A Project

on

DESIGN AND IMPLEMENTATION OF AN APPLICATION FOR CREDIT CARD FRAUD DETECTION.

Submitted in partial fulfillment of the requirement for

the award of the degree of

Bachelor of Computer Application

Under The Supervision of

Dr. Rajnesh Singh

Submitted By

Tannu Sharma
21SCSE1040019

Prerna Singh
21SCSE1040020

Anu Sharma
21SCSE1040042

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

APRIL
2024

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"DESIGN AND IMPLEMENTATION OF AN APPLICATION FOR CREDIT CARD FRAUD DETECTION."** in partial fulfillment of the requirements for the award of the Bachelor of Computer Application submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the Jan 2024 to April 2024 , under the supervision of Dr. Rajnesh Singh Assistant Professor, Department of Computer Science and Engineering, Galgotias University, Greater Noida

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name
Designation

CERTIFICATE

The Final Project Dissertation Viva-Voce examination has been held on_____and her work is recommended for the award .

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:
Place: Greater Noida

ABSTRACT

Credit card plays a very important role in today's economy. It becomes an unavoidable part of household, business and global activities. Although using credit cards provides enormous benefits when used carefully and responsibly, significant credit and financial damages may be caused by fraudulent activities. The rapid growth in E-Commerce industry has lead to an exponential increase in the use of credit cards for online purchases and consequently they has been surge in the fraud related to it .In recent years, For banks has become very difficult for detecting the fraud in credit card system. Machine learning plays a vital role for detecting the credit card fraud in the transactions. For predicting these transactions banks make use of various machine learning methodologies, past data has been collected and new features are been used for enhancing the predictive power.

The performance of fraud detecting in credit card transactions is greatly affected by the sampling approach on data-set, selection of variables and detection techniques used. This project aims to propose an efficient approach that automatic detects fraud credit card related to insurance companies using deep learning algorithm called Autoencoders. The effectiveness of the proposed method has been proved in identifying fraud in actual data from transactions made by credit cards in September 2013 by European cardholders. In addition, a solution for data unbalancing is provided in this paper, which affects most current algorithms. The suggested solution relies on training for the autoencoder for the reconstruction normal data. Anomalies are detected by defining a reconstruction error threshold and considering the cases with a superior threshold as anomalies.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

Et al.	and others
ANN	Artificial Neural Network
AIS	Artificial Immune System
GA	Genetic Algorithm
HMM	Hidden Markov Model
SVM	Support Vector Machine
SQL	Structured Query Language
NSA	Negative Selection Algorithm
KNN	K Nearest Neighbour Algorithm
API	Application Programming Interface
AIN	Artificial immune Network
SOM	Self-Organizing Map
PCA	Principal Component Analysis
CA	Convolutional Autoencoder
VA	Variational Autoencoder
GP	Genetic Programming

CHAPTER 1
INTRODUCTION

Introduction:

A credit card is a thin handy plastic card that contains identification information such as a signature or picture, and authorizes the person named on it to charge purchases or services to his account - charges for which he will be billed periodically. They have a unique card number which is of utmost importance. Its security relies on the physical security of the plastic card as well as the privacy of the credit card number.

There is a rapid growth in the number of credit card transactions which has led to a substantial rise in fraudulent activities. Credit card fraud is a wide-ranging term for theft and fraud committed using a credit card as a fraudulent source of funds in a given transaction. Generally, statistical methods and many data mining algorithms are used to solve this fraud detection problem. Most of the credit card fraud detection systems are based on artificial intelligence, Meta learning and pattern matching.

Fraud detection is a binary classification problem in which the transaction data is analyzed and classified as "legitimate" or "fraudulent". Credit card fraud detection techniques are classified in two general categories: fraud analysis (misuse detection) and user behavior analysis (anomaly detection).

Motivation for Work:

At the current state of the world, financial organizations expand the availability of financial facilities by employing innovative services such as credit cards, Automated Teller Machines (ATM), internet and mobile banking services. Besides, along with the rapid advances of e-commerce, the use of credit cards has become a convenient and necessary part of financial life. Credit card is a payment card supplied to customers as a system of payment. There are lots of advantages in using credit cards such as:

- **Ease of purchase** Credit cards can make life easier. They allow customers to purchase on credit in arbitrary time, location and amount, without carrying the cash. Provide a convenient payment method for purchases made on the internet, over the telephone, through ATMs, etc.
- **Keep customer credit history** having a good credit history is often important in detecting loyal customers. This history is valuable not only for credit cards, but also for other financial services like loans, rental applications, or even some jobs. Lenders and issuers of credit mortgage companies, credit card companies, retail stores, and utility companies can review customer credit score and history to see how punctual and responsible customers are in paying back their debts.
- **Protection of Purchases** Credit cards may also offer customers additional protection if the purchased merchandise becomes lost, damaged, or stolen. Both the buyer's credit card statement and the company can confirm that the customer has bought if the original receipt is lost or stolen. In addition, some credit card companies provide insurance for large purchases.

In spite of all mentioned advantages, the problem of fraud is a serious issue In-banking services that threaten credit card transactions especially. Fraud is an intentional deception with the purpose of obtaining financial gain or causing loss by implicit or explicit trick. Fraud is a public law violation in which the fraudster gains an unlawful advantage or causes unlawful damage. The estimation of amount of damage made by fraud activities indicates that fraud costs a very considerable sum of money. Credit card fraud is increasing significantly with the development of modern technology resulting in the loss of billions of dollars worldwide each year. Statistics from the Internet Crime Complaint Center show that there has been a significant rising in reported fraud in last decade. Financial losses caused due to online fraud only in the US, was reported to be $3.4 billion in 2011.

Fraud detection involves identifying scarce fraud activities among numerous legitimate transactions as quickly as possible. Fraud detection methods are developing rapidly in order to adapt with new incoming fraudulent strategies across the world. But, development of new fraud detection techniques becomes more difficult due to the severe limitation of the ideas exchanged in fraud detection. On the other hand, fraud detection is essentially a rare event problem, which has been variously called outlier analysis, anomaly detection, exception mining, mining rare classes, mining imbalanced data etc. The number of fraudulent transactions is usually a very low fraction of the total transactions. Hence the task of detecting fraud transactions in an accurate and efficient manner is fairly difficult and challengeable. Therefore, development of efficient methods which can distinguish rare fraud activities from billions of legitimate transactions seems essential.

Problem Statement:

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

1.3.1 Tools and Technology Used

System configuration

This project can run on commodity hardware. We ran the entire project on an Intel 8th generation I5 processor with 8 GB Ram, 2GB Graphics Card. First part is the training phase which takes 20-25 mins of time and the second part is the testing part which only takes a few seconds to make predictions.

Hardware Requirements

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

Software requirements

- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.
- Operating System: Windows 7 and above or Linux based OS or MAC OS.

CHAPTER 2

LITERATURE SURVEY

Introduction

Illegal use of a credit card or its information without the knowledge of the owner is referred to as credit card fraud. Different credit card fraud tricks belong mainly to two groups of application and behavioral fraud. Application fraud takes place when fraudsters apply for new cards from banks or issuing companies using false or other's information.

Multiple applications may be submitted by one user with one set of user details (called duplication fraud) or different users with identical details (called identity fraud). Behavioral fraud, on the other hand, has four principal types: stolen/lost card, mail theft, counterfeit card and „card holder not present" fraud. Stolen/lost card fraud occurs when fraudsters steal credit card or get access to a lost card. Mail theft fraud occurs when the fraudster gets a credit card in mail or personal information from the bank before reaching the actual cardholder. In both counterfeit and „card holders not present" frauds, credit card details are obtained without the knowledge of card holders. In the former, remote transactions can be conducted using card details through mail, phone, or the Internet. In the latter, counterfeit cards are made based on card information. Based on statistical data stated in 2012, the high risk countries facing credit card fraud threat is illustrated in Fig.1. Ukraine has the most fraud rate with a staggering 19%, which is closely followed by Indonesia at 18.3% fraud rate. After these two, Yugoslavia with the rate of17.8% is the most risky country. The next highest fraud rate belongs to Malaysia (5.9%), Turkey (9%) and finally the United States. Other countries that are prone to credit card fraud with the rate below than 1% are not demonstrated in Fig 1.

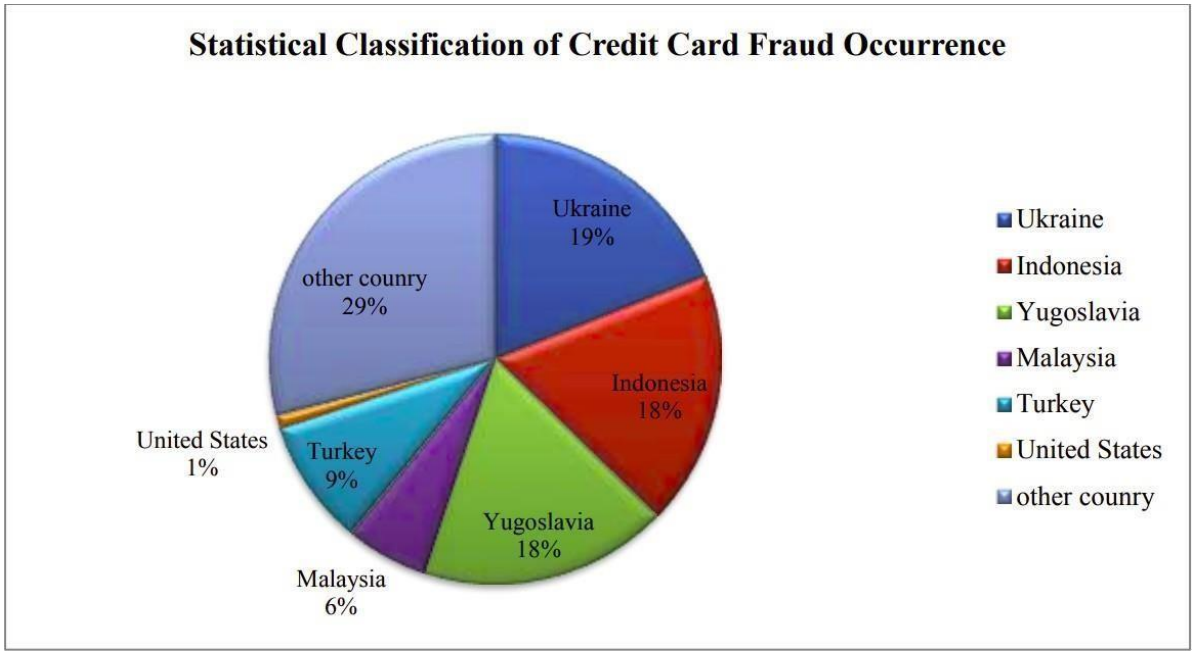

Fig 1. High risk countries facing credit card fraud threat

Difficulties of Credit Card Fraud Detection

Fraud detection systems are prune to several difficulties and challenges enumerated below. An effective fraud detection technique should have abilities to address these difficulties in order to achieve best performance.

- **Imbalanced data:** The credit card fraud detection data has an imbalanced nature. It means that very small percentages of all credit card transactions are fraudulent. This makes the detection of fraud transactions very difficult and imprecise.
- **Different misclassification importance:** in fraud detection tasks, different misclassification errors have different importance. Misclassification of a normal transaction as fraud is not as harmful as detecting a fraud transaction as normal. Because in the first case the mistake in classification will be identified in further investigations.
- **Overlapping data:** many transactions may be considered fraudulent, while actually they are normal (false positive) and reversely, a fraudulent transaction may also seem to be legitimate (false negative). Hence obtaining a low rate of false positives and false negatives is a key challenge of fraud detection systems.
- **Lack of adaptability:** classification algorithms are usually faced with the problem of detecting new types of normal or fraudulent patterns. The supervised and unsupervised fraud detection systems are inefficient in detecting new patterns of normal and fraud behaviors, respectively.
- **Fraud detection cost:** The system should take into account both the cost of fraudulent behavior that is detected and the cost of preventing it. For example, no revenue is obtained by stopping a fraudulent transaction of a few dollars.
- **Lack of standard metrics:** there is no standard evaluation criterion for assessing and comparing the results of fraud detection systems.

