

EVEREST ENGINEERING COLLEGE

Sanepa-2, Lalitpur
(Affiliated to Pokhara University)



A MAJOR PROJECT REPORT ON “INSIDER THREAT DETECTION”

SUBMITTED BY

Aarati Kumari Mahato	[18120001]
Kamana Joshi	[18120035]
Neha Sah	[18120050]
Susmita Khadka	[18120092]

SUBMITTED TO

**DEPARTMENT OF INFORMATION TECHNOLOGY ENGINEERING
EVEREST ENGINEERING COLLEGE
SANEPA, LALITPUR
SEPTEMBER, 2022**

DECLARATION

We hereby declare that the report of the project entitled “Insider threat detection” which is being submitted to the Department of Computer and Information Technology Engineering, Everest Engineering College, Sanepa, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Information Technology Engineering, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this word.

Aarati Kumari Mahato	[18120001]	signature
Kamana Joshi	[18120035]	signature
Neha Sah	[18120050]	signature
Susmita Khadka	[18120092]	signature

CERTIFICATE OF APPROVAL

The project report entitled “Insider Threat Detection” submitted by Aarati Kumari Mahato, Kamana Joshi, Neha Sah and Susmita Khadka in partial fulfillment of the requirement for the Bachelor’s degree in Information Technology Engineering has been accepted as a bonafide record of work independently carried out by the group in the department.

.....

Dinesh Dangol

Associate Professor

Nepal Engineering College

.....

Santa Bahadur Basnet

Project Supervisor

Department of Computer
and IT Engineering

.....

Nischal Regmi

Project Coordinator

Department of Computer and IT
Engineering

.....

Anuj Ghimire

Head of Department

Department of Computer and IT
Engineering

COPYRIGHT

The author has agreed that the library, **Everest Engineering College (EEC)**, Sanepa , Lalitpur may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purposes may be granted by the lecturers, who supervised the project works recorded herein or, in their absence, by the Head of Department wherein the project report was done. It is understood that recognition will be given to the author of the report and to the Department of Information Technology, EEC, in any use of the material in this project report. Copying or publication or other use of this report for financial gain without approval of the Department and author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head of Department of IT, Computer Engineering
Everest Engineering College
Sanepa-2, Lalitpur
Nepal

ACKNOWLEDGEMENT

We have taken effort in this project. However, it would not have been possible without the help and support of many individuals. We would like to extend our sincere thanks to all of them.

We would like to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed supervisor **Er. Santa Bahadur Basnet** for his valuable guidance, encouragement and help for supporting the work. His useful suggestions for this work and cooperative behavior are sincerely acknowledged.

We are highly indebted to EEC and Pokhara University for that project & also creating a friendly environment to the success of our project. We are also appreciative of the effort of HOD **Er. Anuj Ghimire** for supporting us throughout this project.

We are also grateful to our teachers for their constant support and guidance. At the end we would like to express our sincere thanks to all the friends and others who helped us directly or indirectly during this project work.

ABSTRACT

Insider threats constitute one of the most important risk factors for Information Systems and the assets of an organization. For a corporation or organization's IT systems and infrastructure, insider threats are among the most dangerous risk concerns. The discovery of insider threats has drawn the attention of the academic research community worldwide, and numerous remedies have been put up to lessen their potential effects. We may have a lot of solutions for different kinds of threats from outsiders but what the world is now lacking is detecting threats from our own fellow employees. Many organizations are being victims of their own employees and in some cases, external attackers are targeting the employees to override the system through employee credentials. To reduce such threats and also to prevent anomalous behaviors, insider threats become a necessity in cybersecurity. Machine learning (ML) has been used extensively for developing insider threat detection systems in various earlier research and the choice of the best ML classification model for the detection of insider threats is still difficult, though. The use of machine learning algorithms may actually turn the classical method of log analysis into a much more efficient form which is being practiced over the recent period of time. An insider threat detection model has been proposed, where the model has been created using the Isolation Forest and one class SVM. In an Isolation Forest, randomly sampled data is carried out in a tree structure based on randomly selected features and less anomalies are expected to exist in the sample that is deeper within the trees. Similarly, the samples which end up in shorter branches indicate anomalies. The suggested model for insider threat detection has been tested using the CERT dataset.

Keywords: Insider threat, Machine learning, Hybrid, Isolation Forest, One-ClassSVM

TABLE OF CONTENTS

COPYRIGHT	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
CHAPTER 1: INTRODUCTION	1
1.1 Background	1
1.2 Motivation	3
1.3 Project Objectives	4
1.4 Application and Scope	5
CHAPTER 2: LITERATURE REVIEW	6
CHAPTER 3: METHODOLOGY	12
3.1 Approach summary	12
3.2 Choice of Machine Learning Algorithm	12
3.2.1 Isolation Forest Algorithm	13
3.2.2 One Class SVM Algorithm	16
3.3 Requirements	19
3.3.1 Hardware Requirements	19
3.3.2 Software Requirements	19
3.4 Implementation	19
3.4.1 System Block Diagram	19
3.4.2 System Activity Diagram	20
3.5 ML PipeLine	21
3.5.1 Data Cleaning	21
3.5.2 Feature Engineering	22
3.6 Model Training	23

CHAPTER 4: RESULT, ANALYSIS AND RECOMMENDATION	Error! Bookmark not defined.
4.1 Dataset	24
4.2 Result and Analysis	25
4.3 Conclusion and future enhancement	27
REFERENCES	29

LIST OF FIGURES

Fig 3.2.1.1 Steps for isolating instances	16
Fig 3.2.1.2 Example of isolation tree	17
Fig 3.2.2.2 Example of illustration of One Class SVM	18
Fig 3.4.1 System Block Diagram	20
Fig 3.4.2 System Activity Diagram	21
Fig 3.6.1 Model training using isolation forest algorithm	24
Fig 3.6.2 Model training using One Class SVM	24
Fig 4.2.1 Plot of anomaly score using isolation forest	26
Fig 4.2.2 Plot of anomaly score using isolation forest for different set of parameters	27
Fig 4.2.3 Plot of file transfer per user using isolation forest for	28

LIST OF TABLES

Table I: Related works on insider threat detection using machine learning techniques	10
Table 4.1: Format of logon.csv file	25

CHAPTER 1: INTRODUCTION

1.1 Background

An insider threat refers to a cyber security risk that originates from within an organization. It often happens when a current or former employee, contractor, vendor, or partner with valid user credentials abuses their access to the damage of the organization's network systems and data. A person is considered an insider if they have or have had authorized access to or knowledge of an organization's resources, including its staff, information, equipment, networks, and systems. Examples of insider may include:

- A person the organization trusts, including employees, organization members, and those to whom the organization has given sensitive information and access.
- A person to whom an organization has provided access to a computer or a network.
- A person who creates the company's goods and services; this group also includes people who are familiar with the techniques for creating goods that are useful to the company.[1]

Insider threat refers to the potential for an insider to use their authorized access or understanding of an organization to harm that organization. This harm includes malicious, inadvertent actions that have a detrimental impact on the organization's availability, confidentiality, and integrity. Insider threat is defined by the Cyber and Infrastructure Security Agency (CISA) as the risk that an insider will intentionally or unintentionally use their allowed access to harm the department's purpose, resources, people, information, equipment, networks, or systems.

Insider threats can be either intentional or unintentional.

- Unintentional Threat
 - Negligence – Negligent insiders typically are aware of security and or IT policies but choose to ignore them, creating risk for the organization. Examples ignoring notifications to install new updates and security patches, misplacing or losing a portable storage device carrying critical information.

- Accidental – Organizations can successfully work to minimize accidents, but they will occur; they cannot be completely prevented, but those that occur can be mitigated. Examples include mistyping an email address and accidentally sending a sensitive business document to a competitor, unknowingly clicking a hyperlink, opening an attachment that contains a virus within a phishing email.
- Intentional Threats

Threats made with the intent to harm a company for personal gain or to address a Personal issues are known as intentional threats. The intentional insider is often synonymously referenced as a “malicious insider”. The motivation is personal gain or harming the organization.
- Other Threats
 - Collusive Threats - Collusive threats are a subclass of malicious insider threats in which one or more insiders work together with an outside threat actor to compromise an organization. In these situations, cybercriminals usually enlist the help of one or more insiders to commit fraud, intellectual property theft, espionage, or a combination of the three.
 - Third-Party Threats - Third party threats are usually vendors or contractors who are not official employees of a company but who are given access to its resources like facilities, systems, networks, or personnel in order to do their tasks. Their threats may be direct or indirect threats.[2]

The traditional approach of insider threat and anomaly detection used to be via observing log files line to line, which is obviously not an easy process. The use of machine learning algorithms may actually turn that classic method into simpler form which is being practiced over the recent period of time. Analyzing security log data is critical for any information security department of the company. Still, with the growing number of devices and connections, it is becoming a difficult task for security administrators to analyze and review the logs. To create a standalone anomaly detection system and apply different machine learning algorithms to identify the most suitable

algorithms which can be used to log analysis. An average web server receives visitors in thousands per day, generating a log a volume size of more than a gigabyte, and to analyze this data by a system administrator is an impossible task unless a considerable number of man-hours are dedicated to that single task. The majority of the commercial of the self security tools use log-based anomaly detection systems, which are based on the pattern matching which is only effective in detecting pre-programmed or known attack patterns and is not effective in detecting zero- day attacks and takes time from security professionals to update the system with new patterns. Advanced anomaly detecting systems need to detect both known and unknown patterns automatically and identify key areas in a security network where the system administrator can focus on which our project is developed to solve.

