

##### DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

**A PROJECT REPORT ON**

#### “CREDIT CARD FRAUD DETECTION”

*Submitted in the partial fulfillment of the requirements in the 8th semester of*

##### BACHELOR OF ENGINEERING IN

**INFORMATION SCIENCE AND ENGINEERING**

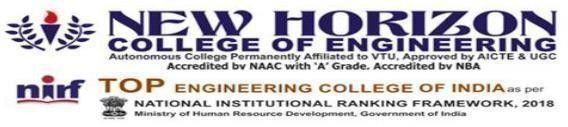
By

##### ASHWINI H V 1NH15IS143

***Under the guidance of***

##### Mrs. BASWARAJU SWATHI

Sr. Assistant Professor, Dept. of ISE, NHCE



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

#### CERTIFICATE

Certified that the project work entitled “**CREDIT CARD FRAUD DETECTION**”, carried out by **Ms. ASHWINI H V, USN-1NH15IS143**, a bonafide student of NEW HORIZON COLLEGE OF ENGINEERING, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Information Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2018-19.It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

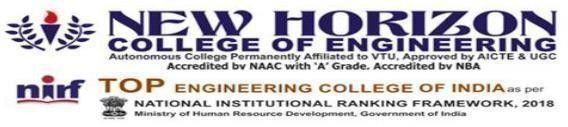
|  |  |  |
| --- | --- | --- |
| ……………………. | ……………………. | ……………………. |
| **Prof. Baswaraju Swathi** | **Dr. R J Anandhi** | **Dr. Manjunatha** |
| **Guide, Dept. of ISE** | **HOD, Dept. of ISE** | **Principal, NHCE** |

**External Viva**

**Name of the Examiners Signature with Date**

###### 1.

**2.**



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

#### DECLARATION

I hereby declare that I have followed the guidelines provided by the Institution in preparing the project report and presented report of project titled “**CREDIT CARD FRAUD DETECTION**”, and is uniquely prepared by me after the completion of the project work. I also confirm that the report is only prepared for my academic requirement and the results embodied in this report have not been submitted to any other University or Institution for the award of any degree.

**Signature of the Student Name: ASHWINI H V USN: 1NH15IS143**

## ABSTRACT

Nowadays the usage of credit cards has dramatically increased. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. Here we model the sequence of operations in credit card transaction processing using a Hidden Markov Model (HMM) and show how it can be used for the detection of frauds. An HMM is initially trained with the nonnal behavior of a cardholder. If an incoming credit card transaction is not accepted by the trained HMM with sufficiently high probability, it is considered to be fraudulent. At the same time, we try to ensure that genuine transactions are not rejected. We present experimental results to show the effectiveness of our approach and compare it with other techniques available in the literature.

## ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project. I would like to take an opportunity to thank them all.

First and foremost I thank the management, **Dr. Mohan Manghnani,** Chairman, New Horizon Educational Institutions for providing us the necessary state of art infrastructure to do Project.

I would like to thank **Dr.Manjunatha**, Principal, New Horizon College of Engineering, Bengaluru, for his valuable suggestions and expert advice.

I would like to thank **Dr. R J Anandhi**, Professor and Head of the Department, Information Science and Engineering, New Horizon College of Engineering, Bengaluru, for constant encouragement and support extended towards completing my Project.

I deeply express my sincere gratitude to my guide **Mrs. Baswaraju Swathi**, Assistant professor, Department of ISE, New Horizon College of Engineering, Bengaluru, for his/her able guidance, regular source of encouragement and assistance throughout my project period.

Last, but not the least, I would like to thank my peers and friends who provided me with valuable suggestions to improve my project.

