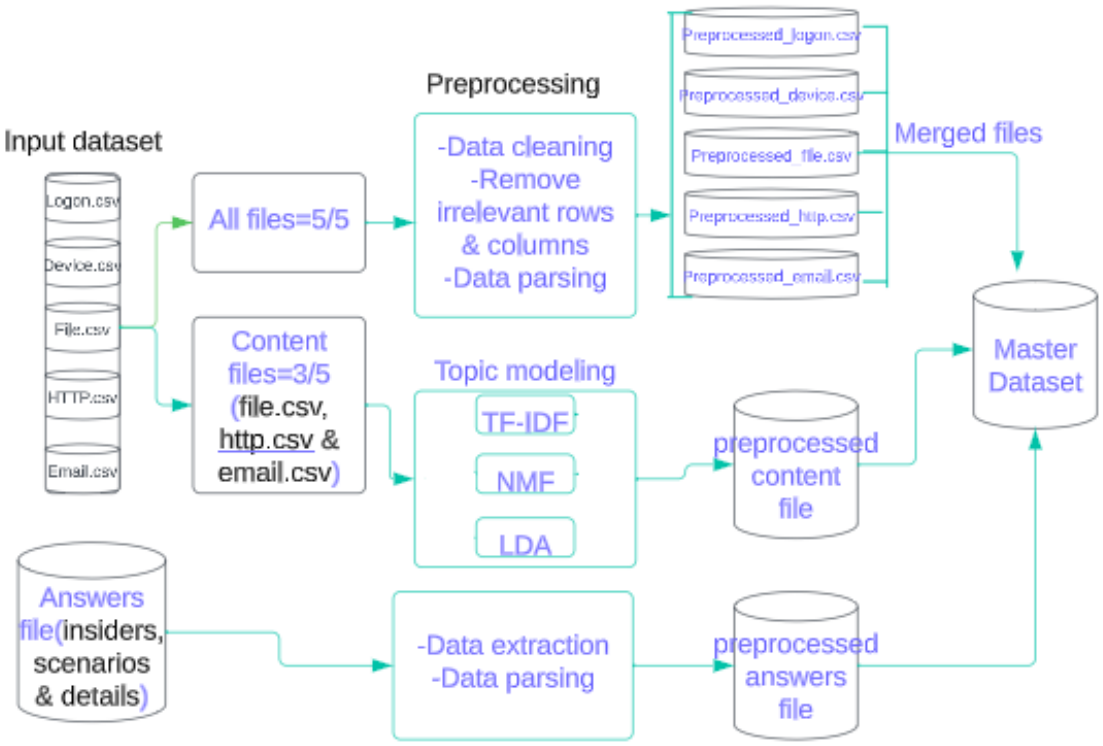

Figure 4.2: Overall Pre-processing Process

4. **Text-vectorization:** To convert the text in the "content" column of content-files into numbers, we perform a text vectorization process. Two used methods for text vectorization are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

Bag of Words (BoW): Using BoW, we create a vocabulary of all unique words (or tokens) in the text corpus. Then, for each ”content” column in the content-files, we represent it as a vector where each element corresponds to the count of a word from the vocabulary in the document.

TF-IDF: Using TF-IDF, we create a vector for each document, where each element represents the importance of a word in the document relative to its importance in the entire corpus. TF-IDF takes into account both the term frequency (how often a term appears in a document) and the inverse document frequency (how unique a term is across all documents).

To give some clarification on how these methods work with the texts in the content-files of our data preprocessing step let us take some sample of texts from the raw dataset and see how they will be represented numerically. Let ”remain a representative of remain the consensus a concert although to connect component to collaborated” be a text that we want to convert into a numeric format. The steps to be followed are:

Tokenization: Split the sentence into individual words or tokens. For example, tokenize the sentence into: [”remain”, ”a”, ”representative”, ”of”, ”remain”, ”the”, ”consensus”, ”a”, “concert”, ”although”, ”to”, ”connect”, ”component”, “to”, ”collaborated”, ”data”, ”into”, ”his”, ”Dropbox”].

Stop Word Removal: Remove common words like ”a”, “of”, ”the”, and ”to” if they are not relevant to the analysis.

Text-vectorization: Applying the BoW to convert the preprocessed sentence into a numerical representation we will have a result as shown in Figure 4.3.

5. **Normalization:** We used a Min-Max scaling normalization technique to re-scale the numerical data in each file to have values within a specified range of between 0 and 1.

6. **Merging:** From the previous steps, we get a numerical representation of each file that has been saved as preprocessed-filename.csv. The merging process is first applied to the all-files that hold the five .csv log files and then to content-files which hold the three .csv log files, finally, we merge the merged content-files with merged all-files horizontally by “id” using inner join keeping only the rows with matching ”id” values in both datasets. This enables us to have a unified preprocessed master dataset forming the bedrock for subsequent feature extraction.

