1. INTRODUCTION

One of the most important unresolved issues in cybersecurity is the detection of malicious be havior. The use and disclosure of sensitive information and data by malicious people is increa sing. According to the 2018 Insider Report, 90% of organizations are vulnerable to insider att acks. Nearly 60 percent of organizations experienced one or more insider attacks in 2019. Be cause insiders were allowed access to the organization's assets, they were more likely than ou tside attackers to compromise the confidentiality, availability or integrity of information.

Most "insider threats" involve malicious employees seeking to harm the company through the ft and violence. However, in practice, careless and negligent employees can often cause serio us harm (66%). Although there are many types of insider that can be found in theory,

The largest regular employees (52%) and privileged IT users (59%), respectively. the increa se in users with unauthorized access, the increase in access devices to confidential informatio n, fast processing, insufficient user information and training, and increased data sensitivity. T he most unsatisfactory fact is that the damage caused by internal performance is between $10 0,000 and $500,000.

Deep learning is a powerful machine learning technique that has shown great promise in detecting insider threat attacks. Insider threats refer to malicious activities that are carried out by employees or contractors within an organization. These attacks can be difficult to detect, as the individuals involved may have legitimate access to sensitive data and systems.

This research focuses on internal discovery based on user behavior. Based on user's activities, their actions are classified as normal or abnormal. For specific selection, situations and activities will be able to analyzed to identify bad behavior. In implementation phase, the model is trained using the selected of vectors.A deep learning method aims to identify insiders with higher accuracy and lower false positives. During training, model is trained by using the behavior of normal users, and here any deviation from normal behavior is classified as malicious or insider information during the trial period.

Our study stands out with the following important results:

- A precise literature review of current methods for internal discovery using machine learning.
- Prepare an internal search based on a simple but non-judgemental in-depth study. Compared to many existing studies, this method is more comprehensive and therefore can provide better results.
- A comparative study of the latest technology and the model we propose.

.Deep learning:

Deep learning is a part of machine learning and part of artificial intelligence. Deep learning is also possible because neural networks are built on the human brain. In deep learning, nothing is specially programmed. It's a machine learning class that uses multiple nonlinear processing units to perform transformation and feature extraction.

The output of each previous layer is used as the input of each subsequent layer.

LSTM:

Long ShortTerm Memory Networks—commonly called "LSTMs"— a special type of RNN with long-term learning capability. LSTMs are specifically designed to avoid long-term problems.

Bi-LSTM:

Bidirectional LSTM (Bi-LSTM) layers learn bidirectional long-term dependencies between time steps in a data series or a time series. These expectations are useful when you want network to learn through full-time processing of at every step.It is a mixed model consisting of two LSTMs: one with forward and one with feedback. BiLSTM is useful for data found on the web and improves what is included in the algorithm (for example, knowing which words come before and after a sentence).

AE(AUTOENCODERS):

Autoencoders are neural networks whose output processes are as diverse as their input processes. In short, the no.of output units in the output layer is equal to the input units in the input layer.Autoencoders copy information from input to output in an unsupervised manner and are therefore sometimes referred to as copy neural networks. The

Autoencoder has three parts:

• Encoder: The encoder is a supported neural network that will compresses input into a hidden representation and encodes input image into an overall expansion. A compressed image is a deformed version of the original image.

• Code: This part of a network has a simple representation of input decoder.

• Decoder: A decoder is an integrated device like an encoder and has a structure similar to an encoder.

This network is the responsible for transforming ideas from numbers to sizes.

1.1 Problem Statement:

Hunting Insider Threat : Detection of Malicious Employees within the organization based on their behaviour.

1.2 Scope of Project:

The project will focus on developing a solution that is highly accurate, scalable, and can be easily integrated with existing security systems. The system will be tested and evaluated on a CERT dataset to measure its effectiveness in detecting insider threats. Ultimately, the goal of the project is to help organizations prevent insider attacks and protect sensitive information from being compromised

1.3Motivation:

As a Computer Science student, we understand the importance of data, especially confidential data within an organization. However, despite an organization's best efforts to keep this data safe from hackers and attackers, there is still a threat from within the organization itself. Malicious employees can potentially cause significant damage and financial losses to the organization, making it imperative to identify and mitigate such threats. To address this challenge, we aim to develop a solution that can help organizations detect malicious employees. Since employees have the advantage of being able to bypass security checks more easily than external attackers, finding malicious insiders is a challenging task. Therefore, we plan to leverage data analytics and Deep learning techniques to analyze user behavior and identify anomalies or patterns that may indicate malicious intent.

2. LITERATURE SURVEY

2.1 Behavioural Based Insider Threat Detection Using Deep Learning:

Authors:- Waseem Iqbal,Mehreen Afzal,Rida Nasir Rabia Latif
(2021)

The authors of this article are trying to identify bad behavior. As shown in the 2018 Insider Report [1], 90% of organizations in today's world are threatened by an insider attack. Nearly 60% of organizations experienced one or more insider attacks in 2019 [2]. This article focuses solely on user behavior based on insider threat detection. Based on the user's activities, their actions are classified as normal or malicious. For specific options, identify situations and activities to identify negative behaviors. In the implementation phase, the model is trained using the selected vectors. A deep learning method is to identify insiders with higher accuracy and lower probabilities. Also, compare the accuracy, F1 score, precision and FPR of different algorithms. The data used here is the CERT dataset. In this article they are using "LSTM-Autoencoder". First collect user data from various csv files, then process the data and use login/logout details, user_role, user_functional_unit, user_department, user_activity, etc. to extract rich events/user roles based on selected features that are then used for model training and evaluation.

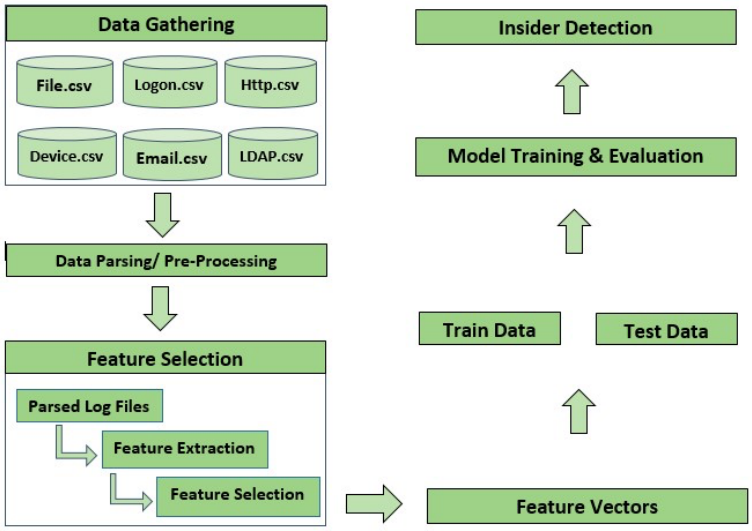

Figure:1 Workflow of process

Structure of Proposed System:

The model is trained with Epochs = 200, batch size = 64, learning rate = 0.0001, the activation function is "relu" and the optimizer used is "adam". The length used for entry and exit is the same. The architecture of LSTM autoencoder is shown in Figure 2.

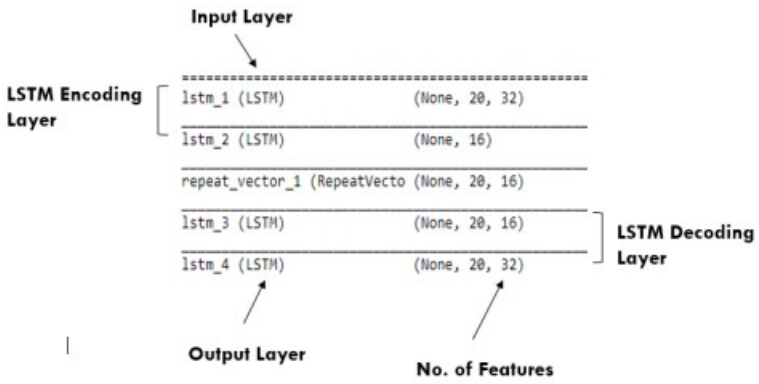

Figure:2 Feature Vector

The difference between output and input is calculated as unemployment. The loss function is calculated as the mean squared error (MSE). In the model training phase, the inputs are converted into outputs. The model will be trained, for example, on normal or negative data. Insiders = 0.If the construction error is high during the test, it is considered as abnormal behavior. The model is tested using good and bad models. Reproducibility errors are very high for internal users compared to regular users. A starting point is to define the difference between bad behavior and bad behavior. If the value is higher than the threshold, it is considered "internal", if the value is lower than the threshold, it is considered "normal", as shown in figure3.

