**CREDIT CARD FRAUD DETECTION**

**CHAPTER 1**

**Abstract:-** Credit card fraud events take place frequently and then result in huge financial losses. Criminals can use some technologies such as Trojan or Phishing to steal the information of other people’s credit cards. Therefore, an effective fraud detection method is important since it can identify a fraud in time whena criminal uses a stolen card to consume. Inthis paper, Random Forest, SVM, KNN and Naïve Bayes is used to train thebehavior features of normal and abnormal transactions. We are using the dataset of the credit card transactions. We analyze the performance on creditfraud detection. The data used in our experiments come froman e-commerce company in China.

***Index Terms*—**Random forest, SVM, KNN, Naïve Bayes, Credit card fraud.

**CHAPTER 2**

**INTRODUCTION**

Credit cards are widely used due to the popularization of ecommerce and the development of mobile intelligent devices. Card-not-present transactions (i.e., online transaction without a physical card) is more popular, especially all credit card operations are performed by web payment gateways, e.g., PayPal and Alipay. Credit card has made an online transaction easier and more convenient. However, there is a growing trend of transaction frauds resulting in a great losses of money every year. It is estimated that losses are increased yearly at double digit rates by 2020. Since the physical card is not needed in the online transaction environment and the card’s information is enough to complete a payment, it is easier to conduct a fraud than before. Transaction fraud has become a top barrier to the development of e-commerce and has a dramatic influence on the economy. Hence, fraud detection is essential and necessary. Fraud detection is a process of monitoring the transaction behavior of a cardholder in order to detect whether an incoming transaction is done by the cardholder or others. Generally, there are two kinds of methods for fraud detection: misuse detection and anomaly detection. Misuse detection uses classification methods to determine whether an incoming transaction is fraud or not. Usually, such an approach has to know about the existing types of fraud to make models by learning the various fraud patterns. Anomaly detection is to build the profile of normal transaction behavior of a cardholder based on his/her historical transaction data, and decide a newly transaction as a potential fraud if it deviates from the normal transaction behavior. However, an anomaly detection method needs enough successive sample data to characterize the normal transaction behavior of a cardholder. This paper is about misuse method. We use random forest to train the normal and fraud behavior features. Random forest is a classification algorithm based on the votes of all base classifiers.

The major contributions of this paper are summarized as follows.

1) To deal with normal/fraud detection problem, the Random Forest algorithm is used to train the normal/fraud behavior features.

2) From the result of experiments, some conclusions are made which would be helpful for future work.

**CHAPTER 3**

**LITERATURE SURVEY**

# 1. A New Framework for Credit Card Transactions Involving Mutual Authentication between Cardholder and Merchant

Electronic Commerce (e-Commerce) and ease in the onsite transactions have led to the exponential growth in the acceptance of credit cards among consumers of all the sections. But despite their remarkable advantages, consumers are still reluctant in their use, especially for online transactions and reason being the increasing credit card fraud rate. A number of security models have been proposed and deployed for secure online transactions but the sharing of sensitive credit card data over the Internet has made online transactions vulnerable to threats. In this paper, we discuss and analyze the current developments in online authentication procedures including biometrics, one-time-password systems and use of mobile device and Public Switched Telephone Network for cardholder authentication. Then we propose a complete new framework for both onsite and online (Internet shopping) credit card transactions. This framework is more secure, robust, enhances user privacy and does not involve the deployment of special hardware systems at the customer's site.

**2. Random forest for credit card fraud detection**

Credit card fraud events take place frequently and then result in huge financial losses. Criminals can use some technologies such as Trojan or Phishing to steal the information of other people's credit cards. Therefore, an effictive fraud detection method is important since it can identify a fraud in time when a criminal uses a stolen card to consume. One method is to make full use of the historical transaction data including normal transactions and fraud ones to obtain normal/fraud behavior features based on machine learning techniques, and then utilize these features to check if a transaction is fraud or not. In this paper, two kinds of random forests are used to train the behavior features of normal and abnormal transactions. We make a comparison of the two random forests which are different in their base classifiers, and analyze their performance on credit fraud detection. The data used in our experiments come from an e-commerce company in China.

## 3.Unsupervised Profiling Methods for Fraud Detection (2001)

Credit card fraud falls broadly into two categories: behavioural fraud and application fraud. Application fraud occurs when individuals obtain new credit cards from issuing companies using false personal information and then spend as much as possible in a short space of time. However, most credit card fraud is behavioural and occurs when details of legitimate cards have been obtained fraudulently and sales are made on a 'Cardholder Not Present' basis. These sales include telephone sales and e-commerce transactions where only the card details are required. In this paper, we are concerned with detecting behavioural fraud through the analysis of longitudinal data. These data usually consist of credit card transactions over time, but can include other variables, both static and longitudinal. Statistical methods for fraud detection are often classification (supervised) methods that discriminate between known fraudulent and non-fraudulent transactions; however, these methods rely on accurate identification of fraudulent transactions in historical databases -- information that is often in short supply or non-existent. We are particularly interested in unsupervised methods that do not use this information but instead detect changes in behaviour or unusual transactions. We discuss two methods for unsupervised fraud detection in credit data in this paper and apply them to some real data sets. Peer group analysis is a new tool for monitoring behaviour over time in data mining situations. In particular, the tool detects individual accounts that begin to behave in a way distinct from accounts to which they had previously been similar. Each account is selected as a target account and is compared with all other accounts in the database, using either external comparison criteria.

