CHAPTER 1: INTRODUCTION

Fraudulent Credit Card Transactions:

Credit card fraud has been around since a long time but the issue has grown substantially over time. In 2013, the number of data breaches were 614 while this number was projected to increase up to a considerable 1579 in 2017. [1] The total monetary loss which parties incurred due to credit card not present (CNP) fraud is expected to reach $6.4 billion in 2018 in the US alone, which itself is a massive three times rise from 2016. [2]

These statistics show how the problem of fraudulent credit card transactions has been increasing in recent years. In the past, before this digital age, credit cards had to be physically stolen, duplicated, or otherwise acquired to carry out a fraud transaction. However, now with the emergence of electronic payments, people can use your credit card by just knowing the card number and security code which can easily be acquired through methods such as phishing, hacking and data breaches, and social engineering among others.

One of the primary concerns associated with credit card fraud is to identify and hence, reverse fraudulent transactions in a timely manner. Since the ratio of fraudulent credit card transactions to real credit card transactions is very small, it is extremely difficult to identify fraud transactions out of millions of real transactions.

The aim of this project is to propose a methodology through concepts of machine learning and data science to efficiently and accurately identify a fraudulent credit card transaction.

What is a Neural Network?

A neural network, more formally known as an artificial neural network, is a machine learning model which based on the structure and functioning of biological neural networks. A neural network can be developed through supervised learning, unsupervised learning, and reinforcement learning. Typically, neural networks are organized in a number of layers which are classified as the *Input Layer*, *Hidden Layer(s)*, and *Output Layer*. Each of these layers is described below:

1. **Input Layer:** This is the layer that deals with the external environment to feed the input into the neural network model. Each independent attribute (variable) in the dataset which has an impact on the output is a neuron in the input layer.

2. **Hidden Layer(s):** This is the layer in which the inputs are processed. It is an intermediary layer between the input layer and output layer in which an activation function, such as the sigmoid function, is applied to process the data fed to the hidden layer from the previous layer.

Theoretically, this is the layer which is responsible for extracting the meaning and required features from the input layer. It is important to mention that there can be more than one hidden layer in a neural network, depending on the requirements and complexity of the dataset. In each hidden layer, the number of neurons can vary. There is no definite formula (or method) known to determine the exact number of hidden layers or neurons per hidden layer in a neural network.

3. **Output Layer:** This is the layer where the processed data is transmitted from the neural network. The number of neurons in the output layer depend on the type of analysis that the neural network is performing and it can vary depending on the desired outcome from the neural network.

In a neural network, the communication between the various layers is carried out through connections between the neurons. Each connection has a weight assigned to it which is typically assigned randomly in the beginning but is then modified and *improved* through a learning rule. An example of a commonly used learning rule is the learning rule in which the weights are recalculated through the backward propagation of the error in the output value. A neural network in which backpropagation occurs is known as a backpropagation-al neural network (BPNN).

Artificial Neural Network (Image): https://cdn-images-1.medium.com/max/1600/1*hczvrCYgU_JQt5sx-UGM1A.gif

As with other machine learning models, a neural network is first trained using a part of the dataset which is referred to as the training dataset. To train the neural network model, the weights are updated several times by applying a learning rule until the weights and outputs are accurate to a satisfactory level.

There are numerous applications of neural networks which include the recognition of patterns or discovery of regularities within a dataset. Generally, a neural network performs very well when the dataset is very diverse, complex, or involves a high number of variables. It is best suited for problems where the relationship between variables is vague and dynamic (non-linear) in nature.

What is Keras?

Keras is an open source machine learning library which has been written in Python. The library is widely used to model neural networks and enable experimentation on neural networks. Keras is commonly used as a wrapper to Tensorflow. It is highly popular because of it is modular, extensible, and user-friendly.

The library contains a number of implementations of all the basic building blocks of a neural network. These include the layers, activation functions, learning rules, optimizers, and objectives in addition to a number of utilities for handling text-based and image-based data. Through Keras, neural network models can be developed on the web, on a Java Virtual Machine (JVM), and smartphones (such as iOS and Android).

The benefit of using Keras is that it simplifies the process of building a neural network. Without Keras, individual components of the neural network have to be developed from scratch and then have to be adjusted according to the problem. Keras enables us to work directly on the neural network and hence making the code simpler, efficient, and cleaner to write, maintain, and troubleshoot.

Here are the advantages of using the Keras machine learning library:

1. Neural network models developed on Keras can be easily transformed into products that can be released on a range of platforms such as mobile and web.
2. There is support for a number of backend engines compatible with Keras which include the Tensorflow backend and CNTK backend.
3. It is developer-centric and not machine-centric. It is easy to use and flexible at the same time.
4. The research community and industry are widely using Keras and it is supported by key companies such as Google and Microsoft.

