Deep Learning for Detecting Adversarial Drift in Insider Threat Detection

Name of the student: Abdulqodir Usmanali

Supervisor: Rogerio de Lemos

Abstract

In the modern digital landscape, the persistent evolution of approaches by insider threats to workplaces necessitates a proactive response to the dynamic threat landscape. Training robust models to counteract these threats is a valid approach to detect these insider threats. The primary objective of this project is to elucidate the presence of adversarial drift within a synthetic insider threat dataset. This task is approached by us through initially training a model on an early version of the dataset and then seeing how that model fares against a later version and then evaluating the success rate and seeing if drift is present. Another experiment is done after through personally creating multiple variations of an earlier insider threat then training a model on that and seeing how those fares against a later version of the insider threat dataset. Given these two large experiments, it should generate the results needed for me to see if drift even is present and how the drift evolves over time as the dataset is generated.

1. Introduction

Insider Threat Test Dataset is a synthetic dataset concerning insider threats as given from the title. Insider threats are users or individuals within an organisation that have private privileged information access to the organisation and pose great cybersecurity risk. Insider threats don't have to be malicious themselves and can inadvertently pose a threat through lack of awareness of security policies in place or unintentional errors. Insider threats are vastly investigated in the world of cyber security and their methods are constantly evolving. Change in pattern of the insider threats can be considered as adversarial drift. Adversarial drift refers to the phenomena where the distribution of data shifts over time or other conditions which affect machine learning models being built to detect them. The shift can occur due to many various changes such as user behaviour, introduction of new security flaw in a system and many others. In context of machine learning, models built to detect previous patterns of malicious actors fail to stand against future patterns. The aim of this research is to explore how well malicious actors change their patterns in a simulated environment. Elucidation of adversarial drift is important within a synthetic dataset since simulating these patterns will end in a more accurate model to evaluate against for various research projects and business proposals since it will emulate a real-world scenario far closer.

Initially in this report, we contextualise the foundational aspects of the research through background research. This research is done by looking at the dataset underpinning our investigation, the idea of adversarial drift, how to handle the large dataset we are working on and related work on the dataset and how we can use that within our research.
Then we develop a fundamental methodology behind our work and exploring the decision-making process to get it by looking into the dataset. This section provides an insight into how we structured the experiments and conducted analysis on the dataset.
Following the establishment of our methodology, we present the findings from three iterations of experiments conducted throughout the research process. While not all outcomes may have been directly impactful, each iteration contributes valuable insights toward addressing the overarching research question. The evaluation of these results serves to provide a comprehensive understanding of the research outcomes.
Finally, we conclude the report by offering reflections on the project as a whole and outlining potential avenues for future research. This concluding section provides a synthesis of the findings, highlights key takeaways, and suggests directions for further inquiry.

2. Background Research

In this research, two primary components require thorough examination: the Dataset itself and the concept of adversarial drift. These areas serve as the cornerstone of our investigation, offering essential insights into the data landscape and the dynamic nature of threat behaviours.

Understanding the Dataset is crucial for laying the groundwork for our experiments and contextualizing the findings within the broader research framework. Similarly, delving into adversarial drift is essential for comprehending the evolving tactics employed by malicious actors and their implications for cybersecurity. By delving deep into these areas, we aim to garner a comprehensive understanding of the underlying dynamics driving insider threat detection and response strategies.

Due to the datasets being extremely large looking into strategies to deal with them is also a good idea since more standard methods won't work unless you load all the data into the memory.

In addition, we explore previous work to identify approaches that can be more fruitful, as well as to gain a more complete understanding of the dataset and strategies used previously to detect insider threats. By leveraging insights from past research efforts, we aim to build upon existing knowledge and enhance the effectiveness of our approach in addressing the challenges posed by insider threats.