**4. Credit Card Fraud Detection using Time Series Analysis**

Credit card usage has been increased tremendously because of the popularity of E-commerce. As the usage of credit card grows the occurrence of fraudulent transactions also increases, thus comes the stipulation of fraud detection. Detection of fraudulent transaction using credit card plays a vital role in financial institutions. In the proposed work, fraud detection is done with data mining approaches. The parameters considered are transaction amount and transaction time. For every cardholder there is always a robust periodic pattern in the spending behaviour, centered on this fact the anomalies in the transaction are detected by analyzing the past history of transactions belonging to an individual cardholder. In this work two levels of detection methods are used. At the first level the fraud is detected by analyzing whether the new incoming transaction is fraud or not by using distance-based method. At the second level the next transaction is predicted by means of label-prediction methodology and compared with the actual transaction, if there is deviation then it is detected to be a fraudulent transaction. If the particular transaction is considered as a fraud then the cardholder is asked to continue the transaction by asking a secret question, if the cardholder does not give correct answer then the transaction will not be allowed to continue further. The approach used in the proposed work has also decreased the false positive situation and hence it is ensured that genuine transaction is not rejected.

**5.Credit Card Fraud Detection System Using Hidden Markov Model and K-Clustering**

Credit card frauds are increasing day by day regardless of the various techniques developed for its detection. Fraudsters are so expert that they engender new ways for committing fraudulent transactions each day which demands constant innovation for its detection techniques as well. Many techniques based on Artificial Intelligence, Data mining, Fuzzy logic, Machine learning, Sequence Alignment, neural network, logistic regression, naïve Bayesian, Bayesian network, metalearning, Genetic Programming etc., has evolved in detecting various credit card fraudulent transactions. A steady indulgent on all these approaches will positively lead to an efficient credit card fraud detection system. This paper presents a survey of various techniques used in credit card fraud detection mechanisms and Hidden Markov Model (HMM) in detail. HMM categorizes card holder’s profile as low, medium and high spending based on their spending behaviour in terms of amount. A set of probabilities for amount of transaction is being assigned to each cardholder. Amount of each incoming transaction is then matched with card owner’s category, if it justifies a predefined threshold value then the transaction is decided to be legitimate else declared as fraudulent. Existing fraud detection system may not be so much capable to reduce fraud transaction rate. Improvement in fraud detection practices has become essential to maintain existence of payment system. In this paper Hidden Markov Model (HMM) is used to model the sequence of operation in credit card transaction processing. If an incoming credit card transaction is not accepted by the trained HMM with sufficiently high probability, it is considered to be fraudulent.

**CHAPTER 4**

**SYSTEM REQUIREMENT SPECIFICATION**

**4.1 Functional Requirements**

The particular necessities are user interfaces. The outside clients are the customers. Every one of the customers can utilize this product for ordering and looking.

* Hardware Interfaces: The outside equipment interface utilized for ordering and looking is PCs of the customers. The PC's might be portable PCs with remote LAN as the web association gave will be remote.
* Software Interfaces: The working Frameworks can be any rendition of windows.
* Performance Prerequisites: The PC's utilized must be atleast pentium 4 machine with the goal that they can give ideal execution of the item.

**4.2 Non-Functional Requirements**

Non utilitarian necessities are the capacities offered by the framework. It incorporates time imperative and requirement on the advancement procedure and models. The non useful prerequisites are as per the following:

* Speed: The framework ought to prepare the given contribution to yield inside fitting time.
* Ease of utilization: The product tought to be easy to understand. At that point the clients can utilize effortlessly, so it doesn't require much preparing time.
* Reliability: The rate of disappointments ought to be less then just the framework is more solid.
* Portability: It thought to be anything but difficult to actualize in any framework

**4.3 Hardware requirements**

The most widely recognized arrangement of prerequisites characterized by any working framework or programming application is the physical PC assets, otherwise called equipment, An equipment necessities list is frequently joined by an equipment similarity list, particularly if there should be an occurrence of working frameworks. A HCL records tried, perfect, and now and then incongruent equipment gadgets for a specific working framework or application. The accompanying sub-segments examine the different parts of equipment prerequisites.

All PC working frameworks are intended for a specific PC design. Most programming applications are restricted to specific working frameworks running on specific structures. In spite of the fact that engineering free working frameworks and applications exist, most should be recompiled to keep running on another design.

The energy of the focal preparing unit (CPU) is a central framework necessity for any product. Most programming running on x86 engineering characterize preparing power as the model and the clock speed of the CPU. Numerous different highlights of a CPU that impact its speed and power, similar to transport speed, store, and MIPS are frequently overlooked. This meaning of energy is regularly wrong, as AMD Intel Pentium CPUs at comparative clock speed frequently have distinctive throughput speeds.

• 10GB HDD(min)

• 128 MB RAM(min)

• Pentium P4 Processor 2.8Ghz(min)

**Software requirements**

Programming necessities manage characterizing programming asset necessities and requirements that should be introduced on a PC to give ideal working of an application.

These necessities or requirements are for the most part excluded in the product establishment bundle and should be introduced independently before the product is introduced.

* Python 3.7 or higher
* Pycharm
* opencv

**Outline of advances**

The innovations utilized is depicted as underneath:

**Python**

* Python is a general purpose high level programming Language (human understandable languages are High level programming languages)
* Python Developed by Guido Van Rossam
* 1989 National Research Institute(NRI) At Netherland
* Officially Python available to the public in 1991 :: FEB 20th 1991

Python was imagined in the late 1980s,[29] and its usage started in December 1989[30] by Guido van Rossum at Centrum Wiskunde and Informatica (CWI) in the Netherlands as a successor to the ABC dialect (itself roused by SETL)[31]capable of exemption dealing with and interfacing with the Amoeba working system.[6] Van Rossum remains Python's chief creator. His proceeding with focal part in Python's advancement is reflected in the title given him by the Python people group:

**Python Feature**

* Simple and easy to learn

Python as only 33 keywords But JAVA as(53) keywords

* Free ware (There is no license we cannot pay anything)

& Open source (we can able to see source code if source is not good I can able to customize our requirements)

* High level programming language (human understandable language)