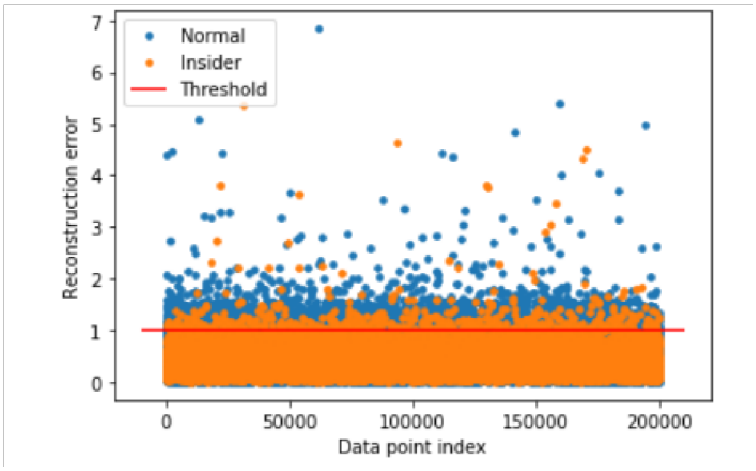

Figure:3 Reconstruction Error for Different Classes

Comparison of Results:

Algorithm	Performance Matrices				
	Accuracy	Precision	F-Score	AUC	Dataset
LSTM-CNN	✗	✗	✗	0.94	V4.2
LSTM-RNN	93.85%	95.12%	✗	✗	V6.2
One-Class SVM	87.79%	✗	✗	✗	V4.2
Multi State LSTM & CNN	85%	✗	✗	0.9047	V6.2
Isolation Forest	79%	✗	✗	✗	V4.2
LSTM based Autoencoder	90%	✗	✗	✗	V4.2
LSTM-AutoEncoder	90%	97%	94%	✗	V4.2

Table:1 Comparison of results

Conclusion:

"LSTM-Autoencoder" for internal threat model was trained and tested on CMU CERT V4.2 using different internal threat scenarios. Platforms used to evaluate ideas Anaconda, Build; In addition, the performance of the proposed method will be compared with other known methods (such as LSTM-CNN, one-class SVM ,Random Forest, LSTM-RNN), Markov Chain model multi-state LSTMs and CNNs. , gated repeating units and transition graphs.The comparison shows that our urine output is good (90.60%), F1 score (94%) and precision (97%).

2.2 Insider Threat Detection Using Deep AutoEncoder and Variational Autoencoder Nueral networks :

Authors: Gueltoum Bendiab, Nicholas Kolokotronis, Starvros Shiaeles, Efthimios Pantelidis,

It is well known that identifying insider threats is a difficult problem that causes many challenges for the research organisation. In this article, we explore Efficiency and Effectiveness of Using Autoencoder and Variational Autoencoder Deployment Learning Techniques to Combat Insider Threats Without Human Intervention.Here the dataset used is public CERT(r4.2) and also compares with other algorithms. From these comparisons we can conclude that Variational Autoencoder neural networks will provides a overall top performance with a low-false rate and high detection accuracy.

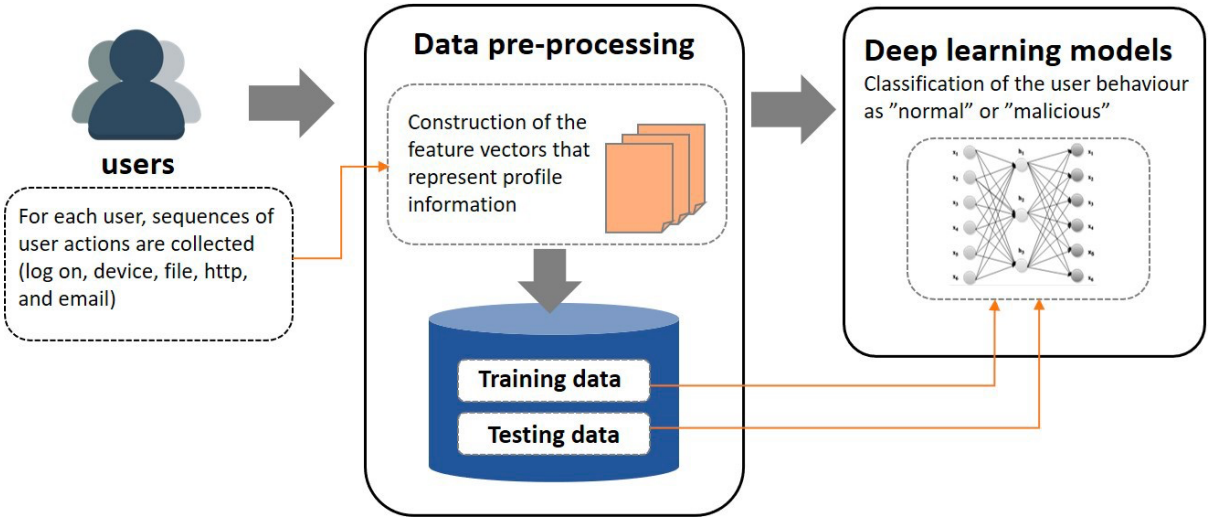

Figure:4 Overview of Proposed model

The r4.2 version of the CERT dataset has the following features.

1) The first scenerio describes how an employee in an organization started using a removable disk to log into the organization's system hours later. The employee shared sensitive information from the agency's systems on the wikileaks.org website and left the agency soon after.

2) The second is to represent the employee in the organization who is looking for a job and is requesting a job from a candidate. He used a USB flash drive to steal sensitive information before it went to another company.

3) The third scenario is the disappointment of the head of the company. So he downloaded the keylogger and sent it to his supervisor's machine using a USB stick. After that, he entered as the overseer and sent a group of earth explosions, causing panic in the organization.

System administrators leave the organization quickly.

Variational autoencoders (VAEs) are key design methods that generate new output data which is similar to the training data. As can be seen, there are two main networks in the structure of VAE as in AE; encoder and decoder.

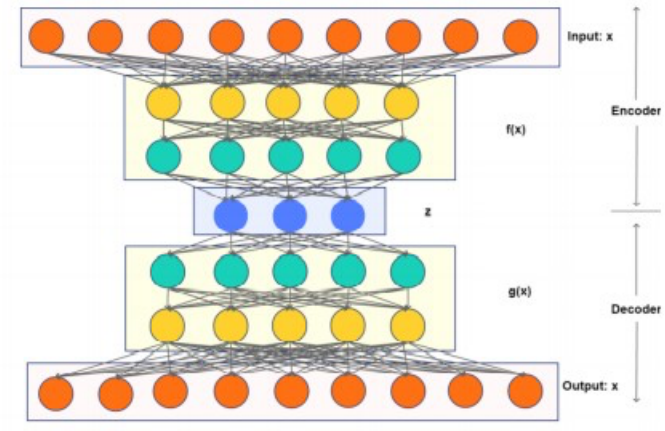

Figure:6 Variational Autoencoder Structure

Results:

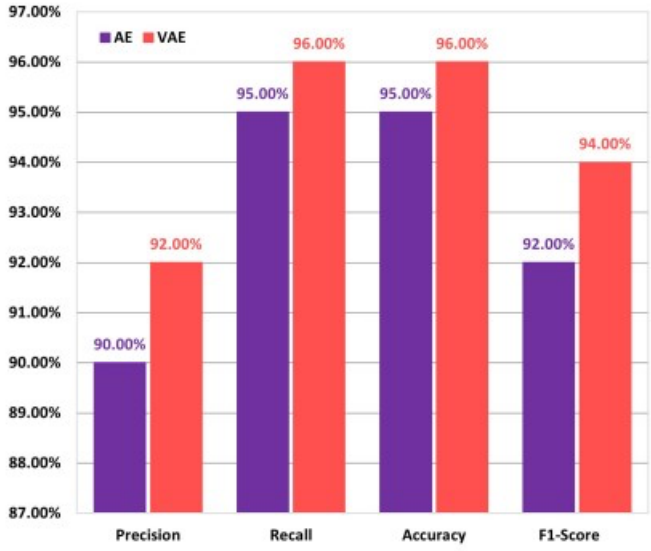

Figure:7 Evaluation Metric Results

Comparison with other Algorithms:

	Precision	Recall	Accuracy	F1-Score
Autoencoder (AE)	90%	95%	95%	92%
Variational Autoencoder (VAE)	92%	96%	96%	94%
CNN [11]	n/a	n/a	90%	n/a
Bio-Inspired models [12] (9 features)	n/a	100%	n/a	70%

Table:2 Comparison of results of various algorithms

Conclusion:

They used both deep neural network autoencoders and variational autoencoders to examine of their performance in detecting insider threats. The test was performed on the public CERT database (CERTr4.2). From the results and comparining with other deep learning methods, the variational autoencoder neural network has proven to be the best method for detecting internal threats with an overall accuracy of 96.00%.On the other side, autoencoders allow slightly better internal threat detect performance (95% accuracy) at a higher cost than negative ones.

