LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

Detecting Security Vulnerabilities using Machine Learning Models

UE18CS390B – Capstone Project Phase – 2

Submitted by:

Rohit Vishwakarma	**PES1201800152**
Sreekanth R Gunishetty	**PES1201801467**
Shamanth R Rao	**PES1201801699**
Anil M S	**PES1201801866**

Under the guidance of

Prof. Guide Name
Designation
PES University

August - December 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

TABLE OF CONTENTS

Section 1	**Common for Prototype/Product Based and Research Projects**
Section 2 & 3	**Applicable for Prototype / Product Based Projects.**
Section 4	**Applicable for Research Projects.**
Appendix	**Provide details appropriately**

1. Introduction

This project is about implementing machine learning models in the field of cybersecurity. We are going to train the machine learning model on the user activities on the website. This tracks their previous activities and predicts their future activities and any deviation to this will trigger alarm. So a cyber analyst doesn't need to go through piles of logs and data everyday, but only a select few high priority ones. The machine learning model trains everyday and reports cases to cyber analysts who then reports if it is an actual case or not. Based on the outcome of this re-trains for the next day and it will be prepared for future attacks.

2. Design Constraints, Assumptions, and Dependencies

We assume that web owners are willing to take risks of sending their user activity data to us complying to laws of the country. We report whatever is given and all the activity of the model can be seen in logs.

We are training a machine learning model, so it needs good computing requirements. The model's accuracy depends on the amount of data web owners are willing to share. A good machine learning model is data hungry.

- **Interoperability requirements**: The requirement for this is that web servers should send the data from machine generated logs, json, csv or any machine generated, automated logs.
- **Interface/protocol requirements:** There are no requirements for interface or for the display of the output as the requirements for such lies in the hands of the web owners who take this service.
- **Data repository and distribution requirements:** The model takes and stores the data given by the web server. But the model resides in the server controlled by the website owners and is processed locally. So no distribution requirement.
- **Efficiency:** The efficiency of the model depends on the machine learning model implemented and the amount of data it gets. It also depends on the proficiency of the human analyst.
- **Performance:** The performance of the model is the time required for the model to train and report the anomalies. The model should be able to report within minutes of seeing the attack in the dataset of millions of logs. The train requirement is, the model needs to be trained at the end of the day for the next day, so, less requirement

- **End-user requirement:** The end user here is not the clients of the web server. Instead are the web owners themselves. So there are no requirements for the end user here.
- **Availability of Resources:** The model can be trained in a separate powerful machine. It requires good computing capacity.
- **Hardware or software environment:** Hardware requirement is as mentioned above. Software requirement is any linux based system and access to the internet.
- **Deployment:** Deployment might require intervention of humans and maynot be just installer. There needs to check for the type and format of the data. The model might require change in the format of data it receives. It also requires to be trained initially for a few days before being deployed.

3. Design Description

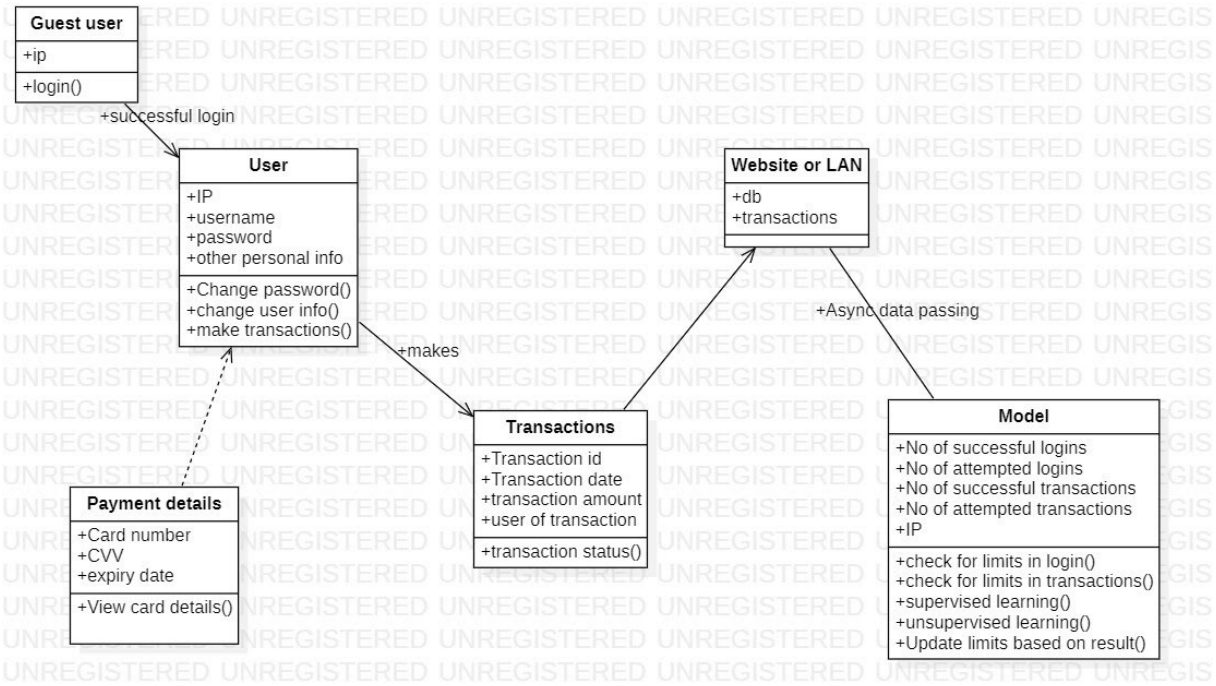
This is a relatively newer concept. Many startups like harvest.ai got acquired by giants like Amazon for improvising in this field. So our design goal is to implement this model on a large scale website and test for efficiency.

The system will be trained to implement both a supervised model and unsupervised model. Unsupervised model is well enough for more noticable and general issues such as account takeover and sudden increase in activities. Supervised model is for complex issues noticeable only by cyber analysts such as fraudulent transactions.

We are going to see training on millions on logs(depending on the dataset). So our model should be efficient enough to detect attacks inside those records in minutes.

Since we train it everyday depending on the attacks detected, we need good computing capacity. There is a separation between web servers and the model. The model works independent of the server. The server sends data asynchronously to the model.

Master Class Diagram



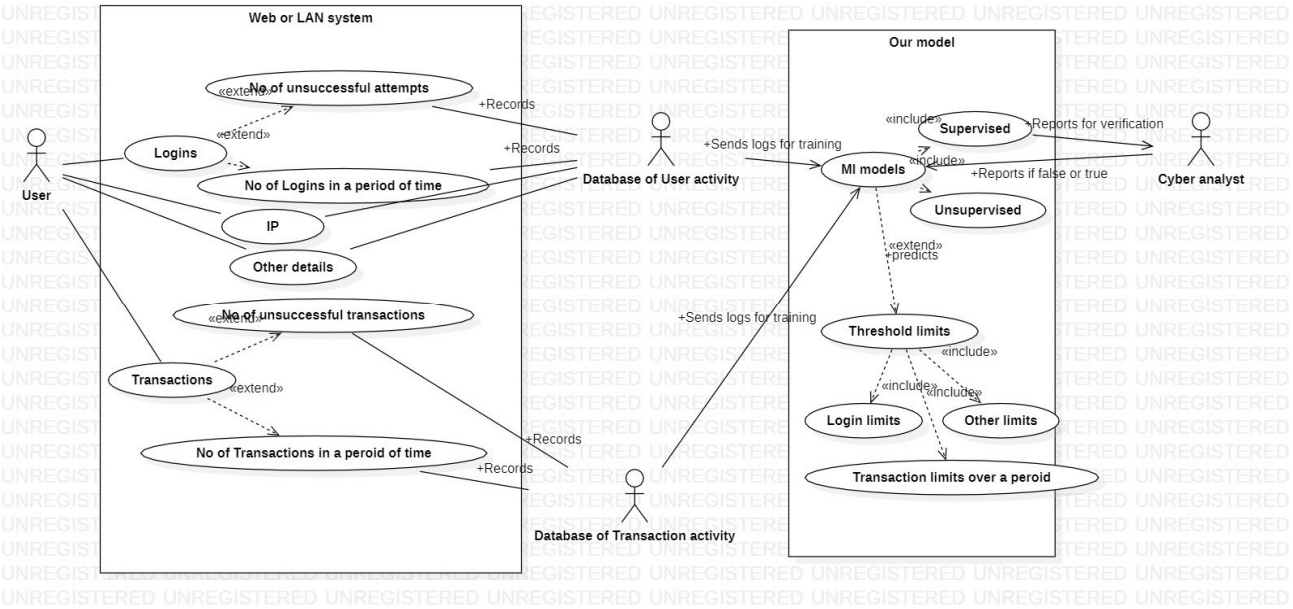
Before describing the relation between the model and the web server, the relation between website and the user needs to be understood before deploying.

A guest user becomes a user after logging in. The number of times logged in and changes in the 2 factor authentication data and related login data is stored and sent to the model.

Same rule applies for transaction data for the account. All the related data is sent for the model for analysis.

The model then sends the result to the analyst for analysis. The result from analysis obtained is again trained by the model.

Use Case Diagram



Above is the overall working of the system. A user is a guest visitor until he logins.

A guest visitor after successful login becomes a user. The no of unsuccessful and successful logins is recorded. They are sent for training for the model. It is individual user based training. It sets the limits based on the usage on the account.

A user can perform as many transactions as he can. The model sets the limits based on previous account activities. Too much increase in transactions sets the trigger for the model.

The model can train based on the input by the human analyst. After it has trained, it sets the limits for individual user accounts based on previous activities. It does both supervised as well as unsupervised training.

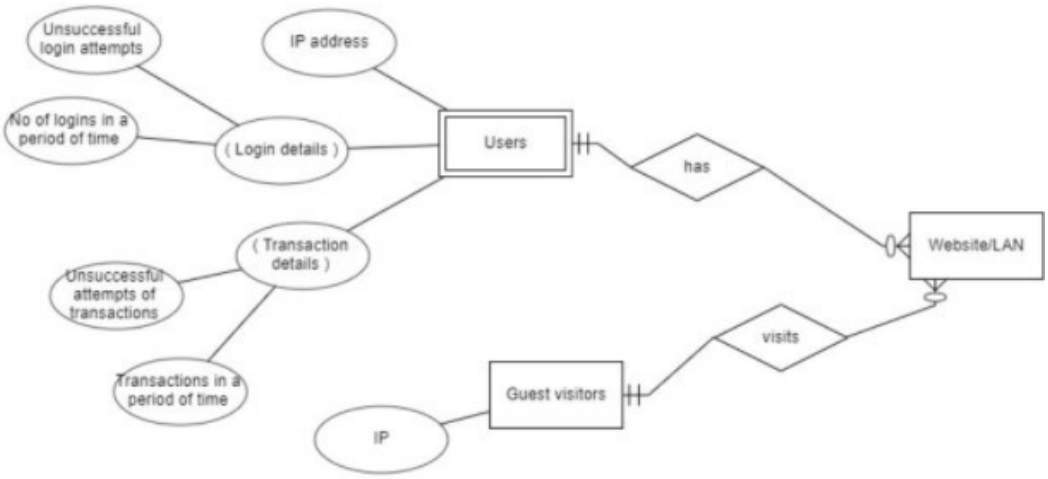
Interface

There are no interfaces required since it doesn't face any users of the website. It just needs to report to a human analyst. The output can be of any format comfortable to the analyst such as json, or the link to the page for the user account monitoring page.

Hardware requirement

Since this trains a machine learning model, the system will require a high throughput machine. This can be kept separate to the web server since data can be passed to this system from the web server.

3.1.1. ER Diagram

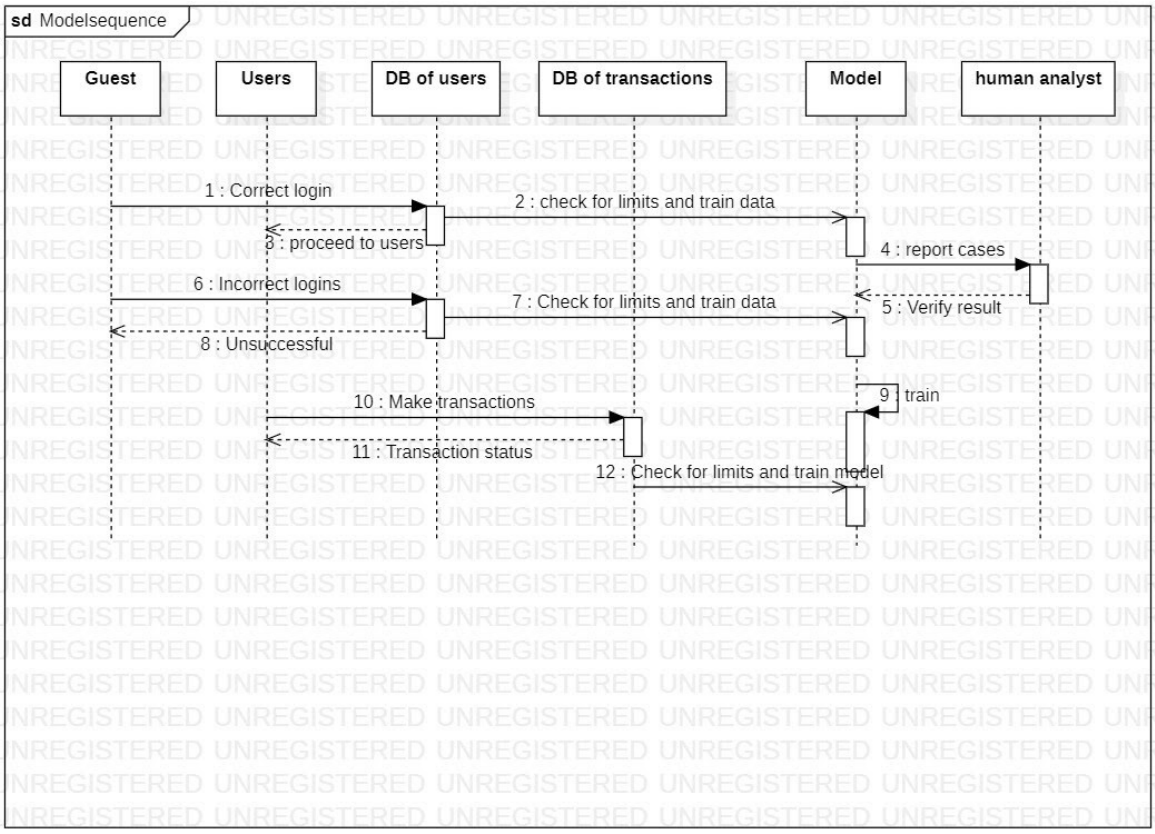


ER diagram above showing typical structure of a web server system. These are the parameters we are going to train on the AI model.

#	Entity	Name	Definition	Type
ENTITIES				
1.	Users	Website accounts	The visitors to the site with the account in the site.	Guest users and logged users
2.	Website/LAN	The website	The website and it's associated database.	website,server or local network
#	**Attribute**	**Name**	**Definition**	**Type (size)**

DATA ELEMENTS				
1.	Login related details	Login data such as login times and 2-factor auth	Has login data for training on account activity	login data and 2-factor data
2.	Transaction related data	Transaction data	Transaction data for fraud detection	Successful,unsuccessful transactions

Sequence Diagram



The Sequence diagram of how the system is going to report to the human analyst.

Whenever a user logins or performs the transactions, the details are sent to the model in the asynchronous manner, so the web server performs with no lag.

3. Proposed Methodology / Approach

4.1 User Interface and Backend Design

User Interface

We have built a responsive and modern looking web application with a stunning landing page which guides and excites the user or analyst about the advantages which the user may get using the product. It has a sleek and classy professional UI. The design is kept as user friendly as possible without doing an compromise on the look and feel.

The design has the following features.

- Landing page.
- About.
- Learn more.
- How to page.
- Input and Output page.

User Interface is built using the following

- Html
- Cascading Style Sheets (CSS)
- JavaScript (Client side)
- Bootstrap 4
- Pixabay pictures.

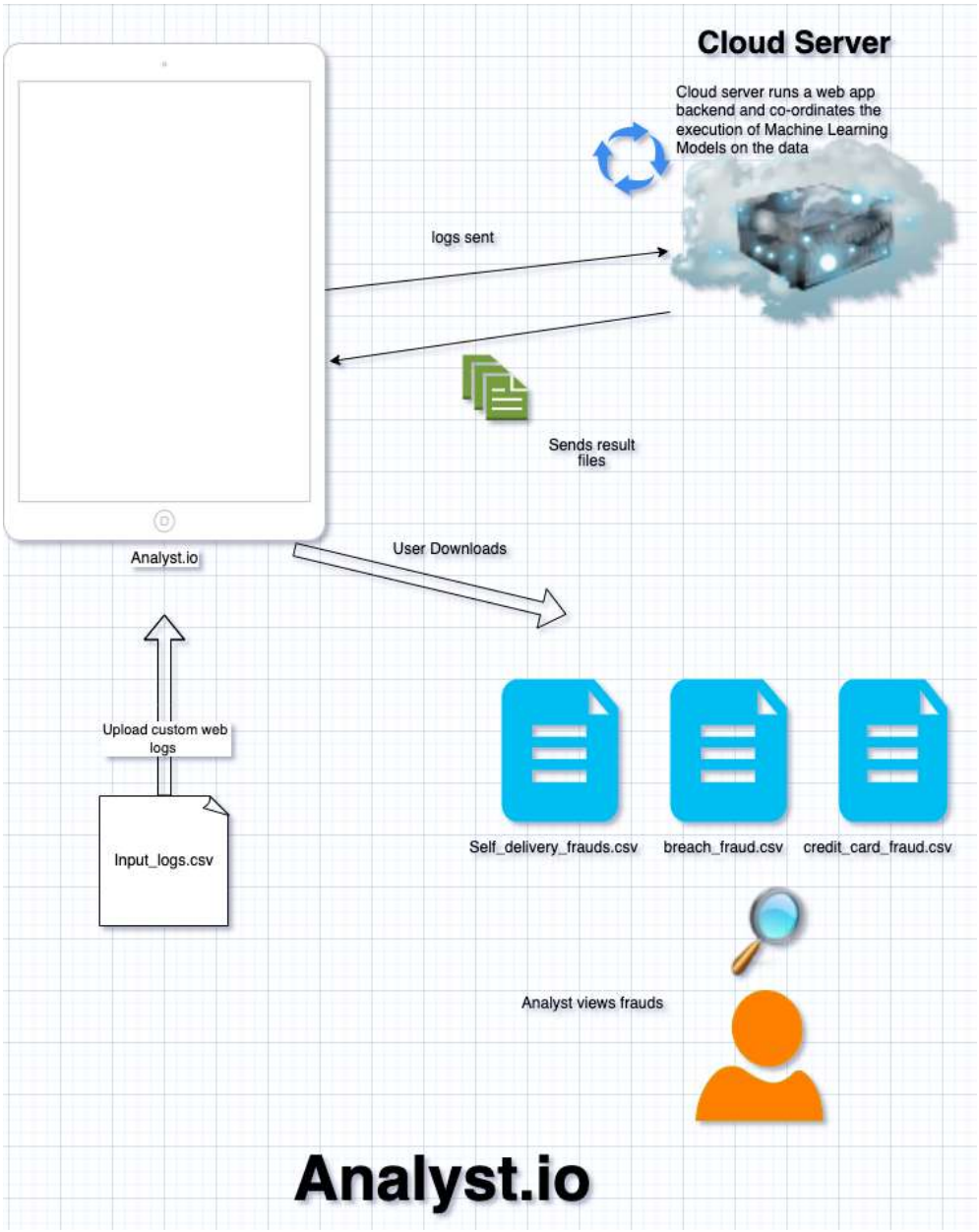
Backend Design

The Backend architecture shows the way the things work at the back end.
The users are supposed to get a copy of the custom E-commerce website logs and upload it safely to the Analyst.io's upload page, then the user clicks the run models button and a request is sent to the backend to execute the Machine Learning Models.

These machine learning models are well equipped to handle logs of size 40 thousand to 1 Lac rows of data with a blazing fast speed and razor sharp accuracy.

Technologies used in the backend.

- Flask microframework
- Requests library



The backend code is well structured and is open to all future extensions. It takes into account all good development practices and the code is a secure code.

4.2 Data Interpretation and Pre-Processing

Breach in terms of use:

- **Data generation:** Every ecommerce site has a limit to the quantity of products by a single user. This is usually done to prevent mass buying and selling by a fraudulent retailer. So to detect this is to detect the users who are buying a product in excess quantity (i.e. more than the average of the quantity bought by genuine users by some excess amount). To generate this data, we create a few users who start buying in excess quantities, which is more than the average of quantities for the same product by a few multiples of standard deviation.
- **Data pre-processing:** Since there are many unwanted columns such as zip code, url, http code, cardnum etc. we remove these columns. We also have a few rows that are not 'orders', so they don't have items or quantity. These rows need to be removed. We also convert quantity and price to numeric.

Self-Delivery Fraud Detection:

- **Data generation**: This is a fraud where a merchant who sells items on the ecommerce site themselves commit fraud. This is done by creating a few fraud accounts where they have the same pincodes. Then they can keep ordering in large quantities of the products sold by the merchant. This gives the merchant a chance to rate his products every time he orders. So he/she can increase the ratings of the products sold by him. To generate this data, we first create a few fraud accounts which have similar pincodes, usually one or two pin codes are used for all accounts. Then the fraud accounts order only the items sold by the merchant and give 5 star ratings. This way the merchant increases ratings for his own products and thus gives fake reviews.
- **Data pre-processing:** We need to look at the parameters that make a user a fraudulent user by his transactions. To perform the fraud, a merchant creates a user who performs large quantities of transactions on products belonging to a specific merchant. He also orders the product to the same pincode so that he can restore it to his inventory. The user also gives 5-star ratings to the product to increase the ratings. So the parameters we need to look at are:
    - % of total ratings by this account: The percentage of ratings given by this account to that of the total percentage of product rating.
    - % of delivery to the pincode of the user to that of all the delivered pin codes for that product.
    - % of transactions for this product to the total number of products bought by the user.
    - % of 5 star rating: Ratings to the specific product to that of total ratings given by the user.
    - % of 4 star rating: % of 4 star rating for the product to that of total ratings given by the user.

- % of 1 star rating: % of 1 star rating for the product to that of total ratings given by the user.

- **Credit Card Fraud Detection**
  - Data generation: Credit card frauds are done by either stealing a credit card that was already being used by a user before or by getting a stolen credit card to perform a large number of transactions on the account. So the data generation step involves reusing the credit card that was already used by an account or hacking into an account and then performing a large number of transactions on the account.
  - Data pre-processing: To detect if the credit card entered is already in use, we make a list of credit cards in use currently, so we know if someone enters the already in use credit card. To detect the large number of fraudulent transactions on an account, we go for a statistical formula known as z-score. Z-score defines how much standard deviations away is the point from the mean. So no matter what the mean is and where the point lies, all points get reduced to a singular metric of score known as z-score. So we define the limit to a credit card based on the z-score of transactions performed on the card.

4.3 ML Modelling

Breach in terms of use

To Detect the Outliers from the large datasets which contain the transaction details. The Breach of Terms of Use Fraud majorly depends on the two attributes namely, Product id and Quantity which the user has bought. We have used several models to detect the outliers from the dataset

Cluster Based Local Outlier Factor (CBLOF)
CBLOF is a meaningful algorithm which provides importance to local data behavior. It can be used as a measure for identifying the physical significance of an outlier. But, It detects the outliers in a very good manner but it frames some of the inliers as fraudulent transactions.

Histogram-Based Outlier Detection (HBOS)
A Histogram-based outlier detection (HBOS) algorithm is presented, which scores records in linear time. It assumes independence of the features making it much faster than multivariate approaches at the cost of less precision.

KNN
KNN works on a principle assuming every data point falling in near to each other is falling in the same class. In other words it classifies a new data point based on

similarity. It detects the outliers during irregular conditions and the pattern of detection doesn't match with Our Dataset fields and attributes.

Isolation Forest
Isolation forest works on the principle of the decision tree algorithm. It isolates the outliers by randomly selecting a feature from the given set of features and then randomly selecting a split value between the maximum and minimum values of the selected feature. It gives us the best fraud detection model for Breach in terms of use.

Self Delivery Fraud Detection

Random Forest
Random Forest is a supervised learning algorithm which has nearly similar hyperparameters as a decision tree or a bagging classifier. It adds additional randomness to the model, while growing the trees. It searches for the best feature among a random subset of features instead of fetching for the most important feature while splitting a node.

Decision Trees
Decision Tree is a model which classifies based on multiple covariates or for developing prediction algorithms. It classifies Our Dataset population into branch-like segments that construct an inverted tree with a root node, internal nodes and leaf nodes.

Credit Card Fraud Detection

We need the Outlier Detection model to detect the associated Credit card frauds in the transactions. We tried to use the Isolation forest model to detect the higher Z Scores transactions. In the Isolation Forest, the anomalies need a lesser number of random partitions to be isolated compared to the so defined normal data points in the dataset. Therefore, the anomalies will be the points of a shorter path in the tree. The path length is assumed as the number of edges traversed from the root node. The Model provides us with an accuracy of 87-89%.

4.4 Further Explorations and Plans

Our Goal is to reduce the amount of logs a human need to go through to detect anomalies in the user accounts. We have combined all these models and made a system with the dashboard and trained the newly available data with the model. So The model becomes more and more accurate overtime. Thus developed models can

adapt to many different sites as it can keep on training and can become more and more accurate overtime.

We would like to extend the number of frauds detected in Our Website Analyst.io and present more insights on the data representation and visualizations to enable the users to grasp the data distribution in a more efficient and convenient manner

Appendix A: Definitions, Acronyms and Abbreviations

- **Web server**: The server that owns and operates the website.
- **Web owner**: The manager handling the web server.
- **Model/our model**: The server system that trains on data provided by the web server and reports anomalies to cyber analysts.
- **Cyber analyst**: The person who has knowledge about cyber security and can identify malicious activities in the web server.

Appendix B: References

- Campus, Kattankulathur. "Credit card fraud detection using machine learning models and collating machine learning models." International Journal of Pure and Applied Mathematics 118.20 (2018): 825-838.
- Maniraj, S., et al. "Credit card fraud detection using machine learning and data science." International Journal of Engineering Research and 8.09 (2019).
- Tripathi, Diwakar, Bhawana Nigam, and Damodar Reddy Edla. "A novel web fraud detection technique using association rule mining." Procedia computer science 115 (2017): 274-281.
- Quah, Jon TS, and M. Sriganesh. "Real-time credit card fraud detection using computational intelligence." Expert systems with applications 35.4 (2008): 1721-1732.
- Kou, Yufeng, et al. "Survey of fraud detection techniques." IEEE International Conference on Networking, Sensing and Control, 2004. Vol. 2. IEEE, 2004.
- MIT ,CSAIL,ai square:Training a big data machine to defend. Kalyan Veeramachaneni and Ignacio Arnaldo. http://people.csail.mit.edu/kalyan/AI2_Paper.pdf
- PatternEx,Virtualanalyst, https://www.youtube.com/watch?v=mLCX5wZ3B40

Appendix C: Record of Change History

[This section describes the details of changes that have resulted in the current Low-Level Design document.]

#	Date	Document Version No.	Change Description	Reason for Change
1.				
2.				
3.				

Appendix D: Traceability Matrix

[Demonstrate the forward and backward traceability of the system to the functional and non-functional requirements documented in the Requirements Document.]

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.
Design goals on training data	section 3.1
Data handling on server	section 3.3
Hardware requirement	section 4
Software requirement	section 12 1.11