Credit Card Fraud Detection Techniques

The credit card fraud detection techniques are classified in two general categories: fraud analysis (misuse detection) and user behavior analysis (anomaly detection). The first group of techniques deals with supervised classification tasks at the transaction level. In these methods, transactions are labeled as fraudulent or normal based on previous historical data. This dataset is then used to create classification models which can predict the state (normal or fraud) of new records. There are numerous model creation methods for a typical two class classification task such as rule induction, decision trees and neural networks. This approach is proven to reliably detect most fraud tricks which have been observed before, also known as misuse detection.

The second approach deals with unsupervised methodologies which are based on account behavior. In this method a transaction is detected as fraudulent if it is in contrast with

The user's normal behavior. This is because we don't expect fraudsters behave the same as the account owner or be aware of the behavior model of the owner. To this aim, we need to extract the legitimate user behavioral model (e.. user profile) for each account and then detect fraudulent activities according to it. Comparing new behaviors with this model, different enough activities are distinguished as frauds. The profiles may contain the activity information of the account; such as merchant types, amount, location and time of transactions. This method is also known as anomaly detection.

It is important to highlight the key differences between user behavior analysis and fraud analysis approaches. The Fraud analysis method can detect known fraud tricks, with a low false positive rate. These systems extract the signature and model of fraud tricks presented in oracle dataset and can then easily determine exactly which frauds, the system is currently experiencing. If the test data does not contain any fraud signatures, no alarm is raised. Thus, the false positive rate can be reduced extremely. However, since learning of a fraud analysis system (i.e. classifier) is based on limited and specific fraud records, it cannot detect novel frauds. As a result, the false negative rate may be extremely high depending on how ingenious the fraudsters. User behavior analysis, on the other hand, greatly addresses the problem of detecting novel frauds. These Methods do not search for specific fraud patterns, but rather compare incoming activities with the constructed model of legitimate user behavior. Any activity that is sufficiently different from the model will be considered as a possible fraud. Though user behavior analysis approaches are powerful in detecting innovative frauds, they really suffer from high rates of false alarm. Moreover, if a fraud occurs during the training phase, this fraudulent behavior will be entered in baseline mode and is assumed to be normal in further analysis. In this section we will briefly introduce some current fraud detection techniques which are applied to credit card fraud detection tasks.

Artificial Neural Network

An artificial neural network (ANN) is a set of interconnected nodes designed to imitate the functioning of the human brain. Each node has a weighted connection to several other nodes in adjacent layers. Individual nodes take the input received from connected nodes and use the weights 6 together with a simple function to compute output values. Neural networks come in many shapes and architectures. The Neural network architecture, including the number of hidden layers, the number of nodes within a specific hidden layer and their connectivity, must be specified by the user based on the complexity of the problem. ANNs can be configured by supervised, unsupervised or hybrid learning methods.

Supervised techniques

In supervised learning, samples of both fraudulent and non-fraudulent records, associated with their labels are used to create models. These techniques are often used in fraud analysis approach. One of the most popular supervised neural networks is back propagation network (BPN). It minimizes the objective function using a multi-stage dynamic optimization method that is a generalization of the delta rule.The back propagation method is

Often useful for feed-forward network with no feedback. The BPN algorithm is usually time-consuming and parameters like the number of hidden neurons and learning rate of delta rules require extensive tuning and training to achieve the best performance. In the domain of fraud detection, supervised neural networks like back-propagation are known as efficient tools that have numerous applications.

Raghavendra Patidar, used a dataset to train a three layers backpropagation neural network in combination with genetic algorithms (GA) for credit card fraud detection. In this work, genetic algorithms were responsible for making decisions about the network architecture, dealing with the network topology, number of hidden layers and number of nodes in each layer.

Also, Aleskerov developed a neural network based data mining system for credit card fraud detection. The proposed system (CARDWATCH) had three layers of auto associative architectures. They used a set of synthetic data for training and testing the system. The reported results show very successful fraud detection rates.

In, a P-RCE neural network was applied for credit card fraud Detection-RCE is a type of radial-basis function networks that usually applied for pattern recognition tasks. Kroenke proposed a model for real time fraud detection based on bidirectional neural networks. They used a large data set of cell phone transactions provided by a credit card company. It was claimed that the system outperforms the rule based algorithms in terms of false positive rate.

Again in a parallel granular neural network (GNN) is proposed to speed up data mining and knowledge discovery process for credit card fraud detection. GEN is a kind of fuzzy neural network based on knowledge discovery (FNNKD).The underlying dataset was extracted from SQL server database containing sample Visa Card transactions and then preprocessed for applying in fraud detection. They obtained less average training errors in the presence of larger training dataset.

Unsupervised techniques

The unsupervised techniques do not need the previous knowledge of fraudulent and normal records. These methods raise alarm for those transactions that are most dissimilar from the normal ones. These techniques are often used in user behavior approach. ANNs can produce acceptable results for enough large transaction dataset. They need a long training dataset. Self-Organizing map (SOM) is one of the most popular unsupervised neural networks learning which was introduced by SOM provides a clustering method, which is appropriate for constructing and analyzing customer profiles, in credit card fraud detection, as suggested in SOM operates in two phases: training and mapping. In the former phase, the map is built and weights of the neurons are 7 updated iteratively, based on input samples, in latter, test data is classified automatically into normal and fraudulent classes through the procedure of mapping. As stated in after training the SOM, new unseen transactions are compared to normal and fraud clusters, if it is similar to all normal records, it is classified as normal. New fraud transactions are also detected similarly.

One of the advantages of using unsupervised neural networks over similar techniques is that these methods can learn from data streams. The more data passed to a SOM

Model, the more adaptation and improvement on result is obtained. More specifically, the SOM adapts its model as time passes. Therefore it can be used and updated online in banks or other financial corporations. As a result, the fraudulent use of a card can be detected fast and effectively. However, neural networks have some drawbacks and difficulties which are mainly related to specifying suitable architecture on one hand and excessive training required for reaching the best performance on the other hand.

Hybrid supervised and unsupervised techniques

In addition to supervised and unsupervised learning models of neural networks, some researchers have applied hybrid models. John ZhongLei proposed hybrid supervised (SICLN) and unsupervised (ICLN) learning networks for credit card fraud detection. They improved the reward only rule of SICLN model to ICLN in order to update weights according to both reward and penalty. This improvement appeared in terms of increasing stability and reducing the training time. Moreover, the number of final clusters of the ICLN is independent from the number of initial network neurons. As a result the inoperable neurons can be omitted from the clusters by applying the penalty rule. The results indicated that both the ICLN and the SICLN have high performance, but the SICLN outperforms well-known unsupervised clustering algorithms.

Artificial Immune System (AIS)

The natural immune system is a highly complex system, consisting of an intricate network of specialized tissues, organs, cells and chemical molecules. These elements are interrelated and act in a highly co- ordinate and specific manner when they recognize, remember disease causing foreign cells and eliminate them. Any element that could be recognized by the immune system is named an antigen. The immune system's detectors are the antibodies that are capable of recognizing and destroying harmful and risky antigens.

The immune system consists of the two main responses of immune and defense: innate immune response and acquired immune response. The body's first response for defense is made of the outer, unbroken skin and the „mucous membranes" lining internal channels, such as the respiratory and digestive tracts. If the harmful cells could pass through innate immune defense the acquired immunity will defense. In fact, adaptive immune response performs based on antigen-specific recognition of almost unlimited types of infectious substances, even if previously unseen or mutated. It is worth mentioning that the acquired immune response is capable of "remembering" every infection, so that a second exposure to the same pathogen is dealt with more efficiently.

There are two organs responsible for the generation and development of immune cells: the bone marrow and the thymus. The bone marrow is the site where all blood cells are generated and where some of them are developed. The thymus is the organ to which a class of immune cells called T-cells migrates and maturates. There exist a great number of different immune cells, but lymphocytes (white blood cells), are the prevailing ones. Their main function is distinguishing self-cells, which are the human body cells, from non-self- cells, the dangerous foreign cells (the pathogens). Lymphocytes are classified into two main types: B-cells and T-cells, both originating in the bone marrow. Those lymphocytes that

Develop within the bone marrow are called B-cells, and 8 those that migrate to and develop within the thymus (the organ which is located behind the breastbone) are named T-cells.
Artificial Immune System (AIS) is a recent sub field based on the biological metaphor of the immune system. The immune system can distinguish between self and non-self-cells, or more specifically, between harmful cells (called as pathogens) and other cells. The ability to recognize differences in patterns and being able to detect and eliminate infections precisely has attracted the engineer's intention in all fields.

Researchers have used the concepts of immunology in order to develop a set of algorithms, such as negative selection algorithm, immune networks algorithm, clonal selection algorithm, and the dendritic cells algorithm.

Negative Selection:

Negative Selection Algorithm or NSA proposed by is a change detection algorithm based on the T-Cells generation process of the biological immune system. It is one of the earliest AIS algorithms applied in various real-world applications. Since it was first conceived, it has attracted many researchers and practitioners in AIS and has gone through some phenomenal evolution. NSA has two stages: generation and detection. In the generation stage, the detectors are generated by some random process and censored by trying to match self-samples. Those candidates that match (by affinity of higher than affinity threshold) are eliminated and the rest are kept as detectors. In the detection stage, the collection of detectors (or detector set) is used in checking whether an incoming data instance is self or non-self. If it matches (by affinity of higher than affinity threshold) any detector, it is claimed as non-self or anomaly.

Brabazon proposed an AIS based model for online credit card fraud detection. Three AIS algorithms were implemented and their performance was standardized against a logistic regression model. Their three chosen algorithms were the unmodified negative selection Algorithm, the modified negative selection algorithm and the Clonal selection algorithm. They proposed the Distance Value Metric for calculating distance between records. This metric is based on the probability of data occurrence in the training set. Where the detection rate increased, but the number of false alarms and missed frauds remained.

Clonal selection:

Clonal selection theory is used by the immune system to explain the basic features of an immune response to an antigenic stimulus. The selection mechanism guarantees that only those clones (antibodies) with higher affinity for the encountered antigen will survive. On the basis of the clonal selection principle, clonal selection algorithm was initially proposed in and formally explained in the general algorithm was called CLONALG.

Gadi in applied the AIRS in fraud detection on credit card transactions. AIRS is a classification algorithm that is based on AIS which applies clonal selection to create detectors. AIRS generate detectors for all of the classes in the database and in detection stage uses k Nearest Neighbor algorithm (also called K-NN) in order to classify eachrecord.They

Compared their method with other methods like the neural networks, Bayesian networks, and decision trees and claimed that, after improving the input parameters for all the methods, AIRS has shown the best results of all, partly perhaps since the number of input parameters for AIRS is comparatively high. If we consider a 9 particular training dataset, and set the parameters depending on the same database, the results indicate a tendency to improve. The experiment was carried out on the Weka package. Soltani in proposed AIRS on credit card fraud detection. Since AIRS has a long training time, authors have implemented the model in the Cloud Computing environment to shorten this time. They had used MapReduce API which works based on the Hadoop distributed file system, and runs the algorithm in parallel.

Immune Network:

The natural immune system is applied through the interactions between a huge numbers of different types of cells. Instead of using a central coordinator, the natural design a machine learning model to detect credit card fraud using the data of past credit card data. Immune systems sustain the appropriate level of immune responses by maintaining the equilibrium status between antibody suppression and stimulation using duotypes and paratopes antibodies. The first Artificial Immune Network (AIN) proposed by Neal M introduced the AISFD, which adopted the techniques developed by CBR (case based reasoning) community and applied various methods borrowed from genetic algorithms and other techniques to clone the B cells (network nodes) for mortgage fraud detection.