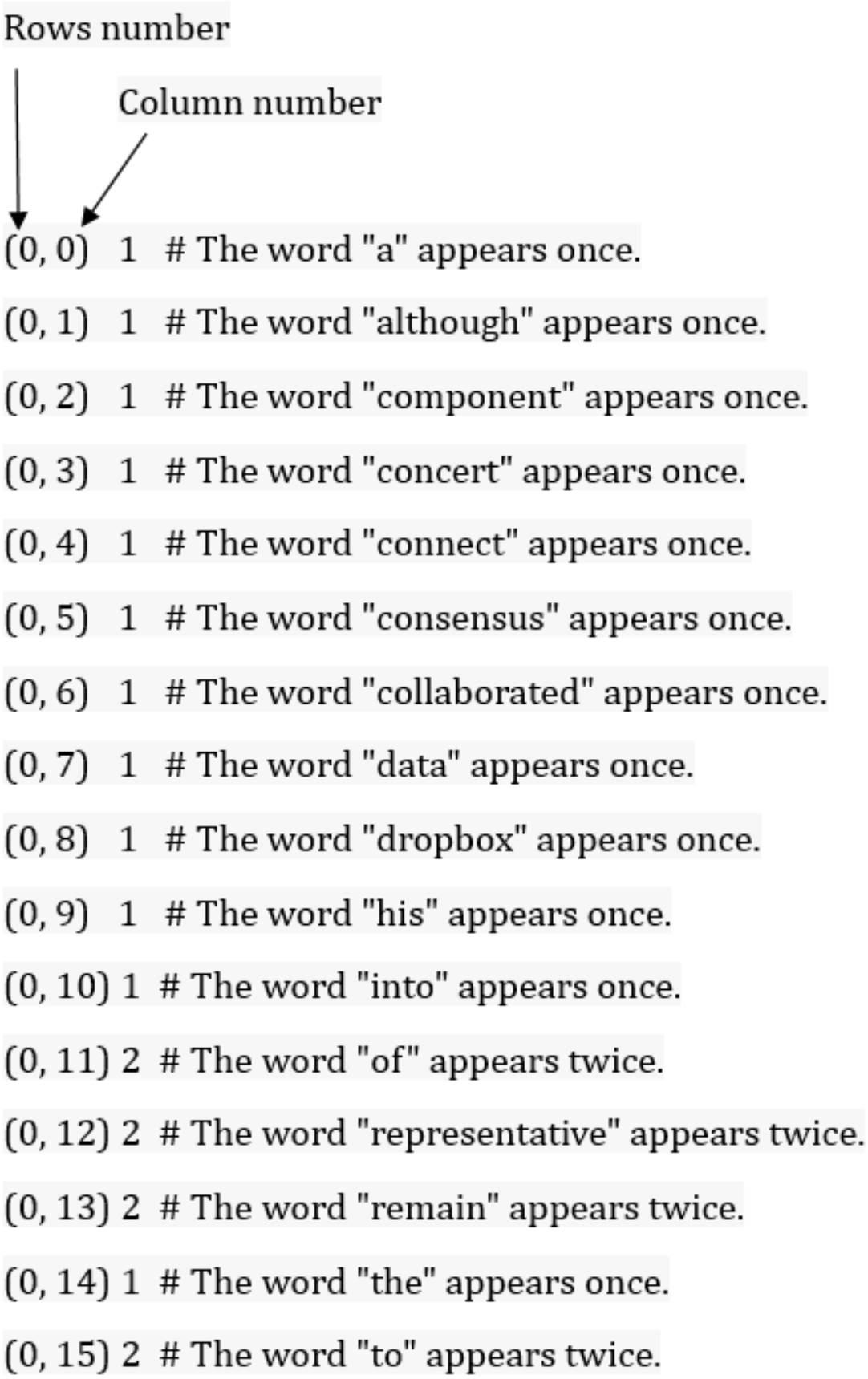

Figure 4.3: Output of text-vectorization example

4.4 Feature Extraction

We do data preprocessing using cleaning, parsing, topic modeling, and Normalization then apply CNN and LSTM separately to extract features from the preprocessed dataset to get corresponding extracted features, CNN extracts spatial patterns from the activity log files, while LSTM captures temporal and sequential behavior patterns. Both models are separately applied to the preprocessed master dataset to generate corresponding extracted feature sets.

Employing the LSTM with four LSTM layers, two dense layers, and CNN with four Conv1D and four max-pooling layers and other parameters as shown in Table 4.1, the research carefully extracts features from the last layers of each model, just before the dense (fully-connected) layers.

Table 4.1: Parameters used in feature extraction

Model	Parameter Name	No. of Parameters	Units
CNN	Conv1D layers	4	128,64,32 &1 6 with kernel=2
	Dense layers	2	16 & 1
	Maxpooling layer	3	poolsize=2
	Dropout rate	3	rate=0.5
	Epochs	10	-
	Number of folds	10	-
	Batch-size	32	-
LSTM	Lstm layers	4	128,64,32 &1 6
	Dense layers	2	16 & 1
	Dropout rate	3	rate=0.5
	Epochs	10	-
	Number of folds	10	-
	Batch-size	32	-

To ascertain whether a user activity is normal or malicious, the pivotal attributes of ”working time” and ”action-id” are utilized as key identifiers. Specifically, ”working time” is precisely defined from 08:00 AM to 5:00 PM. To determine the ”action-id”: ”PC”, and ”Date” features are incorporated to ascertain whether the used PC is self-PC or not. By counting the number of ”Date” that ”PC” was engaged with the specific ”User”, the more engaged ”PC” is considered as ”self-PC”, thereby enabling the system to flag any user logins occurring before/after working time or from different computers rather than ”self-PC” considered as potential malicious activities.

4.4.1 CNN Training for Feature Extraction

The model built by using one dimensional CNN (Conv1D) is trained to extract relevant and informative features from user behavior-based-preprocessed master dataset using Stratified K-Fold Cross-Validation with a Number of folds (num-folds) =10, Epochs = 10, batch-size = 32, learning rate = 0.0001, Dropout rate = 0.5, padding = ”same”, the activation function is “relu” and optimizer used is “Adam”. We use Conv1D because our input data is one-dimensional data which is represented as a sequence of comma-separated value (csv) data. The architecture of CNN used in this thesis is as shown in Figure **??**.