Python Is Platform Independent (It means I can write a program once and run any where(WORA)

* Portability

Moving python program from one platform to another platform without changing any thing

* Dynamically Typed Programming Language

In python we are not required to declare type in Python

* Both Object Oriented and Procedure Oriented Language
* Interpreted Language

It means we are not going to compile

* Extensible

We can use Other Programming Language in Python

**Limitations of python**

* Performance wise it is not up to the mark Beacause its an interpreted language

Interpreter able to see only one line

(JAVA is better performace compare to python in java JIT (just intime compiler) concept is there

* Mobile appications it is not up to the mark

Myth:-python is not suitable large scale enterprise applications

**Flavors of python**

* Cpython :- It can be standard, It can be used to c language python
* Jpython or jpython :- it is for JAVA application
* Iron python:-to work with microsoft .net platform
* Pypy :-Internally JIT (just intime compiler) compiler is there so performance wise too good
* Ruby python:- used for ruby application
* Anaconda python:- To handle Big-data happily go for Anaconda python
* Stackless (python for concurrancy) :-
* parallely you execute (like mutithread) so go for stackless

**Applications of Python**

**1. GUI-Based Desktop Applications:**

Python has simple syntax, modular architecture, rich text processing tools and the ability to work on multiple operating systems which make it a desirable choice for developing desktop-based applications. There are various GUI toolkits like wxPython, PyQt or PyGtk available which help developers create highly functional Graphical User Interface (GUI). The various applications developed using Python includes:

* **Image Processing and Graphic Design Applications:**

Python has been used to make 2D imaging software such as Inkscape, GIMP, Paint Shop Pro and Scribus. Further, 3D animation packages, like Blender, 3ds Max, Cinema 4D, Houdini, Lightwave and Maya, also use Python in variable proportions.

* **Scientific and Computational Applications:**

The higher speeds, productivity and availability of tools, such as Scientific Python and Numeric Python, have resulted in Python becoming an integral part of applications involved in computation and processing of scientific data. 3D modeling software, such as FreeCAD, and finite element method software, such as Abaqus, are coded in Python.

* **Games:**

Python has various modules, libraries and platforms that support development of games. For example, PySoy is a 3D game engine supporting Python 3, and PyGame provides functionality and a library for game development. There have been numerous games built using Python including Civilization-IV, Disney’s Toontown Online, Vega Strike etc.

**2. Web Frameworks and Web Applications:**

Python has been used to create a variety of web-frameworks including CherryPy, Django, TurboGears, Bottle, Flask etc. These frameworks provide standard libraries and modules which simplify tasks related to content management, interaction with database and interfacing with different internet protocols such as HTTP, SMTP, XML-RPC, FTP and POP. Plone, a content management system; ERP5, an open source ERP which is used in aerospace, apparel and banking; Odoo – a consolidated suite of business applications; and Google App engine are a few of the popular web applications based on Python.

**3. Enterprise and Business Applications:**

With features that include special libraries, extensibility, scalability and easily readable syntax, Python is a suitable coding language for customizing larger applications. Reddit, which was originally written in Common Lips, was rewritten in Python in 2005. Python also contributed in a large part to functionality in YouTube.

**4. Operating Systems:**

Python is often an integral part of Linux distributions. For instance, Ubuntu’s Ubiquity Installer, and Fedora’s and Red Hat Enterprise Linux’s Anaconda Installer are written in Python. Gentoo Linux makes use of Python for Portage, its package management system.

**5. Language Development:**

Python’s design and module architecture has influenced development of numerous languages. Boo language uses an object model, syntax and indentation, similar to Python. Further, syntax of languages like Apple’s Swift, CoffeeScript, Cobra, and OCaml all share similarity with Python.

**6. Prototyping:**

Besides being quick and easy to learn, Python also has the open source advantage of being free with the support of a large community. This makes it the preferred choice for prototype development. Further, the agility, extensibility and scalability and ease of refactoring code associated with Python allow faster development from initial prototype.Since its origin in 1989, Python has grown to become part of a plethora of web-based, desktop-based, graphic design, scientific, and computational applications. With Python available for Windows, Mac OS X and Linux / UNIX, it offers ease of development for enterprises. Additionally, the latest release Python 3.4.3 builds on the existing strengths of the language, with drastic improvement in Unicode support, among other new features.

**Versions of python**

* Python 1.0 Introduced in jan 1994
* Python 2.0 Introduced in oct 2000
* Python 3.0 introduced in dec 2008

latest version

python 3.6.3 🡺 2016

python 3.7

Any new version should provide support for old version programs

* There is no- backward compatibility support
* Python 3 is not support to python 2 program

**Python Machine Learning**

Python is a popular platform used for research and development of production systems. It is a vast language with number of modules, packages and libraries that provides multiple ways of achieving a task.

.

Python and its libraries like NumPy, SciPy, Scikit-Learn, Matplotlib are used in data science and data analysis. They are also extensively used for creating scalable machine learning algorithms. Python implements popular machine learning techniques such as Classification, Regression, Recommendation, and Clustering.

Python offers ready-made framework for performing data mining tasks on large volumes of data effectively in lesser time. It includes several implementations achieved through algorithms such as linear regression, logistic regression, Naïve Bayes, k-means, K nearest neighbor, and Random Forest.

**Python in Machine Learning**

Python has libraries that enables developers to use optimized algorithms. It implements popular machine learning techniques such as recommendation, classification, and clustering. Therefore, it is necessary to have a brief introduction to machine learning before we move further.

**What is Machine Learning?**

Data science, machine learning and artificial intelligence are some of the top trending topics in the tech world today. Data mining and Bayesian analysis are trending and this is adding the demand for machine learning. This tutorial is your entry into the world of machine learning.