1.2 Motivation

As organizations' critical assets have been digitized and access to information has increased, the nature and severity of threats have changed. Insiders who work for an organization have more power than ever to abuse their access to crucial organizational resources. Insiders are aware of the locations of valuable assets as well as what is significant and crucial. Their organizations have given them authorized access to these assets and the means to compromise the confidentiality, availability, or integrity of data. One of the biggest difficulties in cybersecurity is the insider threat, which is now a well acknowledged problem. This phenomenon suggests that in order to accurately and quickly identify a malicious insider, dangers call for specialized detection methods, tools, and systems. The insider threat is still regarded as the main security worry in enterprises and has been mentioned as such. A malevolent insider who wants to hurt an organization can do so by, for instance, destroying a crucial IT system or stealing intellectual property to benefit a new employer or a competitor. Organizations rely on cyber systems to support important missions. Insider threat detection discusses the difficulty of detecting insider threats, methods for doing so, and the benefits of using machine learning-enabled software. Additionally, this includes actions taken before an attack that may suggest the possibility of an incident as well as system vulnerabilities that insiders use in their attacks. By tracking all the related information, Insider Threat Detection helps organizations instrument their systems and change their practices to lower both the frequency and the severity of insider attacks and preserve the integrity of their most critical information. Data

sources for insider threat often include cyber observables: things that people can be observed doing on their computer. For example, increased use of web-based email or a cloud-based data-storage application might indicate an intention to illegally distribute intellectual property that belongs to the organization, providing cyber-observable evidence of an attack. Other cyber observables include logon/logoff time, files accessed on a network, building card swipes, and organization email usage, among others. When cyber observables coincide with behavioral indicators, such as poor or erratic job performance, expressed disgruntlement about being passed over for promotions, etc., a profile may emerge of an insider who could be planning to harm the organization. The objective of the Insider Threat Detection project is to increase awareness of these types of observable activities both inside and outside of cyberspace so that organizations can prevent these attacks or mitigate the effect of these attacks.

1.3 Project Objectives

The main objectives of this project are:

1. To build unsupervised machine learning models in order to be able to detect anomalous behaviors and flag them as threats.
2. To generate the anomaly scores of users.
3. To provide various diagrammatic visualizations.

1.4 Application and Scope

Our project is mostly used to detect the insider threat of an organization or an institution. Numbers of threats are increased in different forms like when companies employee's login the companies' user and id outside of the organization with an intention to misuse that access negatively and may send the organizational critical information to the competitor. To track this unnecessary login, we used the insider threat detection system.

The range of insider threat detection is frequently quite unclear. We will outline the topics that are included and excluded from this survey for the sake of clarity. This article largely focuses on systems for identifying malevolent insiders. All of these systems are founded on the idea of spotting unusual or inconsistent employee behavior. The mechanisms differ only in the type of behavior analyzed. For example, some mechanisms analyze employee's printing behaviors, while others may analyze employee's UNIX commands. Our survey explores the behavior types that have been analyzed, and gathers major detection mechanisms of each behavior type. We note that external threat detectors can also detect insider threats, since malicious insiders can use the same attacks as external attackers, such as denial of service, virus planting and spamming. In this article, we won't examine external threat detectors like security information and event management systems, application hardening, firewalls, and network traffic analysis-based solutions, instead concentrating on internal threat detection mechanisms. Insider threat prevention is closely related to insider threat detection. In order to prevent threats, it is important to implement sound policies, procedures, and practices, including monitoring information flow, managing resource utilization, documenting the separation of roles, and using honeypots. This survey will not cover these areas. In a similar way, insider threat forensics—such as the establishment of secure backup and recovery mechanisms and the auditing of employee cyber activities—as well as insider threat mitigation will not be discussed.

CHAPTER 2: LITERATURE REVIEW

In this paper, the author suggested insider-threat detection methods based on user behavior modeling and anomaly detection algorithms. Individual users' varied actions are turned into a structured dataset throughout the user behavior modeling process connecting each row to an instance (user-day, email content, user-week) and each column with input variables for anomaly detection models. They created three datasets, including a daily activity summary dataset, using the CERT database based on user activity logs, an e-mail content dataset based on topic modeling, and an e-mail communication network dataset based on the user's account and sending/receiving information. They constructed insider-threat detection models using these three datasets by employing machine learning-based anomaly detection algorithms to simulate real-world organizations in which only a few insiders' behaviors are potentially malicious. Based on the daily activity summary dataset, the anomaly detection model yielded at most 53.67% of the detection rate by only monitoring the top 1% of suspicious instances. When the monitoring coverage was extended to the top 30% of anomaly scores, more than 90% of actual abnormal behaviors were detected for two roles among the three evaluated. On the evaluation of the e-mail content datasets, at most 37.56% of malicious emails were detected with the 1% cut-off value while the detection rate increased to 65.64% (98.67% at most) when the top 30% of suspicious emails were monitored. Although the proposed framework was empirically supported, there are some limitations in the current research. There is a lack of proper integration between the various anomaly detection outcomes and this approach cannot detect malicious behavior in real time. [3]

In this research paper, the author provided an extensive view and deep understanding of the field of insider threats by surveying and categorizing the existing literature focused on theoretical, technical, and statistical aspects of insider threats as well as analyzed many insider threats incidents and provided statistical information about insiders. According to a 2018 assessment on the insider threat, 53% of threats in the previous year came from within organizations. Additionally, 27% of the firms questioned said that internal attacks were the source of the attacks. It was observed that the chances of insider threats are approximately 66% due to network access and 27% due to physical access. They introduced CERT, NSL KDD OR KDD-99, APEX, RUU, TWOS dataset and presented a taxonomy of contemporary insider types, access, level, motivation, insider profiling, effect security property, and methods used by attackers to conduct attacks and a review of notable recent works on insider threat detection, which covers the analyzed behaviors, machine-learning techniques, dataset, detection methodology, and evaluation metrics. In addition, this survey highlights the challenges faced by other researchers which were performance, insider threat datasets, high dimensionality, physical and cyber behavior, analysis interval, costly and time consuming and the policy and also provide recommendation with a clear picture of research direction for future research which includes hybrid solution, logs, evaluation and validation, human aspect and physical features.[4]

The author presented the UEBA that uses predefined Rules, Anomaly Detection and Machine Learning techniques in its process of hunting insider attacks. According to Insider Threat Report 2019, 68% of organizations are experiencing frequent insider attacks. Data containing valuable information regarding user activities for example, login and logout information, e-mail activities, web searches, files activities, servers accessed, applications, USB etc. is collected from various sources such as system logs, application logs, network devices, network traffic, net flows etc. The raw information gathered from these data sources is then passed through the analytics engine which connects the dots together to formulate a result of ongoing user activities. Anomalous activities of users are detected and assigned a weight according to severity and confidence of the detection system which helps in computing the risk score of the individual users and hence can identify compromised users. In this paper They discussed the UEBA techniques put out in the literature, the broad design and features of the best UEBA solutions now on the market, as well as their advantages and disadvantages. There were several issues that needed to be addressed in development and deployment of the UEBA system which were duration of training data, reducing false positives, risk score calculation. In evaluating UEBA solutions, threat detection capabilities must be taken into account in addition to use cases and scalability. Advanced behavior analytics system architecture that incorporates information from many data sources including social media activities, email and network access logs are also required. [5]

In this method, the Google TensorFlow program was used to create the Convolutional Neural Network (henceforth referred to as CNN) algorithm, which was trained to recognize potential threats from images generated by the supplied dataset. From the examination of the images that were produced and with the help of Machine Learning, the question whether the activity of each user is classified as "malicious" or not for the Information System was answered. By turning activity data into a visual representation, they test the viability of utilizing machine learning algorithms to detect malicious behavior. The procedures they used to finish the task at hand and make conclusions were data sharing and creation of files based on the data of the user under consideration, importing the data files we created in the "D3.js" library, selecting an appropriate image creation plan, examining the application library's patterns and creating images of the user in question that included his/her activity during each day, creating images, implementing and training the CNN algorithm in TensorFlow program and examining user behavior, which we have described as "normal" or "malicious". It had gone through three stages: a) the collection, processing, and classification of the data of the users under consideration, b) the visualization of the extracted data, and c) the use of the CNN algorithm to classify behavior into malicious or normal. The algorithm was trained and tested using a dataset of 860 created images, including both malicious and normal activity and the malicious activity was achieved. The forecasting of the results was completely successful, with a percentage that reached 100%. It should be mentioned that the classification of a user's actions as harmful depends on the Security Policy that the Company or Organization establishes to safeguard its Information System. One of the sets of guidelines for the analysis was that accessing social networking or job-search websites constituted an abnormal activity and should be classified as harmful.[6]

This work describes a method for one class learning, also referred to as unary classification or class modeling, in which the model was only trained on data from most of the classes. The model came to understand what an organization's ideal employee behavior was. The proposed paper attempts to detect the insider threat activities and monitor if any unexpected or suspicious behavior were observed by the model, which produces high reconstruction error within the model and are classified as anomalies. Training of the model implements feature vectors extracted from user log activities in a fixed window per day. This approach implements Gated Recurrent Unit (GRU) based Autoencoder to model user behavior per day and detect anomalous insider threat points. Since the model is overfitted on normal data, the error produced by normal data is extremely low while the autoencoder produces high error on malicious class of abnormal data. The dataset used in the work is Computer Emergency Response Team (CERT) r4.2 and feature vectors are derived according to the number of times a user performs certain activity within a day. Behavior learning through GRU autoencoder is used. At different thresholds, performance of the model was measured and the model demonstrated good distinction with minimum mis-classification for both classes with values of true positive and true negative rates at 79.81%. This research works presented use of one class modeling on normal data of CERT r4.2 data set. When malicious samples are put into the model during the testing phase, the overfitted model on non-malicious normal during the training period data yields substantial reconstruction error. The research focuses on employing GRU units instead of conventional LSTM units in an autoencoder model to simulate non-malicious user behavior.[7]