In the next chapter, we present the studies and solutions that have been proposed by researchers in literature for solving the issue of credit card fraud. We critically evaluate and utilize this research for then describing and proposing a machine learning based solution for the detection of fraudulent credit card transactions.

Aims:

The primary aim of this project is to develop and analyze the performance of a fraud transaction detector which is capable of monitoring, detecting, reversing, and preventing fraudulent transactions. This project aims to improve current systems and methodologies of online credit card fraud detection by incorporating artificial intelligence and machine learning in them. Through the use of these concepts, an intelligent fraud transaction detection system can be developed which is not only reliable but highly accurate as well.

Existing systems for fraud detection make use of pre-defined rules and patterns for identifying fraudulent transactions. Though this is an effective approach, it is not accurate and can report a high number of false

positives. Additionally, there is no prevention or reversal methodology in place with these existing systems. With the situation of credit card fraud becoming more alarming with each passing year, there is a need for a system that is more intelligent. Machine learning can enable a computer system to gather information and add to its set of pre-defined rules to make results more accurate and precise.

This project aims to understand and make use of machine learning in transaction management; identify existing works in literature for online credit card fraud detection; and make use of methodologies proposed in literature (with modifications as needed) to develop a computerized system for fraud transaction detection that can identify and prevent suspicious transactions in real-time i.e. as the transaction occurs. Through this project, the goal is not just to develop a fraud transaction detector but also to analyze how machine learning can help improve methodologies and systems for fraud detection to make them more efficient, reliable, and accurate.

Objectives:

To achieve the aims of the project highlighted in the section above, the following objectives are proposed:

- To implement an improved fraud transaction detection system, an analysis of known works in literature will be carried out. Authors such as Stolfo, J. have proposed solutions for credit-card fraud detection. [3] In this project, we will modify these solutions for use in the proposed online fraud transaction detector.
- A fraud transaction detector will be developed that will connect directly to the databases of credit card companies, banks, and/or other financial institutes. This system will make use of machine learning models such as an *Artificial Neural Network (ANN)* for detecting fraudulent transactions.
- For training the ANN model, training data consisting of safe and fraud transactions will be provided so that the system can learn how to classify transactions. This ANN model will be used to predict and identify fraud transactions in real time.
- Every time a transaction takes place, the fraud transaction detector will analyze the transaction against the ANN model to classify it as safe (in which case the transaction will hold) or classify it as fraud (in which case the transaction will be blocked, and human intervention will be required).
- A web-based and mobile-based interface will be provided to the fraud transaction detector that will allow users (such as the bank managers) to view all transactions that have been classified by the

system. There will be an option provided for filtering out ‘safe’ transactions and fraud transactions. Additionally, it is proposed that an option for unblocking or reversing fraud transactions will also be provided.

- An analysis, in the form of a dissertation, will be written to outline the performance of the fraud transaction detector. The focus of this dissertation will be on the use of artificial intelligence, particularly machine learning, in fraud detection systems. The dissertation will propose metrics and quantitatively analyze how machine learning can improve the accuracy of fraud detection systems in general.

CHAPTER 2: LITERATURE REVIEW

A number of authors have contributed to the literature of data science and machine learning for fraud detection, in general, and credit card fraud detection, in particular. The focus of this chapter is to analyze how machine learning and neural networks can be used for detecting fraud. The studies that have been chosen for this literature review propose solutions based on neural networks for the detection of fraudulent transactions among a pool of fraud and real transactions.

Fraud Detection Using Meta-Learning:

One of the earliest researches in the domain of credit card fraud was conducted by Stolfo. [3] In the paper by Stolfo, the authors comprehensively discuss the method of meta-learning for learning a model based on classified credit card transaction records and then using this model for predicting whether a transaction is fraudulent or real.

The system proposed by Stolfo is based on two key components – the first is a local fraud detection agent and the second is a meta-learning agent. As Stolfo highlights that standalone fraud detection systems employed by individual financial institutions are not sufficient to eradicate the issue of credit card fraud. Thus, a unified and global approach is required to solve this problem. Therefore, the system proposed by Stolfo consists of a local fraud detection agent which is responsible for detecting and preventing fraud within a single financial institution; and a meta-learning system which collects and integrates the information from individual local agents to provide a secure infrastructure for sharing knowledge about fraud transactions between financial institutions. [3]