Machine learning is a discipline that deals with programming the systems so as to make them automatically learn and improve with experience. Here, learning implies recognizing and understanding the input data and taking informed decisions based on the supplied data. It is very difficult to consider all the decisions based on all possible inputs. To solve this problem, algorithms are developed that build knowledge from a specific data and past experience by applying the principles of statistical science, probability, logic, mathematical optimization, reinforcement learning, and control theory.

**Applications of Machine Learning Algorithms**

The developed machine learning algorithms are used in various applications such as:

 Vision processing

 Language processing

 Forecasting things like stock market trends, weather

 Pattern recognition

 Games

 Data mining

 Expert systems

 Robotics

**Libraries and Packages**

To understand machine learning, you need to have basic knowledge of Python programming. In addition, there are a number of libraries and packages generally used in performing various machine learning tasks as listed below:

 **numpy** - is used for its N-dimensional array objects

 **pandas** – is a data analysis library that includes dataframes

 **matplotlib** – is 2D plotting library for creating graphs and plots

 **scikit-learn** - the algorithms used for data analysis and data mining tasks

 **seaborn** – a data visualization library based on matplotlib

**Installation**

You can install software for machine learning in any of the two methods as discussed here:

**Method 1**

Download and install Python separately from **python.org** on various operating systems as explained below:

To install Python after downloading, double click the **.exe** (for Windows) or .**pkg** (for Mac) file and follow the instructions on the screen.

For Linux OS, check if Python is already installed by using the following command at the prompt:

$ python --version. ...

If Python 2.7 or later is not installed, install Python with the distribution's package manager. Note that the command and package name varies.

On Debian derivatives such as Ubuntu, you can use **apt**:

$ sudo apt-get install python3

Now, open the command prompt and run the following command to verify that Python is installed correctly:

$ python3 --version

Python 3.6.2

**Python Machine Learning – Environment Setup**

Similarly, we can download and install necessary libraries like numpy, matplotlib etc. individually using installers like **pip**. For this purpose, you can use the commands shown here:

$pip install numpy

$pip install matplotlib

$pip install pandas

$pip install seaborn

**Method 2**

Alternatively, to install Python and other scientific computing and machine learning packages simultaneously, we should install **Anaconda** distribution. It is a Python implementation for Linux, Windows and OSX, and comprises various machine learning packages like numpy, scikit-learn, and matplotlib. It also includes **Jupyter Notebook**, an interactive Python environment. We can install Python 2.7 or any 3.x version as per our requirement.

To download the free Anaconda Python distribution from Continuum Analytics, you can do the following:

Visit the official site of Continuum Analytics and its download page. Note that the installation process may take 15-20 minutes as the installer contains Python, associated packages, a code editor, and some other files. Depending on your operating system, choose the installation process as explained here:

**For Windows:** Select the **Anaconda for Windows** section and look in the column with Python 2.7 or 3.x. You can find that there are two versions of the installer, one for 32-bit Windows, and one for 64-bit Windows. Choose the relevant one.

**For Mac OS:** Scroll to the **Anaconda for OS X** section. Look in the column with Python 2.7 or 3.x. Note that here there is only one version of the installer: the 64-bit version.

**For Linux OS:** We select the "Anaconda for Linux" section. Look in the column with Python 2.7 or 3.x.

Note that you have to ensure that Anaconda’s Python distribution installs into a single directory, and does not affect other Python installations, if any, on your system.

To work with graphs and plots, we will need these Python library packages: **matplotlib** and **seaborn**.

If you are using Anaconda Python, your system already has numpy, matplotlib, pandas, seaborn, etc. installed. We start the Anaconda Navigator to access either Jupyter Note book or Spyder IDE of python.

After opening either of them, type the following commands:

import numpy

import matplotlib Python Machine Learning

Now, we need to check if installation is successful. For this, go to the command line and type in the following command:

$ python

Python 3.6.3 |Anaconda custom (32-bit)| (default, Oct 13 2017, 14:21:34)

[GCC 7.2.0] on linux

Next, you can import the required libraries and print their versions as shown:

>>>import numpy

>>>print numpy.\_\_version\_\_

1.14.2

>>> import matplotlib

>>> print (matplotlib.\_\_version\_\_)

2.1.2

>> import pandas

>>> print (pandas.\_\_version\_\_)

0.22.0

>>> import seaborn

>>> print (seaborn.\_\_version\_\_)

**Python Machine Learning**

**Machine Learning (ML)** is an automated learning with little or no human intervention. It involves programming computers so that they learn from the available inputs. The main purpose of machine learning is to explore and construct algorithms that can learn from the previous data and make predictions on new input data.

The **input** to a learning algorithm is training data, representing experience, and the **output** is any expertise, which usually takes the form of another algorithm that can perform a task. The input data to a machine learning system can be numerical, textual, audio, visual, or multimedia. The corresponding output data of the system can be a floating-point number, for instance, the velocity of a rocket, an integer representing a category or a class, for example, a pigeon or a sunflower from image recognition.

In this chapter, we will learn about the training data our programs will access and how learning process is automated and how the success and performance of such machine learning algorithms is evaluated.

**Concepts of Learning**

Learning is the process of converting experience into expertise or knowledge.

Learning can be broadly classified into three categories, as mentioned below, based on the nature of the learning data and interaction between the learner and the environment.

 Supervised Learning

 Unsupervised Learning

 Semi-supervised learning

Similarly, there are four categories of machine learning algorithms as shown below:

 Supervised learning algorithm

 Unsupervised learning algorithm

 Semi-supervised learning algorithm

 Reinforcement learning algorithm

However, the most commonly used ones are **supervised** and **unsupervised learning**.

**Supervised Learning**

Supervised learning is commonly used in real world applications, such as face and speech recognition, products or movie recommendations, and sales forecasting. Supervised learning can be further classified into two types: **Regression** and **Classification**.

**Regression** trains on and predicts a continuous-valued response, for example predicting real estate prices.

**Python Machine Learning – Types of Learning**

**Classification** attempts to find the appropriate class label, such as analyzing positive/negative sentiment, male and female persons, benign and malignant tumors, secure and unsecure loans etc.

In supervised learning, learning data comes with description, labels, targets or desired outputs and the objective is to find a general rule that maps inputs to outputs. This kind of learning data is called **labeled data**. The learned rule is then used to label new data with unknown outputs.

Supervised learning involves building a machine learning model that is based on **labeled samples**. For example, if we build a system to estimate the price of a plot of land or a house based on various features, such as size, location, and so on, we first need to create a database and label it. We need to teach the algorithm what features correspond to what prices. Based on this data, the algorithm will learn how to calculate the price of real estate using the values of the input features.

Supervised learning deals with learning a function from available training data. Here, a learning algorithm analyzes the training data and produces a derived function that can be used for mapping new examples. There are many **supervised learning algorithms** such as Logistic Regression, Neural networks, Support Vector Machines (SVMs), and Naive Bayes classifiers.

Common **examples** of supervised learning include classifying e-mails into spam and not-spam categories, labeling webpages based on their content, and voice recognition.

**Unsupervised Learning**

Unsupervised learning is used to detect anomalies, outliers, such as fraud or defective equipment, or to group customers with similar behaviors for a sales campaign. It is the opposite of supervised learning. There is no labeled data here.

When learning data contains only some indications without any description or labels, it is up to the coder or to the algorithm to find the structure of the underlying data, to discover hidden patterns, or to determine how to describe the data. This kind of learning data is called **unlabeled data**.

Suppose that we have a number of data points, and we want to classify them into several groups. We may not exactly know what the criteria of classification would be. So, an unsupervised learning algorithm tries to classify the given dataset into a certain number of groups in an optimum way.

Unsupervised learning algorithms are extremely powerful tools for analyzing data and for identifying patterns and trends. They are most commonly used for clustering similar input into logical groups. Unsupervised learning algorithms include Kmeans, Random Forests, Hierarchical clustering and so on.

1If some learning samples are labeled, but some other are not labeled, then it is semi-supervised learning. It makes use of a large amount of **unlabeled data for training** and a small amount of **labeled data for testing**. Semi-supervised learning is applied in cases where it is expensive to acquire a fully labeled dataset while more practical to label a small subset. For example, it often requires skilled experts to label certain remote sensing images, and lots of field experiments to locate oil at a particular location, while acquiring unlabeled data is relatively easy. **Reinforcement Learning**

Here learning data gives feedback so that the system adjusts to dynamic conditions in order to achieve a certain objective. The system evaluates its performance based on the feedback responses and reacts accordingly. The best known instances include self-driving cars and chess master algorithm AlphaGo.

**Purpose of Machine Learning**

Machine learning can be seen as a branch of AI or Artificial Intelligence, since, the ability to change experience into expertise or to detect patterns in complex data is a mark of human or animal intelligence.

As a field of science, machine learning shares common concepts with other disciplines such as statistics, information theory, game theory, and optimization.

As a subfield of information technology, its objective is to program machines so that they will learn.

However, it is to be seen that, the purpose of machine learning is not building an automated duplication of intelligent behavior, but using the power of computers to complement and supplement human intelligence. For example, machine learning programs can scan and process huge databases detecting patterns that are beyond the scope of human perception.

**CHAPTER 5**

**SYSTEM ANALYSIS**

**5.1 EXHISTING SYSTEM**

Credit cards are widely used due to the popularization of ecommerce and the development of mobile intelligent devices. Card-not-present transactions (i.e., online transaction without a physical card) are more popular, especially all credit card operations are performed by web payment gateways, e.g., PayPal and Alipay. Credit card has made an online transaction easier and more convenient. However, there is a growing trend of transaction frauds resulting in great losses of money every year.

**Disadvantage:**

1. The main disadvantage of the existing system is the detection occurs only after gets a written complaint.

**PROBLEM STATEMENT:**

Credit card fraud detection is challenging task for the users online payment does not require physical card. And if anyone who knows the details of card can make the transactions currently, card holder comes to know only after the fraud transaction is carried out to proper mechanism are there to track the fraud transaction.

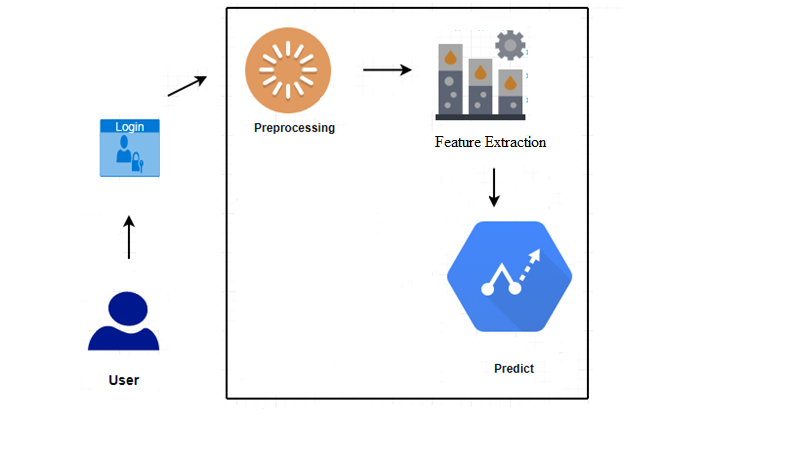
**5.2 PROPOSED SYSTEM**

In proposed system, we use misuse method which can ask the computer to find out whether it’s credit card fraud or not. In this story, we used Random Forest algorithm that analyzes and predicts the fraud and non-fraud/valid transactions.

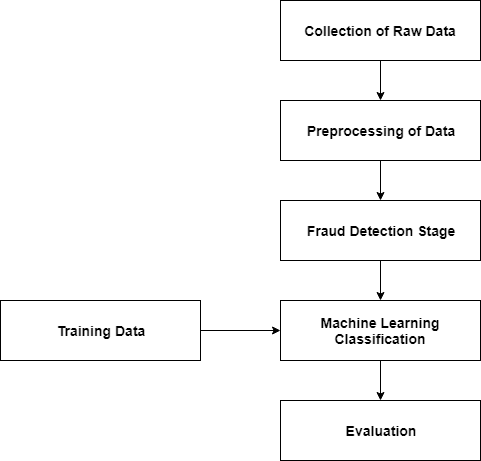
Advantages:

* Performance is good.
* Reduces the time required to predict the output.
* Used for real time predictions of fraud transactions.

**5.3 SYSTEM ARCHITECTURE**

****

**Flow Diagram(Random Forest):**



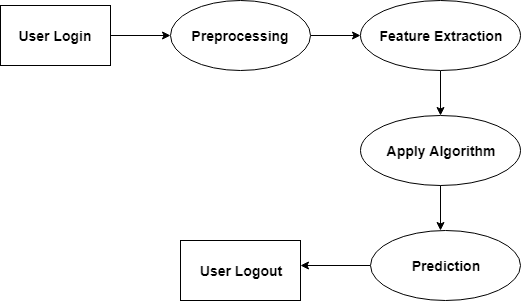
**CHAPTER 6**

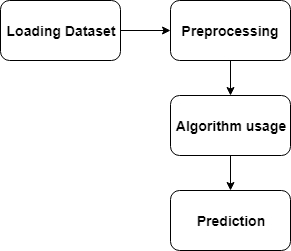
**SYSTEM DESIGN**

System design is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

**6.1 Data Flow Diagram(DFD)**

A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.





**6.2 USE CASE DIAGRAM:**

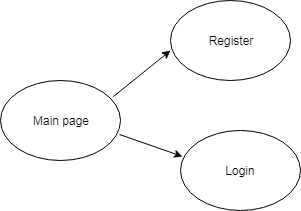
**UML Diagrams**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created, by the Object Management Group.

**Use Case Diagrams**

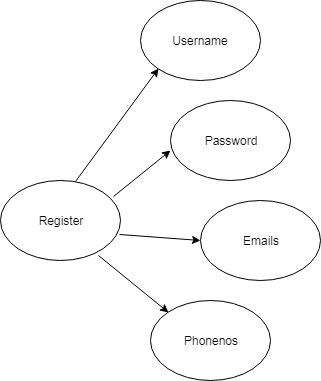
A use case diagram at its simplest is a graphical representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system.

Level-1



In a main program open it will be displayed by on a register and login button.

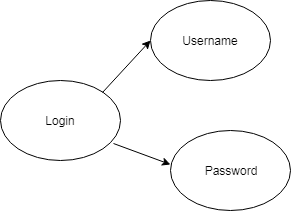
Level-2



If you click on register button fields will be open (username, password, emails, and phonenos) .

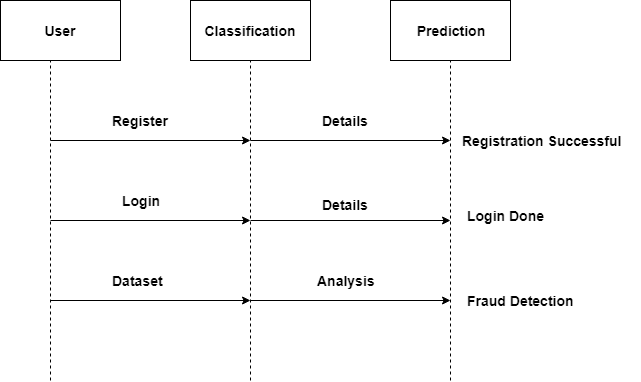
After fill with a fields you can signup. i will show register done then it will open login page.

Level-3



The login page will be displayed then we should enter a username and password (wrong password) I will show error, we have give a exact username and password program will be executed.

**6.3 SEQUENCE DIAGRAM**

****

**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

**List of Algorithms:**

1. Random Forest
2. SVM

Random Forest algorithm works as follows:

* 1. Load the dataset.
  2. Label the data as training and testing set.
  3. Train the classifier using Random Forest algorithm.
  4. Then fit the Random Forest model with the data.
  5. Total count of fraudulent cases and valid transactions is done.
  6. Finally percentage of accuracy is recognized.

Support Vector Machine works as follows:

1. Load the dataset.

2. Label the data as training and testing set.

3. Train the Classifier using Support Vector Machine algorithm.

4. Then fit the Random Forest model with the data.

5. Total count of fraudulent cases and valid transactions is done.

6. Finally percentage of accuracy is recognized.

Naïve Bayes works as follows:

1. Load the dataset.

2. Label the data as training and testing set.

4. Train the classifier using Naïve Bayes algorithm.

5. Then fit the Random Forest model with the data.

6. Total count of fraudulent cases and valid transactions is done.

7. Finally percentage of accuracy is recognized.

K-Nearest Neighbors works as follows:

1. Load the dataset.

2. Label the data as training and testing set.

3. Train the classifier using K-Nearest Neighbors algorithm.

4. Then fit the Random Forest model with the data.

5. Total count of fraudulent cases and valid transactions is done.

6. Finally percentage of accuracy is recognized.

**List of Modules:**

1. **Registration Module:**
2. **Login Module**
3. **Prediction Module**

**1.Registration Module:**

1. In this module the user registers the project.
2. After registration the credentials are saved in the My-SQL
3. Then user can login the project.



**2.Login Module:**

1. In this module the user has the login credentials.
2. He can login with the registered details.
3. Then after logging in successfully he can access the project.

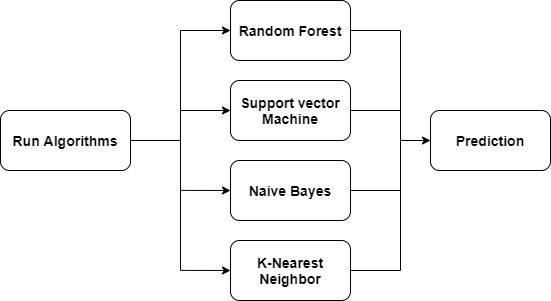


**3.Prediction Module:**

a) In this module the user can access the project.

b) The project is run and the output is successfully achieved.

c) The bar graph is also displayed for the representation of the results.



**CHAPTER 8**

**TESTING:**

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing an transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

**Test objectives**

* Testing is a process of executing a program with the intent of finding an error.
* A good case is one that has a high probability of finding an undiscovered error.
* A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software. Testing can't show the absences of defects are present. It can only show that software defects are present.

**Testing principles**

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

**Testing design**

Any engineering product can be tested in one of two ways:

**White box Testing**

This testing is also called as glass box testing. Inthis testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function.

it is a test case design method that uses the control structure of the procedural design to derive test cases.

**Black box Testing**

Inthis testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**Testing strategies**

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

**Any software testing strategy should have the following characteristics:**

* 1. Testing begins at the module level and works outward toward the integration of the entire computer based system.
  2. Different testing techniques are appropriate at different points in time.
  3. The developer of the software and an independent test group conducts testing.
  4. Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

**Levels of Testing**

Testing can be done in different levels of SDLC. They are:

**Unit Testing**

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules.

The Test cases in unit testing are as follows:

Table I: Unit Test Case 1

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 1 |
| Description | User registered to the system. |
| Input | User entered registration details. |
| Expected output | Registered successfully |
| Actual Result/Remarks | Got the expected output. |
| Passed(?) | Yes. |

Table II: Unit Test Case 2

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 2 |
| Description | User login to the system. |
| Input | User enter login details. |
| Expected output | Login into the system. |
| Actual Result/Remarks | Got the expected output. |
| Passed(?) | Yes |

Table III: Unit Test Case 3

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 3 |
| Description | Click on the prediction option to go to the next window. |
| Input | Click the prediction button. |
| Expected output | Got next window to display the predicted output. |
| Actual Result/Remarks | Got the expected output . |
| Passed(?) | Yes. |

Table III: Unit Test Case 4

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 4 |
| Description | The output window appears and displays the predicted output with graph using Random Forest Alogorithm. |
| Input | The final output window is displayed |
| Expected output | Predicts the fraudulent and non-fraudulent credit card transactions and prints the accuracy. |
| Actual Result/Remarks | Got the expected output with graph. |
| Passed(?) | Yes. |

**Integration Testing**

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

* + - *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards throw the control hierarchy beginning with the main control module.
    - *Bottom-Up Integration*: Bottom up integration as its name implies, begins Construction and testing with automatic modules.
    - *Regression Testing*: In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagated unintended side effects.

**9.5.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Table X: Functional Testing items

|  |  |
| --- | --- |
| Valid Input | Identified classes of valid input must be accepted. |
| Invalid Input | Identified classes of invalid input must be rejected. |
| Functions | Identified functions must be exercised. |
| Output | Identified classes of application outputs must be exercised. |

***Systems/Procedures:*** Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**Validation testing**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title “validation criteria”. Information contained in that section forms the basis for validation testing approach

**Alpha testing**

It is virtually impossible for a software developer to forsee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer’s site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

**Beta testing**

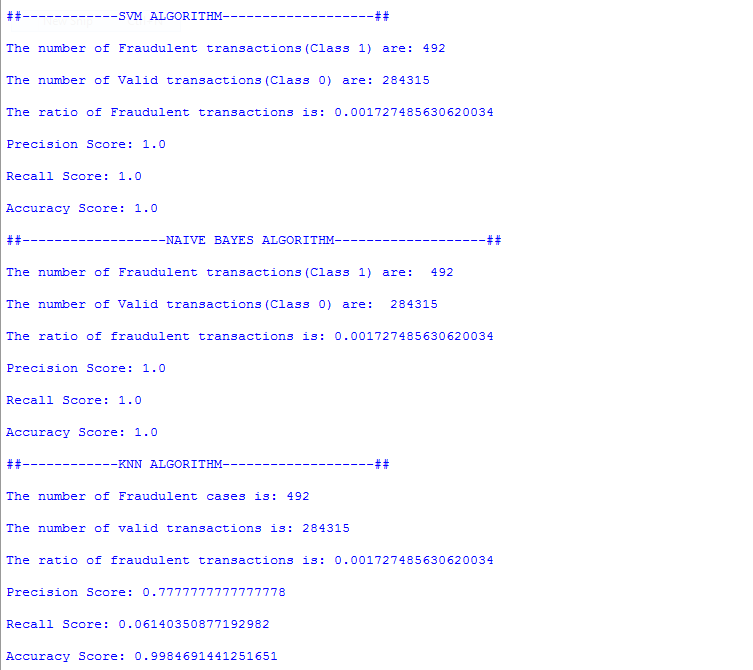
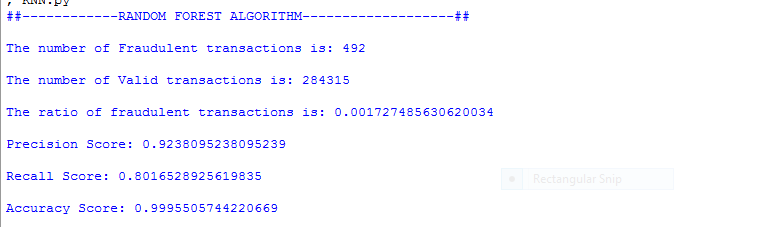
The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