This study employed the CERT dataset. r4.2 along with a series of machine learning classifiers to predict the occurrence of a particular malicious insider threat scenario - the uploading of sensitive information to wikileaks before leaving the organization. These algorithms were combined into a meta-classifier that performed more predictably than the sum of its constituent models. It also defined a methodology for performing pre-processing on organizational log data into daily user summaries for classification, and was used to train multiple classifiers. Overall, the models were evaluated through analysis of their associated confusion matrix and Receiver Operating Characteristic (ROC) curve, and the best performing classifiers were aggregated into an ensemble classifier. This meta-classifier had an accuracy of 96.2% with an area under the ROC curve of 0.988. This paper established a novel pre-processing strategy for insider threat to improve categorization outcomes based on insider threat categories. The resultant data set produced using the established methodology is used to train a series of classifiers which all outperform the predictive performance of previous strategies identified in the research. The most performant of these models were aggregated into a meta-learner algorithm using probability vote. This led to the creation of a model. with a ROC curve containing a greater area underneath than any of the other models that were explored in this work and demonstrated the applicability of this strategy for raising classifier performance as a whole but they did not develop a general model on the data set r4.2. In order to test the hypothesis that instance data tailored to each scenario creates more performant classifiers than one generalized classifier.[8]

In this paper, an attempt to address the gaps including a lack of real data, low accuracy, and a relatively low false alarm by detecting insider threats with the novelties of the present investigation first developed two deep learning hybrid LSTM models integrated with Google's Word2vec LSTM (Long Short-Term Memory) GLoVe (Global Vectors for Word Representation) LSTM. Secondly, the performance of two hybrid DL models was compared with the state-of-the-art ML models such as XGBoost, AdaBoost, RF (Random Forest), KNN (K-Nearest Neighbor) and LR (Logistics Regression). Thirdly, the present investigation bridges the gaps of using a real dataset, high accuracy, and significantly lower false alarm rate. This issue of insider threat detection requires state-of-the-art Artificial Intelligence models and utility. In this paper, an attempt was made to detect insider threats based on the deep learning hybrid model of Word2vecLSTM, GLoVeLSTM, and Machine learning models such as XGBoost, AdaBoost, RF (Random Forest), KNN (K-Nearest Neighbor), and LR (Logistics regression). It was discovered that the ML-based model XGBoost showed an accuracy of 92%, whereas DL-based word2vecLSTM and GLoVeLSTM achieved accuracy values of 73.4% and 74.00%, respectively. The non-technical aspects of insider threats require assimilation with technical issues to complete the ecosystem for better understanding and address the issue better and no framework or standard exists in the current time, which can address the evaluation of insider threat detection systems. [9]

TABLE I: Related work on insider threat detection using machine learning techniques

STUDY	ALGORITHM	DATASET	RESULT	EXPLANATION
J.Kim [3]	Anomaly detection algorithm	CERT dataset	Accuracy> 90%	The method proposes insider-threat detection methods based on user behavior modeling and anomaly detection algorithms. Experimental results show that the proposed framework can work reasonably well to detect insiders' malicious behaviors. Although the proposed framework was empirically verified, there is lack of proper integration between the various anomaly detection outcomes and cannot detect the malicious behavior in real time.
Al.Mhiquani[4]	Anomaly detection algorithm	CERT NSLKDD APEX RUU	NSLKDD Or KDD-99	This study provides an extensive view and deep understanding of the field of insider threat, discuss about the challenges and provides recommendation with a clear picture of research direction for future research.
Khalid [5]	Supervised and Unsupervised Machine Learning algorithms.			The author discussed the different approaches used in UEBA including user and role-based detection, user and entity activity mapping and highlighted the fact that open-source community still lags behind in giving a sophisticated UEBA solution.
Vasileios [6]	CNN	CERT Dataset	Training data=80% Validation data=20%	This study determined whether it was possible to detect harmful behavior by transforming activity reports into a visual representation using machine learning techniques. Using a dataset of 860 generated photos, containing both malicious and legitimate activities, the algorithm was trained and evaluated.
S.Nepal [7]	Neural network algorithm LSTM	CERT Dataset	Precision of Model= 80.1% F1 score =79.4%	This study focused on employing GRU units instead of conventional LSTM units in an autoencoder model to stimulate non-malicious user behavior.

A.J.Hall [8]	Meta-learner algorithm	CERT Dataset	MetaLearner accuracy= 96.2%	This study established a new way for processing insider threat to improve classification outcomes based on insider threat categories but did not develop a general model on data set r4.2 in order to test the hypothesis that instance data tailored to each scenario created more performant classifiers than one generalized classifier.
M. A. Haq [9]	transfer learning KNN		word2vecLS TMaccuracy =73.4% GLoVeLST M= Accuracy= 74.00% XGBoost accuracy =92%	In this paper, an attempt was made to detect insider threats based on the deep learning hybrid model of Word2vecLSTM, GLoVeLSTM, and Machine learning models such as XGBoost, AdaBoost, RF, KNN, and LR to address the gaps including a lack of real data, low accuracy, and a relatively low false alarm. The non-technical aspects of insider threats need to be assimilated with technical issues to complete the ecosystem for better understanding and address the issue better

CHAPTER 3: METHODOLOGY

3.1 Approach summary

Our approach can be summarized by the following steps:

1. Choose a relevant machine learning algorithm for prediction
2. Extract features, data preprocessing.
4. Use these features to determine at which confidence level an inside threat can be predicted
5. Visualize the prediction results.

3.2 Choice of Machine Learning Algorithm

The data set we use contains only 0.23% positives, which makes it a sparse data set. This introduces complexity in that we need to choose an algorithm appropriate to sparse data sets. Many supervised machine learning algorithms do not perform well on sparse data sets since such data sets have too little training data [10][23]. After considering the capabilities of supervised and unsupervised machine learning algorithms, we concluded that an unsupervised approach would be a better fit for our problem.

Unsupervised algorithms have the advantage that they do not need to be trained, which also opens up the door for smaller data sets. There are many different unsupervised algorithms but by reviewing several papers [18][15][30][28] and searching for an algorithm that could handle our classification problem using a sparse data set, we found Isolation Forest. These papers use sparse data sets and Isolation Forest in combination with good results, and averagely it seemed like the best standardized algorithm for the usage with sparse data sets. Furthermore, Isolation Forest has been proven efficient in prediction using sparse data sets [18]. This led us to choose Isolation Forest as a basis for our work, with the possibility of improvement of the work by comparing with other algorithms(secondly we selected OneClassSVM).

3.2.1 Isolation Forest Algorithm

Isolation Forest is a machine learning anomaly detection algorithm created by Fei Tony Liu et al. Isolation forest is an unsupervised model in which randomly sub-sampled data is processed in a tree structure based on randomly selected features. Similar to this, samples that end up on short branches tend to be anomalies since the tree found it easier to distinguish them from other observations. Isolation forest is based on the Decision Tree algorithm. Following steps are followed by an isolation algorithm.

- Among the dataset, a randomly selected subsample of the data is selected and assigned to a binary tree.
- A random feature is initially chosen from the set of all N features, and then the tree is branched based on a random threshold value between the minimum and maximum values of the selected feature.
- If the value of the data point is less than the selected threshold, it goes to the left branch else to the right thus the node is split into left and right branches.
- The node is divided into left and right branches depending on whether the value of the data point is less than the selected threshold or greater.
- The above steps are repeated to construct random binary trees.[10]

Model training is finished after an ensemble of iTress (Isolation Forest) is produced. A data point is sent through each of the trained trees while scoring is being done. Each data point is now given an "anomaly score" based on the depth of the tree that was needed to get there. This score is an accumulation of the depth discovered by each iTree. Based on the contamination, anomalies receive a score of -1 and normal points receive a value of 1. (Percentage of anomalies present in the data).

Once we have an Isolation Forest (a collection of Isolation Trees) the algorithm uses the following anomaly scores given a data point x and a sample size of m :

$$s(x, m) = 2^{\frac{-E(h(x))}{c(m)}}$$

In the equation above, $h(x)$ represents the path length of the data point x in a given Isolation Tree. The expression $E(h(x))$ represents the expected or “average” value of this path length across all the Isolation Trees. The expression $c(m)$ represents the average value of $h(x)$ given a sample size of m and is defined using the following equation.

$$c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

The equation above is derived from the fact that an Isolation Tree has the same structure as a binary search tree. The termination of a node in an Isolation Tree is similar to an unsuccessful search in a binary search tree as far as the path length is concerned. Once the anomaly score $s(x, m)$ is computed for a given point, we can detect anomalies using the following criteria:

1. **If $s(x, m)$ is close to 1 then x is very likely to be an anomaly.**
2. **If $s(x, m)$ is less than 0.5 then x is likely a normal point.**
3. **If $s(x, m)$ is close to 0.5 for all of the points in the dataset then the data likely does not contain any anomalies**

Most existing model-based approaches to anomaly detection construct a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies. Isolation Forest on the other hand uses a different approach that explicitly isolates anomalies instead of profiling normal points. Isolation Forest isolates anomalies using their quantitative properties: they are few and different. They have shown that a tree structure can be constructed effectively to isolate every single instance. In multiple dimensions this is done by randomly constructing hyperplanes

that separates instances from one another. The number of hyperplanes needed in order to separate out a single instance corresponds to the instances' anomaly score. Below we try to exemplify how this would work in two dimensions, where the instances are separated by lines. Illustration can be found in Figure 3.2.1.1.

1. A line is randomly drawn through the collection of instances.
2. An orthogonal line is randomly drawn.
3. The above process continues until all instances are isolated.
4. The iTree is built based on the instances. Each Instance is a leaf in the iTree, see Figure 3.2.1.2. Each line represents a node, in other words a decision point where the choice is made if the specific instance is larger or smaller than the value.

Isolation Forest locates anomalies effectively since they are close to the root of the tree. It works well in high dimensional problems which have a large number of irrelevant attributes and in situations where the data set does not contain any anomalies.