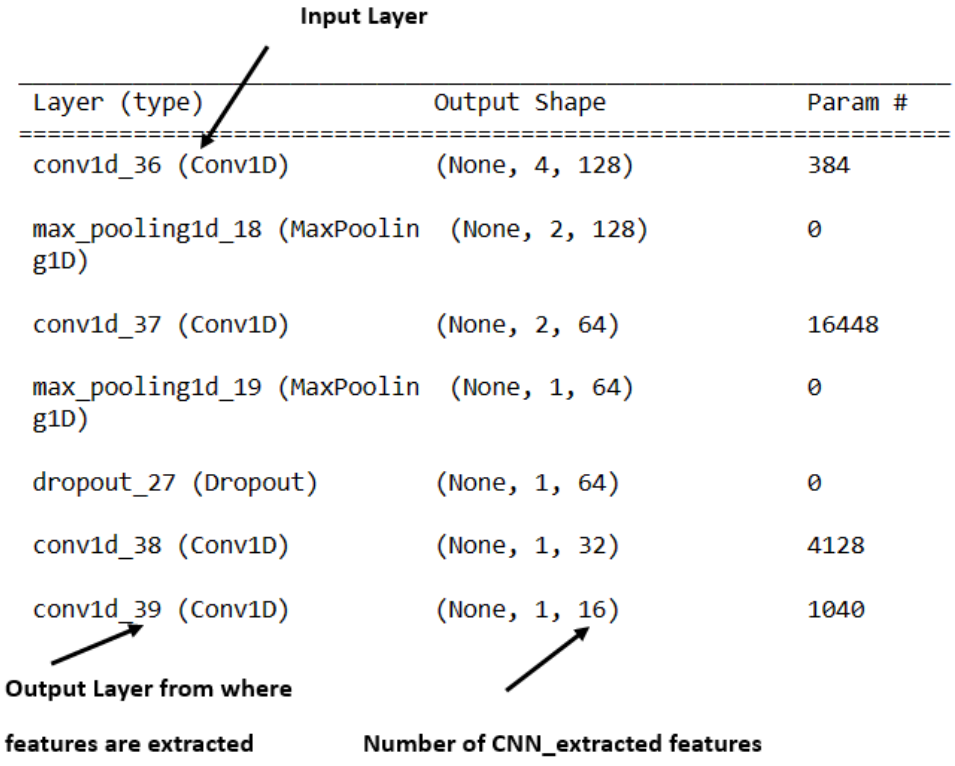

Figure 4.4: CNN model summary

The dissimilarity between the predicted probability distribution and the true binary labels for each data point in the dataset is calculated by using the binary-cross entropy (BCE) loss function. The binary-cross entropy loss of the model over the fold is plotted as shown in Figure 4.5

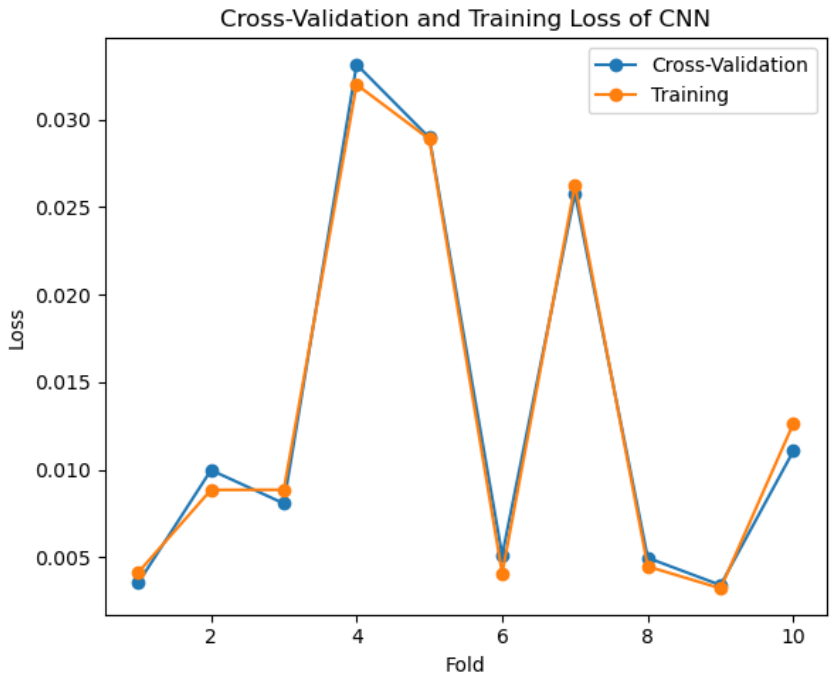

Figure 4.5: binary-cross entropy loss of CNN over fold

4.4.2 LSTM Training for Feature Extraction

The model built by using LSTM is trained again to extract relevant and informative features from a user behavior-based-preprocessed dataset with a Number of folds (num-folds) =10, Epochs = 10, batch-size = 32, learning rate = 0.0001, Dropout rate = 0.5, the activation function is “relu” and optimizer used is “Adam”. The architecture of LSTM is shown in Figure 4.6

The average squared difference between the predicted and true values in the dataset or the loss function is calculated as the Mean Squared Error (MSE). The Mean Squared Error loss of the LSTM model over the fold is plotted as shown in Figure 4.7.