2.1. Insider Threat Test Dataset.

The Insider Threat Test Dataset is a synthetic dataset created by "CERT Division in partnership with ExactData, LLC, and under sponsorship from DARPA I2O, generated a collection of synthetic insider threat test datasets" (Lindauer, 2020). These datasets provide both synthetic background data and data from synthetic malicious actors" (Lindauer, 2020). There are currently 6 versions of the Insider Threat Test Dataset, and each later version evolves and improves the features and the logs introduced previously. As the version number increases, the users also generally increase and the threat scenarios as well. Some version increases also have small incremental changes that involve just lots more scenarios. Versions 4.2 and 5.2 have the greatest number of threats (of 30 and 29 respectively for scenario 1) which lets train models based on those versions.

2.2. Adversarial Drift:

Dataset shift refers to the changes in input data distribution over time or across different environments. This can significantly impact the performance of machine learning models as described in the paper about pitfalls and challenges within application of machine learning on malware detection as touched upon on a paper about the discrepancy between the high performance of machine learning (ML) models for malware detection reported in academic settings and their often less effective real-world application (Cavallaro, et al., 2023)~~-~~. This idea can be extrapolated to this context well due to malware detection being like insider threat detection, so it is equally important in this context which validates the problem we are looking at. Understanding the nature and types of dataset shift is crucial for developing robust models capable of adapting to evolving data landscapes. So, in this context after identifying the shift if there is one, we need to list all the variables that can be the reason behind affecting it.

2.3. Working With Large Datasets

Large datasets are impossible to be dealt by loading it all into memory, so we need datamining tools/libraries to allow us to even start analysing and looking at the dataset. Between the choices of a python library or a simple datamining tool such as Orange, a python library seems like a more appropriate choice here due to versatility of writing code. There are many python libraries for datamining and between researching them all the two that stuck out the most for me were Pyspark and Polars. Pyspark is a very well-integrated data processing tool that has the capability to perform distributed programming which is a process that allows the program to run in a system of nodes that are doing work in parallel. Polars is an upgrade on the current basic beginner standard of data processing of pandas which can't perform distributed programming but can run instructions very fast due to its lazy evaluation nature (only evaluating what's needed when needed) and being built with Rust which is a low-level programming language. Although Pyspark can compute billions of logs effectively compared to polars, polars is much faster in millions of logs which is the number of logs we will be working with according to multiple benchmarks that were

found online (Beach, 2023) (Kv, 2023). So, using polars in my own personal implementations should be imperative wherever possible in this research.
Due to using other people's scripts as well as mine, pandas will also need to used since polars is a later iteration on pandas' principles.

2.4. Related Work

There have been a lot of implementations of supervised and unsupervised machine learning techniques for threat detection in the Insider Threat Test Dataset. It was decided to focus on ones that perform their experiments using libraries and tools rather than ones that develop bespoke software. With that in mind, 2 pieces of work discussing these ideas prevalently have been found showing how limited the research around this topic therefore leading us to design more tests around this dataset to understand it further. One a journal article detailing under sampling techniques and its effects on how the threats were detected using basic ML models (Al-Shehari, et al., 2022) and another one exploring data granularity and feature extraction.
In the journal (Al-Shehari, et al., 2022), authors describe how they carried out experiments in sci-kit learn toolkit and how they were managing to get results through a basic extracted feature set and experimenting with resampling techniques and a mixture of both under sampling and oversampling the data since the Dataset is heavily imbalanced. This method should be explored later as it can potentially aid with our aims of creating the model to test for.
In the paper (D. C. Le, et al., 2020), they explore a user-centered approach and apply various machine learning classifiers as well as an auto encoder into the issue. Authors are mainly investigating the data granularity and how there is a trade-off for how granular the data can be in reference to the user as in how many logs are used to synthesize their type of log that is based on the user. This paper also employs a more realistic conditions through not random sampling thus this can be used to verify the results we are getting as well as research. This paper should also be explored more since it has tools and scripts attached to the paper allowing me to use their scripts to build my own models to test for drift.