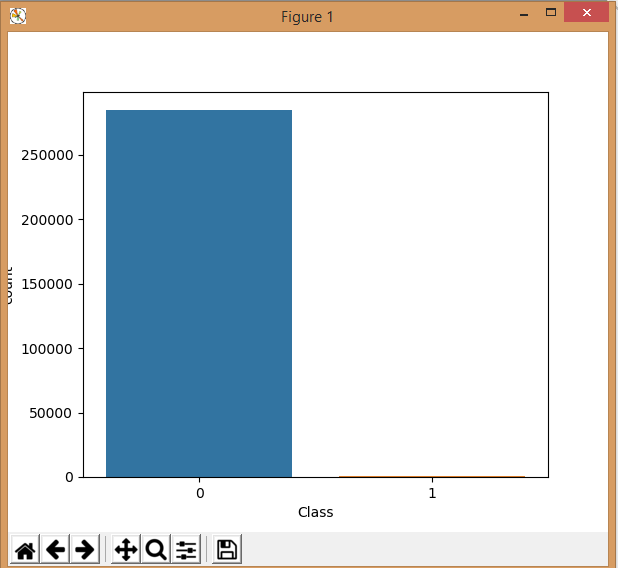
**System Testing and Acceptance Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

**Screenshots**

****



**CHAPTER 9**

**RESULTS**

The Credit Card’s fraudulent and non-fraudulent transactions using Random Forest algorithm, Niave Bayes algorithm, KNN algorithm and SVM algorithm is shown with the graph and the accuracy.

**CHAPTER 10**

**CONCLUSION**

This paper has examined the performance of Random Forest, SVM, Naïve Bayes and KNN algorithms. A real-life B2C dataset on credit card transactions is used in our experiment. Our future work will focus on solving these problems. The algorithm of random forest itself should be improved. Therefore, we also try to make some improvement for this algorithm.

**REFERENCES**

[1] Gupta, Shalini, and R. Johari. ”A New Framework for Credit Card Transactions Involving Mutual Authentication between Cardholder and Merchant.” International Conference on Communication Systems and Network Technologies IEEE, 2011:22-26.

[2] Y. Gmbh and K. G. Co, “Global online payment methods: Full year 2016,” Tech. Rep., 3 2016.

[3] Bolton, Richard J., and J. H. David. ”Unsupervised Profiling Methods for Fraud Detection.” Proc Credit Scoring and Credit Control VII (2001):5–7.

[4] Seyedhossein, Leila, and M. R. Hashemi. ”Mining information from credit card time series for timelier fraud detection.” International Symposium on Telecommunications IEEE, 2011:619-624.

[5] Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit card fraud detection using hidden markov model. IEEE Transactions on Dependable and Secure Computing, 5(1), 37-48.

[6] Drummond, C., and Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: why under-sampling beats oversampling. Proc of the Icml Workshop on Learning from Imbalanced Datasets II, 1–8.

[7] Quah, J. T. S., and Sriganesh, M. (2008). Real-time credit card fraud

detection using computational intelligence. Expert Systems with Applications,

35(4), 1721-1732.

[8] Kundu, A., Panigrahi, S., Sural, S., and Majumdar, A. K. (2009). Blastssaha hybridization for credit card fraud detection. IEEE Transactions on Dependable and Secure Computing, 6(4), 309-315.

[9] Shi, E., Niu, Y., Jakobsson, M., and Chow, R. (2010). Implicit Authentication through Learning User Behavior. International Conference on Information Security (Vol.6531, pp.99-113). Springer-Verlag.

[10] Duman, E., and Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. Expert Systems with Applications, 38(10), 13057-13063.

[11] Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C. (2011). Data mining for credit card fraud: a comparative study. Decision Support Systems, 50(3), 602-613.

[12] Sahin, Y., and Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines. Lecture Notes in Engineering and Computer Science, 2188(1).

[13] Mota, G., Fernandes, J., and Belo, O. (2014). Usage signatures analysis an alternative method for preventing fraud in E-Commerce applications. International Conference on Data Science and Advanced Analytics (pp.203-208). IEEE.

[14] Behdad, M., Barone, L., Bennamoun, M., and French, T. (2012). Natureinspired techniques in the context of fraud detection. IEEE Transactions on Systems Man and Cybernetics Part C, 42(6), 1273-1290.

[15] Ju, W. H., and Vardi, Y. (2001). A hybrid high-order markov chain model for computer intrusion detection. Journal of Computational and Graphical Statistics, 10(2), 277-295.

[16] Bolton, R. J., and Hand, D. J. (2002). Statistical fraud detection: a review. Statistical Science, 17(3), 235-249.

[17] Vlasselaer, V. V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., and Snoeck, M., et al. (2015). Apate : a novel approach for automated credit card transaction fraud detection using network-based extensions. Decision Support Systems, 75, 38-48.

[18] Chan, P. K., Fan, W., Prodromidis, A. L., and Stolfo, S. J. (2002).Distributed data mining in credit card fraud detection. IEEE Intelligent Systems and Their Applications, 14(6), 67-74.

[19] RONG-CHANG CHEN, TUNG-SHOU CHEN, and CHIH-CHIANG LIN. (2006). A new binary support vector system for increasing detectionrate of credit card fraud. International Journal of Pattern Recognitionand Artificial Intelligence, 20(02), 227-239.

[20] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[21] Dietterich, T. G. (2000). Ensemble methods in machine learning. , 1857(1), 1-15.

[22] Abeel, T., de Peer, Y. V. and Saeys, Y. Java-ML: A Machine LearningLibrary, Journal of Machine Learning Research, 2009, 10, 931-934

[23] Quinlan, J. R. (1986). Induction on decision tree. Machine Learning, 1(1), 81-106.

[24] Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. J. (1984). Classification and regression trees. Biometrics, 40(3), 358.