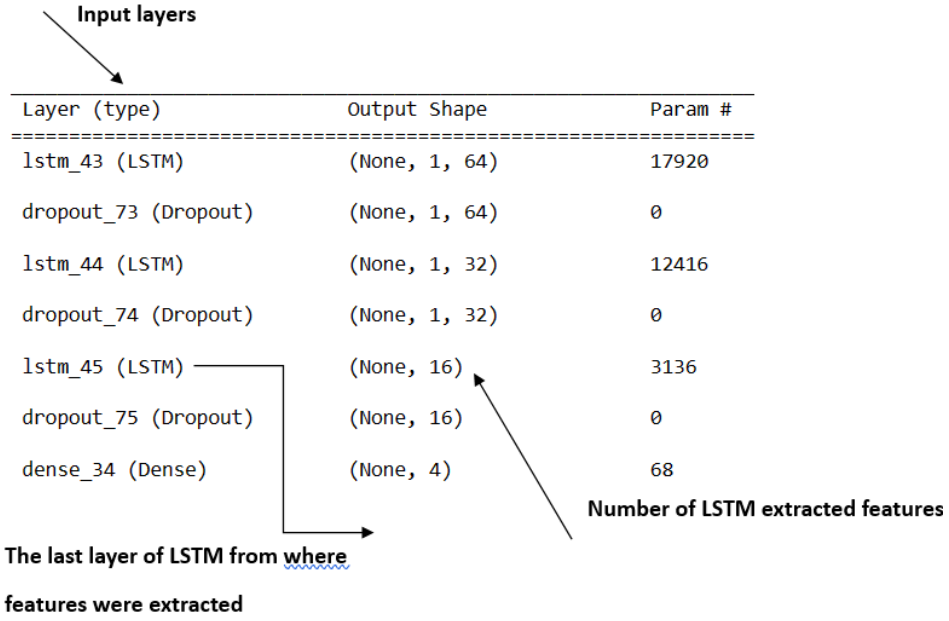

Figure 4.6: LSTM model architecture

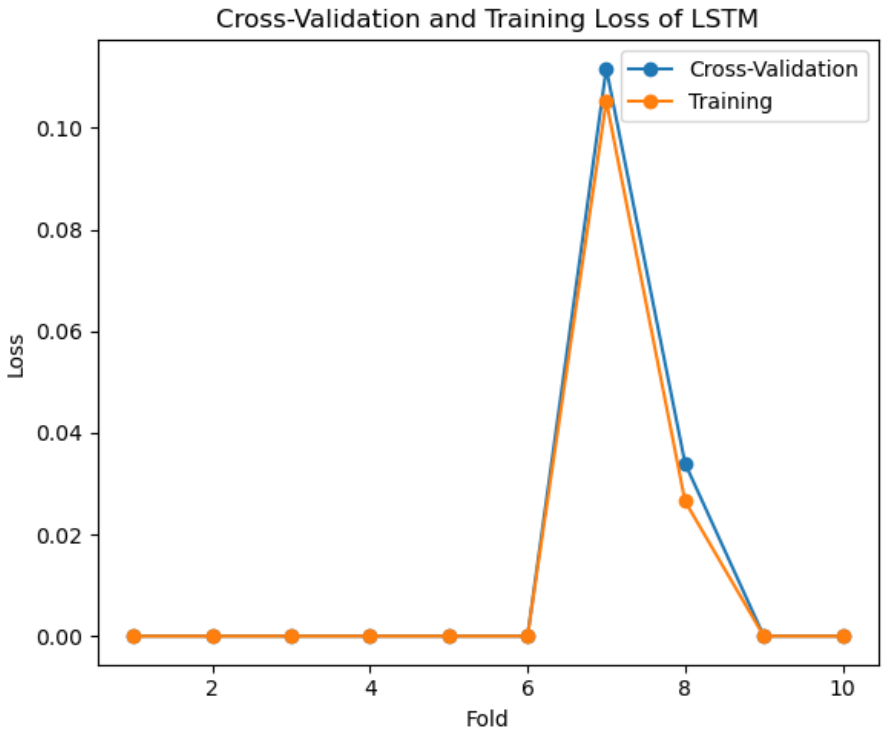

Figure 4.7: MSE loss of LSTM over the folds

During the training phase for feature extraction, the utilization of Stratified K-Fold cross-validation is implemented. This technique is employed to ensure that the training dataset is divided into folds while maintaining the distribution of classes across the folds aiming to maintain the proportionality of class instances in each fold and to prevent any biase in class distribution that could potentially bias the model training process.

In addition to this, we have introduced a separate extractor model that serves the purpose of deriving features from the last layers of both the CNN and LSTM models. As a result of this feature extraction, we have a cumulative total of 32 distinct features 16 CNN-extracted-features and 16 LSTM-extracted-features at our disposal.

4.5 Feature Concatenation & Selection

4.5.1 Feature Concatenation

The two sets of features obtained from CNN and LSTM are concatenated to create a unified and comprehensive feature representation of user behavior. The concatenated feature set captures a rich combination of visual, spatial, temporal, and sequential patterns, offering a more holistic representation of the data. This merging process ensures that the resulting feature set captures a comprehensive representation of user behavior, effectively harnessing the insights gleaned from both models.

4.5.2 Feature Selection

At this point, we have the concatenated and unified file that holds the extracted and concatenated features from both models. Before selecting features we drop the constant features that occur in both files using VarianceThreshold and we left with 28 unique features. From those 28 unique features, we selected 20 of them using the SelectKBest method.

4.6 Insider Threat Detection

The insider threat detection experiment is done by using three datasets: the CNN-extracted features dataset, LSTM-extracted features dataset and selected-features dataset from Section 4.4 and 4.5 are fed into a siamese neural network (SNN), which has two identical subnetworks. The term ”identical” in this context refers to their shared configuration, including their parameters and weights. It compares the feature vectors of the inputs to determine how similar they are. It uses only a few numbers of samples to learn and identify the unseen which operates within the framework of Few-Shot Learning. The siamese neural network's architecture is designed with parameters as shown in Table 4.2 to efficiently learn and classify normal and malicious insider threats, even in scenarios with limited labeled data, making it capable of detecting new and unseen insider threat patterns.

Table 4.2: Parameters used in SNN

Parameters	Number of parameters
Dense layer	2 with units of 128,64
Activation function	Relu and sigmoid
Epochs	30
Batch-size	32

We follow some steps to work with SNN:-

Siamese Architecture: A Siamese neural network consisting of two identical subnetworks (often referred to as ”twins”) that share the same architecture, weights, and parameters is designed by using dense layers with relu and sigmoid activation functions with an embedding dimension of 32. We set the margin and threshold to be 1 and 0.5 respectively. These twin networks take in pairs of data points as input.

Feature Extraction: Each input data point is passed through its respective twin network. These networks perform their own feature extraction and transform the input data into a lower-dimensional feature vector. The goal is to ensure that similar data points have similar feature representations.

Metric Learning: After feature extraction, the Siamese network computes a distance or similarity metric between the feature vectors produced by the twin networks using Euclidean distance. The network learns to minimize this metric for similar pairs and maximize it for dissimilar pairs.

Training: During training, the Siamese network is provided with pairs of data points and their corresponding labels indicating whether the pairs are similar or dissimilar. The dataset is split into train and test using the Sklearn library for train-test-split in the ratio of 80:20. The 80% was for training the model and 20% was for testing the model performance. The network is trained using a contrastive loss function that encourages the feature vectors of similar pairs to be close in the feature space and those of dissimilar pairs to be far apart.