**ASHWINI H V (1NH15IS143)**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **ABSTRACT** | **i** |
| **ACKNOWLEDGEMENT** | **ii** |
| **TABLE OF CONTENTS** | **iii** |
| [**LIST OF FIGURES**](#_bookmark0) | [**vi**](#_bookmark0) |
| **CHAPTER** | **Page No.** |
| **1. PREAMBLE** | **01** |
| 1.1Introduction | 01 |
| 1.2Relevance of the Project | 01 |
| 1.3Problem Statement and Definition | 02 |
| 1.4Purpose | 03 |
| 1.5Objective of the Study | 04 |
| 1.6Existing System | 05 |
| 1.7Proposed System | 06 |
| 1.7.1 Advantages of the System | 06 |
| **2. LITERATURE SURVEY** | **07** |
| **3. SYSTEM REQUIREMENTS AND SPECIFICATIONS** | **09** |
| 3.1General Description of the System | 09 |
| 3.1.1Overview of functional requirements | 09 |
| 3.1.2Overview of data requirements | 11 |
| 3.2Technical Requirements of the System | 13 |

* + 1. Hardware Requirements 13
    2. Software Requirements 13
  1. Language Specifications 14
     1. Python Introduction 14
     2. Machine Learning Features 15

1. SYSTEM REQUIREMENTS AND ANALYSIS 16
   1. [System Architecture 16](#_TOC_250011)
   2. [Data Flow Diagram 17](#_TOC_250010)
      1. DFD for Data Extraction 18
      2. DFD for Classification of Data 18
   3. [Use Case Diagram 19](#_TOC_250009)
2. IMPLEMENTATION 20
   1. [Different Modules of the Project 31](#_TOC_250008)
   2. Flow Chart of the Proposed System 34
3. EXPERIMENTS RESULTS 35
   1. Outcomes of the Proposed System 35
4. TESTING 38
   1. Testing and Validations 38
   2. [Testing Levels 38](#_TOC_250007)
      1. Functional Testing 39
      2. Non-Functional Testing 39
   3. [White Box Testing 41](#_TOC_250006)
   4. [Different Stages of Testing 43](#_TOC_250005)
      1. [Unit Testing 43](#_TOC_250004)
      2. [Integration Testing 44](#_TOC_250003)
      3. [System Testing 46](#_TOC_250002)
      4. [Acceptance testing 47](#_TOC_250001)
5. CONCLUSION AND FUTURE ENHANCEMENT 48

Conclusion 48

Future Enhancement 48

[REFERENCES 49](#_TOC_250000)

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig NO** | **Name** | **Page no** |
| 1 | Data Requirements | 13 |
| 2 | System Architecture | 16 |
| 3 | Data Flow Diagram | 17 |
| 4 | DFD for Data Extraction | 18 |
| 5 | DFD for classification of data | 18 |
| 6 | Use case Diagram | 19 |
| 7 | Training test | 32 |

## CHAPTER 1

* 1. **INTRODUCTION:**

## PREAMBLE

Today, all around the world data is available very easily; from small to big organizations are storing information that has high volume, variety, speed and worth. This information comes from tons of sources like social media followers, likes and comments, user’s purchase behaviors. All this information used for analysis and visualization of the hidden data pattern. Early analysis of big data was centered primarily on data volume, for example, general public database, biometrics, financial analysis.

For frauds, the credit card is an easy and friendly target because without any risk a significant amount of money is obtained within a short period. To commit credit card fraud, fraudsters try to steal sensitive information such as credit card number, bank account and social security number. Fraudsters try to make every fraudulent transaction legitimate which makes fraud detection a challenging problem. Increased credit card transactions show that approximately 70% of the people in the US can fall into the trap of these fraudsters.

Credit card dataset is highly imbalanced dataset because it carries more legitimate transactions as compared to the fraudulent one. That means prediction will get very high accuracy score without detecting a fraud transaction. To handle this kind of problem one better way is to class distribution, i.e., sampling minority classes. In sampling minority, class training example can be increased in proportion the majority class to raise the chance of correct prediction by the algorithm.

##### Relevance of the Project

In general, a fraud is defined as a crime committed with intention to damage a person and is also a violation. Fraud may be committed for various reasons: for entertainment, to exploit a business / an organization, to take revenge, to cause financial loss, to damage identity and etc. Also there are several types of frauds: bankruptcy frauds, identity thefts, health frauds, religious frauds, credit card frauds, insurance frauds, forgery, tax frauds and many more. Here considering only the credit card frauds, they can be of two kinds:

1. Offline credit card frauds and
2. Online credit card frauds.

Offline credit card frauds are those where an individual’s credit card is lost or stolen. If any attacker or

hacker, hack the details and use it to commit illegal actions is referred as online frauds. With the rapidly

developing technology, usage of internet is drastically increasing. Substantially, this is leading to many credit-card fraudulent activities.