Also looking into how supervised machine learning was used to solve this problem on a more technical level would be useful so resulted in investigating this paper (Bin

Sarhan, B. and Altwaijry, N., 2023). This paper investigates how machine learning techniques are developed to test against the dataset which was helpful to inspire the approach done in this paper. They explore mainly 4 machine learning approaches which are SVM, Random Forest (RF), Neural Network (NN), and AdaBoost along with a k-fold cross-validation. They also touched upon the SMOTE resampling techniques used which is an algorithm developed to make underrepresented classes more prevalent within a dataset.

3. Dataset

In this section we will be exploring the dataset more in detail and giving a general methodology

3.1 Structure of the datasets

The datasets are comprised of 6 files, 5 of which acting as logs along with a folder of LDAP which has all the employee information such as their email and what department they belong to along with other information. There is also a file called psychometric which shows a sample of psychometric evaluation done on the synthetic users although this isn't relevant.

The table below shows how 4.2 is laid out which is what we focus on the most as we train the data on this. Even though the features evolve, all the features found in 4.2 are also in later datasets meaning we can use the same ones to test the later datasets.

Log type	Log content	Description
Devices.csv	ID, date, user, PC, activity (connect/disconnect)	Relates to all physical devices being connected and disconnected.
File.csv	ID, date, user, PC, filename, content	Relates to files being accessed by the user.
Email.csv	ID, date, user, PC, to, cc, bcc, form, size, attach-ment count, content	Emails being sent and who is attached to it and bag of words content so a group of words that represent the content of the email, cc and bcc.

Logon.csv	ID, date, user, PC, activity	Logs of users logging in and out of the system
http.csv	ID, date, user, PC, URL, content	logs of what websites the users are accessing along with the kind of content they view based on a group of word representing it.

Figure 1. Description of the logs found in each dataset.

Scenarios (since when they were available in the dataset)	Description
Malicious Leak (found from version 2 onwards)	User uploads sensitive data to WikiLeaks and exits the company.
Insider Theft Prior to Job Switch (found from version 3 onwards)	Employee steals data using a thumb drive before joining a competitor.
Retaliatory Cyberattack by Disgruntled Admin (found from version 4 onwards)	System administrator installs a keylogger to sabotage supervisor through a panic-inducing email.
Persistent Unauthorized Access (found from version 5 onwards)	User repeatedly emails confidential files to personal account over three months.
Post-Layoff Attack (found from version 6 onwards)	A member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain

Figure 2. Description of the scenarios observed in the datasets.

3.2 Logs.

The logs seem mainly useful and will need to be encoded here’s a table demonstrating how these will be encoded for the experiments.

Date	For date we create a feature based on the hour that the activity was done in.

user	Users were all given unique IDs from 0001 to 1000
PC	For PC each pc already had a 4-digit ID so just cut out the PC part and left with the 4 digit ID
Connect/Disconnect	Binary – 0/1 as it can only be in two states.
Logon/Logoff	Binary – 0/1 as it can only be in two states.
HTTP links	Gave all the http links an ID to see which website they are visiting.
Empty cells	The empty cells have been encoded as -1 to tell the ML model the cell is empty since none of the other numbers are negative
Files	Files have been divided into two parts with the extension having its own column and the file itself having a unique id attached to it since the file names are randomised
Email	Blew up the emails into its own columns for cc, bcc and all of them to process it better then assign each email a number to signify which user it was doing it.
Content	Everything related to bag of words was removed since it should be possible to detect patterns without it and it takes up too much memory and strategies relating to bag of words handling are complex
Malicious User	Binary – 0/1 as it can only be in two states

Figure 3. Ideas of encoding features

3.3 General Methodology for testing for Adversarial Drift

Given the structure of the logs and dataset, we can test for drift. Main idea is to train a machine learning model on dataset version 4.2 exclusively regarding the threat actors in the first scenario and test it against 6.1 and 5.2.