By the time of training, we select two pairs, positive pairs, and negative pairs.

Positive Pairs: refers to a pair of input samples that belong to the same class either class-0 (Normal) or class-1 (Malicious) and are fed into the Siamese network together as input. These pairs represent positive samples that should have similar feature representations.

Negative Pairs: refers to a pair of input samples that belong to a different class meaning they are drawn from both classes. These pairs represent negative samples that should have dissimilar feature representations.

Contrastive loss: We use a contrastive loss function to train the Siamese network. This loss encourages positive pairs to have low dissimilarity (small distance) and negative pairs to have high dissimilarity (large distance) in the feature space. The loss function computes the Euclidean distance between the feature vectors of the anchor and comparison samples where the anchor refers to the first input sample in a contrastive loss.

The training objective is to make the Siamese network learn to minimize the contrastive loss. It aims to bring feature representations of similar (positive) pairs closer together while pushing apart feature representations of dissimilar (negative) pairs by a margin.

In the detection phase, the Siamese neural network effectively classifies user behaviors into two categories: normal and malicious insider threats. Siamese networks classify inputs by learning their similarity. We feed a pair of inputs to these networks, and each network computes the features of one input. Then, the similarity of features is computed using their difference or the dot product. For the same class input pairs, the target output is 1, and for different class input pairs, the output is 0.

4.7 Model Evaluation

In this phase, the proposed method's performance was assessed through a comprehensive evaluation using essential measurement metrics, including precision, recall, F1-score, and FNR (False Negative Rate). A detailed comparative analysis is also undertaken, wherein the proposed approach is compared against existing insider threat detection methods, such as Recurrent Neural Network (RNN), eXtreme Gradient Boosting(XGBoost), and Isolation Forest (iF).

4.7.1 Performance Measurement Metrics

In this research four performance measurement metrics: precision, recall, F1 score, and FNR are used to evaluate the performance of the proposed approach.

Precision: Measures the proportion of correctly predicted positive instances out of all instances that were predicted as positive by a model. In other words, it quantifies the accuracy of the positive predictions made by a model. Mathematically:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Where True Positives (TP) are the instances that are correctly predicted as positive. False Positives (FP) are the instances that are incorrectly predicted as positive when they are actually negative.

Recall: Also known as Sensitivity or True Positive Rate, is a metric used in classification tasks to measure the proportion of correctly predicted positive instances out of all actual positive instances. In other words, it quantifies the model's ability to correctly identify all positive instances. The recall formula is:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Where True Positives (TP) are the instances that are correctly predicted as positive. False Negatives (FN) are instances that are incorrectly predicted as negative when they are actually positive.

F1 score: The F1 score, also known as the F1 measure or F1-score, is a metric that combines both precision and recall into a single value, providing a balanced measure of a model's performance in binary classification tasks. It is the harmonic mean of precision and recall, giving equal weight to both metrics.

The F1 score considers both false positives and false negatives, making it a good indicator when the class distribution is imbalanced. The F1 score is calculated using the following formula:

$$F1Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.3}$$

Where Precision is the ratio of true positive predictions to the total predicted positives and Recall is the ratio of true positive predictions to the total actual positives.

FNR (False negative rate): measures the ratio of false negatives to the total number of actual positives in a classification problem. In other words, it quantifies the rate at which the model incorrectly predicts the negative class when the true class is positive. Mathematically, the FNR is calculated as:

$$FNR = \frac{FN}{(FN + TP)} \tag{4.4}$$

Where False Negatives (FN) are the instances that the model incorrectly classified as negative when they are actually positive, True Positives (TP) are the instances that the model correctly classified as positive

Result and discussion

This chapter presents all the results found from testing and evaluating the proposed model and models that are taken for comparative study purposes.

5.1 Experimental Results

In this section, the experimental results which are found from using the three training datasets: CNN-Extracted features dataset, LSTM-extracted features dataset, and Selected-features dataset, then evaluated on the corresponding testing dataset is discussed. To evaluate the performance of the model performance measurement metrics such as Precision, Recall, F1 score, and FNR are used. The experimental results are shown in Figure 5.1.