2.3 “Comparing insider IT sabotage and espionage: A model based analysis”:

This article presents a report examining the brain, function, relationship, and content of belief elements that lead to at least two types of belief in criminal behavior: internal destruction of critical information (IT) systems and espionage. Security experts and policy leaders now see espionage and insider threats as serious concerns, but often treat them as separate issues that need to be addressed with different security measures. In this study, researchers explored the similarities and differences between cybercrime and auditors to isolate the main factors or conditions that lead to the two types of high crime. Using a structural dynamics approach, the team developed a descriptive model that represents a high degree of interrelationship between the two domains as modeled by the domains. This effort made clear the contrast between the two types of trust crimes. Factors found in saboteurs and spies include the inclusion of stress factors on self-interest and insider risk of terrorist activity; the concerns and actions of insiders before or during the attack; Ability to detect or respond to violations; physical organization and management of power outages. Based on the findings and analysis, recommendations and policy implications are also presented.

2.4 “Scenario-based insider threat detection from cyber activities,”:

An internal event is the result of various malicious activities resulting from the unintentional or intentional misuse of an organization's systems, resources, data and networks. Preventing of insider threats is extremely difficult, as they involve community partners of the organization with access to sensitive/confidential resources. Recent research in insider threat detection has focused on the development of behavioral detection techniques to find anomalies or changes in user behavior that were good at the time. But unusual activities are not the same as the type of violence that will be lead to an insider attack. Based on the improvement of existing methods, we present a method for detecting insider threats by dividing time between user activities. Initially, a number of daily properties were calculated from user-defined tasks. Next, construct a time series feature vector based on statistics over time for each day's feature. The label of time series feature vector (either bad or not) is inferred from ground truths. We use a data rate sensitive method that randomly tests parameter classes to determine the instability of actual threat data with only a few adverse events. As a classifier, we use a two-layer deep autoencoder neural network and compare its performance with other classifiers: Random Forest and Multilayer Perceptron.

Our system was evaluated using CMU Insider Threat Data, which is the only available in publicly that threat data containing nearly 14 GB of web analytics calculated with login, connection, routing information and e-mail information, and supporting results were obtained.

2.5 “Insider threat detection with deep neural network,”:

Insider threat detection has received a lot of attention from business and researchers. Most current research focuses on using machine learning techniques to identify insider threats. However this job requires "feature engineering", which is very difficult and time-consuming. It is well known that deep learning can learn powerful features. In these paper, we presenting a new modern method of internal threat detection based on user behavior deep neural network (DNN).Specifically, They will use the LSTM with CNN framework to discover different users. First, we use a short-term neural network (LSTM) to learn user behavior and extract physical features from user actions, similar to natural language patterns. Second, the extracted features are converted into a large set of features that a convolutional neural network (CNN) uses to identify internal threats. We strive to combat publicly available information about insider threats. The experimental results show that our method can detect the internal threat and we get AUC = 0.9449 in best condition.

2.6 “Graph based framework for malicious insider threat detection,”:

In the most cases of security services focus on preventing the attacks from outside the enterprise environment, the real threat comes from within the protected environment. Insider threats have demonstrated their strength, significantly affecting the financial stability, national security and privacy of millions of people. What's in the news is just the tip of the iceberg, there's more to come, and there are some threats that have never been seen before. To address this major cybersecurity threat, we propose an integrated approach based on image analysis and vulnerability detection. Our framework analyzes data inconsistencies to isolate malicious users hiding behind others.

The results show that the framework is useful for distinguishing the majority of users who exhibit positive behavior from the small number of users who exhibit negative behavior..

2.7 Tools, Technologies and Libraries Used

Tools

- **Jupyter Notebook and Anaconda**
  Jupyter and Anaconda are two popular tools used by data scientists and developers for data analysis and scientific computing. Jupyter is an open source and also a web application will allows users to share and create interactive notebooks with visualizations,explanatory text and code. These notebooks can be used for various tasks such as data cleaning, statistical modelling, and machine learning.
- **Python 3.10**
  Python 3.10 was released on October 4, 2021 and contains a variety of new features and optimizations that make it a better, faster, and more secure programming language compared to 3.9.0

Technologies

- **Deep Learning**
  Deep learning has explored in many fields, mainly in computer vision, natural language processing and speech recognisation. It has enabled breakthroughs in areas such as autonomous vehicles, medical imaging, and robotics. Furthermore, the availability of open-source deep learning frameworks, such as PyTorch, TensorFlow, and Keras, has made that easier for researchers and practitioners to experiment with and apply deep learning techniques to a wide range of problems.Despite its successes, deep learning is not a panacea and has its own challenges, including the need becomes more for big amounts of data and computing resources, and risk of overfitting.

Libraries

- **Numpy**
  NumPy is a Python library for computational science that supports many variables and matrices, as well as many numerical functions for manipulating these arrays. NumPy is a package to deal with Python.
- **Pandas**
  Pandas is an open-source data analysis and management library for Python. It provides a powerful and flexible dataframe called DataFrame, which is basically a two-dimensional document with rows and columns that looks like a spreadsheet. It can contain various types of data such as rows, numbers, variables, strings, and even Python objects.
- **Matplotlib**
  Matplotlib provides a simple and intuitive interface for creating a wide range of visualizations, from basic line and scatter plots to more complex heatmaps, histograms, and 3D plots. It also allows for extensive customization of plot aesthetics, including colors, fonts, and annotations.One of the strengths of Matplotlib is its integration with other

Python libraries and tools, such as NumPy and Pandas, which allows for seamless data manipulation and analysis. Matplotlib can also be used in conjunction with interactive data visualization tools, such as Jupyter Notebooks and IPython, to create dynamic and interactive visualizations.

- **Keras**

  Keras is an advanced neural network API written in Python designed to make building and training deep learning models simple and efficient. It builds on low-level libraries such as Keras, TensorFlow, Theano, and CNTK, allowing these libraries to take advantage of their optimization and computing power.

- **Seaborn**

  Seaborn provides a wide range of built-in visualizations, including scatterplots, line plots, bar plots, histograms, heatmaps, and more. These visualizations can be customized with a variety of themes, color palettes, and plot styles, allowing users to create professional-looking visualizations with minimal effort.

- **Sklearn**

  Sklearn includes various machine learning algorithms such as classification, clustering, regression and dimensionality reduction, as well as tools for selection of model, evaluation, and data text first. These algorithms can be used for many tasks such as image classification, text analysis and anomaly detection.

3. METHODOLOGY

We use "BiLSTM-Autoencoder" for internal detection. Our plan has several stages. First collect user data from different csv files, then process data and create state input/output layer, user role, user function, client, user function etc. Extract rich events/user roles based on events including The selected features will be used for model evaluation and training. An overview of our model is shown in Figure 8.

3.1 Workflow of the Proposed System:

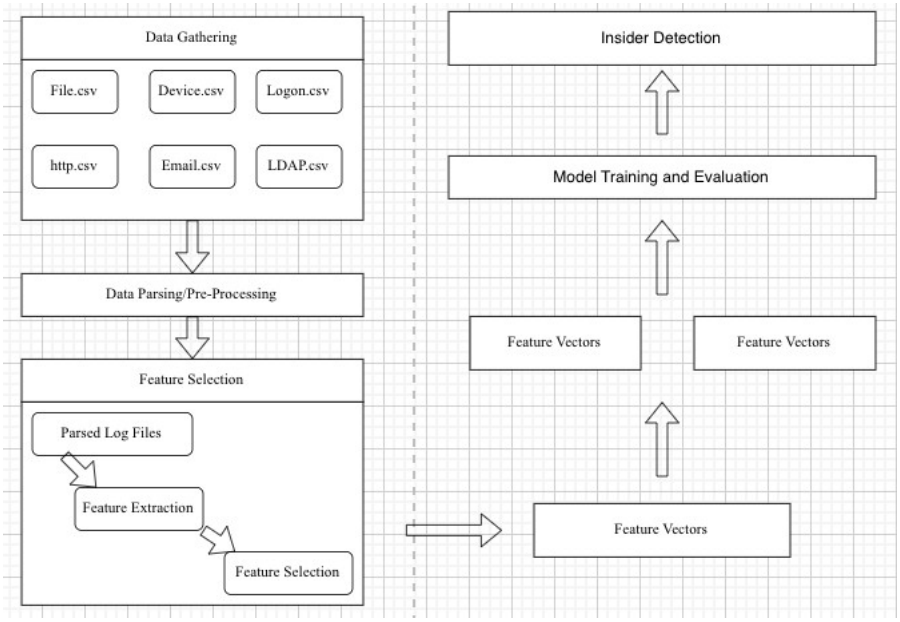

Figure:8 Proposed Methodology

BiLSTM is used for training the model. Since BiLSTM is used to look at historical data to make predictions, it does data up to (t-look back) to make predictions at time t. Takes a 3D array as input.

model × features × review

If the data is ready, it will be divided into two parts i.e., train data and test data. Training data will be used for parameter evaluation and model training, and test data is used for evaluation of model.

1. **Data Gathering**

The Dataset is taken from public CERT(r4.2). Here the dataset contains various files.
Logon.csv file will contains Users login and logout timings.
Device.csv contains users connecting and disconnecting the external devices.
http.csv contains user browser history
email.csv contains email logs
file.csv contains log of the user activity files
psychometric.csv contains user personal attributes.

2. Inside Threat Scenario

a. Users who use an external hard drive or work after hours have not had the usual odds of logging in to an event after hours in the past, using a flash drive, uploading files to wikileaks.org, and leaving the organization shortly after.

b. Users visited the job site and searched for jobs from candidates.Use flash drives before leaving the organization (at a higher frequency than in previous activities) to steal information.

3. Data Preprocessing

The file contains multiple csv files (login, file, HTTP, email, device, LDAP), each with raw data. This information cannot be fed into the algorithm and must be processed first. All csv files are analyzed and written to a csv file, eg. Create a master file containing data from all csv files. The feature set was extracted from this number and letter data.These values should be encoded to used as the input for the algorithm we propose.**Feature Extraction**

Analyzing the whole CSV file and analyzing related data can learn and predict user behavior effectively. It also depends on the internal environment we are dealing with. malware.Psychometric.csv is not used for custom selection. Properties in all other csv files include date, time, pc, user_id, PC, user_role, user_functional_unit, user_department and function attributes. The identity of the attribute is redundant and is not included.

4. Model Training And Evaluation

After the data preprocessing step is complete, each user's sequence of actions is fed to the feature extraction module to predict the next action in a timely manner. In this study, bi-LSTMs are used to learn user behavior language, similar to LSTMs used in natural language processing (NLP). BiLSTM uses a previous set of physical sequences as input to extract time-based features to construct each user's behavioral sequence because LSTM can provide variable length inputs and outputs. In this work, we use dual LSTM so that the network can better understand each user's business process and better predict the next job in real time. Bi-LSTM can be performed in two directions (for example forward and backward).i.e., from the past to the future, from nothing to the past). In biLSTM, the user's state is classified as the user's new action (new input), user's previous action, and previous output. Therefore, this bi-LSTM model is based on an iterative process at each time point for time-context learning sequences. In bi-LSTM forward propagation is used twice: once for the forward unit and once for the backward unit, because it will be easier to use this information in the future to understand the network and prediction of the next action in a timely manner. Figure 9 shows the binary LSTM model to extract data. When

analyzing user behavior with bi-LSTM, three situations are taken into account: forget Ft, enter and leave Ot. Finally, no evaluation data is used to classify users as real users based on their past and present actions (behaviour). Among these, "no categorical feature" was marked as "available" and "no categorical feature" was marked as "0". Among other things, the dataset is analyzed for user feedback.

5. After data-cleaning is performed on the original data, additional data integration, data filtering, data binding and data extraction are performed on data cleaning. Here is a list of users.

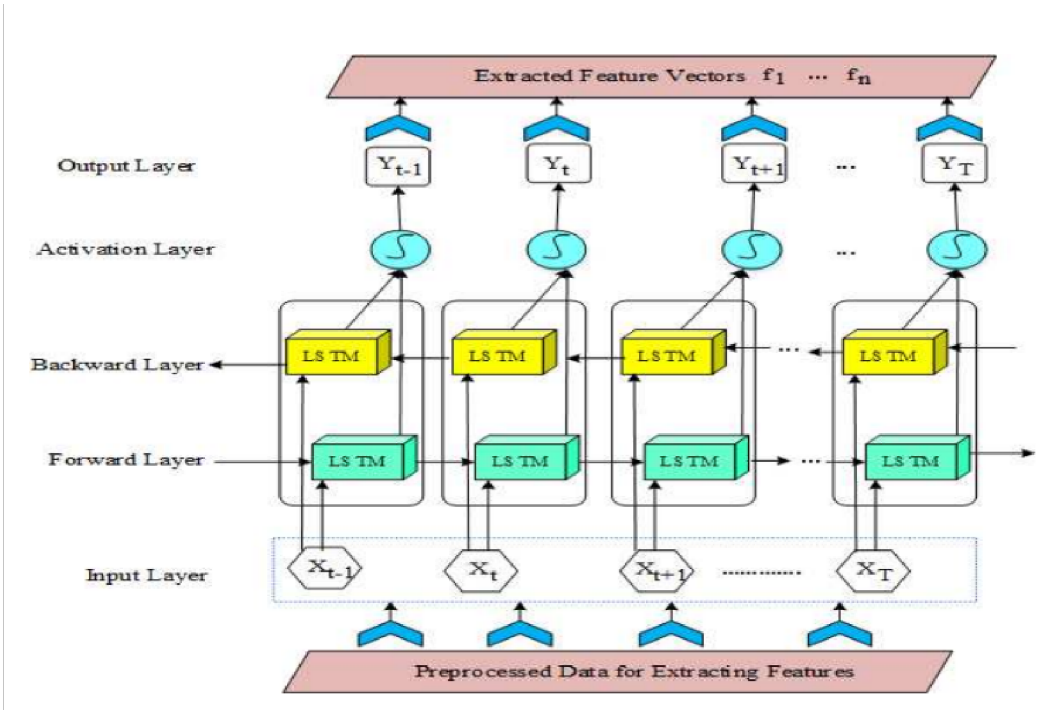

Figure:9 Bi-directional LSTM Architecture

Each time "Ht" means to current use hidden state, "Ct" means to use current cell state, "Xt" means to use current time and "Ht-1" means to use old hidden state. The sequential execution of double LSTMs takes each user's sequence of actions and predicts their next action. In his work, the results of two LSTM networks (forward and backward hidden state) are compared with the real system to reduce the error using the entropy loss value. Finally, the last hidden layer of bi-LSTM is the output, which gives a vector map of each user string. In this work, we consider a given time t and an input vector Xt ϵRnxd, where 'n' is an integer.
user and 'd' is the selected number. Available for all users (single-level operation). Thus, the binary LSTM representation of the forward and backward hidden state at timestamp t will be →-HtϵRnxh and ←HtϵRn ×h, respectively, where h denotes the number. house secret. Autoencoder neural networks have two networks, an decoder network and a encoder network. The encoder is learn how to read input and compress it into an internal representation defined by the bottleneck layer. The decoder network then uses the encoder's output (bottleneck layer) to rebuild the input. Once the autoencoder learns the neural network, the decoder is discarded, we keep the encoder in our hands and used to convert the input data to a vector as the output

of the bottleneck layer. To implement the autoencoder model, we build an input layer, an decoder network, a encoder network, and an output layer.