##### PROBLEM STATEMENT AND EXPLANATION

In the past recent years, credit card breaches have been trending alarmingly. Therefore, it is necessary to develop credit card fraud detection techniques as the counter measure to combat illegal activities.

There are lots of issues that make this procedure tough to implement and one of the biggest problems associated with fraud detection is the lack of both the literature providing experimental results and of real world data for academic researchers to perform experiments on. The reason behind this is the sensitive financial data associated with the fraud that has to be kept confidential for the purpose of customer’s privacy. Now, here we enumerate different properties a fraud detection system should have in order to generate proper results:

The system should be able to handle skewed distributions, since only a very small percentage of all credit card transactions is fraudulent

There should be a proper means to handle the noise. Noise is the errors that is present in the data, for example, incorrect dates. This noise in actual data limits the accuracy of generalization that can be achieved, irrespective of how extensive the training set is.

Another problem related to this field is overlapping data. Many transactions may resemble fraudulent transactions when actually they are genuine transactions. The opposite also happens, when a fraudulent transactions appears to be genuine.

The systems should be able to adapt themselves to newkinds of fraud. Since after a while, successful fraud techniques decreases in efficiency due to the fact that they become well known because an efficient fraudster always find a new and inventive ways of performing his job.

These points direct us to the most important necessity of the fraud detection system, which is, a decision layer. The decision layer decides what action to take when fraudulent behavior is observed taking into account factors like, the frequency and amount of the transaction.

##### Purpose

There are two objectives of credit card fraud detection. It helps merchants and banks reduce the number of payment fraud cases and helps merchants grow their revenues.

To run a sustainable business, merchants need to make a profit, which is what’s left after deducting the

costs of doing business from a company's revenue. Therefore, a business’s tolerance for payment fraud is

a function of, among other things, its gross margin (sell price - cost of goods sold). The lower the margin , the lower the tolerance for payment fraud.

In practice, when fraud occurs, the cardholder disputes the charge and the debit is usually cancelled, which means either the cardholder’s bank or the merchant absorbs the loss (see [1] for more details). Cumulatively, fraud represents a significant financial risk to the merchant and the issuing bank. To reduce fraud, chip and pin technology, 3DSecure, and fraud detection techniques are used.

But if chip and pin technology and 3DSecure exist, why is fraud detection used? There are two main reasons.

First, the total cost of chip and pin technology and 3DSecure is relatively high compared to the cost of fraud detection. e.g., while online merchants care about conversion, 3DSecure reduces it by several percents (> 5%). Hence, when they have the option, many online merchants decide to deactivate 3DSecure and manage the risk of payment fraud themselves.

Second, adding more security layers to the buying process greatly reduces checkout velocity and, in turn, convenience for the buyer. While convenience for buyers may look like a fuzzy concept at first, for companies like Amazon, which pioneered one-click checkout, it’s a marketing argument and a means to convert and grow revenues.

##### Objective of the study

In recent years, topics such as fraud detection and fraud prevention have received a lot of attention on the research front, in particular from payment card issuers. The reason for this increase in research activity can be attributed to the huge annual financial losses incurred by card issuers due to fraudulent use of their card products. A successful strategy for dealing with fraud can quite literally mean millions of dollars in savings per year on operational costs.

Fraud prevention is interesting for financial institutions. The advent of new technologies as telephone, Automated Teller Machines (ATMs) and credit card systems have amplified the amount of fraud loss for many banks. Performing the analysis manually is literally impossible, while automation of this process might present a lot of practical difficulties.

Analyzing every transaction is legitimate or not is very expensive. Moreover it is also time consuming, hence it is not practically possible. Confirming whether a transaction was done by a client or a fraudster is a better option, but by phoning all card holders is cost prohibitive if it is check in all transactions.