Danger Theory:

The novel immune theory, named Danger Theory was proposed in 1994. It embarked from the concept that defined "self-non-self" in the traditional theories and emphasizes that the immune system does not respond to "non-self" but to danger. According to the theory a useful evolutionarily immune system should focus on those things that are foreign and dangerous, rather than on those that are simply foreign. Danger is measured by damage inflicted to cells indicated by distress signals emitted when cells go through an unnatural death (necrosis).

Dendritic cells (DCs), part of the innate immune system, interact with antigens derived from the host tissue; therefore, the algorithm inspired by Danger Theory is named dendritic cell algorithm. Dendritic cells control the state of adaptive immune system cells by emitting the following signals:

- PAMP (pathogen associated molecular pattern)
- Danger
- Safe
- Inflammation

PAMP is released from tissue cells following sudden necrotic cell death; actually, the presence of PAMP usually indicates an anomalous situation.

The presence of Danger signals may or may not indicate an anomalous situation; however the probability of an anomaly is higher than the same, under normal circumstances Safe signals act as an indicator of healthy tissue.

Inflammation signal is classed as the molecules of an inflammatory response to tissue injury. In fact, the presence of this signal amplifies the above three signals.

DCs exist in a number of different states of maturity, depending on the type of environmental signal present in the surrounding fluid. They can exist in immature, Semi-mature or mature forms. Initially, when a DC enters the tissue, it exists in an immature state. DCs which have the ability to present both the antigen and active T-cells are mature. For an immature DC to become mature it should be exposed to PAMP and danger signals predominantly. The immature DCs exposed to safe signals predominantly are termed "Semi-mature"; they produce semi-mature DCs output signaling molecules, which have the ability to de-activate the T-cells. Exposure to PAMP, danger and safe signals lead to an increase in co-stimulatory molecules production, which in turn ends up in removal from the tissue and its migration to local lymph nodes.

Hybrid AIS or methods

Some researchers applied different algorithms (i.e. vaccination algorithm, CART and so on) by AIS algorithm which are presented below:

Wong presents the AISCCFD prototype proposed to measure and manage the memory population and mutate detectors in real time. In their work both the two algorithms the vaccination and negative selection were combined. The results were tested for different fraud types. The proposed method demonstrated higher detection rates when the vaccination algorithm was applied, but it failed to detect some types of fraud precisely.

Huang presented a novel hybrid Artificial Immune inspired model for fraud detection by combining triple algorithms: CSPRA, the dendritic cell algorithm (DCA), and CART. Though their proposed method had high detection rate and low false alarm, their approach was focused on logging data and limited to VoD (video on demand) systems and not credit card transactions. Ayara applied AIS to predict ATM failures. Their approach is enriched by adding a generation of new antibodies from the antigens that correspond to the unpredicted failures.

Genetic Algorithm (GA)

Inspired from natural evolution, Genetic algorithms (GA), were originally introduced by John Holland. GA searches for optimum solutions with a population of candidate solutions that are traditionally represented in the form of binary strings called chromosomes.

The basic idea is that the stronger members of the population have more chances to survive and reproduce. The strength of a solution is its capability to solve the underlying problem which is indicated by fitness. New generation is selected in proportion to fitness among the previous population and newly created offspring. Normally, new offspring will be produced by applying genetic operators such as mutation and crossing over on some fitter

Members of the current generation (parents). As generations progress, the solutions are evolved and the average fitness of the population increases. This process is repeated until some stopping criteria, (i.e. often passing a pre-specified number of generations) is satisfied. Genetic Programming (GP) is an extension of genetic algorithms that represent each individual by a tree rather than a bit string. Due to the hierarchy nature of the tree, GP can produce various types of models such as mathematical functions, logical and arithmetic expressions, computer programs, networks structures, etc.

Genetic algorithms have been used in data mining tasks mainly for feature selection. It is also widely used in combination with other algorithms for parameter tuning and optimization. Due to the availability of genetic algorithm code in different programming languages, it is a popular and strong algorithm in credit card fraud detection. However, GA is very expensive in terms of time and memory. Genetic programming also has various applications in data mining as classification tools.

EkremDuman developed a method for credit card fraud detection. They defined a cost sensitive objective function that assigned different costs to different misclassification errors (e.g. false positive, false negative). In this case, the goal of a classifier will be the minimization of overall cost instead of the number of misclassified transactions.

This is due the fact that the correct classification of some transactions was more important than others. The utilized classifier in this work was a novel combination of the genetic algorithms and the scatter search. For evaluating the proposed method, it was applied to real data and showed promising results in comparison to literature. Analyzing the influence of the features in detecting fraud indicated that statistics of the popular and unpopular regions for a credit card holder is the most important feature. Authors excluded some types of features such as the MCC and country statistics from their study that resulted in less generality for typical fraud detection problems.

K.RamaKalyani presented a model of credit card fraud detection based on the principles of genetic algorithm. The goal of the approach was first developing a synthesizer algorithm for generating test data and then to detect fraudulent transactions with the proposed algorithm.

Bentley developed a genetic programming based fuzzy system to extract rules for classifying data tested on real home insurance claims and credit card transactions.

In authors applied Genetic Programming to the prediction of the price in the stock market of Japan. The objective of the work was to make decisions in the stock market about the best stocks as well as the time and amount of stocks to sell or buy. The experimental results showed the superior performance of GP over neural networks.

Hidden Markov Model (HMM)

A Hidden Markov Model is a double embedded stochastic process which is applied to model much more complicated stochastic processes as compared to a traditional Markov model. The underlying system is assumed to be a Markov process with unobserved states. In simpler Markov models like Markov chains, states with definite transition

Probabilities are only unknown parameters. In contrast, the states of a HMM are hidden, but state dependent outputs are visible.

In credit card fraud detection a HMM is trained for modeling the normal behavior encoded in user profiles. According to this model, a new incoming transaction will be classified to fraud if it is not accepted by model with sufficiently high probability. Each user profile contains a set of information about last 10 transactions of that user like time; category and amount of for each transaction. HMM produces high false positive rate.V. Bhusari utilized HMM for detecting credit card frauds with low false alarm. The proposed system was also scalable for processing a huge number of transactions.

HMM can also be embedded in online fraud detection systems which receive transaction details and verify whether it is normal or fraudulent. If the system confirms the transaction to be malicious, an alarm is raised and the related bank rejects that transaction. The responding cardholder may then be informed about possible card misuse.

Support Vector Machine (SVM)

Support vector machine (SVM) is a supervised learning model with associated learning algorithms that can analyze and recognize patterns for classification and regression tasks. SVM is a binary classifier. The basic idea of SVM was to find an optimal hyper- plane which can separate instances of two given classes, linearly. This
Hyper-plane was assumed to be located in the gap between some marginal instances called support vectors. Introducing the kernel functions, the idea 12 was extended for linearly inseparable data. A kernel function represents the dot product of projections of two data points in a high dimensional space. It is a transform that disperses data by mapping from the input space to a new space (feature space) in which the instances are more likely to be linearly separable. Kernels, such as radial basis function (RBF), can be used to learn complex input spaces. In classification tasks, given a set of training instances, marked with the label of the associated class, the SVM training algorithm finds a hyper-plane that can assign new incoming instances into one of two classes. The class prediction of each new data point is based on which side of the hyper-plane it falls on feature space.

SVM has been successfully applied to a broad range of applications such as
In credit card fraud detection, Ghosh and Reilly developed a model using SVMs and admired neural networks. In this research a three layer feed-forward RBF neural network applied for detecting fraudulent credit card transactions through only two passes required to churn out a fraud score in every two hours.

Tung-shou Chen proposed a binary support vector system (BSVS), in which support vectors were selected by means of the genetic algorithms (GA). In the proposed model self-organizing map (SOM) was first applied to obtain a high true negative rate and BSVS was then used to better train the data according to their distribution.

In a classification model based on decision trees and support vector machines (SVM) was constructed respectively for detecting credit card fraud. The first comparative study among SVM and decision tree methods in credit card fraud detection with

A real data set was performed in this paper. The results revealed that the decision tree classifiers such as CART outperform SVM in solving the problem under investigation. Rongchang Chen suggested a novel questionnaire-responder transaction (QRT) approach with SVM for credit card fraud detection. The objective of this research was the usage of SVM as well as other approaches such as Over-sampling and majority voting for investigating the prediction accuracy of their method in fraud detection. The experimental results indicated that the QRT approach has a high degree of efficiency in terms of prediction accuracy.

Qibei Lu established a credit card fraud detection model based on Class Weighted SVM. Employing Principal Component Analysis (PCA), they initially reduced data dimension to less synthetic composite features due to the high dimensionality of data. Then according to imbalance characteristics of data, an improved Imbalance Class Weighted SVM (ICW-SVM) was proposed.

Bayesian Network

A Bayesian network is a graphical model that represents conditional dependencies among random variables. The underlying graphical model is in the form of directed acyclic graphs. Bayesian networks are useful for finding unknown probabilities given known probabilities in the presence of uncertainty. Bayesian networks can play an important and effective role in modeling situations where some basic information is already known but incoming data is uncertain or partially unavailable. The goal of using Bayes rules is often the prediction of the class label associated to a given vector of features or attributes. Bayesian networks have been successfully applied to various fields of interest for instance churn prevention in business, pattern recognition in vision, generation of diagnostic in medicine and fault diagnosis as well as forecasting in power systems. Besides, these networks have also been used to detect anomaly and frauds in credit card transactions or telecommunication networks.

In two approaches are suggested for credit card fraud detection using Bayesian network. In the first, the fraudulent user behavior and in the second the legitimate (normal) user behavior are 13 modeled by Bayesian network. The fraudulent behavior net is constructed from expert knowledge, while the legitimate net is set up in respect to available data from non-fraudulent users. During operation, legitimate net is adapted to a specific user based on emerging data. Classification of new transactions were simply conducted by inserting it to both networks and then specifying the type of behavior (legitimate/fraud) according to corresponding probabilities. Applying Bayes rule, gives the probability of fraud for new transactions. Again, Ezawa and Norton developed a four-stage Bayesian network. They claimed that lots of popular methods such as regression, K-nearest neighbor and neural networks take too long a time to be applicable in their data.