The benefit of using such a system is that the knowledge of fraudulent transactions of individual financial institutions can be shared in a global domain without disclosing proprietary data of these institutions. This centralized and secure meta-learning system ensures that a unified solution for detecting and preventing fraudulent credit card transactions can be adopted. [3]

For the experiment, the authors make use of a dataset from the FSTC containing 500,000 records with 20% i.e. 100,000 of these transactions classified as fraud. To prepare the dataset for the experiment, the authors removed redundant fields and sampled from the large dataset to reduce the size of data for quicker learning. Furthermore, the authors divided the dataset into two parts i.e. the training dataset from the earlier months and the test dataset from the later months. For determining the accuracy of the learned model, the authors made use of a cost function that considers both the True and False Positive rates. [3]

The authors made use of BAYES, RIPPER, ID3, and CART for meta-learning and used a 50%/50% distribution of the fraud and non-fraud training dataset for learning the model i.e. the ratio of fraud transactions to non-fraud transactions was set to 1. The results of the experiment concluded that BAYES was the most accurate meta-learning algorithm and that the 50%/50% distribution resulted in the highest True Positive rate and lowest False Positive rate. [3]

Though, for this project, we are not concerned with the use of the BAYES algorithm, we can conclude that the use of a 50%/50% distribution as training data works well for classifying credit card transactions as fraud and non-fraud.

In the paper by Chan, with is co-authored by Stolfo, a similar meta-learning system is adopted for detecting credit card fraud. Chan emphasizes on the importance of non-uniform class and cost distributions for learning in the study. The authors highlight that skewed distributions which have a non-uniform cost per error are common in the real-world and credit card fraud detection is an example of such as distribution. The primary concern with datasets related to credit card fraud is that the number of fraudulent transactions are quite small as compared to the overall number of transactions. [4] Various authors in the past, including Fawcett [5] and Kubat [6] have discussed the degradation effect which skewed class distributions have on the learning. The authors recognize that a natural class distribution which is highly skewed will not yield the most effective results and classifiers.

The dataset which was used by Chan for the experiment contained 500,000 transactions out of which 20% were fraudulent transactions and 80% were legitimate transactions. The author

recognizes that in the real-world, the distribution is much more skewed and this 20:80 ratio is a result of the *cleaned* dataset provided by the financial institution. [4]

For learning from the dataset, there were certain steps taken by the author before applying the learning algorithms. The first step was to divide the dataset into two parts: the training dataset consisting of transactions from the first 10 months, and the testing dataset consisting of transactions from the 12^{th} month. The 20:80 skewed distribution is then divided into four subsets, each with a 50:50 distribution. This is done to reduce the skewness and ensure effective results from the learning process. After the division, the learning algorithm is applied on each of the smaller datasets and then the results are combined through meta-learning from the classification behavior. Chan makes use of BAYES, CART, RIPPER, and C4.5 as the learning algorithms for training the model for fraud detection. [4]

There are two primary benefits of dividing the original dataset into smaller datasets, as recognized by the author. Firstly, the smaller datasets have a 50:50 distribution i.e. they are less skewed than the original dataset and thus they yield more effective results. Secondly, the smaller datasets are independent of each other and thus, these can be run in parallel on different processors which can yield significant improvements in speed, particularly when dealing with very large datasets. [4]

The study conducted by Chan provides us with an effective approach for handling very large datasets which are highly skewed. For credit card fraud detection, we can make use of a multi-classifier meta-learning non-uniform cost technique as applied by Chan. [4]

Fraud Detection using Artificial Neural Networks:

Serrano [7] worked with a sample of tax data to detect frauds through the use of an ANN model, NRMSE (normalized-root-mean-square-error), and COD (curse of dimensionality). The aim of the study was to analyze the use of ANN models for detecting fraud in financial systems. The study highlighted how traditional fraud detection solutions are not applicable to such financial systems and that the ANN predictor is the best alternative. The author also highlights how ANN can be effective when the sample data available for training the model is small.

Cannady [8] focuses on the use of ANN for misuse detection in an IDS (intrusion detection system). The study highlights that while ANN offers potential, there is work which needs to be done in choosing the most effective and accurate neural network architecture. Furthermore, the study also highlights how an ANN model can be effective if there is direct input from the data source (which was the network data stream in the author's case). Though the author does not implement and deploy a full-scale neutral network system for the IDS, they recognize that neural networks show significant promise in the field.

Credit Card Fraud Detection using Artificial Neural Networks:

Gulati [9] discusses how credit cards are the most common mode of payment in both offline and online mediums. The author illustrates that the widespread use of these credit cards makes them a target for hackers and fraudsters which puts credit card transactions at the danger of being tampered with or otherwise altered. The author recognizes that the issue with current (traditional) fraud detection systems is that fraud transactions are detected only after the transaction has been completed after which it is very difficult to track, reverse, and prevent the transaction from taking effect. On the other hand, the author suggests that the use of an ANN-based fraud detection system can help identify fraud transactions while they are being processed (i.e. in real-time) to maximize effectiveness and minimize losses. The approach suggested by the author is to train an ANN model to consider the spending pattern of each cardholder (i.e. the way that they manage and make transactions) and then classify transactions which deviate significantly from these patterns as suspicious or fraud. Gulati [7] works with sample data from UCI Machine Learning Repository and makes use of Java to deploy the ANN. The author concludes that through the use of a fully-developed ANN-based fraud detection system, we can classify transactions with an accuracy of up to 80%. The only drawback that the author identifies with this approach is that new cardholders (who have no transactional history) will not have any input data for the ANN model and thus their transactions will not be classified.

Mishra [10] compares the various techniques that have been adopted for fraud detection systems and puts the focus on testing the accuracy of the BR, GDA, and LM techniques which are used commonly with ANN models. The data that the author uses consists of two different samples

which are obtained from the UCI Machine Learning repository. After testing the various techniques of ANN for fraud detection, the author concludes that the BR technique is the most accurate with accuracy scores of 98.745 and 99.01% in the two respective datasets while the GDA technique was the second most accurate and LM was the least accurate. The author also highlights that the accuracy of the BR technique improved as the size of the sample increased. This paper provides a useful insight into how the ANN model should be trained and what technique should be used with the ANN to drive the most accurate results for fraud detection.

Chapter 3: Requirements and Design

Description of the Dataset:

The dataset which we have chosen for this report consists of mobile money transactions which have been simulated from real transactions. The data has been contributed by E.A. Lopez-Rojas and it has been extracted from the logs of a mobile money service provider based in an African country. The PaySim mobile money service provider is a multinational company and it has its services running in 14 countries across the world. The dataset that we have used is a scaled down version which is available on Kaggle and it consists of 5,090,096 mobile money transactions. Out of these transactions, only 0.13% are classified as fraudulent transactions. Therefore, the dataset has a skewed distribution and we need to apply appropriate measures to refine and reorder the dataset. This needs to be done in order to ensure high accuracy and effectiveness of the learned model.

Requirements:

Why Neural Networks?

A question that arises here is: Amongst all the machine learning algorithms which include Decision Tree classifier and BAYES classifier, why are Neural Networks preferred for this dataset? For complex and large datasets in which there is no well-defined relationship between the various variables (i.e. attributes) of the data, Decision Tree classifier and BAYES classifier are not known to yield accurate results. Not just this, but it is increasingly difficult to structure and apply other classifiers to a problem such as this where we are looking to recognize a pattern and make use of this pattern for identifying and preventing fraudulent transactions from occurring in the future. In a real-life problem such as this where the relationships are non-linear and the dataset is skewed to favor a particular class, we can make use of the neural network to apply the necessary mathematical transformations such as the ReLU and Sigmoid function to learn an accurate model from the dataset. The accuracy and efficiency of neural networks in solving complex problems such as this is the primary reason why we have made use of this machine learning algorithm for our report.

Structure of the Neural Network:

In Chapter 1, we outlined the structure of a neural network which is used for learning models and classifiers for making predictions. Here we outline how we will make use of that structure to build our neural network model. For our experiment, the neural network will consist of 3 layers i.e. a simple perceptron. The first layer i.e. the input layer will consist of 9 neurons each of which will be mapped from 9 columns (i.e. features) found in the dataset. For the second layer i.e. the

hidden layer, we will be testing a variable number of neurons to determine the ideal configuration for optimum accuracy. Ultimately, we will be making use of up to 3 hidden layers I the neural network for learning the model from the dataset. The final layer i.e. the output layer will consist of a single neuron which will be used to determine whether a specific transaction is fraudulent or legitimate.

<<insert structure of neural network here>>

Activation Function: Rectified Linear Unit and Sigmoid:

For the activation function which will be used for transforming the inputs at neurons, we will be making use of Rectified Linear Unit (ReLU) and Sigmoid. We have described both of these functions briefly below.

Rectified Linear Unit (ReLU) is a ramp function which was introduced and demonstrated in 2011 due to its advantages over traditional activation functions such as logistic sigmoid. Unlike other activation functions, ReLU does not saturate since the gradient is always either 1 or 0 and it adds non-linearity to the network. ReLU learns faster than traditional activation functions but the drawback is that it does not activate for negative inputs and thus, all negative inputs are regarded as dead neurons with ReLU.

Sigmoid is a traditional activation function which is very commonly used for various kinds of neural networks. Though it does not learn as fast as ReLU, Sigmoid is an effective activation function that can activate both positive and negative inputs.

~~We will make use of these two activation functions in parallel in separate trials of the experiment to compare the accuracy of the learned model for the dataset.~~

Adam Optimizer:

For the choice of optimizer, we are making use of Adam. Adam is a stochastic optimization algorithm which makes use of first-order gradient-based optimization. The reason for choosing the Adam optimizer is because it has low memory requirements, is computationally efficient, and generally well suited for datasets that are large in size or has a high number of parameters. The Adam optimizer inherits properties from both the RMSProp optimizer and AdaGrad optimizer which are two of the best optimizers present.

It is important to mention here that the Keras library which we will be using for this experiment has built-in functionality for the activation function of ReLU and Sigmoid and the Adam Optimizer.

Data Refinement:

Commented [SM1]: Please review this and let me know if there are changes to be made in this.

Commented [PR2R1]: We use 9 features from dataset. And we need to increase the hidden layer size. In general, we can't get the desired result with this simple network architecture.
I am planning to use 2~3 layers for hidden layer.

Commented [SM3R1]: Acknowledged.

Commented [SM4]: Please verify.

Commented [PR5R4]: It's better to remove this sentence. We don't need to verify its usability because it was already proven activation function.

Commented [SM6R4]: Removed.

Commented [SM7]: Please let me know if this should be discussed in more detail.

Commented [PR8R7]: It's enough.
We don't need to describe it in detail.

Commented [SM9R7]: Acknowledged.

For reducing the learning time and improving the accuracy of the learned model, we will be removing redundant, duplicated, incomplete, and otherwise erroneous transactions from the dataset. In addition to this, since the size of the dataset is quite large i.e. 5,000,000+ transactions, we will be randomly sampling transactions from the dataset without replacement.

Commented [SM10]: Please verify.
Commented [PR11R10]: It's fine.
Commented [SM12R10]: Acknowledged.

Multi-Class Meta-Learning:

As with the real-world, the dataset which we are making use of for this experiment has a skewed distribution with only 0.13% of the transactions marked as fraud. Due to the dominance of legitimate transactions in the dataset, there is a high probability of false negatives i.e. fraudulent transactions being marked as legitimate transactions by the model. To diminish the effect of this skewness, we will make use of the multi-class meta-learning model which was proposed by the authors Stolfo and Chan as we discussed in Chapter 2. [3][4]

Ideally, we will be looking to divide the dataset into smaller datasets that have a 70:30 distribution i.e. 70% fraudulent transactions and 30% legitimate transactions. The reason for dividing the data as such is because our objective is to detect fraudulent transactions and thus, minimize the False Acceptance Ratio (FAR). A low FAR means that there are very few fraudulent transactions that are being marked as legitimate. An undetected fraudulent transaction can have a significant cost and therefore we need to minimize the probability of this happening. For this experiment, we can trade-off a high False Rejection Ratio (FRR) for a False Acceptance Ratio (FAR) since the cost of a fraudulent transaction being marked as a legitimate transaction is much higher than the cost of a legitimate transaction being marked as a fraudulent transaction.

~~Ideally, we will be looking to divide the dataset into smaller datasets that have a 50:50 distribution i.e. 50% fraudulent transactions and 50% legitimate transactions.~~

Commented [SM13]: Please suggest.
Commented [PR14R13]: No problem for 5:5
But I'd like to suggest 7:3 or 8:2 for fraudulent and genuine. In this way, we can reduce the FAR (false acceptance ratio). The main purpose of this project is to detect the fraud transaction. We can accept the result if our engine treats the genuine as fraud. But we have to detect every fraud transaction because it is very harmful. We have to try to reduce the FAR.
Commented [SM15R13]: Acknowledged and changed with reasons.

Accuracy Considerations and Cost Function:

Credit card fraud is a sensitive matter and a single set of fraudulent transactions can lead to loss worth millions of dollars for the bank, accountholders, and other stakeholders. For this reason, we cannot make use of a simple ratio between correct predictions over total predictions for the accuracy of the learned model. Instead, we have to evaluate the cost which is associated with a False Negative i.e. when a transaction is fraudulent, but it is marked as legitimate and with a False Positive i.e. when a transaction is legitimate, but it is marked as fraudulent. We will make use of a cost matrix for this purpose where we will evaluate the Precision, Recall, and F-Measure for the learned model to determine how accurate, consistent, and effective the predicted results are.

Commented [SM16]: Please review.
Commented [PR17R16]: I will update later.
Commented [SM18R16]: Acknowledged.