The rationale for developing a model based on Dataset Version 4.2 is twofold: firstly, it leverages the comprehensive data volume provided by this iteration; and secondly, it addresses the issue of class imbalance, largely due to the significant representation of threat actors in this version. The purpose of this model is to undergo a thorough evaluation against the latest version of the dataset, Dataset Version 6, with the aim of identifying and examining potential shifts in adversarial tactics.

Initially, we employ Dataset Version 4.2 to train a model, effectively capturing the behaviour patterns of threat actors within that specific period. This model is designed to encapsulate the intricacies and distinct characteristics of attacker behaviour as observed in Dataset Version 4.2. Dataset Version 4.2 was used since it has the earliest case of most abundant number of insiders and it will offset the already heavily imbalanced dataset slightly. Earlier versions cannot be used since they only have 1 instance of a threat actor per scenario along with around 25 datapoints regarding each threat actor.

Training the model, it was decided to only go with class weights as a form of balance technique since other forms of balance seemed more intrusive and unnecessary given the nature of the data. Given the data, the models were trained in two ways, one where all the log indexes were completely randomised and another where the data was treated as sequential. These were important to do as they both can highlight and reflect different parts of the dataset. One being an ideal ML environment where the threat actions are found sporadically in the dataset without having much structure, but same users can be inside the test dataset and the training dataset. Another one where the data is treated like a stream so based on only a part of the stream, the decision tree builds its model making it easier to detect other threat actors since it’s less prone to overfitting and more realistic.

In summary, our methodology involves two critical steps:

1. Fusion and encoding of large dataset.
2. Model Generation/Evaluation

3.4 Machine learning models to use.

Others will be considered and attempted but these are the ones that are preferred to use due to previous works using them.

SVM

SVMs (Support Vector Machines) are known for their effectiveness in separating data points into two classes by finding an optimal hyperplane that maximizes the margin between the two classes in the feature space so in context of this paper it can be malicious and benign users.

Decision Tree

In a Decision Tree Classifier, the algorithm learns a series of if-else decision rules based on the input features to classify data points into different categories or classes. These classifiers work by recursively splitting the dataset into subsets based on the values of specific features, ultimately assigning a class label to each leaf node.

3.5 Evaluation metrics to use.

Since we will be experimenting with the datasets, we need to establish a few evaluation metrics to compare this research to previous as well as to establish links and create a thorough evaluation of the results. This project mainly needs the measures to compare how the model has performed against others therefore only precision, recall and f1-score are needed.

Precision- Calculated as ratio of positive predictions to the total number of positive predictions made. Measures accuracy of the models.

Recall- Calculated as ratio of true positive predictions to the total number of positive instances. Measures proportion of actual positive results.

F1-score – average of recall and precision. Measures the performance of the model.

We can also support the general effectiveness through a confusion matrix where we can see how many correctly and incorrectly identified insiders are found since a confusion matrix shows how many true positives, false positives, false negatives, and true negatives are predicted. These will be used more in the Corpus rather than here since general performance of the model is enough.

4. Feature removal

Since the dataset is quite big, cutting of sections of the dataset is needed down so it can run it faster as well as store less data.

We decided to perform an experiment where an entire version of the dataset is encoded, stack all the features in columns then see if they had any impact if removed any and how many can be removed before it will have adverse effects on the models produced. Findings of this information how it can help us draft a general methodology and what can we do further.

Methodology for this experiment

Fusion and encoding of large dataset.

The dataset 4.2 was used. The rest of the columns were all merged into a single file through a python script using polars library and using mainly lazy evaluation which means the information from the files is grabbed in an on-demand basis and not much is loaded into the memory.

Model Generation/Evaluation

After being merged, the data was used to train a machine learning model with the insiders being predicted and the model was based on various classifiers at first. Finally, the model is tested on the other half of the dataset and see the results and repeat it with other features not included.

Problems Encountered

Couldn’t use Random Forest or SVC classification since they were taking far longer. This is due to the large size of the dataset and can be waited out, but it has not done anything.