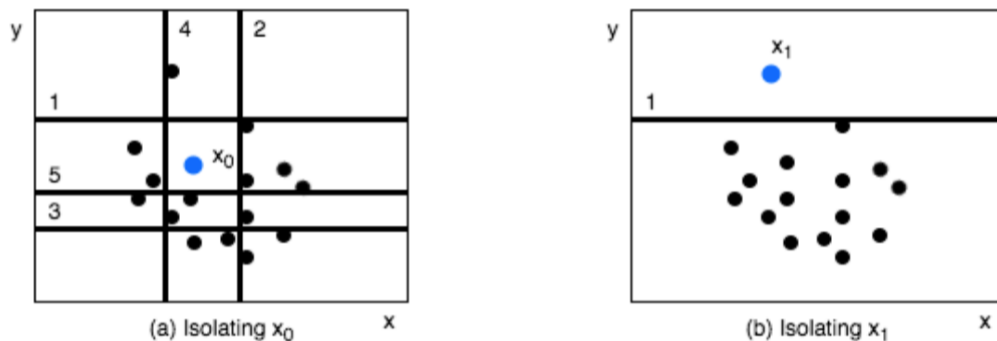


Figure 3.2.1.1: Steps for isolating instances. A normal instance (a) x_0 has an isolation path of five, and an anomaly (b) x_1 has a shorter isolation path of two.

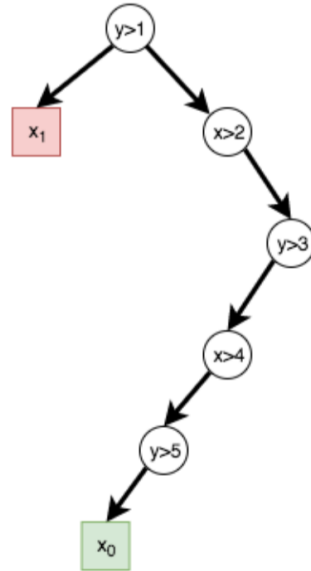


Figure 3.2.1.2: Example of the beginning of an Isolation tree, only containing x_0 and x_1 , isolating x_1 (Anomaly instance) only takes one step, which is found closer to the root of the tree compared to the normal instance x_0 found further away. The numbers represent the lines in Figure 3.2.1.1

3.2.2 One Class SVM Algorithm

One-Class SVM is an unsupervised learning technique where the specific data elements in the dataset are separated out into a single category. Outlier detection and novelty detection are examples of one-class classification where the outlier elements are detected separately from the rest of the data elements. 1-SVM works on the basic idea of minimizing the hypersphere of the single class of examples in training data and considers all the other samples outside the hypersphere to be outliers or out of training data distribution. The figure below shows the image that demonstrates the hypersphere formed by 1-SVM to learn the ability to classify out of training distribution data based on the hypersphere. One-Class Support Vector Machine models approximate a hyperplane that separates samples belonging to different classes during the training. During the classification step the labels are assigned based on which side of the plane the samples

lie. The one-class variant can be used for anomaly detection, with the hyperplane encapsulating the inliers. One-class Support Vector Machines (OCSVM), which are a type of SVM in which the algorithm attempts to separate the data points from the origin via a hyperplane of maximum distance from the origin.

The mathematical expression to compute a hypersphere with center c and radius r is

$$\min_{r,c} r^2 \text{ subject to, } \|\Phi(x_i) - c\|^2 \leq r^2 \quad \forall i = 1, 2, \dots, n$$

The expression above tries to minimize the radius of a hypersphere. However, the above formulation is very restrictive to outliers so, the more flexible formulation to tolerate outliers to an extent is given by

$$\min_{r,c,\zeta} r^2 + \frac{1}{\nu n} \sum_{i=1}^n \zeta_i$$

$$\text{subject to, } \|\Phi(x_i) - c\|^2 \leq r^2 + \zeta_i \quad \forall i = 1, 2, \dots, n$$

Here the function phi is the hypersphere transformation of x samples. The figure below shows how the formulation of a hypersphere forms a hypersphere by minimizing the radius r , center c .

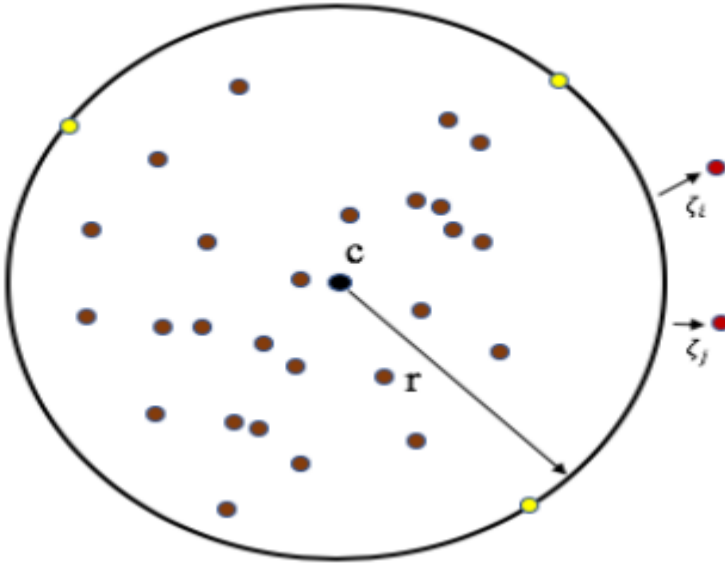


Figure 3.2.2.2: Example illustration of One-Class SVM

One-Class SVM(OCSVM) can be used for both kinds of anomaly detection applications i.e., outlier detection and novelty detection.

The SK-learn provides a class known as ‘OneClassSVM’ that internally implements the mathematical modeling of minimizing the hypersphere through training from data samples.

3.3 Requirements

3.3.1 Hardware Requirement

- A PC (min RAM 4GB, HDD 500GB, SSD adds value)

3.3.2 Software Requirement

- Any operating system (Linux, Windows, Mac).
- Web Browser: Chrome, Firefox, Opera etc.
- Code Editor: VS Code, Jupyter, Google Collab etc.
- Spreadsheet: Excel

3.4 Implementation

3.4.1 System Block Diagram

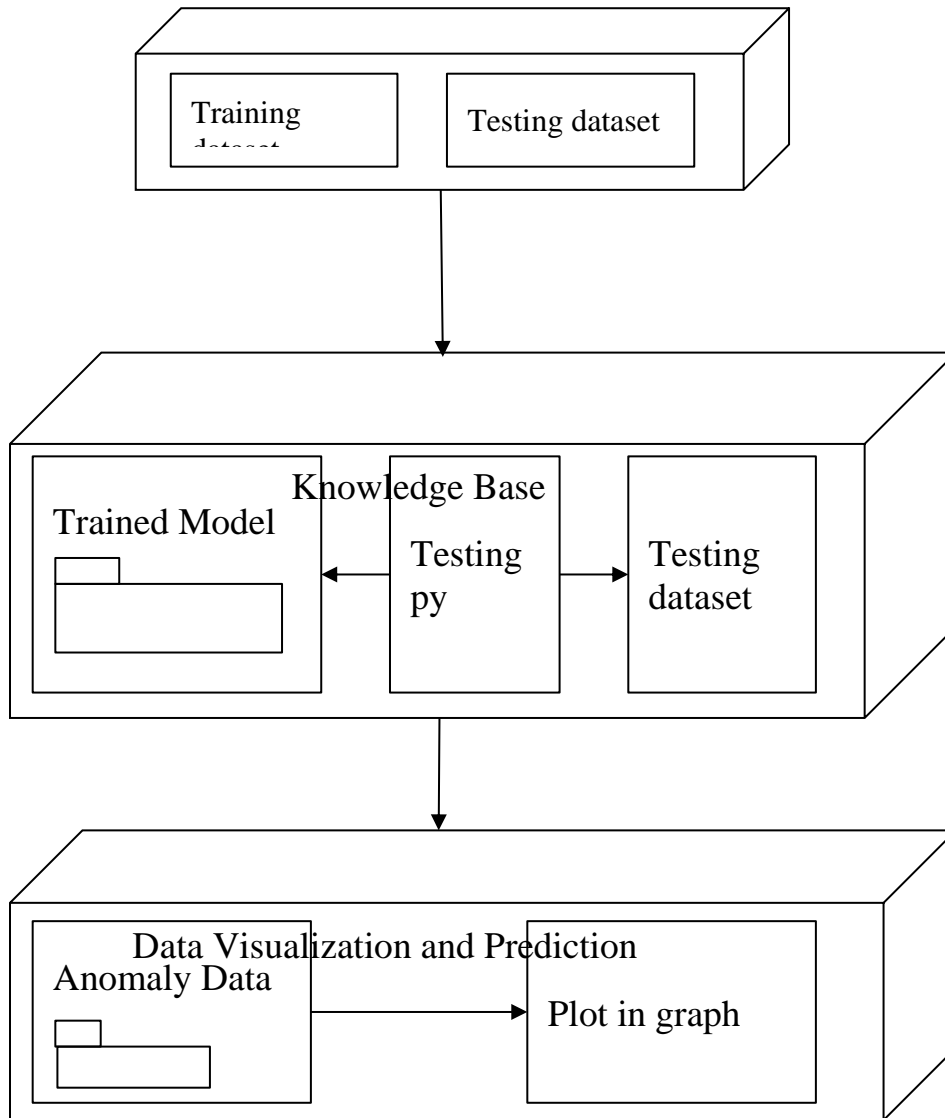


Figure 3.4.1: System Block Diagram

3.4.2 System Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

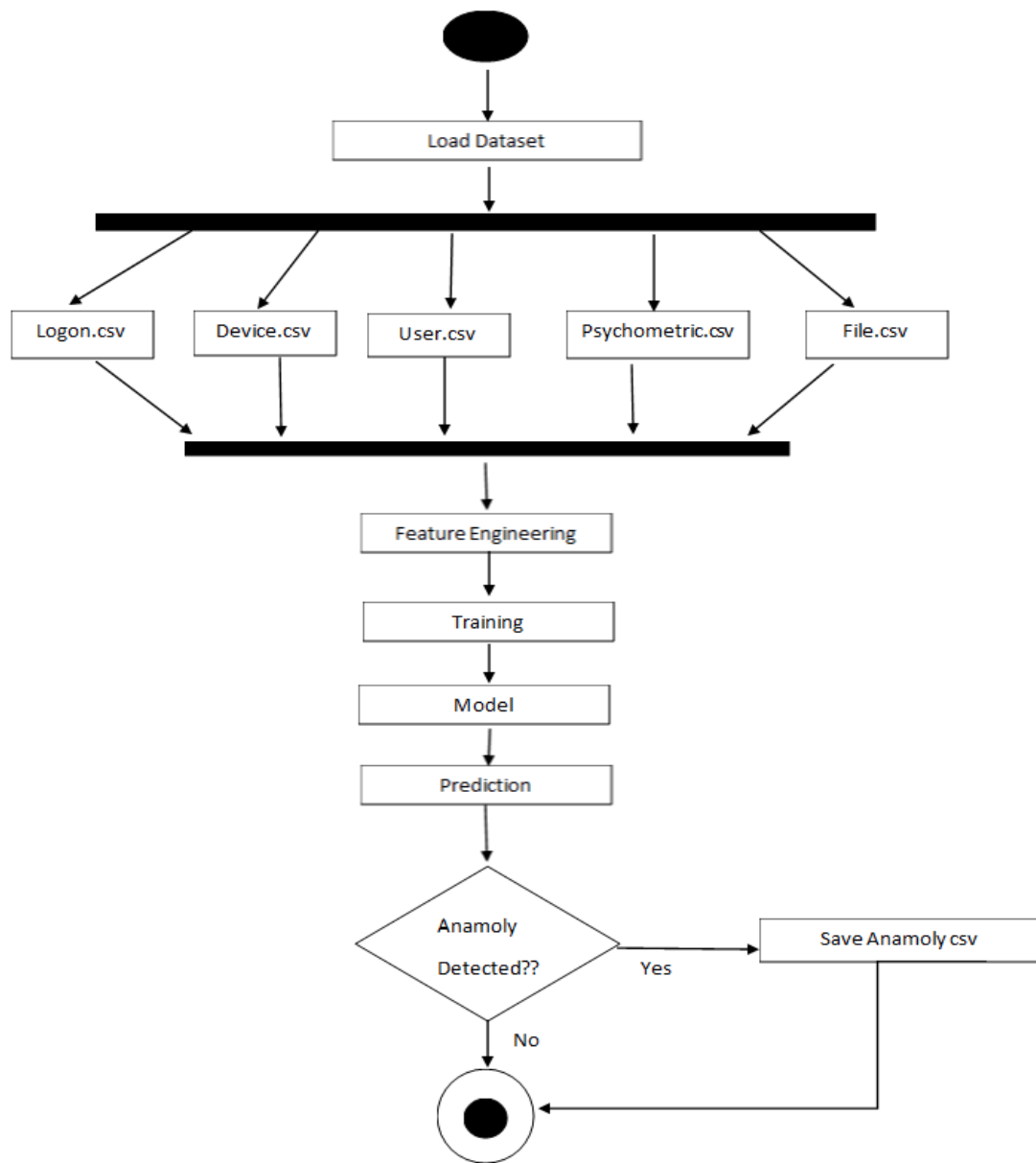


Figure 3.4.2: System Activity Diagram

3.5 ML PipeLine

3.5.1 DATA CLEANING

CERT datasets consist of data in comma separated csv files. Such files include data in numeric as well as categorical data which needs to be cleaned. Cleaning process removes unnecessary features as well as removes rows where data is missing. Example, the id column has no significance in our model therefore it is removed.

df_users

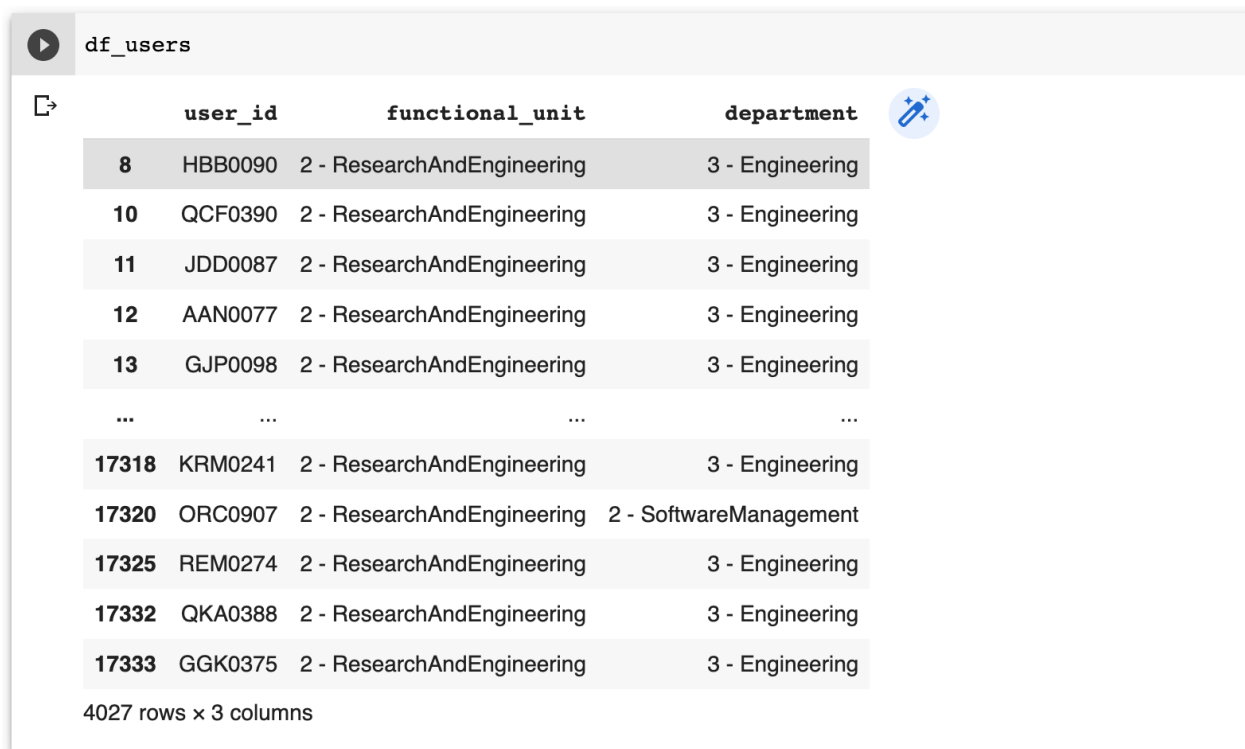
	user_id	functional_unit	department
8	HBB0090	2 - ResearchAndEngineering	3 - Engineering
10	QCF0390	2 - ResearchAndEngineering	3 - Engineering
11	JDD0087	2 - ResearchAndEngineering	3 - Engineering
12	AAN0077	2 - ResearchAndEngineering	3 - Engineering
13	GJP0098	2 - ResearchAndEngineering	3 - Engineering
...
17318	KRM0241	2 - ResearchAndEngineering	3 - Engineering
17320	ORC0907	2 - ResearchAndEngineering	2 - SoftwareManagement
17325	REM0274	2 - ResearchAndEngineering	3 - Engineering
17332	QKA0388	2 - ResearchAndEngineering	3 - Engineering
17333	GGK0375	2 - ResearchAndEngineering	3 - Engineering

4027 rows x 3 columns

Fig 3.5.1: Cleaned data sample

3.5.2 FEATURE ENGINEERING

After cleaning data we are left with features which may or may not be useful for our model. Therefore feature engineering is done. In the feature engineering process we select the features either based on the result of principal component analysis or simply perform other statistical operations on data to get meaningful features. In our case we perform the later one and obtain min, max, mode and mean of various columns.



	user_id	functional_unit	department
8	HBB0090	2 - ResearchAndEngineering	3 - Engineering
10	QCF0390	2 - ResearchAndEngineering	3 - Engineering
11	JDD0087	2 - ResearchAndEngineering	3 - Engineering
12	AAN0077	2 - ResearchAndEngineering	3 - Engineering
13	GJP0098	2 - ResearchAndEngineering	3 - Engineering
...
17318	KRM0241	2 - ResearchAndEngineering	3 - Engineering
17320	ORC0907	2 - ResearchAndEngineering	2 - SoftwareManagement
17325	REM0274	2 - ResearchAndEngineering	3 - Engineering
17332	QKA0388	2 - ResearchAndEngineering	3 - Engineering
17333	GGK0375	2 - ResearchAndEngineering	3 - Engineering

4027 rows x 3 columns

Fig 3.5.2.1 Feature extracted data from users.csv file

In users.csv file, we have 6 features(employee_name, user_id, email, role, functional_unit and department). Among them, we selected data with functional_unit of ResearchAndEngineering and the department of Engineering only for our evaluation.

df_device.head()

	id	date	user	pc	activity
372228	{A3E5-U0UK49TX-0846AICK}	03/31/2011 15:28:26	MAF0199	PC-9927	Disconnect
189765	{O5N0-Z0VW34JB-1321TGWD}	08/17/2010 17:49:18	NVM0015	PC-5313	Disconnect
218940	{X0C4-P9YN10VU-5814XEWN}	09/22/2010 14:59:45	RKP0922	PC-3127	Connect
111893	{B0F6-I7GU31SS-8230DYOZ}	05/14/2010 11:58:11	LPM0520	PC-0884	Connect
43601	{L4T4-M0RG78QV-6844PGTW}	02/22/2010 15:32:43	GJP0098	PC-6435	Disconnect

Fig 3.5.2.2 Feature extracted data from device.csv file

df_file_1.head()

	id	date	user	pc	filename	content
0	{M8R9-I5RI11PG-5467DBVR}	01/01/2010 06:51:00	RPM0600	PC-9164	5B9VCBIU.doc	D0-CF-11-E0-A1-B1-1A-E1 m45 adss arxiv 1128955...
1	{K7D4-F3MJ16HT-2340NUGJ}	01/01/2010 08:09:28	CSD0242	PC-8696	111WA4EL.txt	58-38-59-4B barbiger detmoldii labillardire 97...
2	{B1G0-C3CC52KM-8002OOIA}	01/01/2010 08:17:52	CSD0242	PC-8696	GSQEMB1R.doc	D0-CF-11-E0-A1-B1-1A-E1 rearers broody fattene...
3	{D7F8-W4CG21DB-5694PJWH}	01/01/2010 08:18:33	WXW0044	PC-9422	RDAOYBUK.txt	4B-4E-43-54 alboin salona turisindus audoin el...
4	{B7Y6-R0HX46WQ-8863CNGR}	01/01/2010 08:20:17	CSD0242	PC-8696	D92Z9FA8.pdf	25-50-44-46-2D wreckhouse till31122007 59 nia ...

Fig 3.5.2.3 Feature extracted data from file.csv file

df_logon_1.head()

	id	date	user	pc	activity
0	{Q9R3-U1PE27LQ-2164BJLT}	01/01/2010 06:20:00	SLW0616	PC-0110	Logon
1	{R1B6-S0LQ34EP-3361QPTV}	01/01/2010 06:28:00	RPM0600	PC-9164	Logon
2	{Y3R9-H0JY37RA-5902HXNX}	01/01/2010 07:04:00	DAM0170	PC-1525	Logon
3	{D5B1-I6BP26NF-3845HZNZ}	01/01/2010 07:23:00	WXW0044	PC-9422	Logon
4	{I0N6-I9PB69SZ-5272SLAT}	01/01/2010 07:31:00	RLD0349	PC-1290	Logon

Fig 3.5.2.4 Feature extracted data from logon.csv file

```
[38] df_psychometric.head()
```

	employee_name	user_id	O	C	E	A	N
29	Kendall Raven Conner	KRC0720	35	40	46	27	
535	Brynne Cassandra Bartlett	BCB0629	33	38	33	25	33
695	Hayfa Phyllis Leon	HPL0787	30	38	14	19	13
557	Ferris Ezra Delacruz	FED0275	26	16	28	13	30
836	Callum Forrest Page	CFP0581	24	47	37	42	33

Fig 3.5.2.5 Feature extracted data from psychometric.csv file

3.6 Model Training

Feature engineering leaves data in format which then can be fed into algorithms to get the output. We use the Isolation Forest Algorithm for prediction. Isolation Forest takes input in the form of a numpy array, therefore it is necessary to convert our data into that format. We convert the dataframe after feature engineering into a matrix which contains a numpy array upon which the model is fit.

```
[ ] #user pc
    forest = IsolationForest(bootstrap=False, contamination=0.1,max_features=1.0,
                             max_samples='auto',n_estimators=100, n_jobs=1, random_state=None,
                             verbose=0)
    forest.fit(df_user_pc_ct)
```

```
IsolationForest(contamination=0.1, n_jobs=1)
```

```
[ ] user_pc_a_score = forest.decision_function(df_user_pc_ct)
    print(user_pc_a_score[1:10])

[0.08861535 0.08861535 0.00259553 0.08861535 0.08861535 0.08861535
 0.08861535 0.08861535 0.08861535]
```

```
▶ user_pc_result = pd.DataFrame()
  user_pc_result['user'] = df_user_pc['user']
  user_pc_result['ascore'] = user_pc_a_score
  print(user_pc_result)
```

Figure 3.6.1: Model training using Isolation Forest Algorithm

```
[ ] # device (-user)
one_class_svm_device = OneClassSVM(kernel = 'rbf', gamma = 0.001, nu = 0.03)
one_class_svm_device.fit(device_params)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:598: FutureWarning: np.matrix usage is deprecated in 1.0
FutureWarning,
OneClassSVM(gamma=0.001, nu=0.03)

[ ] import pickle
with open('device_params.pkl', 'wb') as file :
    pickle.dump(one_class_svm_device, file)

▶
dev_file_ascore = one_class_svm_device.decision_function(device_params)
print(dev_file_ascore)
```

Figure 3.6.2: Model training using One Class SVM Algorithm

CHAPTER 4: RESULT, ANALYSIS AND RECOMMENDATION

4.1 Dataset

The CERT dataset[1] is a synthetic dataset that was created as part of a project at Carnegie Mellon University to answer a specific problem in the insider threat field: the difficulty of obtaining data. The dataset was created using several interdependent systems to create log behaviors of a virtual organization. The attacks in the CERT data were developed manually after consultations with counter-intelligence experts and included during the dataset creation.

There are several releases of the CERT dataset, varying in number of attacks, users, length, how advanced the text generation methods were, and others. We decided to use the release 4.2 (which we refer to as the original dataset), with 501 days of activity logs and 1000 users — 70 of whom performed attacks of 3 possible scenarios. While there are newer releases of the CERT data available, r4.2 is often chosen due to the number of anomalies it contains. For example, at the time of writing, the newest dataset, r6.2, contains only 5 attacks

The CERT dataset (r4.2) contains behavior data for 1000 users over a period of a year and a half. We separated data into parts containing actions of single users. Each part consisted of samples representing a 24-hour day of a user's actions. We refer to samples and days of behavior interchangeably. A sample is a sequence of actions, sorted by the timestamps in the CERT dataset. We enumerated and encoded actions as numbers in a sequence. The sequence was changed into a fixed length feature vector containing counts of every action performed, for models requiring other types of input. The possible actions were: logging in, logging off, connecting a removable drive, disconnecting a removable drive, email events, file manipulation events, http access events (including browsing, down- loading and uploading) and a parsing error event accounting for any unknown actions or dirty data.

Table 4.1: Format of the logon.csv file

id	datetime	user	pc	activity
{X1D9-S0ES98JV-5357PWMI}	01/02/2010 06:49:00	NGF0157	PC-6056	Logon
{G2B3-L6EJ61GT-2222RKSO}	01/02/2010 06:50:00	LRR0148	PC-4275	Logon
{U6Q3-U0WE70UA-3770UREL}	01/02/2010 06:53:04	LRR0148	PC-4124	Logon

Source: [1] <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099>

4.2 Result and Analysis

According to the decision function in the `sklearn.ensemble.IsolationForest` library the more abnormal the data is, the lower the anomaly score it will have.

The figure below shows the plot of anomaly score and users for the evaluation of logon parameters.

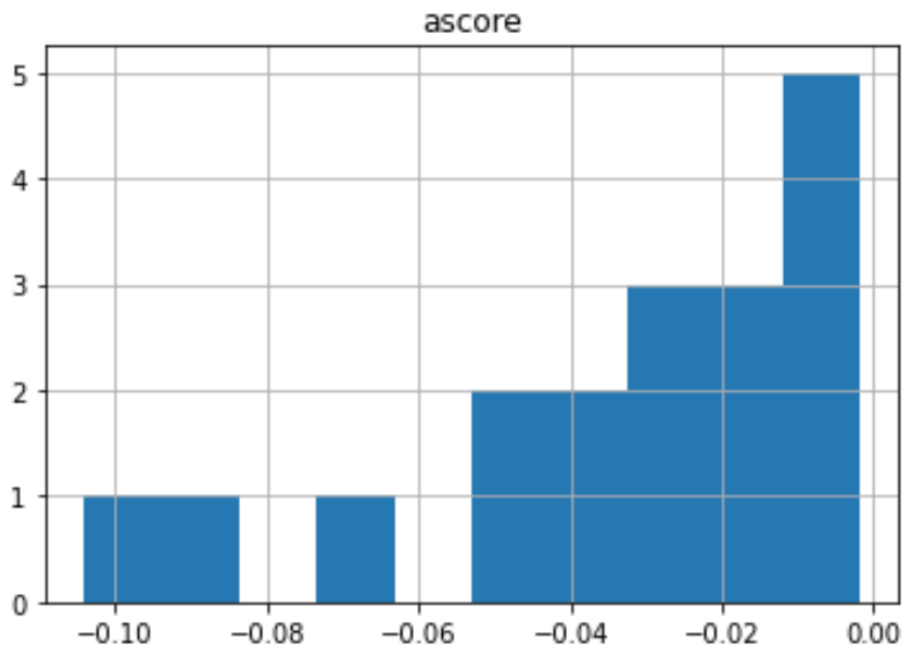


Figure 4.2.1 Plot of anomaly score using isolation forest for logon parameters

The figure below shows the histogram plot of the anomaly score using isolation forest for the parameters of the `psychometric.csv` file. It can be seen that about 20 users have the anomaly score of -0.1 which shows that these users are risky because the anomaly score is less than 0.

```
df_psychometric_result.loc[df_psychometric_result['ascore'] < 0].hist()
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8e9c2d3c10>]],
      dtype=object)
```

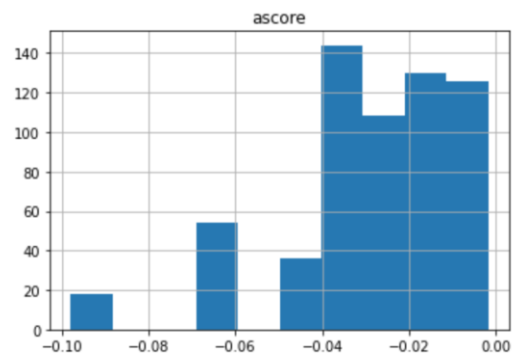


Figure 4.2.1 Plot of anomaly score using isolation forest for psychometric parameters

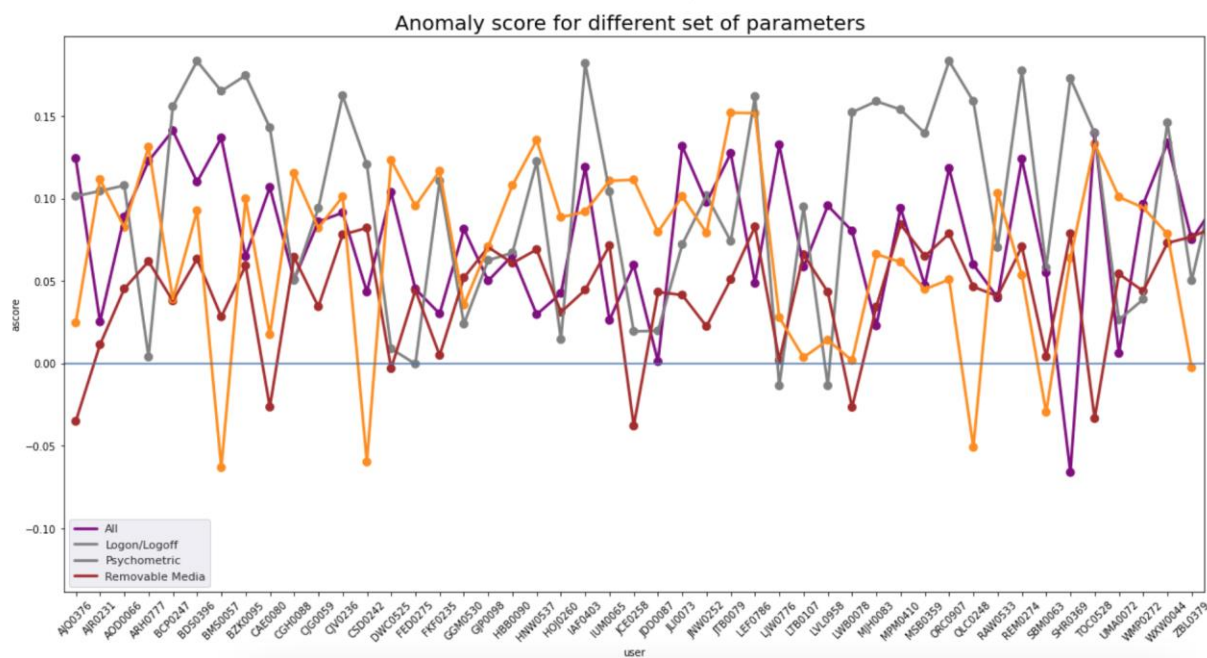


Figure 4.2.2: Plot of anomaly score using isolation forest for different set of parameters

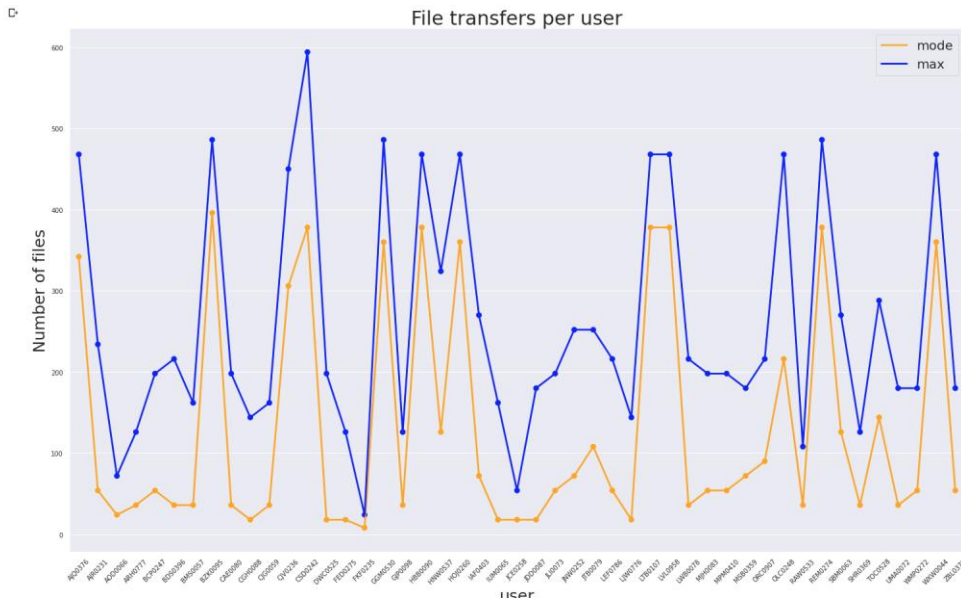
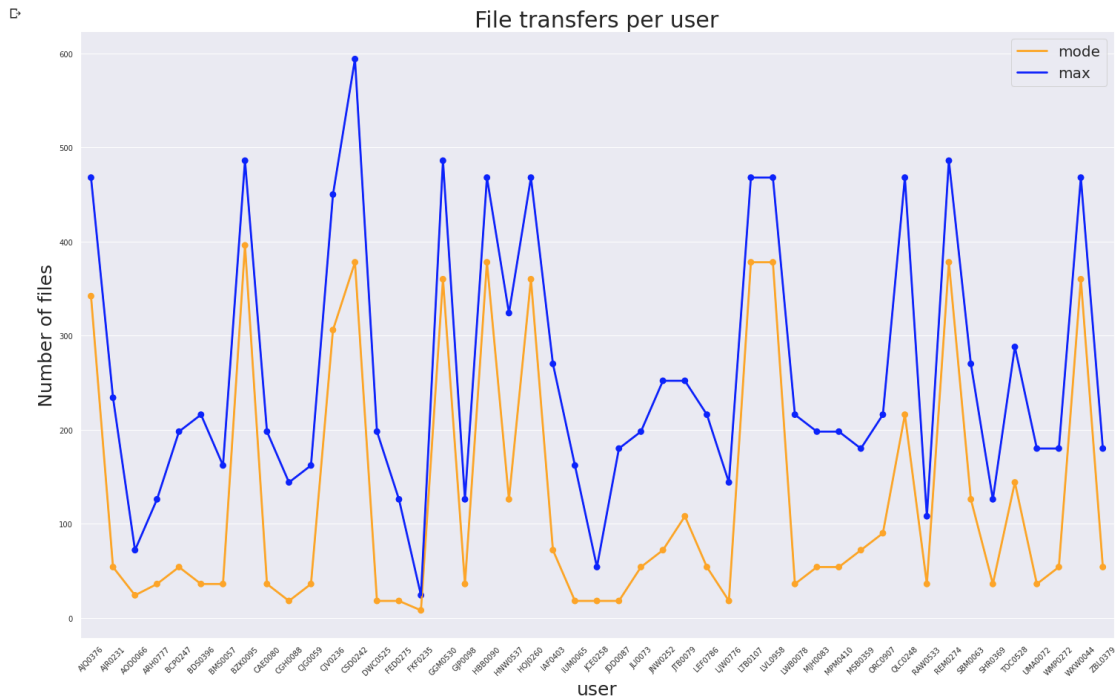
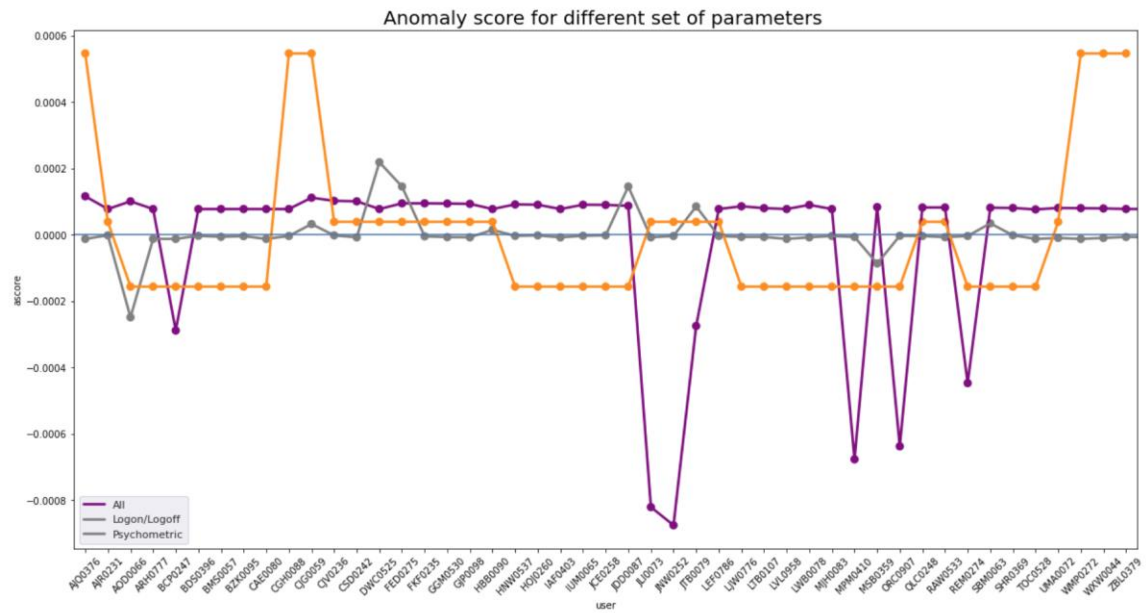


Figure 4.2.2: Plot of file transfer per user using isolation forest

The above figure shows the max point plot of number of files transferred versus user. It shows the normal value of file transfer per user. For example, the user CJV0236 transfers files in the range of 390-400 and the maximum file transferred by the same user is around 600. It shows that the normal range of file transferred for that user is in the range of 400 but the maximum file transferred is about 600, which proves that each user is risky as his behavior is different from the normal behavior.



4.3 Conclusion and Future Enhancement

The study identified that algorithm Isolation Forest performed better than One-Class SVM for anomaly detection. As we have referenced the CERT dataset, the pipeline for development for other datasets might be different. Additionally, log files having hidden patterns must be parsed in order to get the processable datasets. Some of the extensions that we can have in near future are:

1. Integrated log parser to parse the log files with hidden patterns so as to generate the processable datasets.
2. Better deployment of the system in dedicated deployment platform
3. Specialized UI/UX for different functional organizations.