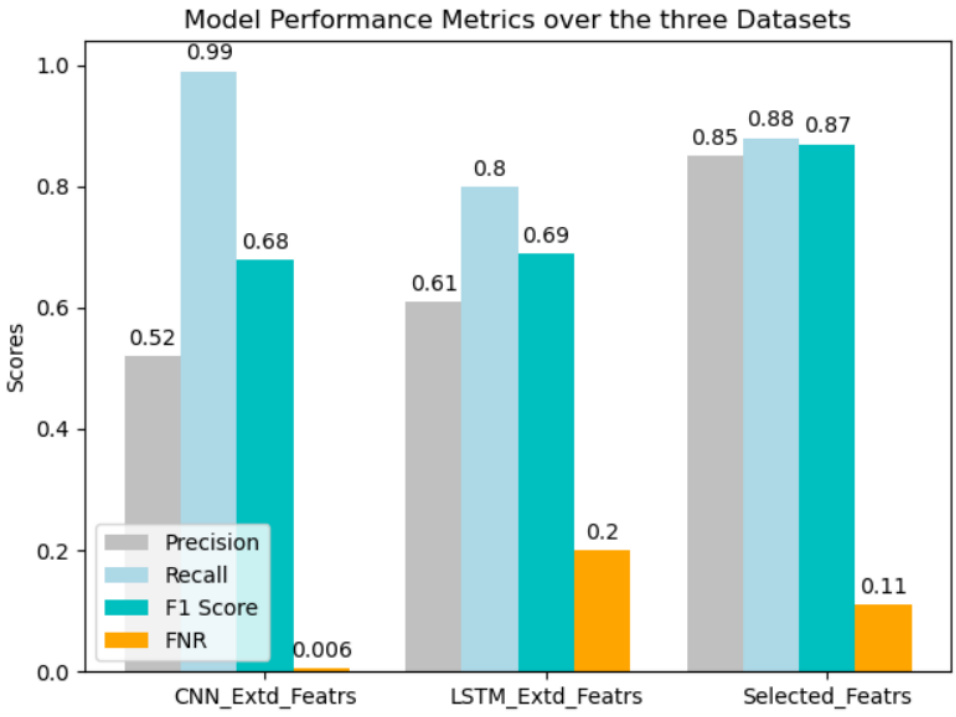

Figure 5.1: Performance evaluation of SNN over three datasets

The given result shown with the bar chart of Figure 5.1, shows that with the experiment done using CNN-extracted features datasets, SNN performs with an F1 score of 68% and an FNR (False Negative Rate) value of 0.006. In the second experiment which is done using LSTM-extracted features datasets, the results of SNN in terms of an F1 score is 69% and an FNR value of 0.2. The last result is obtained using the Selected-features dataset and SNN best performs in terms of F1 score with 87% and in terms of FNR, it has the lowest value 0.11.

5.1.1 Comparative Study

To compare the proposed approach's performance to models that are recently used in the insider threat detection realm, we run isolation Forest, XGBoost, and RNN using the three datasets, and the result is shown in the Table 5.1

This table shows the performance of the proposed model and baseline models, where the 'Dataset' column refers to the result for which dataset, the 'Models' column indicates the employed models, while the 'Performance metrics' column shows the models' performance result with the corresponding dataset for a given measurement metrics.

Table 5.1: Comparative study Results

Dataset	Models	Performance Metrics			
		Precision	Recall	F1 score	FNR
CNN-extracted	iF	0.50	0.50	0.50	0.90
	RNN	0.83	0.57	0.60	0.87
	XGBoost	0.78	0.59	0.63	0.81
	SNN	0.52	0.99	0.68	0.006
LSTM-extracted	iF	0.58	0.57	0.58	0.81
	RNN	0.92	0.62	0.68	0.76
	XGBoost	0.90	0.58	0.62	0.84
	SNN	0.61	0.80	0.69	0.20
Selected-featurs	iF	0.85	0.74	0.78	0.51
	RNN	0.97	0.70	0.78	0.59
	XGBoost	0.92	0.70	0.77	0.59
	SNN	0.85	0.88	0.87	0.11

In the case of the CNN-extracted dataset, the worst performance was obtained with isolation forest (iF) at 50% and 0.9 in terms of both F1 score and FNR, respectively. The best performance was obtained with siamese neural network (SNN) at 68% and 0.006 both in terms of F1 score and FNR respectively.

In the case of the LSTM-extracted dataset, the worst performance was again obtained with isolation forest (iF) at 58% and 0.81 in terms of both F1 score and FNR, respectively. The best performance was obtained with siamese neural network (SNN) at 69% and 0.20 both in terms of F1 score and FNR respectively.

In the case of the Selected-features dataset which holds the features extracted from both CNN and LSTM together, the worst performance was obtained with XGBoost at 77% and 0.59 in terms of both F1 score and FNR, respectively. The best performance was obtained with siamese neural network (SNN) at 87% and 0.11 both in terms of F1 score and FNR respectively.

According to the experimental result summary, the proposed approach outperforms in each and every one of the three experiments with the highest F1 score percentage and lowest FNR value.

5.2 Discussion

In this work, insider threat detection task has been carried out using Deep Learning algorithms. From the result, we can observe that if we use combined Deep Learning algorithms such as LSTM and CNN for feature extraction we can have features with the associated ability of CNN to learn spatial features and the capability of LSTM to learn the temporal dependencies. As we can see from the experimental results, having these two leveraged abilities of the deep learning models makes a difference in detecting malicious user activities.

The research question raised in section 1.3 is:

What is the effect of using combined CNN and LSTM for feature extraction with a few-shot learning model for insider threat detection?

CNN and LSTM are powerful deep-learning models that can extract meaningful features from raw data. We have seen in the result that using these models for feature extraction improves the performance of insider threat detection. CNN and LSTM are computationally intensive models that require a lot of data and computing power to train. However, by using few-shot learning, we can reduce the amount of data required for training, which can significantly speed up the training process. Few-shot learning is a technique that allows models to learn from a small number of examples. By combining few-shot learning with CNN and LSTM, we improved the generalization of the model and made it more robust to new and unseen threats.

Conclusion and Future Work

6.1 Conclusion

Insider threat is the risk when an insider will use their access to an organization or their familiarity with it to cause harm to that organization. The existing approaches suffer from data scarcity, class imbalance, and feature representation. To address these gaps we proposed an approach that uses CNN and LSTM as feature extractors and the Siamese Neural Network of Few-shot learning for insider threat detection tasks. By using CNN and LSTM we can have features that comprise the spatial features and features learned from LSTM the ability to learn temporal dependencies. Using SNN of few-shot learning enables us to detect insider threats with a small number of labeled datasets for malicious insiders in imbalanced class scenarios. To test the performance of our model we use the CERT insider threat dataset r4.2. We evaluate the performance of the proposed approach using precision, recall, F1 score, and FNR. We use the existing baseline models that have been used for insider threat detection such as RNN, XGBoost, and isolation forest to compare our proposed approach's performance with them.

The experimental result shows that with the experiment done using CNN-extracted features datasets, the proposed model best performs with an F1 score of 68% and an FNR (False Negative Rate) value of 0.006, using LSTM-extracted features datasets, the results of SNN in terms of an F1 score is 69% and an FNR value of 0.2, and with the Selected-features dataset, the proposed model outperforms in terms of F1 score having 87% and 0.11 FNR.

6.2 Future Work

Insider threats have dynamic user behavior and have the potential to do a lot of damage in a very short time. Therefore, to reduce this effect, real-time insider threat detection is crucial. We recommend future researchers who are interested in this research area, to study real-time insider threat detection.

References

[1] DEFINING INSIDER THREATS. https://www.cisa.gov/defining-insider-threats. Accessed: 20-10-2022.

