Project Report: Insider Threat Detection using Unsupervised Learning

Project submitted for the partial fulfillment of the requirements for the course

CSE 337L: Cryptography Lab

Offered by the

Department Computer Science and Engineering

School of Engineering and Sciences

Submitted by

1) Sreemanth V,AP22110010048

2) Bhargav K,AP22110010051

3) Naresh B,AP22110010148

4) Bhaskar Y,AP22110010438

5)Pardhiv P,AP22110010785

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[April, 2025]

1. Introduction

Purpose of the Project

The increasing complexity of digital systems and the rise of remote work have escalated the risk of **insider threats**—malicious activities conducted by individuals within an organization, such as employees or contractors. These threats are often more difficult to detect than external attacks because insiders typically operate with legitimate credentials and access.

The goal of this project is to develop a **machine learning-based system** that can automatically identify suspicious behavior patterns in a large-scale organizational environment. Specifically, the project focuses on analyzing activity logs from the **CERT Insider Threat Dataset**, which simulates real-world organizational behavior, including emails, file accesses, and device usage.

The objective is twofold:

1. **Detect anomalous behavior patterns** using unsupervised machine learning algorithms.

2. **Compare the effectiveness** of different models—namely **Isolation Forest** and **K-Nearest Neighbors (KNN)**—in terms of anomaly detection quality, execution time, and clustering performance.

Why Unsupervised Learning?

In real-world cybersecurity applications, obtaining labeled insider threat data is extremely rare due to privacy concerns and the rarity of known incidents. Therefore, **unsupervised learning** is a practical approach that doesn't rely on prior labels. Instead, it learns what is "normal" and flags data points that deviate from this norm.

Significance

Insider threats can cause severe damage, including:

- Data leaks
- Sabotage of internal systems
- Financial losses
- Reputation damage

With massive logs generated daily, **manual detection is infeasible**. By using machine learning models like Isolation Forest and KNN, organizations can automate this detection process and **proactively identify potential threats** before any serious damage occurs.

2. Background

Why It Is Relevant

1. Real-World Impact of Insider Threats

Insider threats are among the most challenging and **damaging cyber risks** faced by modern organizations. While external attackers can be stopped with firewalls or intrusion detection systems, insiders already have **legitimate access** to sensitive systems and data.

Example Scenario:

Imagine an employee at a financial institution with access to client data. If this employee starts emailing encrypted attachments to external domains after working hours, it could be a **data exfiltration attempt**. Manual detection is difficult because:

- The employee has valid access
- Activity may seem normal if viewed in isolation
- Large organizations generate **millions of logs** per day

This is where **unsupervised anomaly detection** plays a crucial role—it automatically flags patterns that **deviate from normal behavior**.

2. Increasing Digital Footprint & Remote Work

With remote work and BYOD (Bring Your Own Device) policies, the **attack surface has expanded**, increasing the potential for insider misuse. Behavioral monitoring through ML can act as a **non-intrusive, continuous defense mechanism**.

Possible Ways to Solve Insider Threat Detection

Several approaches exist, each with its own pros and cons:

1. Rule-Based Monitoring

Description: Static rules are defined (e.g., flag if file > 10MB is emailed).

- Easy to implement
- Fails to detect unknown behavior patterns

- High false positive rate

2. Supervised Machine Learning

Description: Uses labeled data to learn normal vs malicious behavior.

- High accuracy if labeled data is available
- Insider threat labels are **rare** and **sensitive**
- Doesn’t generalize to unseen behaviors

3. Unsupervised Anomaly Detection (used in this project)

Description: Learns normal behavior, flags deviations.

- No labels needed
- Can detect **new attack types**
- Scales well with large log datasets
- Requires tuning and feature engineering

4. Deep Learning-Based Detection

Description: Models like LSTMs or Autoencoders learn sequences or representations.

- Captures complex behavior patterns
- Computationally expensive
- Less interpretable ("black-box")

3. Proposed Approach

Algorithm 1: Isolation Forest

Isolation Forest works by **randomly selecting features and splitting values**. Anomalies are isolated quickly due to their rarity and distinctiveness.

Steps:

1. Randomly select a feature and split value
2. Repeat the process to build trees
3. Average path length for each point is calculated
4. Shorter paths indicate anomalies

Pros:

- Scalable to large data
- Doesn’t assume distribution
- Effective with high-dimensional data

Algorithm 2: KNN-Based Anomaly Detection

KNN computes the **distance to the k-nearest neighbors**. Points far from others are flagged as anomalies.