3.2 Activity Diagram:

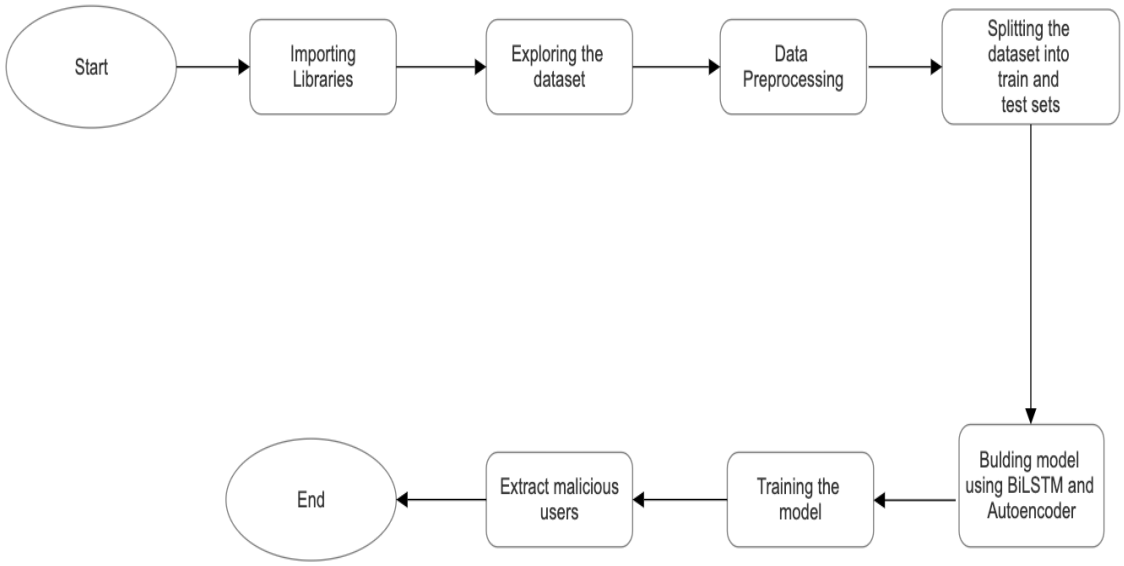

Figure:10 Activity Diagram

3.3 UML Use Case:

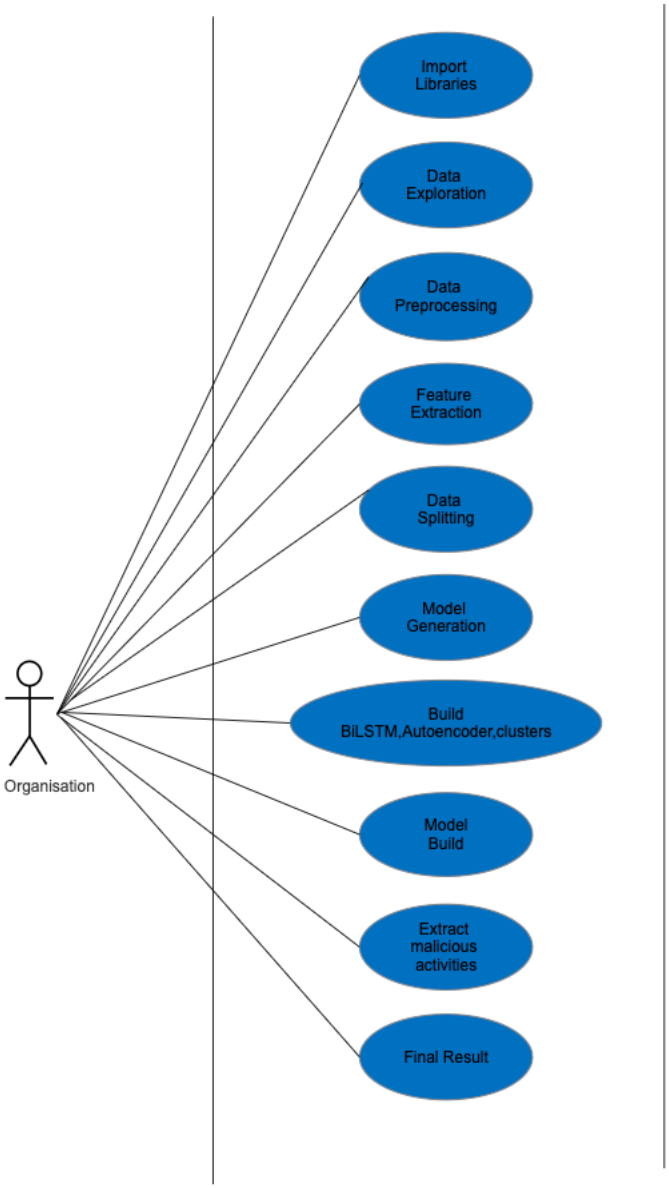

Fig:11 Use Case Diagram

4. IMPLEMENTATION

4.1 Algorithms

4.1.1 Hunting insider based on their behaviours using BiLSTM and Autoencoder

Procedure for hunting insider threat
Step1: Take CERT v4.2 as input
Step2: Initilization:merge all the files on ‘user id’
Step3: Data Pre-processing
Data Cleaning
Data transformation
Data reduction
Step4: Feature Extraction
Step5: split final dataset into train and set
Step6:Train the model using BiLSTM
Step7: Test the model
Step8:Result 0 if user is normal or 1 if user is malicious
Step9: end

4.1.2 ENCODING

Step1:Create LabelEncoder object for each categorical variablele_
Code:

```
logon = LabelEncoder()
le_job_title = LabelEncoder()
le_department = LabelEncoder()
le_business_unit = LabelEncoder()
```

Step2:Fit and transform each categorical variable using their respective LabelEncoder object

```
df_master_f['activity'] =le_logon.fit_transform(df_master_f['activity'])
df_master_f['role'] = le_job_title.fit_transform(df_master_f['role'])
df_master_f['functional_unit']=le_department.fit_transform(df_master_f['functional_unit’]
df_master_f['department'] = e_business_unit.fit_transform(df_master_f['department'])
```

Step3:Print the mapping of the categorical values with their respective encoded values

```
print('Logon:', dict(zip(le_logon.classes_,le_logon.transform(le_logon.classes_))))
print('JobTitle:', dict(zip(le_job_title.classes_,le_job_title.transform(le_job_title.classes))))
print('Department:',dict(zip(le_department.classes_,le_department.transform(le_departmemt.c
lasses)
print('BusinessUnit:', dict(zip(le_business_unit.classes_,le_business_unit.transform(le_
```

step4:Create label encoder for user and pc activities

```
user = LabelEncoder()
pc = LabelEncoder()

step5: Fit and transform user activity using their respective LabelEncoder object
df_master_f['pc_x'] = pc.fit_transform(df_master_f['pc_x'])

step6:Extract date, time, and day of the week from datetime column
df_master_f['Date'] = df_master_f['date_x'].dt.date

step7: extract hours from time
df_master_f['logof_Time'] = df_master_f['logof_Time'].apply(lambda x: x.hour)
df_master_f['logon_Time'] = df_master_f['logon_Time'].apply(lambda x: x.hour)

step8:create labelencoder for a day and fit and transform the day variable using label encoder object.
day = LabelEncoder()
df_master_f['day'] = day.fit_transform(df_master_f['day'])

step9:drop duplicates and reset the columns
df_master_f.drop_duplicates(inplace=True)
df_master_f.reset_index(inplace=True)
df_master_f.drop(columns=['index'],axis=1,inplace=True)
```

4.1.3 Building model

```
Step1:Define the dataframe
df = df_master_f

step2:Split  DataFrame into the training and test sets
x_train,  x_test =  train_test_split(df, test_size=0.2, random_state=42)

step3:split  training set into the training and validation sets
x_train, x_val = train_test_split(x_train,  test_size=0.2, random_state=42)
x_train.shape

step4:Assume intial train data is an empty list
X_train = []

Step5:Assuming df_master is your DataFrame with 8 columns
 for i in range(60, 1088): X_train.append(x_train.iloc[i-60:i, :])

step6:convert into an numpy array
X_train = np.array(X_train)
```