[2] CERT Definition of 'Insider Threat'. https://insights.sei.cmu.edu/blog/cert-definition-of-insider-threat-updated/. Accessed: 20-10-2022.

[3] Rida Nasir, Mehreen Afzal, Rabia Latif, and Waseem Iqbal. Behavioral based insider threat detection using deep learning. *IEEE Access*, 9:143266–143274, 2021.

[4] Veriato insider threat report 2018. https://www.veriato.com/resources/whitepapers/insider-threat-report-2018.pdf. Accessed: 26-10-2022.

[5] 2022 Cost of Insider Threats Global Report. https://protectera.com.au/wp-content/uploads/2022/03/The-Cost-of-Insider-Threats-2022-Global-Report.pdf. Accessed: 30-10-2022.

[6] Fangfang Yuan, Yanan Cao, Yanmin Shang, Yanbing Liu, Jianlong Tan, and Binxing Fang. Insider threat detection with deep neural network. In *Computational Science – ICCS 2018: 18th International Conference, Wuxi, China, June 11–13, 2018, Proceedings, Part I*, page 43–54, Berlin, Heidelberg, 2018. Springer-Verlag.

[7] Ahmed Saaudi, Zaid Al-Ibadi, Yan Tong, and Csilla Farkas. Insider threats detection using cnn-lstm model. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 94–99, 2018.

[8] Abbas Yazdinejad, Ali Dehghantanha, Reza Parizi, Gautam Srivastava, and Hadis Karimipour. Secure intelligent fuzzy blockchain framework: Effective threat detection in iot networks. *Computers in Industry*, 144:103801, 01 2023.

[9] Angad Pal Singh and Ankit Sharma. A systematic literature review on insider threats, 2022.

[10] Elie Alhajjar and Taylor Bradley. Survival analysis for insider threat. *Computational and Mathematical Organization Theory*, 28(4):335–351, 2022.

[11] Duc C. Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021.

[12] Filip Bartoszewski, Michael Just, Michael A. Lones, and Oleksandr Mandrychenko. Anomaly detection for insider threats: An objective comparison of machine learning models and ensembles. In Audun Jøsang, Lynn Futcher, and Jan Hagen, editors, *ICT Systems Security and Privacy Protection. SEC 2021*, volume 625 of *IFIP Advances in Information and Communication Technology*, pages 367–381. Springer, 2021.

[13] R. G. Gayathri, Atul Sajjanhar, Yong Xiang, and Xingjun Ma. Anomaly detection for scenario-based insider activities using cgan augmented data. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 718–725, 2021.

[14] Liu Liu, Chao Chen, Jun Zhang, Olivier De Vel, and Yang Xiang. Doc2vec-based insider threat detection through behaviour analysis of multi-source security logs. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 301–309, 2020.

[15] Junhong Kim, Minsik Park, Haedong Kim, Suhyoun Cho, and Pilsung Kang. Insider threat detection based on user behavior modeling and anomaly detection algorithms. *Applied Sciences*, 9(19), 2019.

[16] Xuebin Wang, Qingfeng Tan, Jinqiao Shi, Shen Su, and Meiqi Wang. Insider threat detection using characterizing user behavior. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 476–482, 2018.

[17] Faisal Janjua, Asif Masood, Haider Abbas, and Imran Rashid. Handling insider threat through supervised machine learning techniques. *Procedia Computer Science*, 177:64–71, 2020.

[18] Naghmeh Moradpoor Sheykhkanloo and Adam Hall. Insider threat detection using supervised machine learning algorithms on an extremely imbalanced dataset. *Int. J. Cyber Warf. Terror.*, 10(2):1–26, apr 2020.

[19] Balaram Sharma, Prabhat Pokharel, and Basanta Joshi. User behavior analytics for anomaly detection using lstm autoencoder - insider threat detection. New York, NY, USA, 2020. Association for Computing Machinery.

[20] Ahmed Saaudi, Zaid Al-Ibadi, Yan Tong, and Csilla Farkas. Insider threats detection using cnn-lstm model. 04 2019.

[21] Degang Sun, Meichen Liu, Meimei Li, Zhixin Shi, Pengcheng Liu, and Xu Wang. Deepmit: A novel malicious insider threat detection framework based on recurrent neural network. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 335–341, 2021.

[22] Vasileios Koutsouvelis, Stavros Shiaeles, Bogdan Ghita, and Gueltoum Bendiab. Detection of insider threats using artificial intelligence and visualisation. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 437–443, 2020.

[23] Jordan Richard Schoenherr and Robert Thomson. Insider threat detection: A solution in search of a problem. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–7, 2020.

[24] Obinna Igbe and Tarek N. Saadawi. Insider threat detection using an artificial immune system algorithm. *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 297–302, 2018.

[25] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Hanghang Tong. Few-shot insider threat detection. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*, CIKM '20, page 2289–2292, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Rene Y. Choi, Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational vision science & technology*, 9(2):14–14, 2020.

[27] Harsh Kukreja, N Bharath, CS Siddesh, and S Kuldeep. An introduction to artificial neural network. *Int J Adv Res Innov Ideas Educ*, 1:27–30, 2016.

[28] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022.

[29] Jianxin Wu. Introduction to convolutional neural networks. 2017.

[30] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[31] Yann Lifchitz, Yannis Avrithis, and Sylvaine Picard. Few-shot few-shot learning and the role of spatial attention. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2693–2700, 2021.

[32] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning, 2020.

[33] Rohit Kundu. Everything you need to know about few-shot learning, 7 2023.

[34] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

[35] Mingcheng Hou and Issei Sato. A closer look at prototype classifier for few-shot image classification, 2022.

[36] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[37] William Grant Hatcher and Wei Yu. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018.

[38] Yu Wang, Gu-Yeon Wei, and David Brooks. A systematic methodology for analysis of deep learning hardware and software platforms. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 30–43, 2020.