Steps:

1. Compute Euclidean distance to k nearest neighbors
2. Average these distances
3. Threshold the top $X\%$ largest distances as anomalies

Pros:

- Intuitive and simple
- Captures local density changes

System Design

Modules:

4. **Data Ingestion**: Reads email.csv and psychometric.csv
5. **Data Preprocessing**:
    a. Convert timestamps, extract features (e.g., hour of day)
    b. Count number of recipients in to, cc, bcc
6. **Feature Engineering**:
    a. One-hot encoding, label encoding for user IDs
    b. Merge psychometric scores (O, C, E, A, N)
7. **Scaling**:
    a. StandardScaler applied for uniformity across features
8. **Anomaly Detection Models**:
    a. Isolation Forest
    b. KNN (using nearest neighbor distances)
9. **Evaluation & Visualization**:
    a. Distribution plots
    b. Bar chart comparing anomaly counts
    c. Silhouette score for cluster separation

Data Flow Chart

```
Start
 |
 ▼
Load CERT Email Dataset + Psychometric Dataset
 |
 ▼
Preprocess Data
 ├─ Convert timestamp to hour
 ├─ Count 'to', 'cc', 'bcc' recipients
 ├─ Handle missing psychometric values
 └─ Encode user ID
 |
 ▼
Merge Email Data with Psychometric Scores
 |
 ▼
Feature Engineering
 ├─ Extract relevant columns (hour, size, attachments, OCEAN, etc.)
 └─ Scale features using StandardScaler
 |
 ▼
Apply Anomaly Detection Algorithms
 ├─ Isolation Forest
 |    └── Fit model → Predict anomalies → Label 0 (normal), 1 (anomaly)
 └─ KNN-Based Detection
    ├─ Compute k-nearest neighbors
    └─ Label top X% of farthest points as anomalies
 |
 ▼
Analyze & Compare Results
 ├─ Count anomalies from both methods
 ├─ Visualize anomaly distributions
 └─ Compute Silhouette Score for quality assessment
 |
 ▼
Interpret Results & Conclude
 |
 ▼
End
```

4.Results & Discussion:

Anomaly Detection Summary

Two models were implemented and compared:

Model	Total Data Points	Anomalies Detected	Normal Data Points
Isolation Forest	2,631,979	**26,300**	2,605,679
KNN (Distance-based)	2,631,979	**131,499**	2,500,480

Observation:

- Isolation Forest identified fewer anomalies, suggesting it is more conservative.
- KNN detected more, possibly flagging behavior that's unusual but not necessarily malicious.

Silhouette Scores

To evaluate the clustering quality of anomalies:

Model	Silhouette Score
Isolation Forest	0.52
KNN	0.46

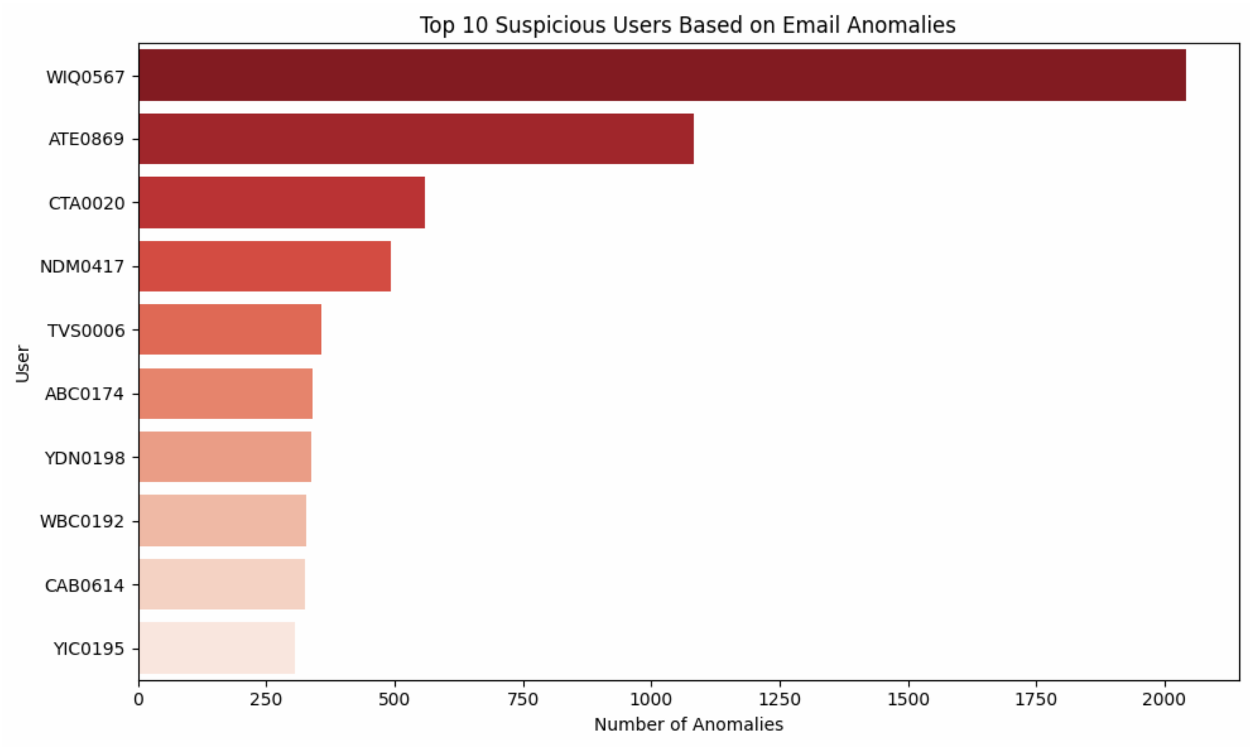

Figure 1 : Top 10 suspicious users based on email anomalies

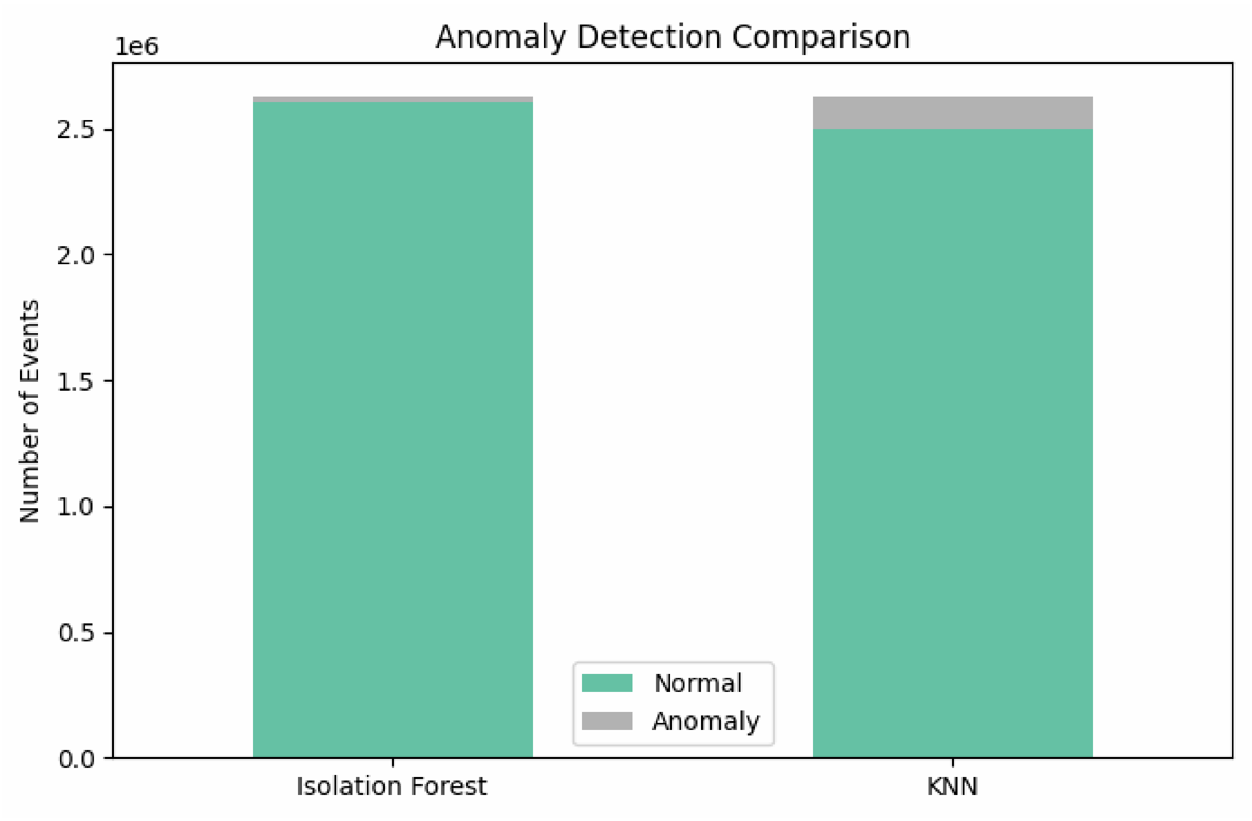

Figure 2 : Anomaly Detection comparison

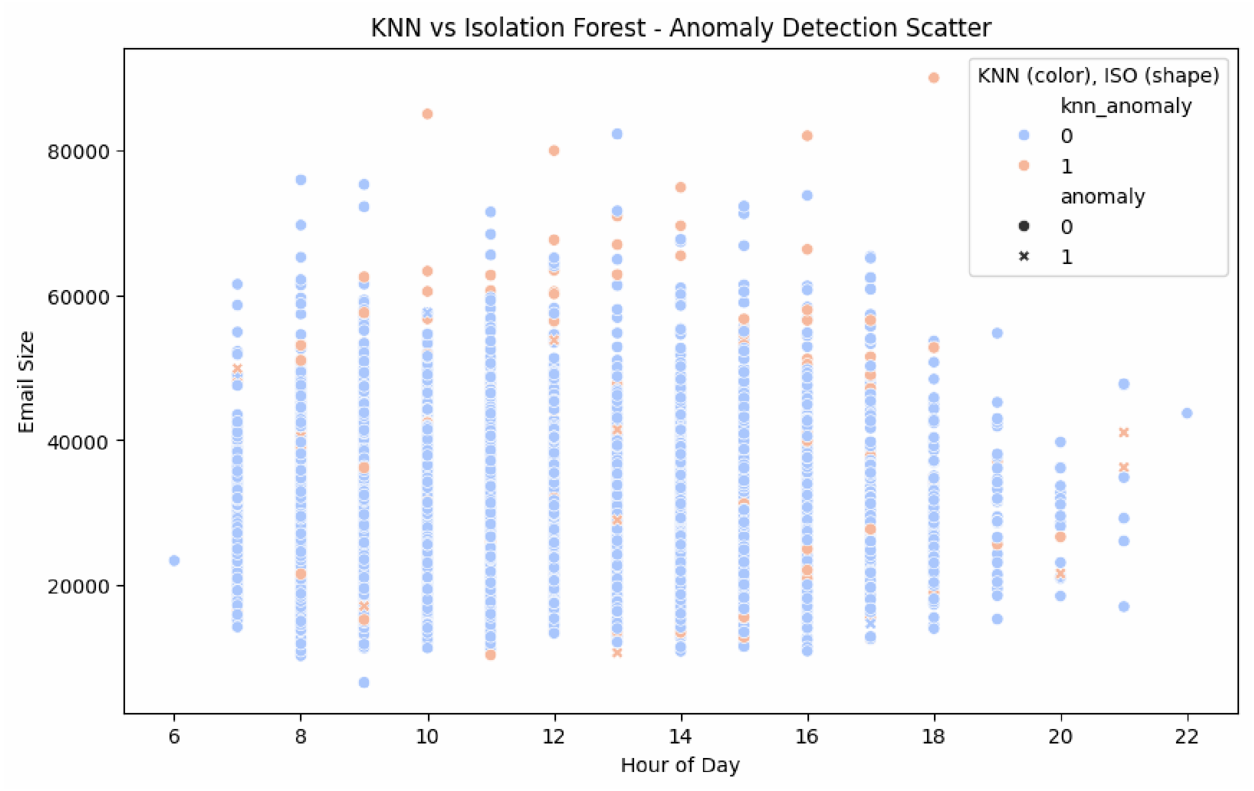

Figure 3 : KNN vs Isolation Forest

5.Conclusion

In this project, we successfully developed an unsupervised anomaly detection system to detect insider threats using the CERT Kaggle dataset. We implemented two different anomaly detection models: **Isolation Forest** and **K-Nearest Neighbors (KNN)**, comparing their performance in terms of anomaly detection quality and execution time.

The key findings of our comparison are:

- **Isolation Forest** proved to be more conservative, detecting fewer anomalies, and showed a slightly better **Silhouette Score** (0.52) compared to KNN (0.46).

- **KNN**, on the other hand, flagged more anomalies, suggesting that it may detect unusual behavior, even if it's not strictly malicious.

Despite the promising results, the project does face some limitations:

- The need for **feature engineering** to better capture the underlying patterns in user behavior.

- The **false positive rate** could be high in both models, which is common in anomaly detection tasks, especially without labeled data.

- The methods may not generalize well to unseen data, as the CERT dataset may not fully represent all real-world insider threat scenarios.

6.References

1. CERT Insider Threat Dataset. (n.d.). Retrieved from https://www.kaggle.com/datasets/nitishabharathi/cert-insider-threat
2. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf