HARARE INSTITUTE OF TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

BTECH (HONS) INFORMATION TECHNOLOGY

Real Time Credit Card Fraud Detection

Using Ensemble Methods

BY

Nyasha Chizampeni

H170209V

HIT 400

SUPERVISOR: Mr. Sumburero

**This project report was submitted to Harare Institute of technology in partial fulfilment of the bachelor of technology (Hons) degree in information technology**

DECLARATION

I hereby declare that this document is my original authentic work, which I have worked out with the help of my supervisor. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

ACKNOWLEDGEMENT

l sincerely express gratitude to the esteemed and revered guide of Mr. P Sumburero of Department of Information Technology for his invaluable guidance, supervision, and encouragement throughout the work. I also would like extend our gratitude to all HIT students for their assistance in some of the aspects of my proposal especially on surveys, support.

ABSTRACT

Fraud is the illegal act of impersonating an individual in order to discredit them or acquire benefits or privileges meant for the said individual. Fraud detection is a strong pillar of FinTech, used as a layer of security to ensure that client transact in a safe, controlled and monitored environment. Since the innovation of e-business and online transactions, new attack vectors have opened up and people with malicious intent are discovering ways to manipulate these new systems and steal from unsuspecting individuals.

There have been significant developments in the research to fight fraud, from rule-based algorithms for dealing with stagnant and slow evolving economic climates to machine learning and artificial intelligence for constantly evolving economies, wide array of solutions has been researched and evaluated for this problem.

The economic climate in Zimbabwe is ever changing and very few systems can cope with it. Only huge organizations and financial institutes have the resources to implement more complex and accurate solutions that use machine learning and deep learning. This leaves the huge majority of SME’s exposed to this threat, resulting in innocent Zimbabweans being duped of their hard-earned money every day. The proposed solution intends to provide a complex and highly accurate solution in a very simple and affordable manner.

Keywords: FinTech, Machine Learning, Deep Learning, Artificial Intelligence, Rule Based Algorithms

Contents

[DECLARATION 3](#_Toc10523854)

[CHAPTER 1 INTRODUCTION 8](#_Toc10523855)

[1.1Introduction 8](#_Toc10523856)

[1.2 Background 8](#_Toc10523857)

[1.3 Problem Statement 9](#_Toc10523858)

[1.5 Hypothesis 11](#_Toc10523860)

[1.6 Technical Objectives 11](#_Toc10523861)

[1.7 Expected Results 11](#_Toc10523862)

[1.8 Ethics Consideration 12](#_Toc10523863)

[1.9 Conclusion 12](#_Toc10523864)

[CHAPTER 2 REQUIREMENTS ANALYSIS 13](#_Toc10523865)

[2.1 Introduction 13](#_Toc10523866)

[2.1.1 Evaluate Alternatives 13](#_Toc10523867)

[2.1.2 Outsource 13](#_Toc10523868)

[2.1.3 Improvement 14](#_Toc10523869)

[2.1.4 Development 14](#_Toc10523870)

[2.2 User Requirements 14](#_Toc10523871)

[2.2.2 Technical Feasibility 15](#_Toc10523872)

[2.2.3 Hardware 16](#_Toc10523873)

[2.2.4 Software 16](#_Toc10523874)

[2.2.5 Technical Expertise 16](#_Toc10523875)

[2.3 Economic Feasibility 16](#_Toc10523876)

[2.3.1 Cost Benefit Analysis 17](#_Toc10523877)

[2.3.2 Tangible Benefits 18](#_Toc10523878)

[2.4.1 Schedule Feasibility 20](#_Toc10523879)

[CHAPTER 3 SYSTEM ANALYSIS 21](#_Toc10523880)

[3.1 Introduction 21](#_Toc10523881)

[3.2. Description of current system 21](#_Toc10523882)

[3.3.1 Context diagram of the existing system 21](#_Toc10523883)

[3.3.2 Weaknesses of current system 22](#_Toc10523884)

[3.4 Description of the Proposed Solution 22](#_Toc10523885)

[CHAPTER 4: SYSTEM DESIGN 28](#_Toc10523887)

[4.1 Introduction 28](#_Toc10523888)

[4.4 Database Modelling 31](#_Toc10523889)

[4.4.1 DATABASE TABLES 32](#_Toc10523890)

[4.4.1 E-R Diagram 34](#_Toc10523891)

[34](#_Toc10523892)

[4.5 Algorithm Design 35](#_Toc10523893)

[4.6 Interface Design 35](#_Toc10523894)

[Input design 35](#_Toc10523895)

[4.7 Security Design 36](#_Toc10523896)

[CHAPTER 5 IMPLEMENTATION AND TESTING 37](#_Toc10523897)

[5.1 Introduction 37](#_Toc10523898)

[5.2 Coding Strategy 37](#_Toc10523899)

[5.3 Coding Review 38](#_Toc10523900)

[5.4 Types of Testing and Results 42](#_Toc10523901)

[5.4.2 Non-Functional Testing 43](#_Toc10523902)

[5.5 Test Cases 44](#_Toc10523903)

[5.6 Levels of Testing and Results 44](#_Toc10523904)

[5.6.1 Unit Testing 44](#_Toc10523905)

[5.6.2 Integration Testing 46](#_Toc10523906)

[5.6.3 System Testing 46](#_Toc10523907)

[5.7 Installation 46](#_Toc10523908)

[5.7 1 User training 47](#_Toc10523909)

[5.8 Conclusion 47](#_Toc10523910)

[CHAPTER 6 CONCLUSION 48](#_Toc10523911)

[6.1 Scope for Future Extension 48](#_Toc10523912)

[6.2 Maintenance 48](#_Toc10523913)

[6.2.1 Interval System Review 48](#_Toc10523914)

[6.2.2 Maintenance Activities 48](#_Toc10523915)

[6.2.3 Disaster Recovery 48](#_Toc10523916)

[6.3 Recommendations 48](#_Toc10523917)

[APPENDIX 49](#_Toc10523918)

CHAPTER 1

Introduction

Online transaction fraud is defined as the illegal use of a system or, criminal activity through the use of account information without the knowledge of the account holder. Everyday unsuspecting individuals lose their money to fraudsters who have mastered the weakness of the current fraud detection systems. The current local solutions are either too rigid to adjust to changing fraud trends or they are not available commercially. The purpose of this project is to harness the power machine learning in order to provide a lasting effective solution to fraud detection.

Fraud Detection reinforces client confidence in a business, and also ensures smooth flow of business processes. Online transactions have been the main target for fraudsters in recent years. Attackers study and master the system’s weaknesses and turn them into attack surfaces. There is need for a flexible, accurate and robust solution that can adapt to the way the nature of transactions changes over time. The proposed solution intends to harness the power of machine learning’s ensemble methods and artificial intelligence to develop a system that can combat this growing problem.

Background

The 1990’s were the beginning of the commercialization of the Internet and soon after several types of fraud emerged. The first fraud trend was the use of well-known names for example celebrities to commit fraud. Fraudster would use third-party stolen credit cards with the name of the celebrity of the day. The only security feature available was the completion of an authorization form, the name used in the purchase was not checked and fraudsters used this to their advantage. Merchants got smarter and implemented rules to check the name being used. Since there are so many possible names, and so many people with the same name, it was only a temporary solution. Likewise, the fraudsters moved on to new attacks. Next the fraudsters came up with card-generator applications for real credit card numbers, using these they defrauded merchants over and over again. Over time new forms of fraud emerged each time solutions were brought forward.

The fraudsters could now test the credit cards and go on a shopping rampage over the internet up to date. In addition, a 2015 research note from Barclays stated that African counties are responsible for 47 percent of the world’s card fraud in spite of only accounting for less than 40% percent of total worldwide card volume. The high level of debit and credit card fraud in the United States and China impacts other countries. Across the border fraud occurs when fraudsters use a consumer's credit or debit card data in one country to make fraudulent transactions in another country. That may finally be what it takes for us to guard ourselves, as the number of fraud victims is estimated to reach 14 million in 2018. The number of clients who experience a credit card breach and fraud in the same year is expected to raise 34 percent between 2014 and 2018.

Motivation

The Zimbabwean economy has been on the down fall for a long time now and this has bred a conducive environment for fraudsters to thrive. With the advancements in technology, these attackers have found new and more dangerous ways to attack the system. There is a need to use the same technology to create a safe and legal cyberspace for all consumers.

Most Zimbabweans are considered laggards in terms of technology adoption. This is partly because of the risks associated with exploring an unfamiliar space. It is because of risks such as online transaction fraud that demotivates Zimbabweans citizens and businesses to engage with new technologies. By finding ways to mitigate these risks, more Zimbabweans will feel comfortable adopting new technologies in their businesses.

Problem Statement

Both the card issuer and the merchant are liable in the event of fraud, most of the fraud detection efforts are being directed towards the card issuer (the Bank) and merchants are left with no real tools to protect themselves. Most local SME’s still use rule-based techniques to detect fraud. This solution was relevant back then when Zimbabwe had a stable economy, now these complex rules have to be constantly be rewritten at every turn to avoid incorrect classification of transactions. Bigger organisations can afford develop in house solutions that are not available to the public leaving SME’s exposed to attackers. As a result of this most businesses in Zimbabwe hesitate in the adoption of technology in their business processes because of fear of the risks associated with transacting online.

Hypothesis

If we observe the nature and properties of online transactions, and classify them into fraudulent and non-fraudulent classes, we can use our observations to determine where future transactions are safe or not before we allow them to go through.

Technical Objective

* To develop and deploy a model that can accurately classify transactions as fraudulent and clean

Expected Results

* A model that can classify transactions as normal or fraudulent in real time
* A model with an accuracy score of at least 80%
* The model is to be deployed using a REST API

Ethical Consideration

The data used to train the machine learning model is actual transaction data from customers and has to be handled with care and privacy to maintain confidentiality. Some critical fields of the data have been removed to this effect.

Conclusion

This chapter puts across the problem that this study is solving, a brief description of what is already happening locally and globally with regards to the field of study in form of an introduction, the objectives of the study, justification of the study and its scope. In the following chapter the researcher is going to review literature that has already been written by other researchers in the area of credit card transaction fraud detection.

CHAPTER 2

Introduction

This chapter defines the concept of Credit Card transaction fraud detection using a machine learning model developed using ensemble methods. This chapter will investigate this topic looking at the different ways in which different systems of the same context were implemented.

# A Genetic Programming Approach for Fraud Detection in Electronic Transactions

The researcher made use of a Latin American dataset based on UOL PagSeguro electronic payment system. The research made use of the genetic programming algorithm which is very efficient on problems with a large search space. The research was also focused on companies that are transactions proxies like PayPal. The GP algorithm generates random classification rules at first, and each rule is reproduced, crossed over and mutated in order to receive a fitness value. The process is repeated to obtain the best classification model.

The GP algorithm is rather difficult to understand and requires extensive knowledge to setup and operate.

# Implementation of Hidden Markov Model for Credit Card Fraud Detection

The researcher proposed a solution that would create clusters by identifying the spending habits of the cardholder and choose an initial set of probabilities based on the spending profile. The system would also then construct sequences for training data and build and a model. The HMM is a twofold stochastic process with two pecking order levels. It can be utilized to model muddled stochastic procedures when contrasted with a conventional Markov model. A Hidden Markov Model has a limited arrangement of states represented by an arrangement of progress probabilities. HMM utilizes cardholder's spending conduct to recognize fraud.

The HMM is highly expensive, yields low accuracy and is not scalable when presented with very large datasets.

# An Approach to Identify Credit Card Frauds through Support Vector Machine using Kernel Trick

The researcher made use of a German transaction dataset and the solution was proposed it improve the previous system with made use of a combination of the HMM, k-clustering and OTP verification. The proposed solution attempted to use Support Vector machines to classify between fraudulent and legit credit card transactions. The data was pre-processed and converted into a svmlib format. A polynomial kernel function was selected and used to find the best parameters for speed and accuracy. Finally, the SVM classifier was trained and validated with accuracy was collected and documented. This result was also tested against other kernels such as Linear and RBF.

While the accuracy of the SVM was impressive, it lacked transparency of results and failed to scale with the increase in the size of the dataset.

# A New User-Based Model for Credit Card Fraud Detection Based on Artificial Immune System

The AIS method of fraud detection is based on the user transaction behaviour. It combines two methods which are general thresh holding and account behaviour. The researcher did not specify the dataset used in the development of the system. AIS generates normal memory cells suing the user’s transaction records and fraud cells using the fraudulent records. To get a more accurate result, analysis on the training dataset has to be performed in order to control the number of memory cells produced. One good property of this method is its ability of adaptive learning

During the test phase the user’s transaction details are presented, to the normal memory cells together with the fraud cells. This method has more false positives since it based on the user’s transaction patterns, but most people have irregular spending habits.

# INSURANCE FRAUD DETECTION USING RANDOM FOREST

In this paper fraud detection using Machine learning is done by deploying the classification and regression algorithms, the researcher uses the supervised learning Random forest algorithm to classify the fraud. Random forest is advanced version of Decision tree. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by picking only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by fixing a stopping criterion for node splits, Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest. The Random forest algorithm’s performance will be deterred when dealing with a small data set.

# Hybrid Design using Counter Propagation Neural Network-Genetic Algorithm Model for the Anomaly Detection in Online Transaction

CPNN, a variant of ANN was used for classification due to its capacity for generalization because of its refined network and experimentally proven better learning rate. GA’s optimization was integrated into this system in order to optimize the CPNN training parameters so that the best chromosome having optimal parameter setting can be obtained, and used by CPNN for classification purposes. The system operated in two stages; in the first phase, GA formed clusters. Clustering was done by dot product, while in second phase, the weights between the cluster units and the output units were adjusted. Minimizing error function; error function being the average error incurred when CPNN classifies large input data was considered. Initial weights were randomly selected between 0 and 1, with an assumed initial population size.

It is rather difficult to confirm the structure or high processing time for large neural networks, it lacks results transparency and it is difficult to setup.

Conclusion

After an extensive research of the various methods of credit card transaction fraud detection, l concluded that sklearn’s ensemble methods Random Forest algorithm would be used to implement the proposed solution. I decided to use the Random Forest algorithm because:

* **Performance:** It can make better predictions and achieve better performance than any single contributing model.
* **Robustness**: It reduces the spread or dispersion of the predictions and model performance.

CHAPTER 3

Introduction

In this chapter, l am going to analyse an existing system stating its weakness, describe the proposed solution by use of diagrams, and clearly state the functional and non-functional requirements of the proposed system and conduct a feasibility test.

Existing System

One of the current solutions to fraud detection in the market right now is the rule-based method. Rule based fraud detection systems use correlation and logical comparison of data to identify potential ‘acts of fraud’ based on insights gained from previous (known) fraud incidents. They generally use traditional methods of data analysis and require complex and time-consuming investigations that deal with different domains of knowledge like financial, economics, business practices and behavior. Fraud often consists of many instances or incidents involving repeated transgressions using the same method. Fraud instances can be similar in content and appearance, but usually are not identical. Rule based systems rely on identifying a known fraud pattern.

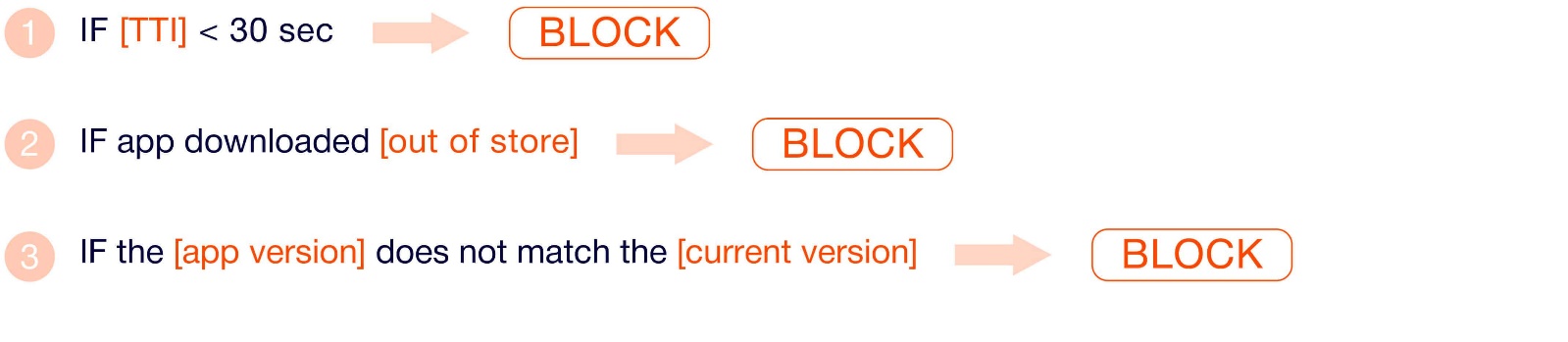


Fig 1.1

Above is a snippet of an example of a rule-based method, unfortunately fraudsters do not follow rules

Context Diagram



Fig 1.2

Process Flow Diagram



Fig 1.3

In this process flow the client initiates a transaction and the details are verified, once verified the transaction is then classified and stored in the database, if the transaction is clean, the system returns an ok message to the client, but if it is fraudulent, the transaction is flagged and the admin is alerted.

User Case Diagram



Fig 1.4

# Feasibility Analysis

A feasibility analysis is an important tool that help in the assessment of the viability of a project.

# Technical Feasibility

The tools to be used to develop the proposed solution are open source and freely available on the internet and once it is developed client systems will only need access to the internet to make use of the proposed solution.

Based on the size of the project, it can be considered a one-man job since it has a development period of over 6 months.

# Economic Feasibility

As stated before, the tools used to develop the proposed solution are open source and therefore free of charge and it will be deployed for free using the Heroku or pythonanywhere hosting service. Considering the size and complexity of the project outsourcing technical labour is not necessary.

# Requirements Analysis

This analysis goes into detail about how the system is expected to respond to certain parameters and situations as illustrated below under functional as well as non-functional requirements.

# Functional Requirements

Below is a table stating the functional requirements.

|  |  |
| --- | --- |
| **Number** | **Functional Requirement** |
| 1 | Generate unique alphanumeric credentials |
| 2 | Classify transactions in real time |
| 3 | Return model performance analytics on request |
| 4 | Generate a dataset of user transactions on demand |

Table 1.1

DFD Level 1



Fig 1.5

Use Case Diagram



Fig 1.6

# Non-Functional Requirements

Below is a table stating the non-functional requirements

|  |  |
| --- | --- |
| **Area** | **Non-Functional Requirement** |
| Availability | System to be available 24/7 |
| Performance | Classification to occur in near real time or real time |
| Security | Critical Data to be encrypted |
| Usability | Error messages should be clear and concise |

Table 1.2

# Interface Requirements

The classification model is going to be deployed as a REST API, in order to allow for smooth interfacing with other systems. The Client will also be provided with a web interface to manage and monitor their account and data.

# Requirements of the REST API Interface

* An Internet connection
* Ability to make HTTP requests

# Requirements of the Web Interface

* A functional browser
* An internet connection

# Hardware Requirements

|  |  |
| --- | --- |
| Processor Type | Core i3 or better |
| Processor Speed | 1.70 Ghz or better |
| HDD | 20 Gb or more |
| Monitor | VGA or higher resolution 800x600 or higher resolution |
| Pointing Device | Mouse or compatible pointing device |
| RAM | 4Gb or better |

Table 1.3

# Software Requirements

|  |  |
| --- | --- |
| Environment | Python 3.6 or later |
| OS | Windows 7 or later, Linux Ubuntu 17.04 or better |
| IDE | Any suitable IDE |
| Framework | Flask |
| packages | Numpy, pandas, sklearn, flask-restful, requests, seaborn, matplotlib, secrets |

Table 1.4

CHAPTER 4

This CHAPTER is mainly concerned with designing of the proposed system. The implementation of the solution and how the different software components interact.

# Proposed Solution

In order to detect fraudulent transactions in real time, a dataset consisting of local transactions was obtained, the data will be cleaned and transformed in order to train a machine learning model namely the Random Forest classifier from sklearn’s ensemble methods. The trained model will be evaluated using accuracy score, classification report and the confusion matrix. The model will then be extracted and deployed within a REST API to enable interfacing with client systems.

# Solution Architecture



Fig 1.7

# Machine Learning Process

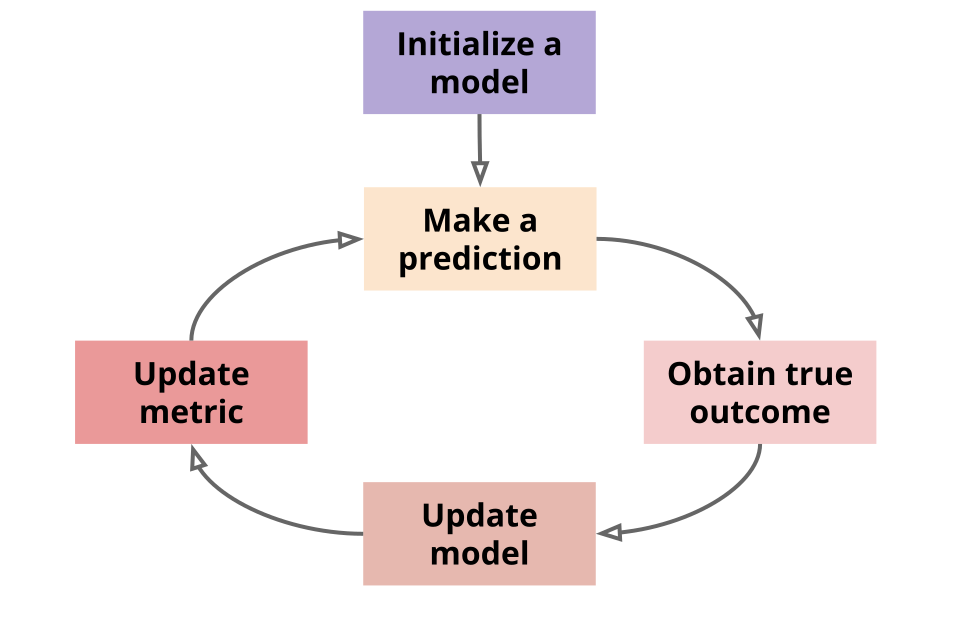


Fig 1.8

# System Architecture



Fig 1.9

The system has 4 major components:

**Client System (Grey)**

This is the system that will make use of the classification API, it is responsible for collecting transaction data and transaction processing

**REST API (Grey)**

This is responsible for bridging the classification model and the client system. It handles requests from the client system, and returns an appropriate response acquired from the classification model

**Data Processor (Light Blue)**

This is responsible for converting the processed data into a data stream required as input the adaptive random forest classifier

**Data Store (light Green)**

This is responsible for the storage of all transaction data

**Retraining Module (Light Orange)**

Since the system follows the KAPPA architecture, the model is trained every time it receives a new data point in order learn new trends.

**How it Works**

With the kappa architecture, the data is treated as a stream. The system does not have to store a historical training set and retrain then model every so often. Another benefit of is that your model is always up-to-date. The natural consequence is that your model is able to deal with concept drift, which happens when the data’s distribution evolves as time goes on. Moreover, the model lifecycle is easier to think about because the learning and prediction steps both process one input at a time. This is architecture was selected because of its similarity to the production environment, where transaction data is served to the API in the form of a data stream.

# Security Design

The proposed solution makes use of token-based authentication for security. In the Token based approach, the client application first sends a request to Authentication server with a valid credentials. Authentication server send an Access token to the client as a response. This token contains enough data to identify a particular user and it has expiry time. The client application then uses the token to access resources in next requests till the token is invalid or expired. When the Access token expires, the client application can request for new access token by using Refresh token.

# Security Design Model

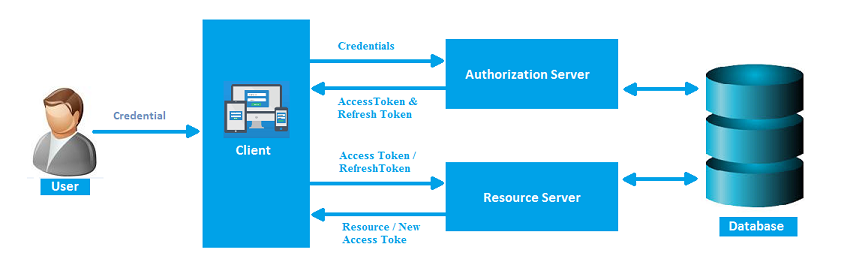


Fig 2.0

UML Activity Diagram



Fig 2.1

UML Sequence Diagram



Fig 2.2

# Database Modelling

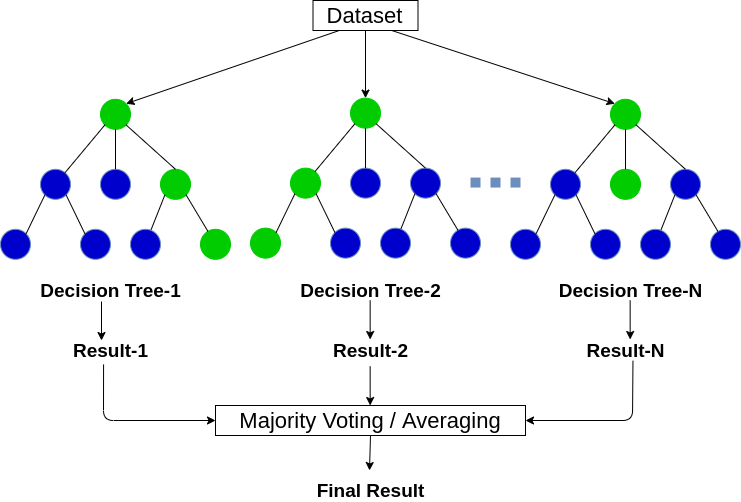
An Object Relational Database is responsible for data storage in this project. SQLite is a light weight file based database that is capable of handling the database requirements of this project.

ER- Diagram



# Algorithm Design

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.



# Interface Design

The proposed solution provides two interfaces to the client, REST API and a web interface. Both these interfaces are designed following well established industry standards to provide clients with simplicity, understandability and ease of use.

# REST API

This interface is meant for business phasing, where the system interfaces with other systems without human involvement. The interface has 4 main methods that can be accessed through ‘GET’

* Authentication - to get the client access token
* Classification - to classify a transaction
* Analytics - to get model performance information
* Data – to retrieve all user transaction data in JSON format

# Web Interface

This was developed with JavaScript HTML and css using the flask framework in order to enable client to manage and monitor their accounts, payments, data and classification model.

# Conclusion

This chapter has described the proposed solution using various diagrams and also described the functional and non-functional requirements of the system.

Chapter 5

# Introduction

This chapter focuses on the strategies that were used to implement the proposed solution and the type of tests done on the final system with their corresponding results. It also explores the functional and non-functional tests done on the final system.

# Coding Strategy

Concepts of functional and object-oriented programming were used. Major functionalities of the system were wrapped in functions and classes. These functions all return a standard json response complete with a status message. This was done to provide more information to the client to assist them during debugging. The use of functions and classes also improves code reusability and speeds up the development process.

# Coding Review

Due to the prevailing COVID-19 pandemic, code review had to be handled by peers in different geological locations which resulted in the use of online repositories like Kaggle for the notebook and GitHub for the code base.

Functional Testing

# Unit Testing

The system is made up 4 main components as stated in fig 1.9. During unit testing each component is tested separately and the results were recorded.

# Rest API

The REST API has 4 main methods

|  |  |  |
| --- | --- | --- |
| Method | Test Case | Result |
| Authentication | Authentication.tc | Successful |
| Classification | Classification.tc | Successful |
| Analytics | Analytics.tc | Successful |
| Data | Data.tc | Successful |

# Data Store

The data store is responsible for storing all transaction data that goes through the system. The data store has two main functions

* Load data
* Retrieve data

|  |  |  |
| --- | --- | --- |
| Method | Test Case | Result |
| Load Data | Load data.tc | Successful |
| Retrieve Data | Retrieve data.tc | Successful |

# Data Processor

The data processor transforms data into a format suitable for the classifier and it as one method preprocess

|  |  |  |
| --- | --- | --- |
| Method | Test Case | Result |
| Preprocess | Preprocess.tc | Successful |

# Retrain Model

This is responsible for retraining the model with new data. It involves calling the data retrieve method and the preprocess method. It has one method Retrain.

|  |  |  |
| --- | --- | --- |
| Method | Test Case | Result |
| Retrain | Retrain.tc | Successful |

# Integration Testing

There are pairs of components that are tightly coupled and they have to be tested together and evaluate the relationship between them. The pairs are:

* Rest API – Data Store
* Rest API – Preprocessing
* Retrain – Data Store
* Retrain – Preprocessing

|  |  |  |
| --- | --- | --- |
| Components | Test Case | Results |
| Rest API – Data Store | RA-DS.tc | Successful |
| Rest API – Preprocessing | RA-PP.tc | Successful |
| Retrain – Data Store | R-DS.tc | Successful |
| Retrain – Preprocessing | R-PP.tc | Successful |

# System Testing

This was done on the complete system to test if all the components function as one unit. This is when we parse test data in the client interface and observe the responses of the system as a whole.

|  |  |  |
| --- | --- | --- |
| Component | Test case | Results |
| Complete System | Complete system.tc | Successful |

# Non-Functional Testing

This is when we test the non-functional requirements of the proposed solution. These include:

* Availability
* Security
* Reliability
* Usability

|  |  |  |
| --- | --- | --- |
| Component | Test Case | Result |
| Availability | Availability.tc | Successful |
| Security | Security.tc | Successful |
| Reliability | Reliability.tc | Successful |
| Usability | Usability.tc | Successful |

Test Cases

# Authentication.tc

This test case tests the authentication method of the REST API. During authentication the client obtains their client id and client token from their web portal and use them to get their API key. They will use this API key to authenticate all their API calls.

|  |  |  |
| --- | --- | --- |
| Client ID | Client token | Output |
| Wrong | Wrong | Invalid client credentials |
| Wrong | Right | Invalid client credentials |
| Right | Wrong | Invalid client credentials |
| Right | right | API key |

Tests were also done on the API calls

|  |  |  |
| --- | --- | --- |
| API Call | API key | Output |
| Get classification | Wrong | {'class': None, 'message': 'Invalid Key'} |
| Get classification | Right | {'class': pred, 'message': 'success'} |
| Get analytics | Wrong | {‘error’: invalid key} |
| Get analytics | Right | {‘analytics\_data’: data} |
| Get Data | Wrong Right | {‘transaction\_data’: data} |
| Get data | Wrong | {‘error’: invalid key} |

# Classification.tc

Tests were run on the classification method of the rest API. Two sets of test data were used during this test case

|  |  |  |
| --- | --- | --- |
| Input Data | API Call | Output |
| Invalid data | Get Classification | {'class': 'None', 'message’:’ invalid input data, refer to docs'} |
| Valid data | Get Classification | {'class': pred, 'message': 'success'} |

# Analytics.tc

This Rest API method is responsible for returning information about the performance of the model.

|  |  |  |
| --- | --- | --- |
| API Call | API Key | Output |
| Get analytics | Wrong | {'analytics\_data': None , ‘message’: “Invalid key”} |
| Get analytics | Right | {'analytics\_data': data , ‘message’: “Success”} |

# Data.tc

This REST API method returns all the client’s transactional data to date

|  |  |  |
| --- | --- | --- |
| API Call | API Key | Output |
| Get data | Right | {'transactional\_data': data, message:”success” } |
| Get data | Wrong | {'transactional\_data': None, message:”invalid key” } |

# Load data.tc

This is a method that loads transactional data into the data Store

|  |  |  |
| --- | --- | --- |
| Method | Input | Output |
| Load\_data(data) | Invalid data | {'message': ‘data is invalid, refer to docs’ } |
| Load\_data(data) | valid data | {'message': ‘data load successful’ } |

# Retrieve Data.tc

This is a method that retrieves transactional data from the data Store

|  |  |
| --- | --- |
| Method | Output |
| retrieve\_data() | {'transactional\_data': data, message: ”success”} |
| retrieve\_data() | {'transactional\_data': None, message: ”data retrival error”} |

# Preprocess.tc

This method transforms data into a format suitable for the classifier

|  |  |  |
| --- | --- | --- |
| Method | Input | Output |
| Preprocess(data) | Invalid data | {‘proccessed\_data’: None, ‘message’: ‘invalid input data’} |
| Preprocess(data) | valid data | {‘proccessed\_data’: data, ‘message’: ‘success’} |

# Retrain.tc

This method retrains the model

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Input Model | Input Data | Output |
| Retrain | Model | Valid data | {‘model’: model.pkl, ‘message’ = ‘success’} |
| Retrain | model | Invalid data | {‘model’: None, ‘message’ = ‘invalid data’} |

# RA-DS.tc

This is an integration test case of two components the Rest API and the Data Store. The client parses transaction data to the Rest API and it sends data to the data store.

|  |  |  |  |
| --- | --- | --- | --- |
| Data | REST API | Data Store | Output |
| Invalid data | {‘error’: ‘invalid data input’} | n/a | {‘error’: ‘invalid data input’} |
| Valid data | {‘transaction\_data’: data} | {'message': ‘data load successful’} | {'message': ‘data load successful’} |

# RA-PP.tc

This is an integration test case of two components the Rest API and the Data preprocessor. The client parses transaction data to the Rest API and it sends data to the data preprocessor.

|  |  |  |  |
| --- | --- | --- | --- |
| Data | REST API | Data Preprocessor | Output |
| Invalid data | {‘transaction\_data’: None,‘message’: ‘invalid data input’} | n/a | {‘transaction\_data’: None,‘message’: ‘invalid data input’} |
| Valid data | {‘transaction\_data’: data, ‘message’: ‘success’} | {'processed\_data': data} | {'processed\_data': data, ‘message’: ‘success’} |

# R-DS.tc

This is an integration test case of two components the model retrainer and the Data store. When the retrainer method is called, it requests data from the data store.

|  |  |  |
| --- | --- | --- |
| Method | Data Store | Output |
| Retrain | {‘transaction\_data’: data, ‘message’: ‘success’} | {‘new\_model’: model.pkl, ‘message’:’success’ } |
| Retrain | {‘transaction\_data’: None,‘message’: ‘invalid data input’} | {‘new\_model’: None, ‘message’: ‘ train data not parsed’} |

# R-PP.tc

This is an integration test case of two components the model retrainer and the Data preprocessor. When the retrainer method obtains data from the data store, it parses it to the preprocessor method and is trained using the result.

|  |  |  |
| --- | --- | --- |
| Method | Data Preprocess | Output |
| Retrain | {‘ processed \_data’: data, ‘message’: ‘success’} | {‘model’: model.pkl,‘message’: ’success’ } |
| retrain | {‘ processed \_data’: None, ‘message’: ‘invalid data’} | {‘model’: None, ‘message’: ’invalid train data’ } |

# Complete system.tc

This test case is responsible for testing how the system works as a whole. It takes transaction data from the client and returns a classification.

|  |  |  |
| --- | --- | --- |
| Client ID | Client token | Output |
| Wrong | Wrong | Invalid client credentials |
| Wrong | Right | Invalid client credentials |
| Right | Wrong | Invalid client credentials |
| Right | Right | API key |

Tests were also done on the API calls

|  |  |  |
| --- | --- | --- |
| API Call | API key | Output |
| Get classification | Wrong | {'class': None, 'message': 'Invalid Key'} |
| Get classification | Right | {'class': pred, 'message': 'success'} |
| Get analytics | Wrong | {‘error’: invalid key} |
| Get analytics | Right | {‘analytics\_data’: data} |
| Get Data | Wrong Right | {‘transaction\_data’: data} |
| Get data | Wrong | {‘error’: invalid key} |

# Availability.tc

This is a non-functional test case where the system’s availability was tested. The system was deployed on a test server for a week and test scripts were written that made API calls at irregular intervals, which enabled the collection or system reports.

# Security.tc

This is a non-functional test case where the system’s security was tested. Due to the token-based authentication, the system is secured from standard security intrusions. Client data is only compromised in the event that the client exposes their client ID and client token to unauthorized parties.

# Reliability.tc

This is a non-functional test case where the system’s reliability was tested. The system has a minimum accuracy score of 80% and due to the incremental learning component, it is guaranteed to maintain or improve this accuracy score.

# Usability.tc

This is a non-functional test case where the system’s usability was tested. The system makes use of the standard REST API interface design with the get methods for simplicity. The system also makes use of a standard errors and messages through out in the form of json, making it easy for clients to familiarize themselves with the system.

# Installation

This system will interaction with other systems using an API. There is need for integration with other systems in order to fully utilize the system’s business logic. This is done with two easy steps,

1. Authentication

2. Your API Call

# Authentication STEP 1

The client has to create an account with the system’s web portal in order to obtain the client ID and client token. They will also use this web portal to manage their API usage.

# Authentication STEP 2

After obtaining the credentials the client will make their initial API Call in order to obtain a

48 bit alphanumeric API key they will use to authenticate all their API calls.

**NB:** This API key is to be kept secret, do not show it to anyone. The code snippet below contains the API Call and the response

**import requests**

**link = ‘localhost:5000/authenticate/’**

**credentials = {**

**'client\_id': '1fddd7bd47a7f9efcd15a5602bd6462e1',**

**'client\_token': '9c10eb2b52ffa06a123e3369f833f74f'**

**}**

**response = requests.get(link, credentials)**

**Print(response.json())**

**Successful**

**{'api key': 'd7311caa7e3c8e6db309d9fe617fff192bddc5bd8177686d2cbdbeaf05d77aa6766d84a2716f8dc3427e91c34a6e24cc'}**

**Failure**

**{'message': 'invalid client credentials'}**

# Get Classification

In order to get a transaction classified, the client has to parse the transaction data to the REST API

Below is the description of the expected input

|  |  |  |
| --- | --- | --- |
| Input | Description | Example |
| age | Age of the account in days | 250 |
| cvv | Card verification | 435 |
| amount | Total transaction amount | 3000 |
| card number | Last 4 card digits | 3466 |
| location | city | Harare |
| bank | Name of bank | CBZ |
| card type | Credit or debit | Credit |

The following is a code snippet of an API call to classify a transaction and the corresponding response is below

**import requests**

**link = ‘localhost:5000/classification/’**

**api\_key = ‘d7311caa7e3c8e6db309d9fe617fff192bddc3bd8877686d2cbdbeaf05d77aa6766d84a2716f8dc3427e91c34a6e24cc’**

**data = {**

 **'api\_key':api\_key,**

**'account\_age':1305,**

**'avs':8475,**

**'amount':1000,**

**'card\_number':1272,**

**'location':'Kadoma',**

**'account\_type':'Credit',**

**'bank':'Standard Bank',**

**'connection\_type':'http',**

**'cvv':'y',**

**'broswer':'Mozilla/5.0 ',**

**'gender':'female',**

**'entry\_type':'chip',**

**'transaction\_time':64,**

**'account\_balance':674,**

**'holder\_age':28**

**}**

**response = requests.get(link, data)**

**print(response.json())**

Successful

**{'class': ‘normal’, 'message': 'classification successful'}**

Failure

**{'class': ‘None’, 'message': error message}**

# Get Analytics

In order to get analytic data of the classification model, the client has to parse their API key. The following code snippet will obtain the analytic data.

**import requests**

**link = ‘localhost:5000/analytics/’**

**api\_key = ‘d7311caa7e3c8e6db309d9fe617fff192bddc5bd887744686d2cbdbeaf05d77aa6766d84a2716f8dc3427e91c34a6e24cc’**

**auth = {‘api\_key’: api\_key}**

**response = requests.get(link, auth)**

**print(response.json())**

**Response**

**{**

**‘f1\_score’: f1\_score,**

**‘recall’: recall,**

**‘precision’: precision,**

**‘accuracy’: accuracy,**

**‘transactions processed’: number of transactions,**

**‘Normal transactions’: number of normal transactions,**

**‘Fraudulent transactions’: number of fraudulent transcations,**

**‘number of retrains’: number of retrains,**

**}**

# Get Data

In order to get the transaction data, the client has to parse their API key. The following code snippet will obtain the transaction data.

**import requests**

**link = ‘localhost:5000/data/’**

**api\_key = ‘d7311caa7e3c8e6db309d9fe617fff192bdd23c5bd8877686d2cbdbeaf05d77aa6766d84a2716f8dc3427e91c34a6e24cc’**

**auth = {‘api\_key’: api\_key}**

**response = requests.get(link, auth)**

**print(response.json())**

**Response**

**{age,cvv,Amount,CardNo,Label,location,card\_type,bank,**

**435.0,346.0,21.0,18.0,True,Nyanga,Credit,Cabs Bank**

**436.0,312.0,79.0,40.0,False,Harare,Credit,Banc ABC**

**436.0,335.0,76.0,49.0,False,Hwange,Credit,AgriBank**

**436.0,268.0,76.0,171.0,False,Nyanga,Credit,NMB Bank**

**436.0,268.0,76.0,171.0,False,Bulawayo,Debit,FBC Bank}**

# Conclusion

All of the test carried out in system testing were successful and those with errors were rectified, therefore the system is fully functional and ready to be deployed.

Chapter 6

# Introduction

The project has been a success in relation to the fulfilment of the desired objectives. The project has been well planned as referenced by the few problems encountered during the project development. Although it has been a success and is ready to be deployed. This chapter will look at the scope of future work and the recommendations to run the system.

# Scope of Future Work

Though the system has been a success, there is always a room for improvement. The thing that are going to be done in the future include the following:

1. To implement KYC (Know Your Customer) models that learn the spending habits of each customer and flag transactions that deviate from normal procedures.
2. To implement the system as a dependency supported by several different programming languages

# Recommendation

* Clients are encouraged to refresh their API key regularly for security reasons, this was not automated since it would interrupt business processes if it changed abruptly.