Step7:repeat steps 4,5,6
Step8:prepare the input data
input_data = X_train

step9:Define BiLSTM autoencoder architecture
input_dim = input_data.shape[-1]
*data*latent_dim = input_dim

step10:Assuming input_data has shape (num_samples, input_dim)
num_timestamps=60
input_data_3d = input_data.reshape(input_data.shape[0], 60, 8)

4.1.4 Applying Clusters

Step1:Import required libraries
Step2:Define Hyperparameters
epochs = 400
batch_size = 64
learning_rate = 0.0001
activation = 'relu'
optimizer = Adam(lr=learning_rate)
timesteps = 60
input_dim = 8
latent_dim = 64
hidden_dim = 64
threshold = 0.5

step2:encode the data
inputs = Input(shape=(timesteps, input_dim))
encoder = Bidirectional(LSTM(hidden_dim,
activation=activation),merge_mode='concat')(inputs)
encoder = Dropout(0.2)(encoder)encoder = Dense(latent_dim,
activation=activation)(encoder)
encoder_model = Model(inputs, encoder)

step3:now use decoder
decoder_inputs = Input(shape=(latent_dim,))
decoder = Reshape((1, latent_dim))(decoder_inputs)
decoder = Bidirectional(LSTM(hidden_dim, activation=activation,
return_sequences=**True**),merge_mode=(’concat’)(decoder)

```
decoder = Dropout(0.2)(decoder)
decoder = LSTM(input_dim, activation=activation, return_sequences=True)(decoder)
decoder_model = Model(decoder_inputs, decoder)

step4: Autoencoder
autoencoder = Model(inputs, decoder_model(encoder_model(inputs)))
autoencoder.compile(optimizer=optimizer, loss='mse')
history=autoencoder.fit(input_data,input_data,epochs=epochs,batch_size=batch_size,validati
on_data=(X_val,X_val))
step5:import matplot library

step6:plot the autoencoder loss during training
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Autoencoder Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'])
plt.show()

step7: get latent representations from encoder
latent_representations = encoder_model.predict(input_data)

step8:performing clustering on latent representations
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters)
kmeans.fit(latent_representations)

step9: get cluster labels and print
cluster_labels=kmeans.labels_
print(“cluster Labels: ”,cluster_labels)
```

4.2 Libraries used

- Import pandas as pd
- Import numpy as np
- Import warnings
- Import matplotlib.pyplot as plt
- Warnings.filterwarnings(‘ignore’)
- From keras.models import Model
- From keras.layers import Input,LSTM,Bidirectional
- From sklearn.preprocessing import LabelEncoder
- From keras.optimizers import Adam
- From keras.models import Model
- From sklearn.cluster import KMeans
- From sklearn.model_selection import train_test_split
- From kera.layers import Input,Dense,Bidirectional,LSTM,Dropout,Reshape
- From keras.models import Model
- From sklearn.metrics import r2_score

4.3Data Set Description

In this paper, we utilize the Insider Threat Dataset provided by Carnegie Mellon University's CERT project to conduct our experiments. Specifically, we use the CERT-r4.2 dataset, which consists of real-world insider attacks that were recorded over a one-and-a-half year period across multiple business sites. The dataset contains a wealth of information, including host data, network data, employee psychological assessments, and human data, among other things. This comprehensive dataset is designed to provide detailed insights into insider threat detection, and specific details regarding the dataset can be found in the accompanying table.

The CERT-r4.2 dataset simulates three main types of insider attacks, including information theft, vandalism, and insider fraud. By processing data and insights, the behavior patterns of users in different time domains and behavioral domains will change. Most of the time, the movements of both users are bad in terms of body and behavior. However, some of the changes in the user's behavior are not obvious and most of the behavior is minor, which requires us to create the correct user settings and show the negative.

Name	domain	No. of Columns
Logon	id, date, user, pc_number, activity (logon, logoff)	5
File	id, date, user, pc_number, filename, activity (open, write, copy, delete),to_removable_media, from_removable_media, content	9
HTTP	id, date, user, pc_number, url, activity (visit, download, upload), content	7
Device	id, date, user, pc_number, file_tree, activity (connect,disconnect)	6
Email	id, date, user, pc_number, to, cc, bcc, from, activity (send, view), size, attachments, content	12
Psychometric	employee_name, id, email, role, projects, function_unit, business_unit, department, team, supervisor	10
LDAP	employee_name, id, O-score, O-score, Cscore, E-score, A-score, N-score	7

Table:3 Description of different files

5. RESULTS

5.1 intial Dataset:

df_psychometric=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\psychometric.xlsx')
df_device=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\device.xlsx')
df_email=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\email.xlsx')
df_file=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\file.xlsx')
df_http=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\http.xlsx')
df_logon=pd.read_excel(r'C:\Users\premc\insider_threat_detection\insider_threat_detection\dataset\logon.xlsx')

Printing Data of Device File:
code: df_device

	id	date	user	pc	activity
0	{P1I9-X2UT34LR-6485PDJX}	2010-02-01 05:15:32	WCR0044	PC-9174	Connect
1	{S2D2-V1NT65KD-8108KBSZ}	2010-02-01 05:17:09	WCR0044	PC-9174	Disconnect
2	{G5Y9-L4GP40WG-0372GEOX}	2010-02-01 06:39:10	WCR0044	PC-5494	Connect
3	{A4Q5-A1FR45TY-9562ISJI}	2010-02-01 06:40:30	WCR0044	PC-5494	Disconnect
4	{M9Q0-O7UM21OB-5937NRWH}	2010-02-01 07:02:33	BHV0556	PC-6254	Connect
...	...	...	...	...	...
994	{O9M7-W0VD02FK-2056LBRS}	2010-04-01 13:40:24	BCS0212	PC-1682	Disconnect
995	{R5B7-W5VU13JX-1034HBPG}	2010-04-01 13:40:36	MLF0962	PC-2902	Disconnect
996	{I1O7-I2LL01RU-0755XBFS}	2010-04-01 13:40:42	HGM0226	PC-5309	Disconnect
997	{Y5A3-W4FU51KE-3184VPPG}	2010-04-01 13:40:53	WRC0885	PC-2240	Connect
998	{Z0D6-Z2SI26WX-1324IVWS}	2010-04-01 13:41:07	BFM0761	PC-9718	Connect

999 rows × 5 columns

Figure:12 Data of Device CSV

5.2 Ratio of different user role in the organisation.

Code:

```
lst = df_copy.role.value_counts()
lbs = pd.unique(df_copy.role)
fig = plt.figure(figsize=(8, 8))
plt.pie(lst, labels=lbs, autopct='%1.1f%%', radius=1.1)
plt.show()
```

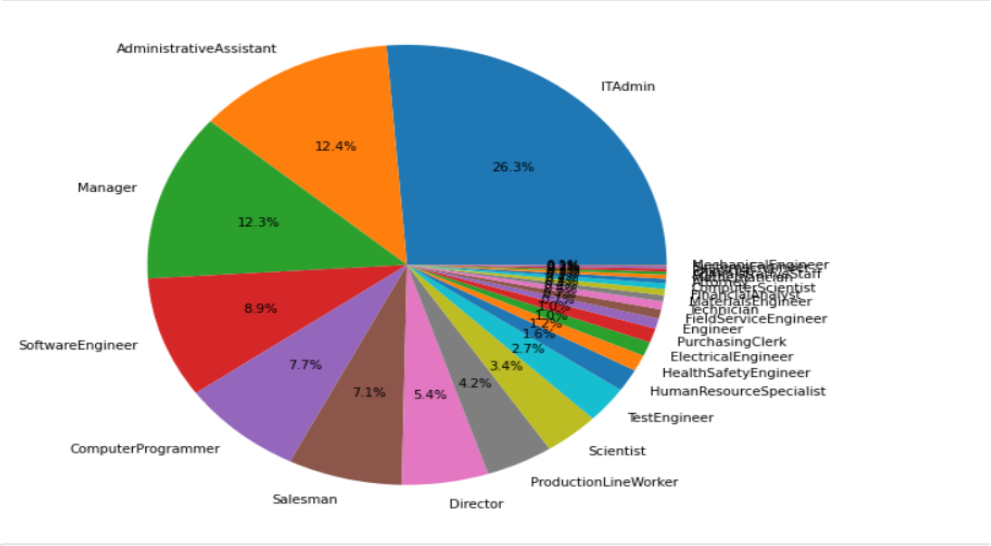

Figure:13 Different roles of users

Code:

```
lst = df_copy.functional_unit.value_counts()
lbs = pd.unique(df_copy.functional_unit)
fig = plt.figure(figsize=(8, 8))
plt.pie(lst, labels=lbs, autopct='%1.1f%%', radius=1.1)
plt.show()
```