System Architecture

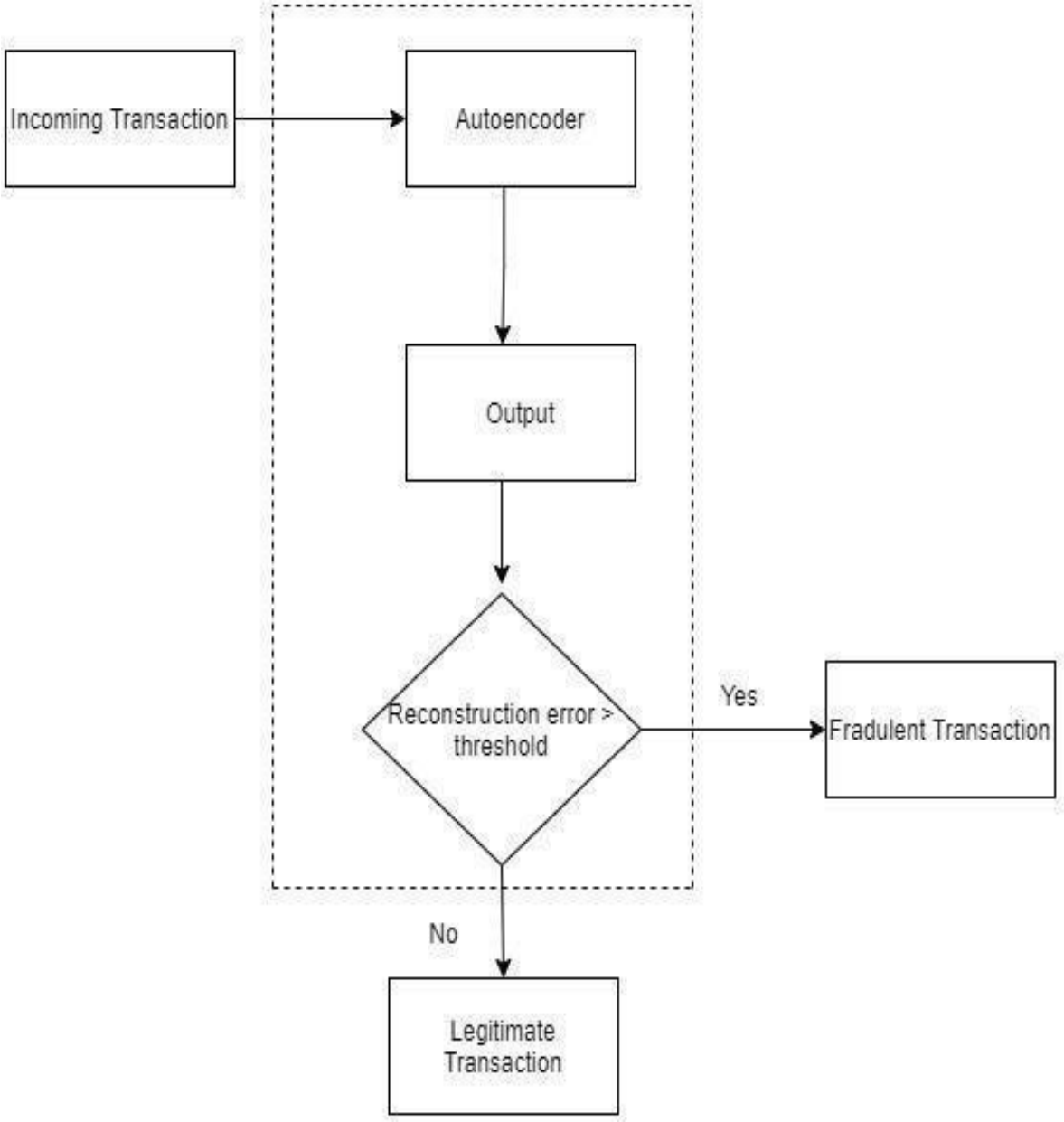

Fig 2. System Architecture

CHAPTER 3
METHODOLOGY ADOPTED

Proposed Systems:

The model needs to classify the incoming transactions into fraudulent or normal transactions. There are several methods to build a binary classifier. We are proposing to use Autoencoder which are unsupervised learning model which reconstructs the compressed input for better classification and reduce the noise in the input data.

Autoencoders:

Autoencoders are neural networks. Neural networks are composed of multiple layers, and the defining aspect of an autoencoder is that the input layers contain exactly as much information as the output layer. The reason that the input layer and output layer have the exact same number of units is that an autoencoder aims to replicate the input data. It outputs a copy of the data after analyzing it and reconstructing it in an unsupervised fashion.

The data that moves through an autoencoder isn't just mapped straight from input to output, meaning that the network doesn't just copy the input data. There are three components to an autoencoder: an encoding (input) portion that compresses the data, a component that handles the compressed data (or bottleneck), and a decoder (output) portion. When data is fed into an autoencoder, it is encoded and then compressed down to a smaller size. The network is then trained on the encoded/compressed data and it outputs a recreation of that data.

The autoencoders reconstruct each dimension of the input by passing it through the network. It may seem trivial to use a neural network for the purpose of replicating the input, but during the replication process, the size of the input is reduced into its smaller representation. The middle layers of the neural network have a fewer number of units as compared to that of input or output layers. Therefore, the middle layers hold the reduced representation of the input. The output is reconstructed from this reduced representation of the input.

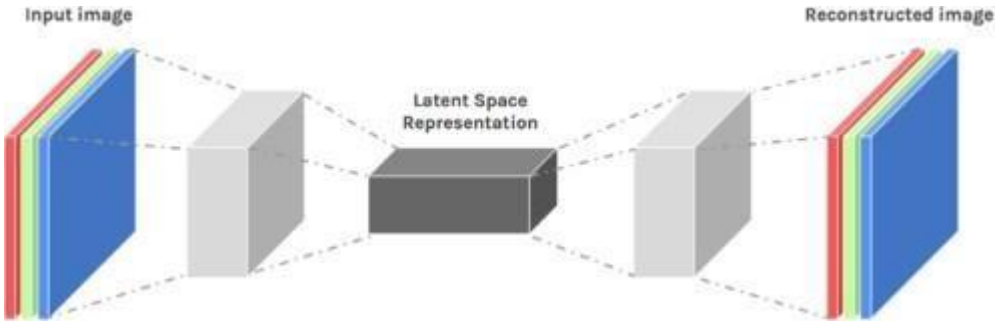

Fig 3. Autoencoder

We have a similar machine learning algorithm ie. PCA which does the same task. Autoencoders are preferred over PCA because:

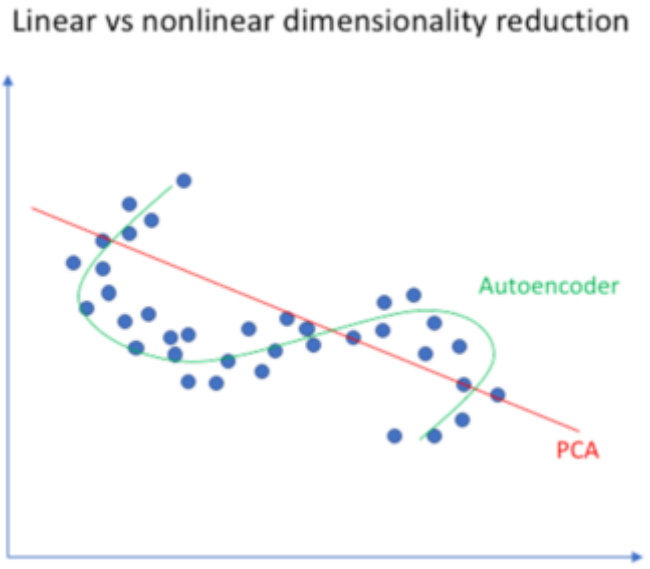

Fig 4. Autoencoder vs PCA

- An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers.
- It doesn't have to learn dense layers. It can use convolutional layers to learn which is better for video, image and series data.
- It is more efficient to learn several layers with an autoencoder rather than learn one huge transformation with PCA.
- An autoencoder provides a representation of each layer as the output.
- It can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder.

Autoencoder Architecture

An autoencoder can essentially be divided up into three different components: the encoder, a bottleneck, and the decoder.

The encoder portion of the autoencoder is typically a feedforward, densely connected network. The purpose of the encoding layers is to take the input data and compress it into a latent space representation, generating a new representation of the data that has reduced dimensionality.

The code layers, or the bottleneck, deal with the compressed representation of the data. The bottleneck code is carefully designed to determine the most relevant portions of the observed data, or to put that another way the features of the data that are most important for data reconstruction. The goal here is to determine which aspects of the data need to be preserved and which can be discarded. The bottleneck code needs to balance two different considerations: representation size (how compact the representation is) and variable/feature

relevance. The bottleneck performs element-wise activation on the weights and biases of the network. The bottleneck layer is also sometimes called a latent representation or latent variables.

The decoder layer is what is responsible for taking the compressed data and converting it back into a representation with the same dimensions as the original, unaltered data. The conversion is done with the latent space representation that was created by the encoder.

The most basic architecture of an autoencoder is a feed-forward architecture, with a structure much like a single layer perceptron used in multilayer perceptrons. Much like regular feed-forward neural networks, the auto-encoder is trained through the use of backpropagation.

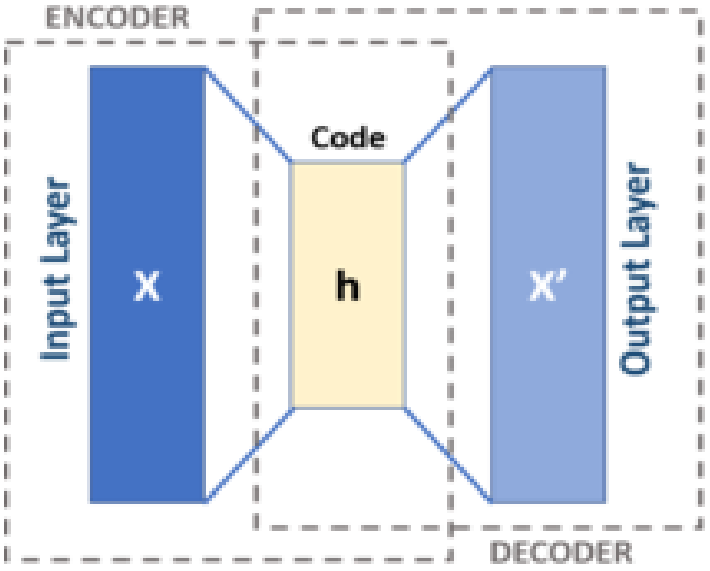

Fig 5 .Autoencoder Architecture [93]

The simplest form of an autoencoder is a feedforward, non-recurrent neural network similar to single layer perceptron's that participate in multilayer perceptron's (MLP) – employing an input layer and an output layer connected by one or more hidden layers. The output layer has the same number of nodes (neurons) as the input layer. Its purpose is to reconstruct its inputs (minimizing the difference between the input and the output) instead of predicting a target value **Y** given inputs **X** . Therefore, autoencoders are unsupervised learning models. (They do not require labeled inputs to enable learning).

An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and Ψ, such that:

$\phi: X \rightarrow F$

$\psi: F \rightarrow X$

$$\phi, \psi = \underset{\phi,\psi}{arg\,min} ||X - (\psi \circ \phi)X||^2$$

An Autoencoder consist of three layers:

1. Encoder
2. Code
3. Decoder

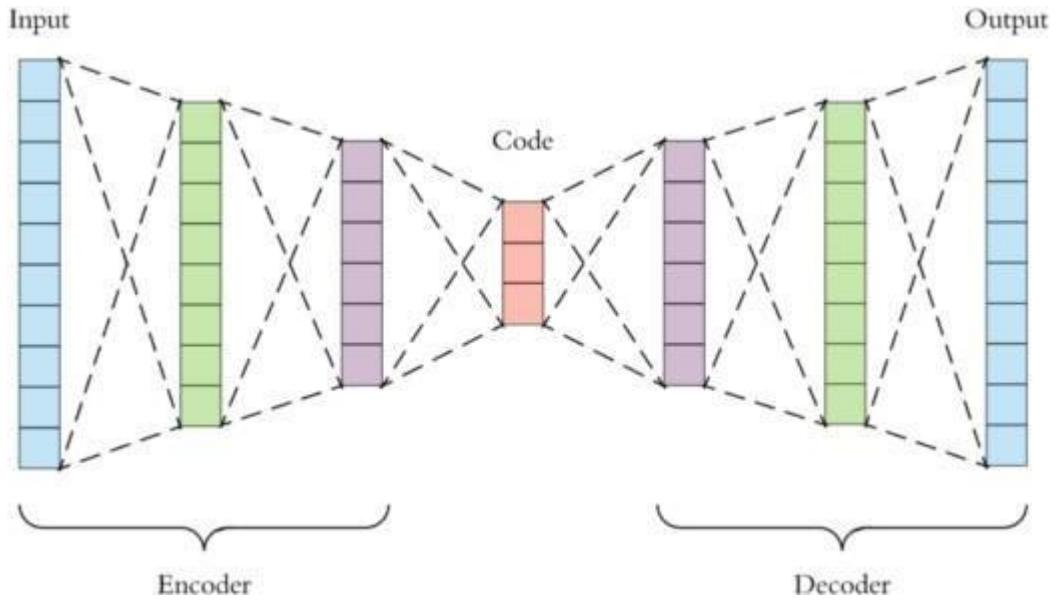

Fig 6: Encoder and decoder

- **Encoder:** This part of the network compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
- **Code:** This part of the network represents the compressed input which is fed to the decoder.
- **Decoder:** This layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.