Results

The results of this experiment show that we could remove all the file and email logs alongside the features introduced in them to create a smaller dataset. This shows how the attack only relies on the other 3 files and using them only would be better as well as faster. The experiment also shows how RandomForestClassifiers and SVMs shouldn't be used as they couldn't give any meaningful results after even adjusting the dataset using a standard scaler and they would take way too long to run.

5 – Initial idea for testing for adversarial drift.

The initial idea was to use the large dataset given and see how it performs given the methodology described before.

Methodology

The methodology was just as described before with the justifications, following is the method applied:

1. merging the dataset using polars lazyframe evaluation
2. cutting out ID, Date, and some other useless features
3. create a decision tree model and some other models.
4. test the models against later versions.

Problems encountered.

Datasets 5,2 and 6.1 both couldn't be fully loaded to test.

For dataset 5.2, the dataset was divided into sections of 10 million with there being 7 total since there are 61 million logs and then test each chunk separately and congregate the results into a single table.

6.1 was extremely big (more than 180 million logs) so only a chunk of it was used rather than the whole dataset and it would still accurately reflect the performance of the model since there are still over 15 million non-malicious users.

Results

The result of SVM was inconclusive since it was taking 6hrs+ to process the data we couldn't let the SVM finish even when given days this is due to the scale of the dataset.

The result of Decision Trees was interesting (found under Drift1 in the Appendices). Within 4.2 it could detect 71% of all the threat actors within an idealised set but within a more realistic setting it could only detect 2% of the threat actors. Then extracting the dataset to 6.1, it could not detect any from either model. This can be due to lots of factors like overfitting and lack of features.

Given the results of this experiment we need to narrow down what's wrong with it. This can be done through feature engineering (and better feature selection) as well as test if it was because of overfitting by comparing it to another project using the same dataset that uses balancing techniques.

6. Compare to another methodology used

After the results of the first experiment, it was decided to compare a methodology done by someone else (Al-Shehari, et al., 2022) and see how that holds up with the code produced since it could be personal incompetence.

Methodology

Fusion and encoding of large dataset.

Fusion method still holds the same as previous attempts with slight changes. Http, device, and logon all collected into one row called activity and there is a new feature called epoch_timestamp which is just a timestamp of each activity.

Model Generation/Evaluation

This section has stayed the same although SVM and Random Forrest were no longer used due to them actively taking lots of time and producing poor results.

Problems encountered.

Due to this approach being largely like the last one no new problem have been experienced and results were generated as expected.

Results

The results were largely like the last one. Now we can clearly tell that the main issue is the model developed is overfit due to decision trees being prone to overfitting and how the model developed was largely like the last one. The second approach of testing might not be overfit, but it also produces poor results meaning large-scale feature manipulation is required to make use of this data.

Stage 3 – Feature engineering and manipulation

For Stage 3 the methodology is completely different than the previous iterations using a feature extraction tool available for this dataset. This feature extraction and engineering tool creates 43 different features for the dataset 4.2 and records logs in a different way which is through daily/weekly/other logs of data performed by 1 user.

Methodology

Fusion and encoding of large dataset.

A script that congregates the data into granular user logs was used. It was decided to use daily logs since they were the fastest ones to generate that still gave good results based on the attached paper. These daily logs encoded the model in a far more unique way. These involve features like the duration of the USB being plugged into devices as well as Booleans that see if the activity was after work and other conditions.

Alongside the dozens of features introduced which can be seen in the corpus. The dataset also encodes http more effectively by identifying the type of website visited and uses LDAP more effectively for its features.

The script can be found in the corpus alongside my attempt at explaining it as well as the analysis of the research paper.

Model Generation/Evaluation

New models have been considered as the tool comes along with it.

Deep Learning - Multi Layer Perceptron:

We use deep learning to build models for anomaly detection in a massive dataset containing over a million data points, with just a few hundred insider threats. In this example the feature extraction tool comes with a sample multi-layer perceptron in the form of an auto encoder using the same features as the ML but takes 200 users as input and tests them against others. The neural network has 3 hidden layers to detect more non-linear properties found in the dataset.

Random Forest:
Ensemble classifier which constructs multiple decision trees to reach predictions.

Alongside these models we can use the previous SVMs and Decision Trees as well.

Problems encountered.

The first problem was that program couldn't even run it and that was due to the program being developed for Linux meaning I had to install a new operating system to use the tool. Also, a partition of exFAT type encoding had to be created so I could access the data from linux as windows has an encoding that's not compatible with Linux.

The second problem was the models took days to compile which although it was manageable but had to wait it all out which took days to complete.

After that, it was realised that later models were leaking memory no matter what we did with the tool so then had to reverse engineer the tool and change the tool, so it only took the first scenario along with only using the files http, devices and logon which fixed the issue although took ages.

Final problem was that the datasets 5.2 and 6.1 were too big to fit into memory so had to edit the code to include a chunking method in python using pandas since all the code was optimised for it and made it comparable to my own results.

Results

The results were good and although the decision tree overfit the model for 6.1 , the other models fared better and displayed that there was not much drift occurring as the percentages stayed similar although a dip was expected in 6.1 as it had only 3 entries to detect due to the logs being in a daily format per user. The results are very conclusive and show this dataset does not have drift since the performance barely changed.

Conclusions and Future Work

Findings

This paper has investigated the presence of adversarial drift of the CERT Insider Threat Test Dataset which was done through 3 iterations. This paper also looked at how removing certain log files gave a boost to the classification of insider threats due to the redundant nature of some of the logs. First two being not fruitful although useful to understand the dataset. Last part confirmed the inexistant nature of adversarial drift in this dataset.

Reflection

The project worked well and achieved the ideas set out but getting there was very challenging therefore unsatisfied with the results. As I got deeper into the project further questions have been appearing with less time to answer them. There are lots of avenues to further cement the idea that adversarial drift does not exist in this dataset.

The idea of my project is novel as it hasn't been attempted before since there aren't any adversarial drift papers on this dataset. The results obtained on detecting insider threats are on par with similar research on ML models on this dataset.
The guidance I would give to other people about this dataset would be that feature engineering is extremely important as the dataset is barebones and with the features extracted it can achieve far more. Another very important and challenging process

was processing and manipulating the dataset, as time evolves the tools needed will evolve as faster more powerful tools will be created. This was felt quite a bit in this research as at the beginning I realised the Orange datamining tool was currently being updated with out of memory tools that are currently in the beta.

Future Work

For future work, a good extension of this project would be to create a generative model that creates threat actors inside of an earlier dataset based on the patterns picked up on the dataset; then training a classification model on it and comparing it to the later datasets.

Another topic that would be interesting to further research would be exploring temporal data and its effects on drift. This can be important as you can create a sin/cos function that deviates from 0 and 1 to reflect the cyclability of time. This can be applied to the date in many ways like the hour of the day making 11pm close to 1am and other features like the week setting where Monday is closer to Friday.

If more time could be had, looking into other scenarios and how they can be used to create models to detect adversarial drift would be a logical continuation.

Acknowledgements:

Rogerio De Lemos

Data granularity Paper authors whose scripts helped tremendously.

Bibliography

Beach, D., 2023. Data Engineering Central. [online] Available at: https://dataengineeringcentral.substack.com/p/spark-vs-polars-real-life-test-case [Accessed 2 February 2024].

Cavallaro, L., Kinder, J., Pierazzi, F. and Pendlebury, F., 2023. Are Machine Learning Models for Malware Detection Ready for Prime Time?. IEEE Security & Privacy, 21(2), pp.53-56.

Kv, V., 2023. Medium. [online] Available at: https://blog.det.life/pyspark-or-polars-what-should-you-use-breakdown-of-similarities-and-differences-b261a825b9d6 [Accessed 2 February 2024].

Lindauer, B., 2020. Insider Threat Test Dataset. [online] Available at: https://insights.sei.cmu.edu/library/insider-threat-test-dataset/ [Accessed 1 February 2024].

Al-Shehari, T. and Alsowail, R.A., 2023. Random resampling algorithms for addressing the imbalanced dataset classes in insider threat detection. International Journal of Information Security, 22, pp.611–629.

Le, D.C., Zincir-Heywood, N. and Heywood, M.I., 2020. Analyzing Data Granularity Levels for Insider Threat Detection Using Machine Learning. IEEE Transactions on Network and Service Management, 17(1), pp.30-44.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, pp.321–357. Available at: http://dx.doi.org/10.1613/jair.953 [Accessed March 2024].

Bin Sarhan, B. and Altwaijry, N., 2023. Insider Threat Detection Using Machine Learning Approach. Applied Sciences, 13(1), p.259. Available at: https://doi.org/10.3390/app13010259 [Accessed March 2024].

Appendices

The test data can be found in the corpus in a form of screenshots or sample of files due to the size of the datasets generated.

Drift 1:

linear approach

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00  14847096
           1       0.12      0.01      0.02       235

    accuracy                           1.00  14847331
   macro avg       0.56      0.50      0.51  14847331
weighted avg       1.00      1.00      1.00  14847331
```

random sample:

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8908289
           1       0.71      0.59      0.64       110

    accuracy                           1.00   8908399
   macro avg       0.85      0.80      0.82   8908399
weighted avg       1.00      1.00      1.00   8908399
```

database 5.2

linear approach:

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00  16384876
           1       0.06      0.01      0.01       235

    accuracy                           1.00  16385111
   macro avg       0.53      0.50      0.51  16385111
weighted avg       1.00      1.00      1.00  16385111
```

sampled approach:

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00  16384876
           1       0.00      0.00      0.00       235

    accuracy                           1.00  16385111
   macro avg       0.50      0.50      0.50  16385111
weighted avg       1.00      1.00      1.00  16385111
```

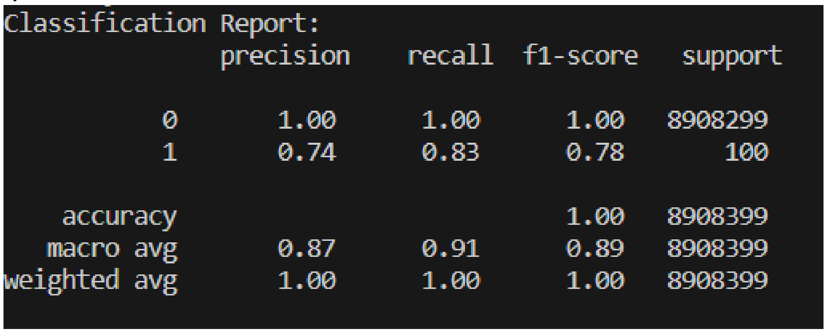
dataset 6.1:

```
           0       1.00      1.00      1.00  156414421
           1       0.00      0.00      0.00         17

    accuracy                           1.00  156414421
   macro avg       0.50      0.51      0.50  156414421
weighted avg       1.00      1.00      1.00  156414421
```

Drift2:

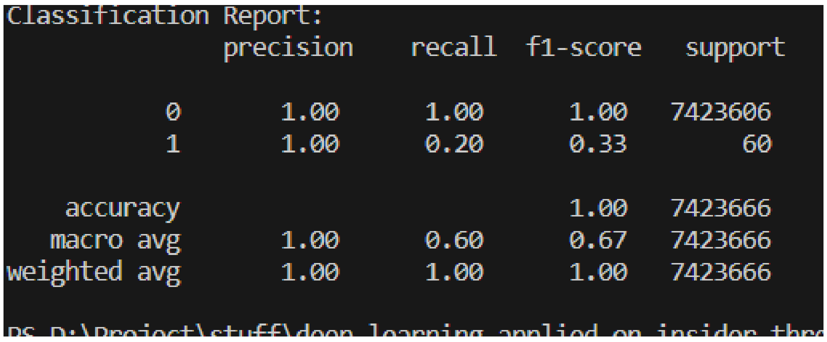
4.2 validation :



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   8908299
           1       0.74      0.83      0.78       100

    accuracy                           1.00   8908399
   macro avg       0.87      0.91      0.89   8908399
weighted avg       1.00      1.00      1.00   8908399
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   7423606
           1       1.00      0.20      0.33        60

    accuracy                           1.00   7423666
   macro avg       1.00      0.60      0.67   7423666
weighted avg       1.00      1.00      1.00   7423666

PS D:\Project\stuff\deep learning applied on insider thre
```

5.2 test : both results

```
              precision    recall  f1-score   support


           0       1.00      1.00      1.00  61607178
           1       0.00      0.02      0.00       325


    accuracy                           1.00  61607503
   macro avg       0.50      0.51      0.50  61607503
weighted avg       1.00      1.00      1.00  61607503
```

6.1 : both results

Classification Report:

```
•              precision    recall  f1-score   support
•
•           0       1.00      1.00      1.00  156414421
•           1       0.00      0.00      0.00         17
•
```

```
•      accuracy                    1.00  156414421
•     macro avg      0.50     0.50     0.50  156414421
•   weighted avg      1.00     1.00     1.00  156414421
•
```

Drift3:

3 part 1:

```
              precision    recall  f1-score   support


       False       1.00      1.00      1.00    154493
        True       1.00      0.80      0.89        56


    accuracy                           1.00    154549
   macro avg       1.00      0.90      0.95    154549
weighted avg       1.00      1.00      1.00    154549



              precision    recall  f1-score   support


       False       1.00      1.00      1.00    154493
        True       0.61      0.80      0.69        56


    accuracy                           1.00    154549
   macro avg       0.80      0.90      0.85    154549
weighted avg       1.00      1.00      1.00    154549
```

3 part 2:

svm -

```
  precision    recall  f1-score   support
```

```
           0       1.00      0.67      0.80    692558
           1       0.00      0.68      0.00        85

    accuracy                           0.67    692643
   macro avg       0.50      0.68      0.40    692643
weighted avg       1.00      0.67      0.80    692643
```

rf-

```
precision    recall  f1-score   support

           0       1.00      1.00      1.00    692558
           1       1.00      0.74      0.85        85

    accuracy                           1.00    692643
   macro avg       1.00      0.87      0.93    692643
weighted avg       1.00      1.00      1.00    692643
```

dt

```
precision    recall  f1-score   support

           0       1.00      1.00      1.00    692558
           1       0.64      0.74      0.68        85

    accuracy                           1.00    692643
   macro avg       0.82      0.87      0.84    692643
weighted avg       1.00      1.00      1.00    692643
```

auto encoder r5.2

```
UC score:  0.9085583673481016
Detection rate at different budgets:
0.1%, DR = 0.00%
```

```
1.0%, DR = 76.32%
5.0%, DR = 78.23%
10.0%, DR = 81.36%
20.0%, DR = 84.66%
```

R6.1

Random forest

```
precision    recall  f1-score   support


         0       1.00      1.00      1.00   1395874
         1       0.67      0.67      0.67   3


    accuracy                         1.00   1395877
   macro avg       0.84      0.84      0.84            1395877
weighted avg       1.00      1.00      1.00   1395877
```

Decision trees:

```
precision    recall  f1-score   support


         0       1.00      1.00      1.00   1395874
         1       0.00      0.00      0.00   3


    accuracy                         1.00   1395877
   macro avg       0.5      0.5      0.5       1395877
weighted avg       1.00      1.00      1.00   1395877
```

auto encoder:

```
UC score:  0.9085583673481016
Detection rate at different budgets:
0.1%, DR = 0.00%
```

1.0%, DR = 33.33%

5.0%, DR = 66.66%

10.0%, DR = 66.66%

20.0%, DR = 66.66%