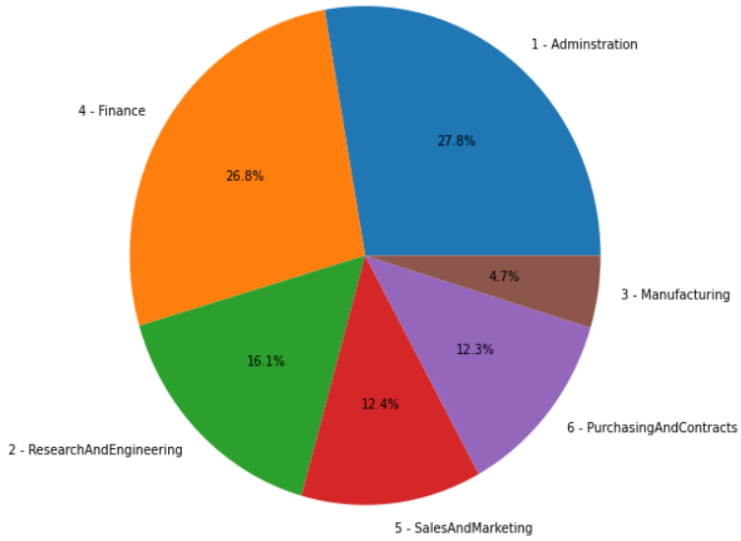

Figure:14 Different Functional Unit

5.3 Ratio logon and logoff Time

Code:

```
lst = df_copy.activity.value_counts()
lbs = pd.unique(df_copy.activity)
fig = plt.figure(figsize=(8, 8))
plt.pie(lst, labels=lbs, autopct='%1.1f%%', radius=1.1)
plt.show()
```

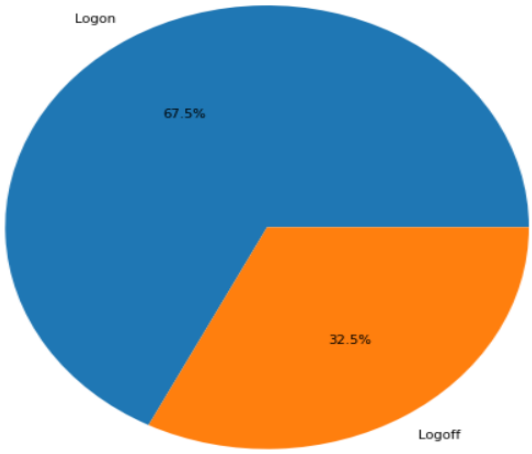

Figure:15 percentage of logoff and logon timings

5.4 Final Data Set

Code:

```
df_logoff_mean = df_user_logoff.groupby('user')['hour'].mean().reset_index()
df_logoff_mean['hour'] = df_logoff_mean['hour'].astype(int)
df_logoff_mean['hour'] = pd.to_datetime(df_logoff_mean['hour'], format='%H').dt.time
df_user_logoff_stats['mode'] = df_logoff_mode['time']
df_user_logoff_stats['mean'] = df_logoff_mean['hour']
df_copy['time'] = pd.to_datetime(df_copy['date']).dt.time
df_copy.drop_duplicates(inplace=True)
df_copy.rename(columns= {'activity':'status'}, inplace=True)
df_copy=pd.merge(df_copy,df_device, on=['user'],how="outer")
df_copy.shape
df_copy.dropna(inplace=True)
df_copy.drop(columns=['date_y', 'pc_y'],axis=1,inplace=True)
df_copy.drop_duplicates()
```

Out[64]:

	user	role	functional_unit	department	status	date_x	pc_x	time	id	activity
0	WCR0044	ITAdmin	1 - Adminstration	6 - Security	Logon	2010-02-01 05:02:50	PC-9174	05:02:50	{P1I9-X2UT34LR-6485PDJX}	Connect
1	WCR0044	ITAdmin	1 - Adminstration	6 - Security	Logon	2010-02-01 05:02:50	PC-9174	05:02:50	{S2D2-V1NT65KD-8108KBSZ}	Disconnect
2	WCR0044	ITAdmin	1 - Adminstration	6 - Security	Logon	2010-02-01 05:02:50	PC-9174	05:02:50	{G5Y9-L4GP40WG-0372GEOX}	Connect
3	WCR0044	ITAdmin	1 - Adminstration	6 - Security	Logon	2010-02-01 05:02:50	PC-9174	05:02:50	{A4Q5-A1FR45TY-9562ISJI}	Disconnect
4	WCR0044	ITAdmin	1 - Adminstration	6 - Security	Logon	2010-02-01 05:02:50	PC-9174	05:02:50	{E2I3-F5GK49LK-9646XIBH}	Connect
...	...	...	...	...	...	...	...	...	...	...
2668	TDS0246	MechanicalEngineer	2 - ResearchAndEngineering	3 - Engineering	Logon	2010-04-01 07:51:00	PC-9565	07:51:00	{U3V7-R0LH83EF-2509XPJA}	Disconnect
2669	AVW0409	SystemsEngineer	2 - ResearchAndEngineering	3 - Engineering	Logon	2010-04-01 07:52:00	PC-4038	07:52:00	{L6G9-Y8KH41BO-1264JZHD}	Connect
2670	AVW0409	SystemsEngineer	2 - ResearchAndEngineering	3 - Engineering	Logon	2010-04-01 07:52:00	PC-4038	07:52:00	{L6Z5-K8EW80SW-6255SZXZ}	Disconnect
2671	AVW0409	SystemsEngineer	2 - ResearchAndEngineering	3 - Engineering	Logon	2010-04-01 07:52:00	PC-4038	07:52:00	{C0Z6-E2TE83JL-6970FMRN}	Connect
2672	AVW0409	SystemsEngineer	2 - ResearchAndEngineering	3 - Engineering	Logon	2010-04-01 07:52:00	PC-4038	07:52:00	{D8O2-D8HS17YL-6254DVEE}	Disconnect

2673 rows × 10 columns

Figure:16 Final Dataset

5.5 Label Encoding the Categorical data

|:

	user	logof_Time	logon_Time	role	functional_unit	department	date_x	pc_x	activity
0	WCR0044	05:02:50	05:19:09	8	0	12	2010-02-01 05:02:50	PC-9174	0
1	WCR0044	05:02:50	05:19:09	8	0	12	2010-02-01 05:02:50	PC-9174	1
2	WCR0044	05:02:50	05:19:09	8	0	12	2010-02-01 05:19:09	PC-9174	0
3	WCR0044	05:02:50	05:19:09	8	0	12	2010-02-01 05:19:09	PC-9174	1
4	WCR0044	05:02:50	05:19:09	8	0	12	2010-02-01 06:22:11	PC-5494	0
...	...	...	...	...	...	...	...	...	...
5543	BJW0369	04:53:11	05:07:24	8	0	12	2010-04-01 04:31:52	PC-9328	1
5544	BJW0369	04:53:11	05:07:24	8	0	12	2010-04-01 04:53:11	PC-5172	0
5545	BJW0369	04:53:11	05:07:24	8	0	12	2010-04-01 04:53:11	PC-5172	1
5546	BJW0369	04:53:11	05:07:24	8	0	12	2010-04-01 05:07:24	PC-5172	0
5547	BJW0369	04:53:11	05:07:24	8	0	12	2010-04-01 05:07:24	PC-5172	1

5548 rows × 9 columns

Figure:17 Encoded Dataset

5.6 Training The Model.

```
Epoch 1/400
17/17 [==============================] - 17s 165ms/step - loss: 275340.8750 - val_loss: 2454.6370
Epoch 2/400
17/17 [==============================] - 1s 69ms/step - loss: 78520.8203 - val_loss: 1573.8989
Epoch 3/400
17/17 [==============================] - 241s 15s/step - loss: 49496.4570 - val_loss: 1021.9841
Epoch 4/400
17/17 [==============================] - 1s 82ms/step - loss: 65234.5156 - val_loss: 1139.6537
Epoch 5/400
17/17 [==============================] - 1s 88ms/step - loss: 111399.0078 - val_loss: 396.0569
Epoch 6/400
17/17 [==============================] - 1s 72ms/step - loss: 154046.9531 - val_loss: 327.8267
Epoch 7/400
17/17 [==============================] - 1s 59ms/step - loss: 107575.2891 - val_loss: 219.2857
Epoch 8/400
17/17 [==============================] - 1s 59ms/step - loss: 63239.1875 - val_loss: 188.9457
Epoch 9/400
17/17 [==============================] - 1s 82ms/step - loss: 34268.3125 - val_loss: 183.4028
Epoch 10/400
17/17 [==============================] - 1s 70ms/step - loss: 21369.1250 - val_loss: 194.7276
```