Application of autoencoders

Data Compression

Although autoencoders are designed for data compression, they are hardly used for this purpose in practical situations. The reasons are:

- **Lossy compression**: The output of the autoencoder is not exactly the same as the input, it is a close but degraded representation. For lossless compression, they are not the way to go.
- **Data-specific**: Autoencoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different from a standard data compression algorithm like jpeg or gzip. Hence, we can't expect an autoencoder trained on handwritten digits to compress landscape photos.

Since we have more efficient and simple algorithms like jpeg, LZMA, LZSS(used in WinRAR in tandem with Huffman coding), autoencoders are not generally used for compression. Although autoencoders have seen their use for image denoising and dimensionality reduction in recent years.

Image Denoising

Autoencoders are very good at denoising images. When an image gets corrupted or there is a bit of noise in it, we call this image a noisy image.To obtain proper information about the content of the image, we perform image denoising.

Dimensionality Reduction

The autoencoders convert the input into a reduced representation which is stored in the middle layer called code. This is where the information from the input has been compressed and by extracting this layer from the model, each node can now be treated as a variable. Thus we can conclude that by trashing out the decoder part, an autoencoder can be used for dimensionality reduction with the output being the code layer.

Feature Extraction

Encoding part of Autoencoders helps to learn important hidden features present in the input data, in the process to reduce the reconstruction error. During encoding, a new set of combinations of original features is generated.

Image Generation

Variational Autoencoder(VAE) discussed above is a Generative Model, used to generate images that have not been seen by the model yet. The idea is that given input images like images of face or scenery, the system will generate similar images. The use is to:

generate new characters of animation
generate fake human images

Image colourisation

One of the applications of autoencoders is to convert a black and white picture

CHAPTER 4

WORKING ON PROJECT

Modules with Sample Code

- Data Loading
- Class wise Analysis
- Data Modelling
- Model Training
- Model Evaluation
- Web App

Data Loading

The dataset was obtained from Kaggle. It contains 28 attributes,which have been scaled and modified.However their description has not been given.
The only attributes known to us are
1.Amount
2. Time
3. Output class[0 for a normal transaction and 1 for a fraudulent transaction].
The dataset contains float data values for every class except the Output class which is of int. The data from the dataset in csv format was loaded into the dataframe of Pandas Python package.Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

```
#Data Loading...

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

df=pd.read_csv('/content/drive/MyDrive/creditcard.csv')

df.head()

df.info()

df.shape
```

Class Wise Analysis

The Output class with 0 is a normal transaction and 1 is a fraudulent transaction.
The total dataset has 284807 rows and 31 columns. Out of these, the normal transactions were 284315 and fraudulent transactions were 492.
The distribution of fraudulent points in the dataset accounts to 0.17% of the total dataset.
The number of normal and fraudulent transactions were plotted as bar graphs using MatPlotlib Python library.
As Time and Amount are the only known attributes, we plotted the graphs with relations between Time and Amount among fraudulent and normal transactions.

```
"""###Class Wise Analysis
"""
normal = df[df['Class']==0]
fraud = df[df['Class']==1]
print("Normal DataPoints: ",normal.shape[0])
print("Fraud DataPoints: ", fraud.shape[0])
print(("Distribution of fraudulent points:
{:.2f}%".format(len(df[df['Class']==1])/len(df)*100)))
sns.countplot(df['Class'])
plt.title('Class Distribution')
plt.xticks(range(2),['Normal','Fraud'])
plt.show()
```

```
[6] normal = df[df['Class']==0]
    fraud  = df[df['Class']==1]
    print("Normal DataPoints: ",normal.shape[0])
    print("Fraud DataPoints:  ", fraud.shape[0])

    Normal DataPoints:  284315
    Fraud DataPoints:   492
```

```
print(("Distribution of fraudulent points: {:.2f}%".format(len(df[df['Class']==1])/len(df)*100)))
sns.countplot(df['Class'])
plt.title('Class Distribution')
plt.xticks(range(2),['Normal','Fraud'])
plt.show()
```

```
Distribution of fraudulent points: 0.17%
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following
  FutureWarning
```

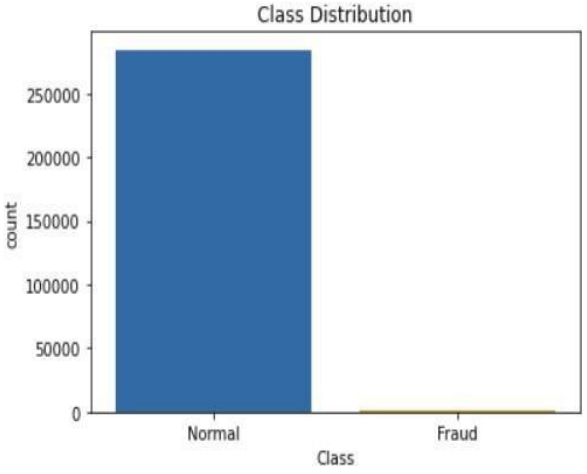

Fig7. Class Wise Analysis

```
df.describe()
normal['Amount'].describe()
fraud['Amount'].describe()
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize = (10,10) )
f.suptitle('Amount per transaction by class')
bins = 10
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
ax1.grid()
ax2.grid()
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```

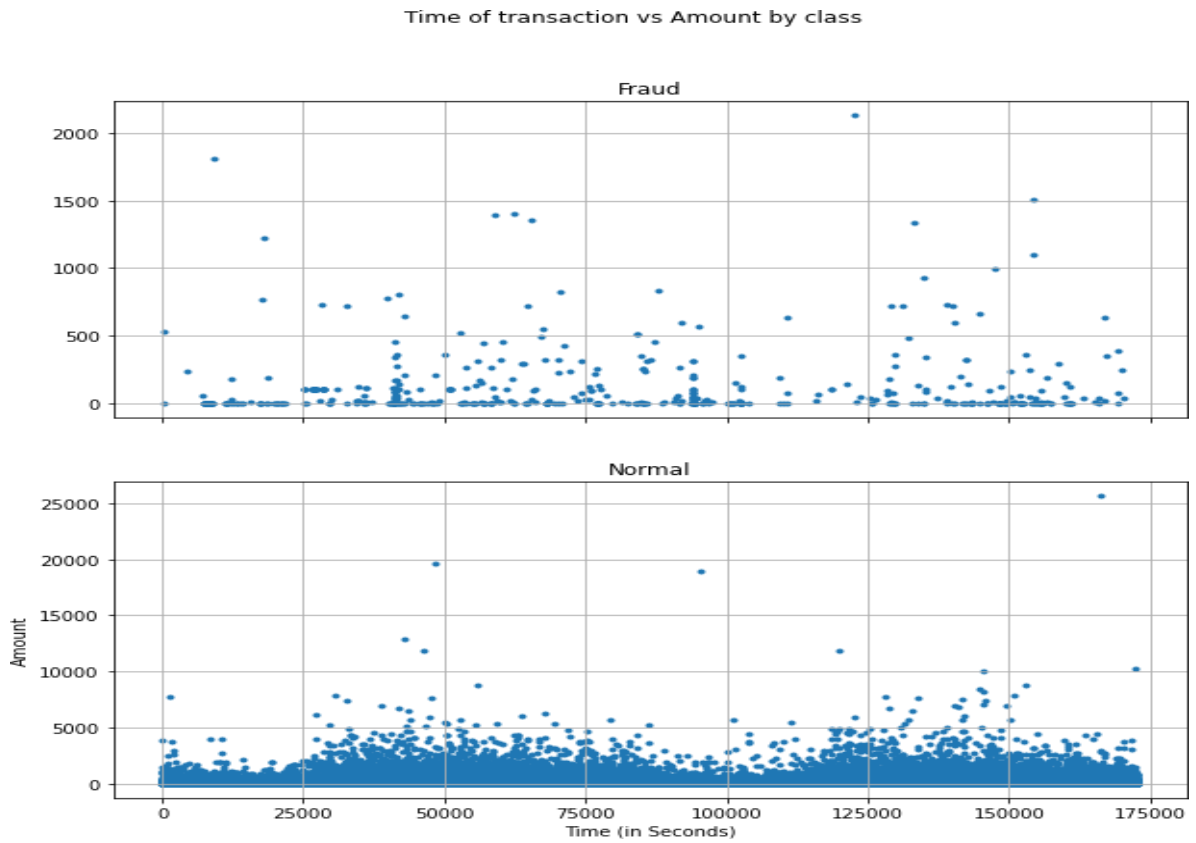

Fig 8.The relation between time of transaction versus amount by fraud and normal class.

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10,10))
f.suptitle('Time of transaction vs Amount by class')
```

```
ax1.scatter(fraud.Time, fraud.Amount, marker='.')
ax1.set_title('Fraud')
ax1.grid()
ax2.scatter(normal.Time, normal.Amount, marker='.')
ax2.set_title('Normal')
ax2.grid()
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```

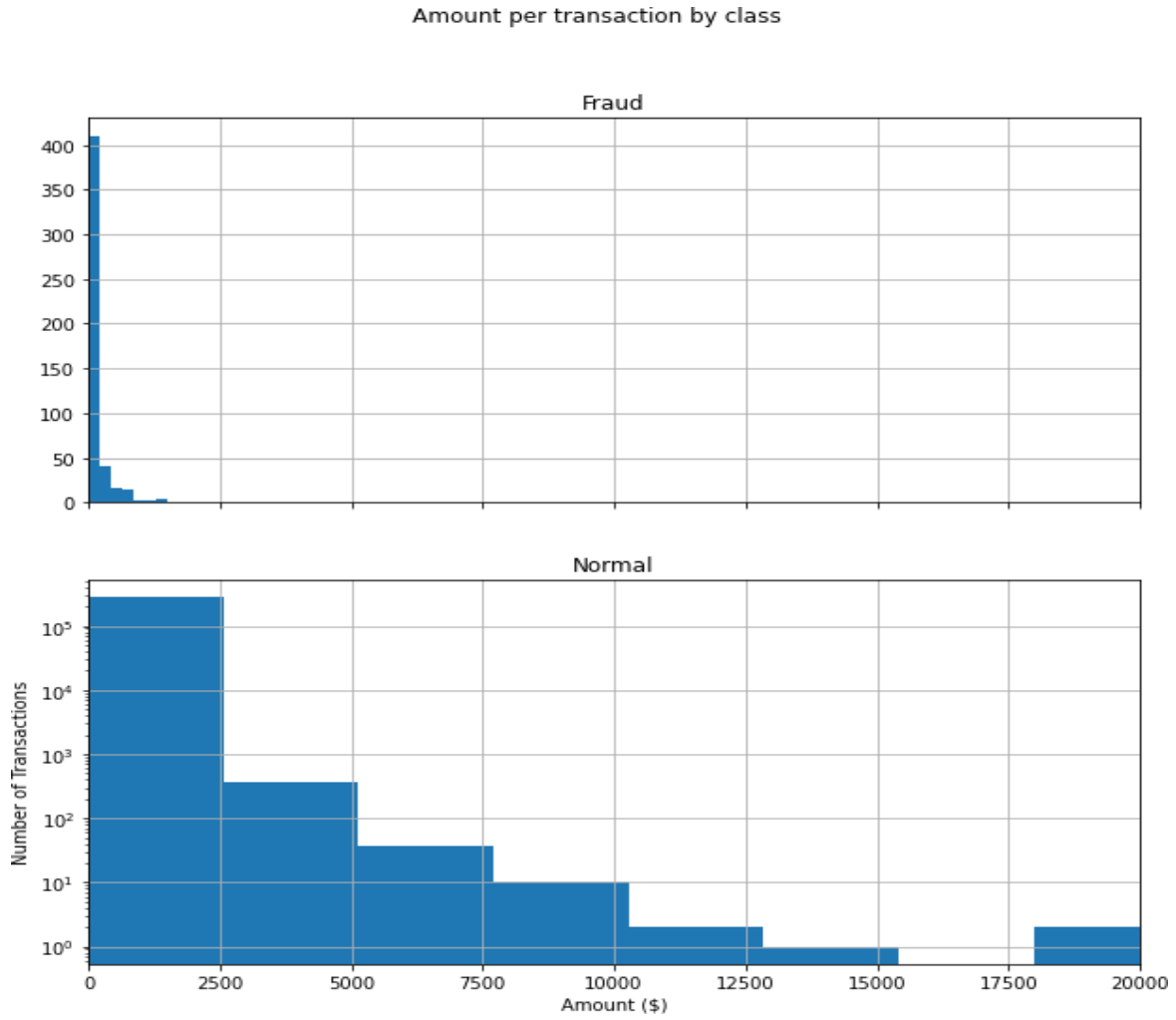

Fig 9. The amount per transaction by fraud and normal class.

Data Modelling

Removal of the “Time” attribute since it has no contribution towards the prediction of the class.
Division of train and test data in the existing dataset , with 80% training data and 20% testing data.