REFERENCES

- [1] Z. (2017) Asadullah, M. Niaz and Wahhaj, “Kent Academic Repository,” *Eur. J. Soc. Psychol.*, vol. 40, no. 2, pp. 366–374, 2017.
- [2] T. O. Oladimeji, C. K. Ayo, and S. E. Adewumi, “Review on Insider Threat Detection Techniques,” *J. Phys. Conf. Ser.*, vol. 1299, no. 1, 2019, doi: 10.1088/1742-6596/1299/1/012046.
- [3] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang, “Applied Sciences Insider Threat Detection Based on User Behavior Modeling and Anomaly Detection Algorithms,” 2019.
- [4] M. N. Al-mhiqani, R. Ahmad, Z. Z. Abidin, and W. Yassin, “applied sciences A Review of Insider Threat Detection : Classification , Machine Learning Techniques , Datasets , Open Challenges , and Recommendations”.
- [5] S. Khaliq, Z. U. Abideen Tariq, and A. Masood, “Role of User and Entity Behavior Analytics in Detecting Insider Attacks,” *1st Annu. Int. Conf. Cyber Warfare. Secur. ICCWS 2020 - Proc.*, 2020, doi: 10.1109/ICCWS48432.2020.9292394.
- [6] V. Koutsouvelis, S. Shiales, B. Ghita, and G. Bendiab, “Detection of insider threats using artificial intelligence and visualization,” *Proc. 2020 IEEE Conf. Netw. Softwarization Bridg. Gap Between AI Netw. Softwarization, NetSoft 2020*, pp. 437–443, 2020, doi: 10.1109/NetSoft48620.2020.9165337.
- [7] S. Nepal and B. Joshi, “User Behavior Analytics for Insider Threat Detection using Deep Learning,” vol. 8914, pp. 232–238, 2021.
- [8] A. J. Hall, N. Pitropakis, W. J. Buchanan, and N. Moradpoor, “Predicting Malicious Insider Threat Scenarios Using Organizational Data and a Heterogeneous Stack-Classifer,” *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 5034–5039, 2019, doi: 10.1109/BigData.2018.8621922.
- [9] M. A. Haq, M. A. R. Khan, and M. Alshehri, “Insider Threat Detection Based on NLP Word Embedding and Machine Learning,” *Intell. Autom. Soft Comput.*, vol. 33, no. 1, pp. 619–635, 2022, doi: 10.32604/iasc.2022.021430.
- [10] N. M. Sheykhkanloo and A. Hall, “Insider threat detection using supervised machine learning algorithms on an extremely imbalanced dataset,” *Int. J. Cyber Warf. Terror.*, vol. 10, no. 2, pp. 1–26, 2020, doi: 10.4018/IJCWT.2020040101.
- [11] I. A. Gheyas and A. E. Abdallah, “Detection and prediction of insider threats to cyber security: a systematic literature review and meta-analysis,” *Big Data Anal.*, vol. 1, no. 1, 2016, doi: 10.1186/s41044-016-0006-0.
- [12] S. Yuan and X. Wu, “Deep learning for insider threat detection: Review, challenges and opportunities,” *Comput. Secur.*, vol. 104, p. 102221, 2021, doi: 10.1016/j.cose.2021.102221.

[13] Abdulla Amin Aburomman, and Mamun Bin Ibne Reaz. “IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC).” *Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection*, 2016, pp. 636-640.

APPENDIX

```
df_user_log_result.head()  
df_user_log_result.loc[df_user_log_result['ascore'] < 0].hist()  
  
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8e9c3a4d90>]],  
      dtype=object)
```

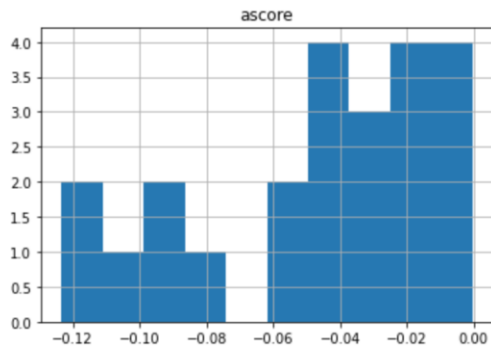


Figure: Plot of anomaly score using isolation forest for logon and user parameters

```
[166] df_device_file_full_result.loc[df_device_file_full_result['ascore'] < 0].hist()  
  
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8e9c40fbd0>]],  
      dtype=object)
```

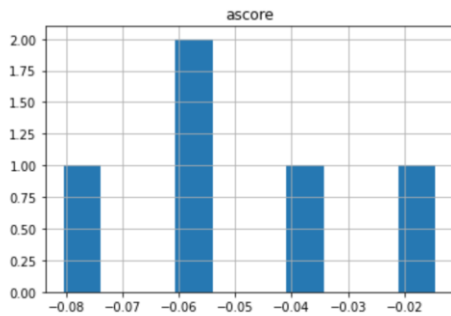


Figure: Plot of anomaly score using isolation forest for device and file parameters

```
df_psychometric_result.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8e9c5fdbd0>]],  
      dtype=object)
```

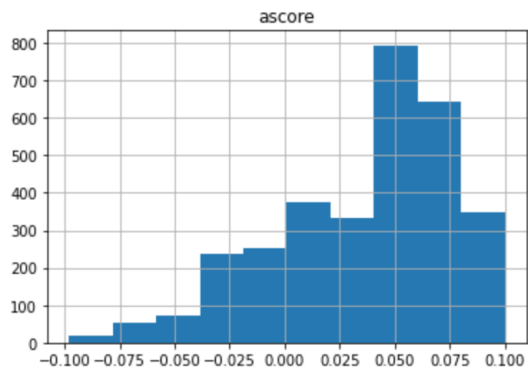


Figure: Plot of anomaly score using isolation forest for psychometric parameters

```
df_device_file_full_result.loc[df_device_file_full_result['ascore'] < 0].hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8e9c40fbd0>]],  
      dtype=object)
```

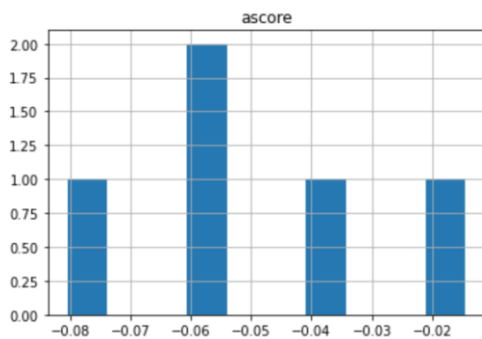


Figure: Plot of anomaly score using isolation forest for device and file parameters

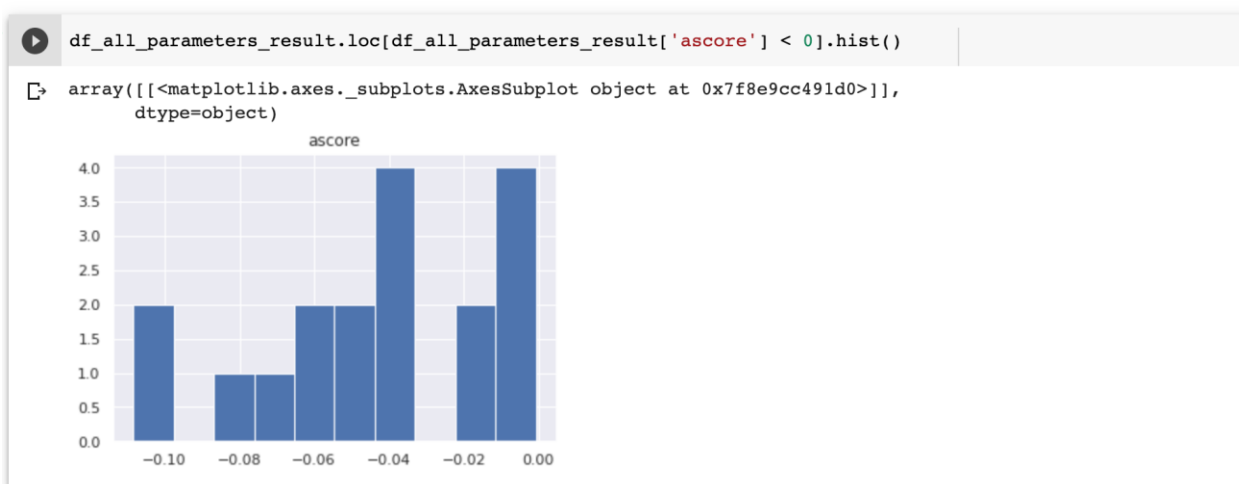


Figure: Plot of anomaly score using isolation forest for all combined parameters

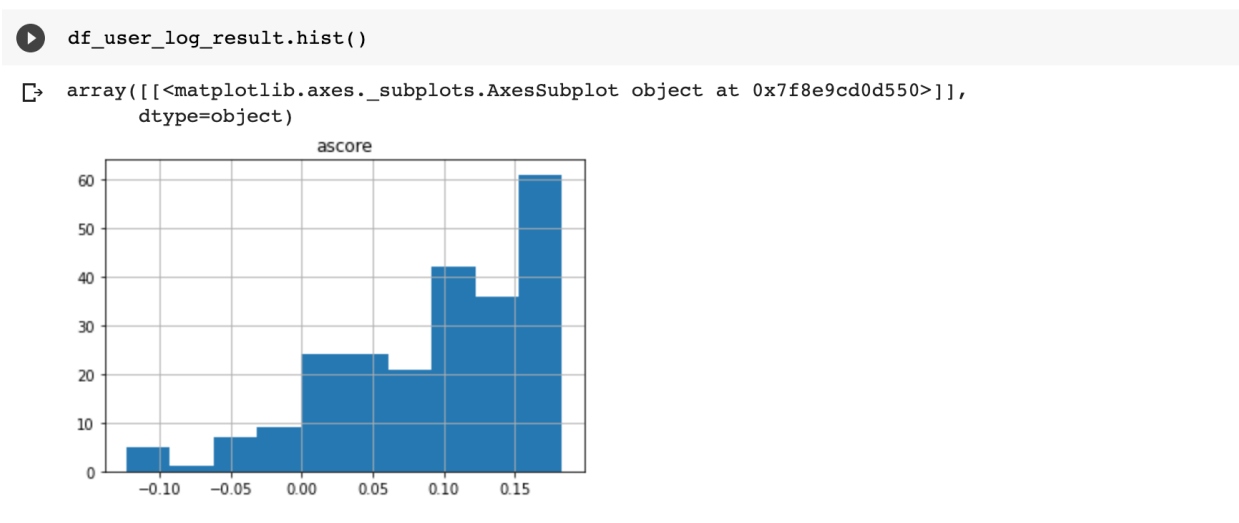


Figure: Plot of anomaly score using isolation forest for user and log parameters