Figure:18 Training the model

5.7 Final Output

if result is '0' then the user is Normal else the user is Malicious.

	user	result
60	SAM0359	0
61	PHJ0041	0
62	PHJ0041	0
63	WCR0044	0
64	WCR0044	1
...	...	...
336	SAM0359	1
337	SAM0359	0
338	PHJ0041	0
339	LKH0051	0
340	SAM0359	0

281 rows × 2 columns

Figure:19 Final output of malicious and normal users

5.8 Plotting the Graph on the Results.

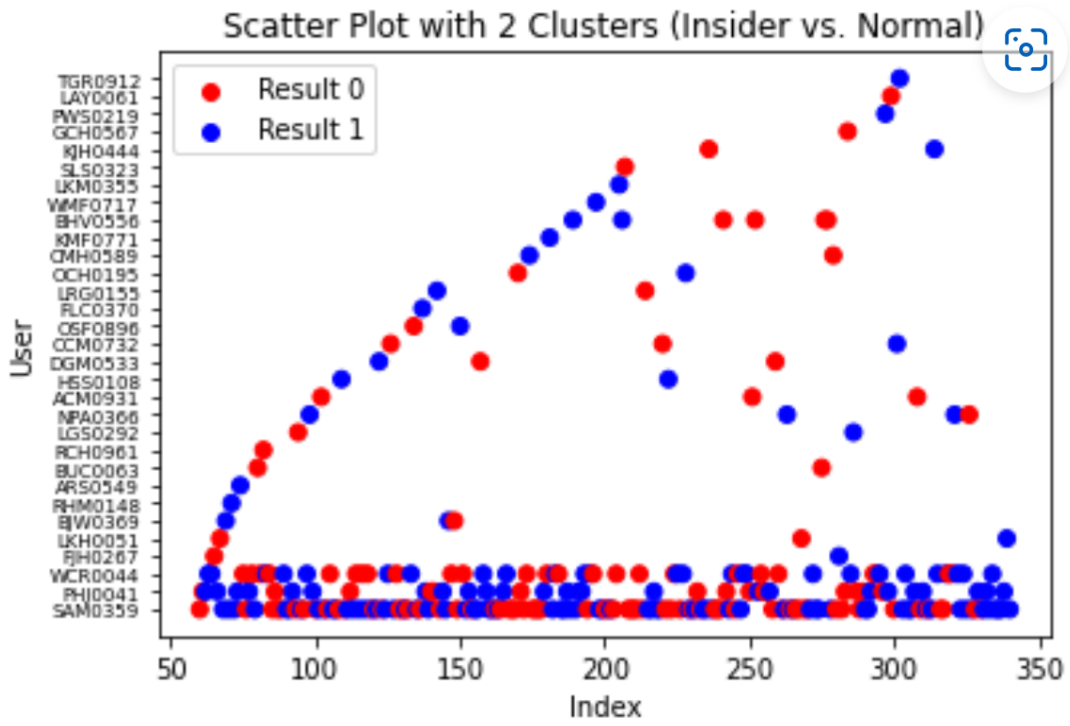

Figure:20 Final Result of malicious and normal users

In above graph,

- Result 0 indicates the normal user
- Result 1 indicates the malicious user

6. CONCLUSION

We examined the insider threat issue and found that mitigating it was a difficult task. Today, insider threats are mitigated using user controls, user behavior monitoring, and physical security controls. In this study, deep learning based on internal intrusion detection is proposed. The main purpose in creating this strategy is to use it for data users in organizations with limited processing and memory. In addition, the design process is simple and flexible, with minimal need for domain knowledge.

"BiLSTM-Autoencoder" is used by using internal threat scenario to detect insiders. The model has been trained and tested on CMU CERT r4.2. The platform used for the evaluation program is Anaconda 2018.12, Build: py37_0 using Jupyter Notebook 5.7.4.

FUTURE WORK:

In the future, we aim to explore and analyze all possible scenarios for detecting insider threats and develop a comprehensive solution to address them. This solution will include the development of an intuitive and user-friendly interface for organizations that can alert them whenever a potentially malicious activity is detected. Our ultimate goal is to provide organizations with a powerful tool to prevent insider attacks, protect sensitive information, and minimize financial and reputational damage caused by malicious employees.

REFERENCES

[1] Behavioral Based Insider Threat Detection Using Deep Learning.-Rida Nasir, Mehreen Afzal,Rabia Latif,Waseem Iqbal. (2021)

[2] Insider Threat Detection Using Deep AutoEncoder and Variational Autoencoder Nueral networks-Efthimios Pantelidis,Gueltoum Bendiab, Starvros Shiaeles, Nicholas Kolokotronis. (2021)

[3] Insider Threats Detection using CNN-LSTM Model – Ahmed Saudi, Zaid Al-ibadi, Yang Tong, Csilla Farkas. (2018)

[4] DeepMIT: A Novel Malicious Insider Threat Detection Framework Based on Recurrent Neural Network – Degang Sun, Meimei Li, Meichen Liu, Zhixin Shi. (2021)

[5] Insider Report 2018,CA Technol., New York,NY,USA. (2018)

[6] Insider Threat Report 2019,CA Technol., San Jose,CA, USA. (2019)

[7] S.R Band D.M Capelli L Fischer A.P.Moore E.D. Shaw and R.F. Trezeeiak,"Comparing insider IT sabotage and espionage: A model based analysis," Carnegie Mellon Univ.,Softw.Eng.Inst. Pittsburg PA,USA,Tech. Rep. CMU/SEI-2006TR-026. (2006)

[8] F. Meng,F. Lou,Y.Fu and Z.Tian,"Deep learning based attribute classification insider threat detection for data security". (2018)

[9] F. Januja , A. Masood,H. Abbas and I. Rashid,"Handling Insider threat through supervised machine learning techniques," Procedia Computer Science. (2020)

[10] D. Zhang,Y. Zhang,Y.Wen,Y.Xu,J.Wang,Y.Yu and D. Meng,"Role based log analysis applying deep learning for insider threat detection," (2018)

[11] S. Horchreiter and J. Schmidhuber,"Long short-term memory," Neural computation.vol. 9, no. 8, pp.1735-1780 (1997)

[12] Singh,Malvika,Babu M.Mehtre and S.Sangeetha."User behavior profiling using ensemble approach for insider threat detection. (2019)

[13] Zaytsev A.S., and A.A.Malvuk "Identifying a potential insider using classification models. (2020)

[14] Lv, Bin, Dan Wang, Yan Wang, Qiujian Lv, and Dan Lu. "A hybrid model based on multi-dimensional features for insider threat detection." (2017)

[15] Majeed, Abdul, Raihan ur Rasool, Farooq Ahmad, Ma- soom Alam, and Nadeem Javaid. "Near-miss situation based visual analysis of SIEM rules for real time network security monitoring." (2019)

[16] Junhong Kim, Minsik Park and Haedong Kim, et al. “Insider Threat Detection Based on User Behavior Modeling and Anomaly Detection Algorithms,” (2019)

[17] Ahmed Saaudi,Zaid Al-Ibadi and Yan Tong. “Insider threats detection using CNN-LSTM model,” (2018)

[18] Jiuming Lu and Raymond K. Wong. “Insider Threat Detection with Long Short-Term Memory,” in the Australasian Computer Science Week Multiconference, (2019)

[19] Mohammed Nasser Al-Mhiqani1, Rabiah Ahmad1 and Z. Zainal Abidin et al. “A Review of Insider Threat Detection: Classification, Machine Learning Techniques, Datasets, Open Challenges, and Recommendations” (2020)

[20] Anagi Gamachchi,Li Sun and Serdar Boztas. “A Graph Based Framework for Malicious Insider Threat Detection,” (2020)