```
"""###Data Modelling"""

data = df.drop(['Time'], axis =1)

X_train, X_test = train_test_split(data, test_size=0.2, random_state=42)

X_train = X_train[X_train.Class == 0]

X_train = X_train.drop(['Class'], axis=1)

y_test = X_test['Class']

X_test = X_test.drop(['Class'], axis=1)

X_train = X_train

X_test = X_test

print(X_train.shape)

#(227451,29)

print(X_test.shape)

#(56962,29)

print(y_test.shape)

#(56962)

scaler = StandardScaler().fit(X_train.Amount.values.reshape(-1,1))

X_train['Amount'] = scaler.transform(X_train.Amount.values.reshape(-1,1))

X_test['Amount'] = scaler.transform(X_test.Amount.values.reshape(-1,1))

X_train.shape

#(227451,29)
```

Model Training

Steps Involved in Model Training

Step1: Encode the input into another vector h. h is a lower dimension vector than the input.

Step2: Decode the vector h to recreate the input. Output will be of the same dimension as the input.

Step3: Calculate the reconstruction error L.

Step4: Back propagate the error from output layer to the input layer to update the weights.

Step5:Repeat the steps 1 through 4 for each of the observations in the dataset.

Step6: Repeat more epochs.

Reconstruction Error

The parameters of the autoencoder model is optimized in such a way that reconstruction error is minimized.

An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and Ψ, such that:

$\phi: X \rightarrow F$

$\psi: F \rightarrow X$

$$\phi, \psi = \arg\min_{\phi,\psi} ||X - (\psi \circ \phi)X||^2$$

Building The Model

Our Autoencoder uses 4 fully connected layers with 14,7,7 and 29 neurons respectively.

The first two layers are used for encoder and the last two go for the decoder.

Additionally, L1 regularization will be used during training.

The activation function used in our model-

4.1.4.3.4.1 tanh

4.1.4.3.4.2 relu

Training The Model

Trained the model for 100 epoch with a batch size of 32 samples and saved the best performing model to a file.

The ModelCheckpoint provided by Keras is really handy for such tasks.

Additionally, the training progress will be exported in a format that TensorBoard understands.

```
"""###Model Training:"""
from keras.layers import Input, Dense
from keras import regularizers
from keras.models import Model, load_model
from keras.callbacks import ModelCheckpoint, TensorBoard
input_dim = X_train.shape[1]
encoding_dim = 14
input_layer = Input(shape=(input_dim,))
encoder = Dense(encoding_dim, activation="tanh",
activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder = Dense(int(encoding_dim / 2), activation="relu")(encoder)
decoder = Dense(int(encoding_dim / 2), activation='tanh')(encoder)
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
X_train.shape
#(227451,29)
nb_epoch = 100
batch_size = 32
autoencoder.compile(optimizer='adam',
                    loss='mean_squared_error',
                    )
checkpointer = ModelCheckpoint(filepath="model.h5",
                               verbose=0,
                               save_best_only=True)
tensorboard = TensorBoard(log_dir='./logs',
                          histogram_freq=0,
                          write_graph=True,
                          write_images=True)

history = autoencoder.fit(X_train, X_train,
                          epochs=nb_epoch,
                          batch_size=batch_size,
                          shuffle=True,
                          validation_split=0.3,
                          verbose=1,
                          callbacks=[checkpointer, tensorboard]).history
```

Model Evaluation

There are a variety of measures for various algorithms and these measures have been developed to evaluate very different things. So there should be criteria for evaluation of various proposed methods. False Positive (FP), False Negative (FN), True Positive (TP), and True Negative (TN) and the relation between them are quantities which are usually adopted by credit card fraud detection researchers to compare the accuracy of different approaches.
The definitions of mentioned parameters are presented below:

- **FP**: the false positive rate indicates the portion of the non-fraudulent transactions wrongly being classified as fraudulent transactions.
- **FN**: the false negative rate indicates the portion of the fraudulent transactions wrongly being classified as normal transactions.
- **TP**: the true positive rate represents the portion of the fraudulent transactions correctly being classified as fraudulent transactions.
- **TN**: the true negative rate represents the portion of the normal transactions correctly being classified as normal transactions.

Table 1shows the details of the most common formulas which are used by researchers for evaluation of their proposed methods. As can be seen in this table some researchers had used multiple formulas in order to evaluate their proposed model.

Measure	Formula	Description
Accuracy (ACC)/Detection rate	TN + TP/TP + FP + FN + TN	Accuracy is the percentage of correctly classified instances. It is one the most widely used classification performance metrics
Precision/Hit rate	TP/TP + FP	Precision is the number of classified positive or fraudulent instances that actually are positive instances.
True positive rate/Sensitivity	TP/TP + FN	TP (true positive) is the number of correctly classified positive or abnormal instances. TP rate measures how well a classifier can recognize abnormal records. It is also called sensitivity measure. In the case of

		credit card fraud detection, abnormal instances are fraudulent transactions.
True negative rate /Specificity	TN/TN + FP	TN (true negative) is the number of correctly classified negative or normal instances. TN rate measures how well a classifier can recognize normal records. It is also called specificity measure.
False positive rate (FPR)	FP/FP+TN	Ratio of credit card fraud detected incorrectly
ROC	True positive rate plotted against false positive rate	Relative Operating Characteristic curve, a comparison of TPR and FPR as the criterion changes
Cost	Cost = 100 * FN + 10 * (FP +TP)	
F1-measure	2 × (Precision ×Recall)/(Precision +Recall)	Weighted average of the precision and recall

Table 1. Evaluation criteria for credit card fraud detection

The aim of all algorithms and techniques is to minimize FP and FN rate and maximize TP and TN rate and with a good detection rate at the same time.

```
from sklearn.metrics import (confusion_matrix, precision_recall_curve, auc,
                             roc_curve, recall_score, classification_report, f1_score,
                             precision_recall_fscore_support, roc_auc_score)
groups = error_df.groupby('true_class')
fig, ax = plt.subplots(figsize = (20,8))

for name, group in groups:
    ax.plot(group.index, group.reconstruction_error, marker='+', ms=10, linestyle='',
            label= "Fraud" if name == 1 else "Normal")
ax.hlines(threshold, ax.get_xlim()[0], ax.get_xlim()[1], colors="r", zorder=100,
label='Threshold')
```

```
ax.legend()
plt.title("Reconstruction error for different classes")
plt.ylabel("Reconstruction error")
plt.xlabel("Data point index")
plt.grid()
plt.show();
```

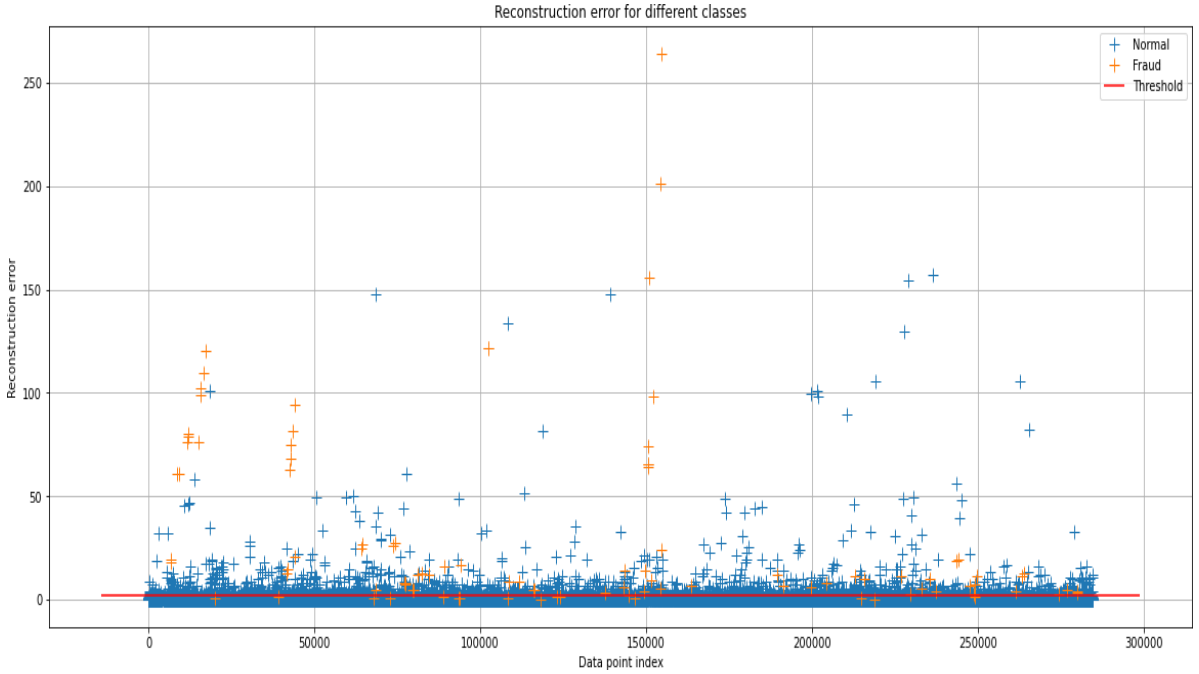

Fig 10. Reconstruction error for different classes

```
fpr, tpr, thres = roc_curve(error_df.true_class, error_df.reconstruction_error)
plt.plot(fpr, tpr, label = 'AUC')
plt.plot([0,1], [0,1], ':', label = 'Random')
plt.legend()
plt.grid()
plt.ylabel("TPR")
plt.xlabel("FPR")
plt.title('ROC')
plt.show()
```

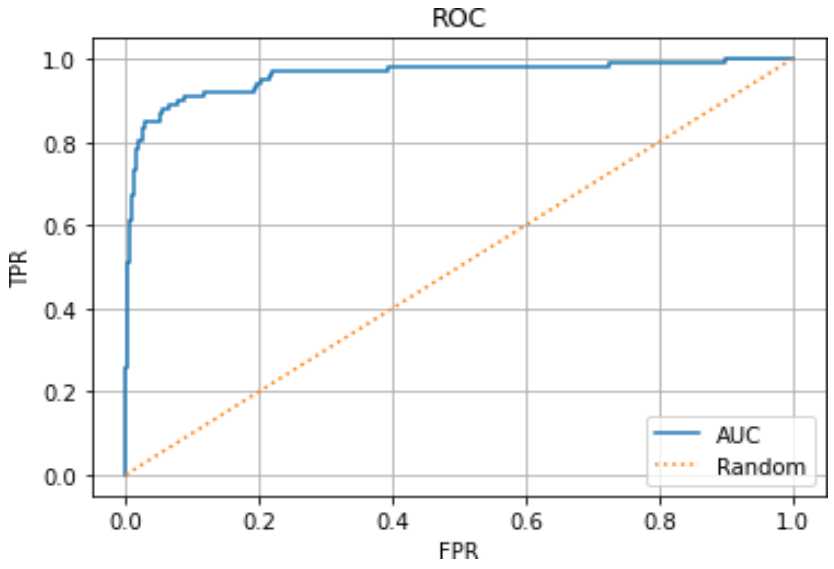

Fig 11. Relative Operating Characteristic curve

```
LABELS = ['Normal', 'Fraud']
threshold = 2
y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
conf_matrix = confusion_matrix(error_df.true_class, y_pred)
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True,
fmt="d", cmap='Greens');
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

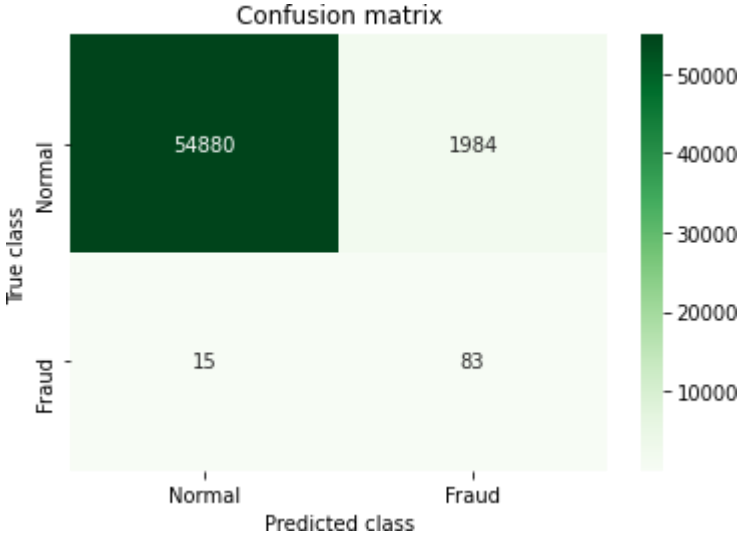

Fig 12. Confusion Matrix for threshold=2

```
print(classification_report(error_df.true_class,y_pred))
```

	precision	Recall	f1-score	support
0	1.00	0.97	0.98	56864
1	0.04	0.85	0.08	98
accuracy			0.96	56962
macro avg	0.52	0.91	0.53	56962
weighted avg	1.00	0.96	0.98	56962

Table 2. Classification Report for Threshold=2

```
threshold = 3.1
y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
conf_matrix = confusion_matrix(error_df.true_class, y_pred)
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True,
fmt="d", cmap='Greens');
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
print(classification_report(error_df.true_class,y_pred))
```

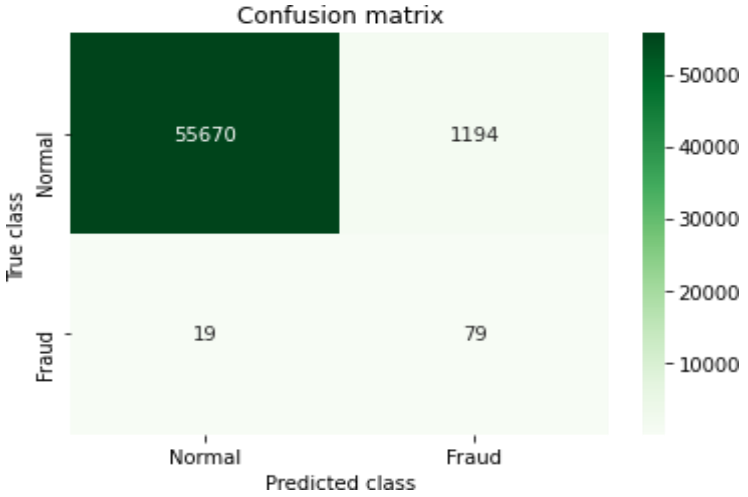

Fig 13. Confusion Matrix for threshold=3.1

	precision	Recall	f1-score	support
0	1.00	0.97	0.98	56864
1	0.04	0.85	0.12	98
accuracy			0.98	56962
macro avg	0.53	0.89	0.55	56962
weighted avg	1.00	0.98	0.99	56962

Table 3. Classification Report for Threshold=3.1

Web App

We have developed a Flask Web App using the model by saving it with pickle.It contains a form to take the input of 30 attributes and predict if the given value is fraudulent or not by predicting the output with the trained model and showing the results.

requirements.txt:

Pillow==8.2.0

pyparsing==2.4.7

python-dateutil==2.8.1

pytz==2021.1

scikit-learn==0.24.2

scipy==1.6.3

seaborn==0.11.1

six==1.16.0

sklearn==0.0

threadpoolctl==2.1.0

Werkzeug==2.0.1

app.py:

```
from flask import Flask, render_template, url_for, request
import numpy as np
import pickle
import os
# load the model from disk r
filename = 'model.pkl'
clf = pickle.load(open(filename, 'rb'))

app = Flask(_name_)
path=os.getcwd()
@app.route('/')
def home():
  return render_template('home.html')

@app.route('/predict', methods = ['POST'])
def predict():
  if request.method == 'POST':
      me = request.form['message']
      me=me.rstrip()
      try:
          message = [float(x) for x in me.split(",")]
      except:
          return render_template('result.html',prediction = 2)
      if len(message)!=30:
          return render_template('result.html',prediction = 2)
      print(message)
      print(len(message))
      vect = np.array(message).reshape(1, -1)
      my_prediction = clf.predict(vect)
```

```
        if my_prediction == 1:
            f1=open('fraudvalues.csv','a')
            s="\n"+me+",1"
            f1.write(s)
            f1.close()
        else:
            f1=open('validvalues.csv','a')
            s="\n"+me+",0"
            f1.write(s)
            f1.close()
    return render_template('result.html',prediction = my_prediction)
@app.route('/data')
def data():
    return render_template('csv.html')
if __name__== "_main_":
    app.run()
```

template/home.html:

```
<!DOCTYPE html>
<html>
 <head>
   <title>Credit Card Fraud Detection Web App</title>
   <link rel="stylesheet" type="text/css" href="../static/css/styles.css">
   <link
     href="https://fonts.googleapis.com/css2?family=Quicksand:wght@500&display=swap"
     rel="stylesheet"
   />
  <style>
      body{
 font-size: 16px;
 font-family: Open Sans, sans-serif;
```

```
 display: flex;
 flex-direction: column;
 align-items: center;
 text-align: center;
 line-height: 1.6;
 padding: 20px;
 border: 4px solid limegreen;
 margin: 2% 10%;
 background: lightyellow;
}

header{
 font-size: 1.6em;
}

header h1{
 font-size: 1.8em;
}

.btn-info{
 border: 1.5px solid black;
 padding: 10px 15px;
 margin: 15px;
 box-shadow: 0 0 4px #eee;
 cursor: pointer;
}

.btn-info:hover{
 background-color: transparent;
 box-shadow: 0 0 10px #eee;
```

```
}

.ml-container{
 font-size: 1.4em;
}

.ml-container textarea{
 width: 100%;
}
  </style>
 </head>
 <body>
   <header>
     <div class="container">
       <h1>
         <p style="text-align: center">Credit Card Fraud Detection(11C)</p>
       </h1>
     </div>
   </header>

   <div class="ml-container">


     <form action="{{ url_for('predict')}}" method="POST">
       <!-- <input type="text" name="comment"/> -->
       <div class="justify">
         <textarea name="message" rows="8" cols="50"></textarea>
       </div>
       <br/>
```

```
      <input type="submit" class="btn-info" value="Predict" />
    </form>
  </div>
 </body>
</html>
```

template/result.html:

```
<!DOCTYPE html>
<html>
 <head>
  <title></title>
  <link
   href="https://fonts.googleapis.com/css2?family=Quicksand:wght@500&display=swap"
   rel="stylesheet"
  />
 <style>body{
 font-size: 16px;
 font-family: Open Sans, sans-serif;
 display: flex;
 flex-direction: column;
 align-items: center;
 text-align: center;
 line-height: 1.6;
 padding: 20px;
 border: 4px solid limegreen;
 margin: 2% 10%;
 background: lightyellow;
}

header{
 font-size: 1.6em;
```

```
}

header h1{
 font-size: 1.8em;
}

.btn-info{
 border: 1.5px solid black;
 padding: 10px 15px;
 margin: 15px;
 box-shadow: 0 0 4px #eee;
 cursor: pointer;
}

.btn-info:hover{
 background-color: transparent;
 box-shadow: 0 0 10px #eee;
}

.ml-container{
 font-size: 1.4em;
}

.ml-container textarea{
 width: 100%;
}</style>
 </head>
 <body>
   <header>
     <div class="container">
```

```
    <div id="brandname">
      <h1 style="color: black">
      Credit Card Fraud Detection Results
      </h1>
    </div>
    <!-- <h2><p style="text-align: center">Detection</p></h2> -->
  </div>
</header>
<h2 style="color: blueviolet;">
  <b>Validation Completed.</b>
</h2>
<div class="results">
  {% if prediction == 0 %}
  <h1 style="color: green">
      According to our model, the provided transaction is NOT a Fraud transaction.
  </h1>
  {% elif prediction == 1 %}
  <h2 style="color: red">
      According to our model, this transaction is a Fraud transaction.
  </h2>
  {% elif prediction == 2 %}
  <h2 style="color: blue">
      According to our model, The Data is invalid(not a transaction).
  </h2>
  {% endif %}
</div>
<a href="/">
  <button class="btn-info">Retest</button>

</a> </body></html>
```

Sample Input and Output

INPUT:

V1	0.339812
V2	-2.743745
V3	-0.134070
V4	-1.385729
V5	-1.451413
V6	1.015887
V7	-0.524379
V8	0.224060
V9	0.899746
V10	-0.565012
V11	-0.087670
V12	0.979427
V13	0.076883
V14	-0.217884
V15	-0.136830
V16	-2.142892
V17	0.126956
V18	1.752662
V19	0.432546
V20	0.506044
V21	-0.213436
V22	-0.942525
V23	-0.526819
V24	-1.156992
V25	0.311211
V26	-0.746647
V27	0.040996

V28 0.102038

Amount 1.693166

Name: 49906, dtype: float64

AUTOENCODER OUTPUT:

[[0. 0. 0. 0. 0. 1.1093647

0. 0. 0.93837786 0. 0. 0.

0.3775134 0. 0. 0. 0. 1.1749479

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 2.3275394]]

FINAL OUTPUT: 1

Web Pages

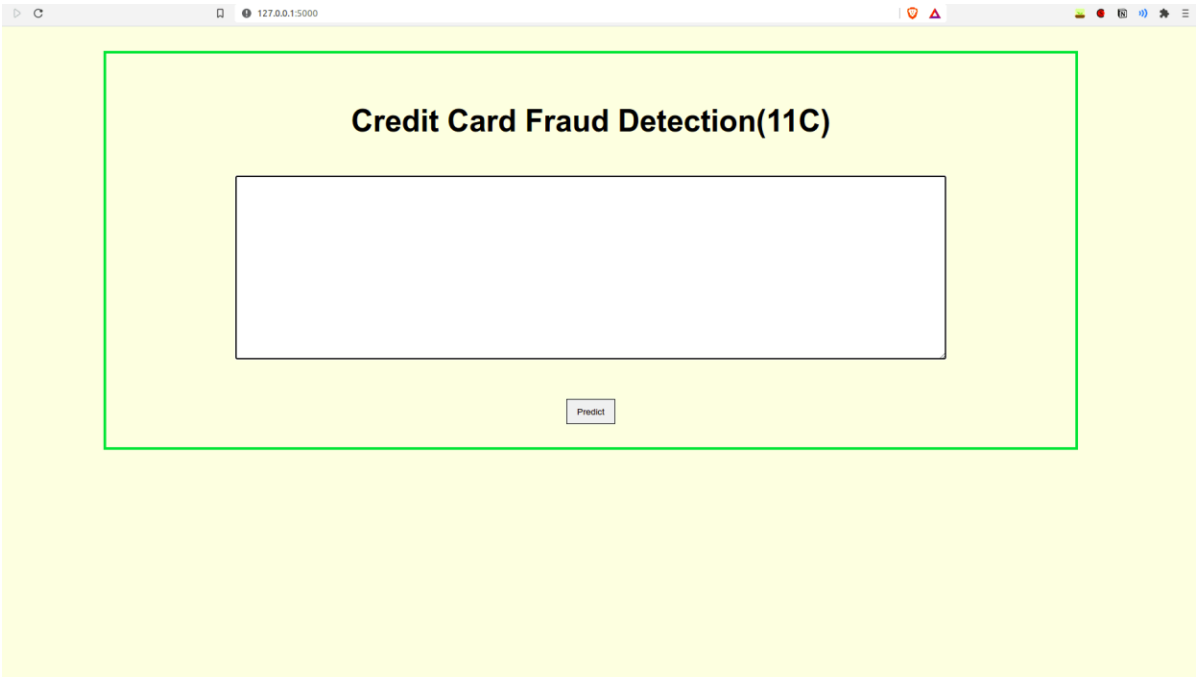

Fig 14. Homepage

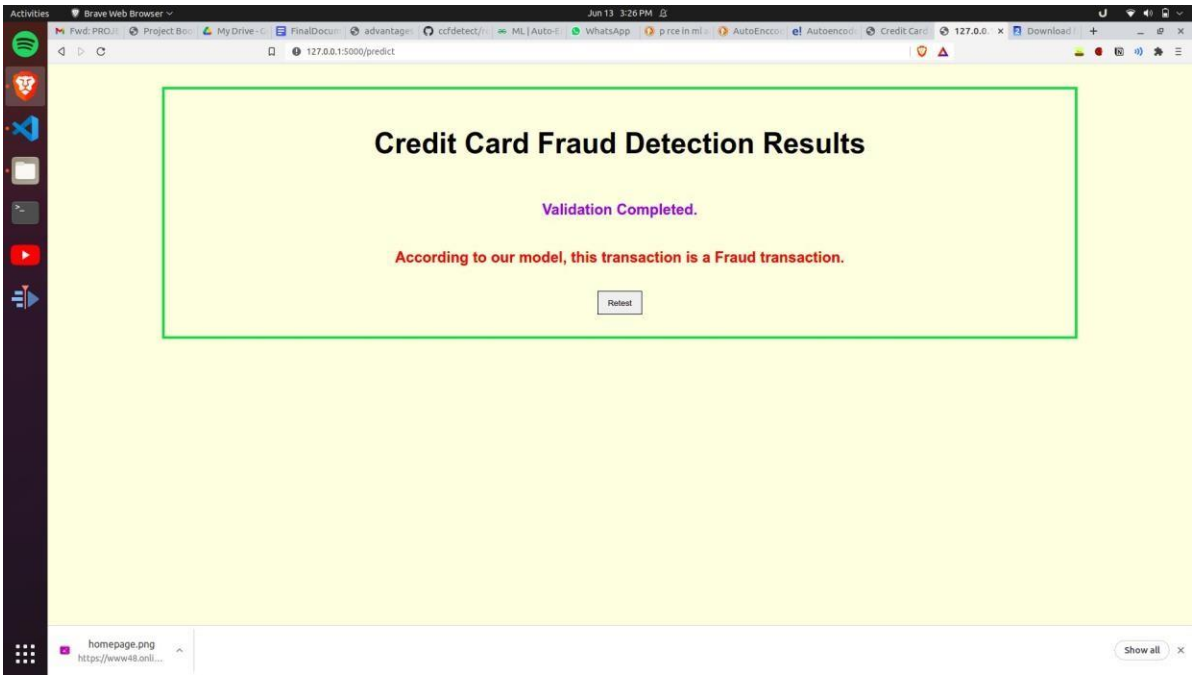

Fig 15. Result Page predicting Fraudulent Transaction

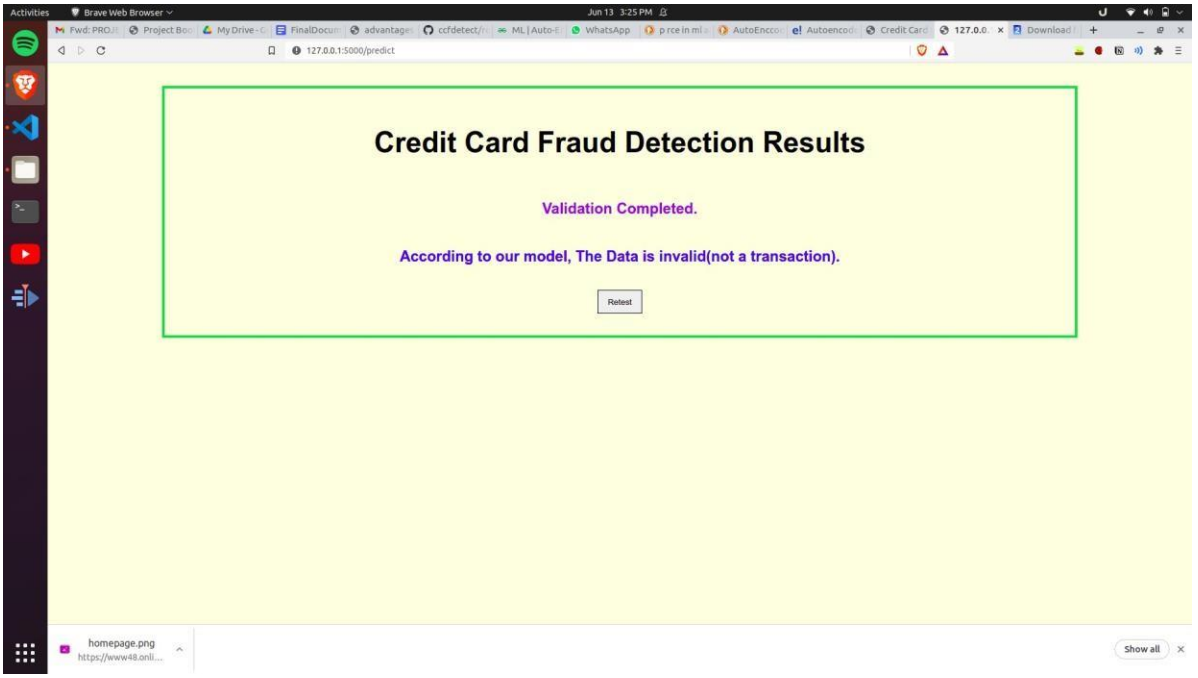

Fig 16. Result Page showing error “Invalid Data” Transaction

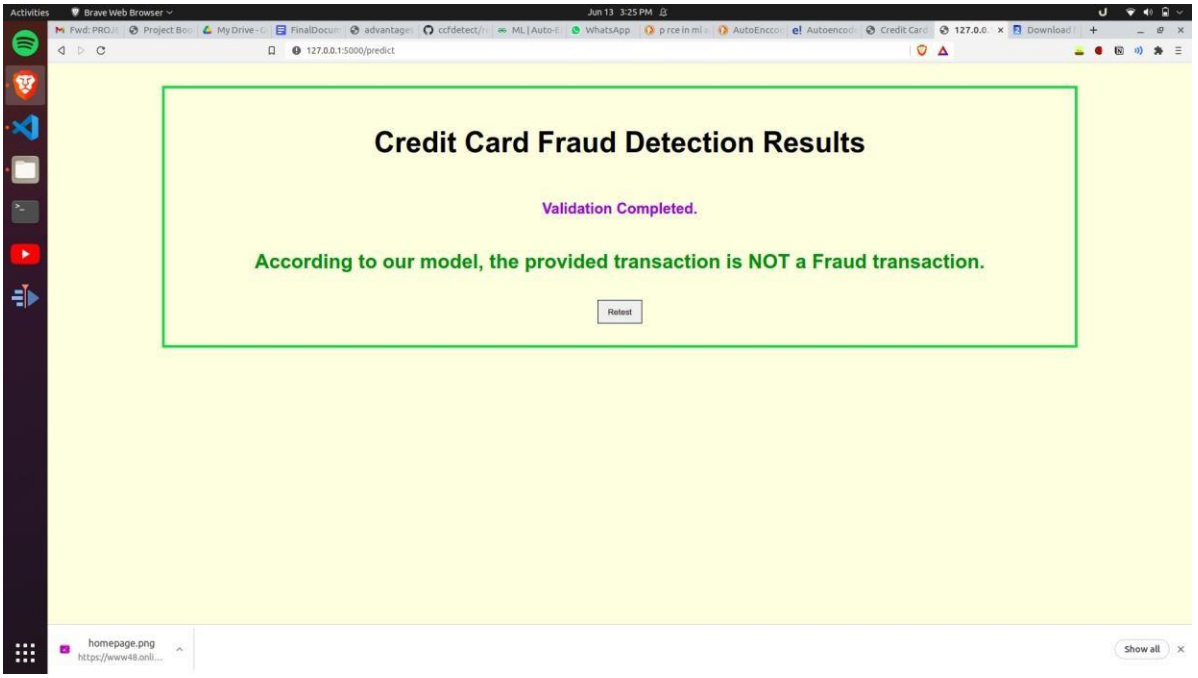

Fig 17. Result Page predicting non Fraudulent Transaction

CHAPTER 5
RESULT AND DISCUSSION

The data set is highly skewed, consisting of 492 frauds in a total of 284,807 observations. This Resulted in only 0.172% fraud cases. This skewed set is justified by the low number of fraudulent transactions. The dataset consists of numerical values from the 28 ‘Principal Component Analysis (PCA)’ transformed features, namely V1 to V28. Furthermore, there is no metadata about the original features provided, so pre-analysis or feature study could not be done.

- The ‘Time’ and ‘Amount’ features are not transformed data.
- There is no missing value in the dataset

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

Conclusion

In this project we have used an autoencoder to encode the given data and then decode it into original data and then calculated the reconstruction error to classify into normal or fraudulent transactions. We have also saved the trained autoencoder model and then loaded with pickle into the flask application.

Future Work

- Deployment of this model into the cloud applications like Heroku
- Extend the model for other datasets
- Create an API which takes transactions into it and predicts the type of transaction.

REFERENCES:

[1] KhyatiChaudhary, JyotiYadav, BhawnaMallick, " A review of Fraud Detection Techniques: Credit Card", International Journal of Computer Applications Volume 45–No.1 2012.

[2] Linda Delamaire, Hussein Abdou, John Pointon, "Credit card fraud and detection techniques: a review", Banks and Bank Systems, Volume 4, Issue 2, 2009.

[3] Maes S. Tuyls K. Vanschoenwinkel B. and Manderick B.; "Credit Card Fraud Detection Using Bayesian and Neural Networks"; Vrije University Brussel – Belgium; 2002.

[4] Soltani, N., Akbari, M.K., SargolzaeiJavan, M., "A new user-based model for creditcard fraud detection based on artificial immune system," Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on., IEEE, pp. 029-033, 2012.

[5] S. Ghosh and D. L. Reilly, "Credit card fraud detection with a neural-network", Proceedings of the 27th Annual Conference on System Science, Volume 3: Information Systems: DSS/ KnowledgeBased Systems, pages 621-630, 1994. IEEE Computer Society Press.

[6] MasoumehZareapoor, Seeja.K.R, M.Afshar.Alam, "Analysis of Credit Card Fraud Detection Techniques: based on Certain Design Criteria", International Journal of Computer Applications (0975 – 8887) Volume 52– No.3, 2012.

[7] Fraud Brief – AVS and CVM, Clear Commerce Corporation, 2003, http://www.clearcommerce.com.

[8] All points protection: One sure strategy to control fraud, Fair Isaac, http://www.fairisaac.com, 2007.

[9] RaghavendraPatidar, Lokesh Sharma, "Credit Card Fraud Detection Using Neural Network", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, 2011.

[10] Holland, J. H. "Adaptation in natural and artificial systems." Ann Arbor: The University of Michigan Press. (1975).

[11] SushmitoGhosh, Douglas L. Reilly, Nestor, "Credit Card Fraud Detection with a NeuralNetwork", Proceedings of 27th Annual Hawaii International Conference on System Sciences, 1994.

[12] Vesanto, J., &Alhoniemi, E. (2000). "Clustering of the self-organizing map".IEEE Transactions on Neural Networks, (2009). 11; (586–600).

[13] Img Src:https://en.wikipedia.org/wiki/File:Autoencoder_schema.png