[39] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[40] M. Mutlu Yapıcı and Nurettin Topaloğlu. Performance comparison of deep learning frameworks. *Computers and Informatics*, 1(1):1–11, 2021.

[41] François Chollet et al. Keras: Deep learning library for theano and tensorflow. 2015. Accessed on [08/07/2023].

[42] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark, 2022.

[43] Johannes Lederer. Activation functions in artificial neural networks: A systematic overview, 2021.

[44] Anurag Aggarwal and Renuka Sharma. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[45] Xinru Li. Differences of tanh, sigmoid and relu activation function in artificial neural network. *International Journal of Scientific Progress & Research*, 80(6):31035–31038, 2019.

[46] Diganta Mishra and Prasanth Ramachandran. Activation functions in deep learning: A comprehensive survey. *arXiv preprint arXiv:2109.14545*, 2021.

[47] Anurag Sharma, Rajesh Kumar Singh, and Rajesh Singh. Efficient implementation of tanh: A comparative study of new results. *Computer Science & Information Technology (CS & IT)*, 13(7):01–10, 2021.

[48] Juan Terven, Diana M. Cordova-Esparza, Alfonzo Ramirez-Pedraza, and Edgar A. Chavez-Urbiola. Loss functions and metrics in deep learning. a review, 2023.

[49] Timothy O. Hodson, Thomas M. Over, and Sydney S. Foks. Mean squared error, deconstructed. *Journal of Advances in Modeling Earth Systems*, 13, 2021.

[50] David Shulman. Optimization methods in deep learning: A comprehensive overview, 2023.

[51] Faisal Mehmood, Shabir Ahmad, and Taeg Keun Whangbo. An efficient optimization technique for training deep neural networks. *Mathematics*, 11(6), 2023.

[52] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[53] Yanzhao Wu, Ling Liu, Juhyun Bae, Ka-Ho Chow, Arun Iyengar, Calton Pu, Wenqi Wei, Lei Yu, and Qi Zhang. Demystifying learning rate policies for high accuracy training of deep neural networks, 2019.

[54] Raz Rotenberg. How to break gpu memory boundaries even with large batch sizes. *Towards Data Science*, 01 2020.

[55] Jiahui Yu and Konstantinos Spiliopoulos. Normalization effects on deep neural networks, 2022.

[56] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization, 2019.

[57] Maheswara Reddy. Scaling, min-max scaling, normalization. Medium, 2021. URL: https://maheswararedypr.medium.com/scaling-min-max-scaling-normalization-fbb8d736f784.

[58] *Integrated Development Environment*, pages 19–28. Springer Netherlands, Dordrecht, 2007.

[59] Zakieh Alizadehsani, Enrique Goyenechea Gomez, Hadi Ghaemi, Sara Rodríguez González, Jaume Jordan, Alberto Fernández, and Belén Pérez-Lancho. Modern integrated development environment (ides). In Juan M. Corchado and Saber Trabelsi, editors, *Sustainable Smart Cities and Territories*, pages 274–288, Cham, 2022. Springer International Publishing.

[60] Anaconda documentation.

[61] Anne Bonner. Getting started with google colab, 2019. Website.

[62] Jupyter Development Team. Jupyter notebook documentation. https://jupyter-notebook.readthedocs.io/en/latest/notebook.html, 2023.

[63] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *2013 IEEE Security and Privacy Workshops*, pages 98–104, 2013.

[64] Duc C. Le, Nur Zincir-Heywood, and Malcolm I. Heywood. Analyzing data granularity levels for insider threat detection using machine learning. *IEEE Transactions on Network and Service Management*, 17(1):30–44, 2020.

[65] Brian Lindauer. Insider threat test dataset, 2020.

[66] Brian Lindauer. Insider threat test dataset, 2020-09-30. Retrieved from https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1.

[67] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. *2013 IEEE Security and Privacy Workshops*, pages 98–104, 2013.

[68] Salvador Garca, Julin Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[69] Salvador García, Julián Luengo, and Francisco Herrera. *Introduction*, pages 1–17. Springer International Publishing, Cham, 2015.

[70] Hadeel S. Obaid, Saad Ahmed Dheyab, and Sana Sabah Sabry. The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, pages 279–283, 2019.

[71] Ilker Etikan and Kabiru Bala. Sampling and sampling methods. *Biometrics & Biostatistics International Journal*, 5(6):00149, 2017.

[72] Pooja Bhardwaj et al. Types of sampling in research. *Journal of the Practice of Cardiovascular Sciences*, 5(3):157, 2019.

[73] Yassine Hamdaoui. TF(Term Frequency)-IDF(Inverse Document Frequency) from scratch in python . — towardsdatascience.com. `https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b` [Accessed 29-08-2023].

[74] Sergey I Nikolenko, Sergei Koltcov, and Olessia Koltsova. Topic modelling for qualitative studies. *Qualitative Research*, 17(5):554–570, 2017.

[75] Towards Data Science Harshil Patel. What is feature engineering? importance, tools, and techniques for machine learning, 2023.

[76] Hassan Eldeeb, Shota Amashukeli, and Radwa ElShawi. Bigfeat: Scalable and interpretable automated feature engineering framework. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 515–524, 2022.

[77] MathWorks. Feature extraction, 2023.

[78] Suresh Dara and Priyanka Tumma. Feature extraction by using deep learning: A survey. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1795–1801, 2018.

[79] Thomas Rincy N and Roopam Gupta. Feature selection techniques and its importance in machine learning: A survey. In *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, pages 1–6, 2020.

[80] scikit-learn contributors. scikit-learn feature selection documentation, 7/27/2023. Accessed: Date.

[81] Lingli Lin, Shangping Zhong, Cunmin Jia, and Kaizhi Chen. Insider threat detection based on deep belief network feature representation. In *2017 International Conference on Green Informatics (ICGI)*, pages 54–59, 2017.

[82] Xing Wan. Influence of feature scaling on convergence of gradient iterative algorithm. *Journal of Physics: Conference Series*, 1213(3):032021, 2019.