Further it might also lead to customer dissatisfaction. Fraud prevention by automatic fraud detections is where the well-known classification methods can be applied, where pattern 89 recognition systems play a very important role. One can learn from past (fraud happened in the past) and classify new instances (transactions). Past data about the customer is available in huge amounts, which can be mined for useful data. This old data can be analyzed and the buying behavior of the user can be obtained. This pattern can be used for comparing with the current transactions and determining the legitimacy of the transaction.

Fraud detection model is among the most complicated models used for the credit card industry. Skewness of the data, search space dimensionality, different cost of false positive and false negative, durability of the model and short time-to-answer are among the problems one has to face in developing a fraud detection model.

##### Existing System

In general, credit card fraud detection has been known as the process of identifying whether transactions are genuine or fraudulent. As the data mining and machine learning techniques are vastly used to counter cyber-criminal cases, scholars often embraced those approaches to study and detect credit card fraud activities. While data mining focused on discovering valuable intelligence, machine learning is rooted in learning the intelligence and developing its own model for the purpose of classification, clustering or so on.

LIMITAIONS

* + - **Imbalanced data:**

The credit card fraud detection data has imbalanced nature. It means that very small percentages of all credit card transactions are fraudulent. This causes the detection of fraud transactions very difficult and imprecise.

* + - 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 low rate of false positive and false negative 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.

* + - **Lack of standard metrics:**

##### Proposed system

There is no standard evaluation criterion for assessing and comparing the results of fraud detection systems.

Our proposed system applies supervised machine learning algorithms to detect credit card fraudulent transactions using a real-world dataset. Furthermore, we employ these algorithms to implement a classifier using machine learning methods. We identify the most important variables that may lead to higher accuracy in credit card fraudulent transaction detection.

Advantages of Proposed System

* + - More accurate result.
    - Able to detect different fraudulent behavior.
    - Cost and Time efficient.

## CHAPTER 2

**LITERATURE SURVEY**

1. A Novel Approach for Credit Card Fraud Detection

In this research, “A Novel Approach for Credit Card Fraud Detection” is designed. Credit Card frauds are increasing as there are millions of users worldwide. To stop these fallacious transactions a technique is designed which uses the combination of Hidden Markov Model, Behavior Based Technique, and Genetic Algorithm. Each and every transaction is tested with above mentioned technique and Fraud Detection system test the transaction and detects fraud. The goal is to detect least and accurate false fraud detection.

1. Implementation of Novel Approach for Credit Card Fraud Detection

In this research work, it is tried to develop a technique for ‘Credit Card Fraud Detection’. Credit Card can be accepted for each online and offline in today’s world. There are combinations of methods used. Firstly, Shopping Behavior is based on which type of products customer buys. Secondly, Spending Behavior in this the fraud is detected based on the maximum amount spent. Thirdly, Hidden Markov Model in this technique profiles are maintained and statistics of a particular user and statistics of different fraud scenarios are clustered. Genetic Algorithm is used for calculation of threshold and accurate frauds. Finally average is taken out by summing the result. The main task of this research work is to explore different views of the same problem and see what can be learned from the application of each different technique.

1. Credit card fraud detection using Machine Learning Techniques

Financial fraud is an ever growing menace with far consequences in the financial industry. Data mining had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons – first, the profiles of normal and fraudulent behaviors change constantly and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used. This paper investigates the performance of naïve bayes, k-nearest neighbor and logistic regression on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,807 transactions. A hybrid technique of under-sampling and oversampling is carried out on the skewed data. The three techniques are applied on the raw and preprocessed data. The work is

implemented in Python. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision, Matthews’s correlation coefficient and balanced classification rate. The results shows of optimal accuracy for naïve bayes, k-nearest neighbor and logistic regression classifiers are 97.92%, 97.69% and 54.86% respectively. The comparative results show that k-nearest neighbor performs better than naïve bayes and logistic regression techniques.

1. Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier

Credit card fraud is increasing considerably with the development of modern technology and the global superhighways of communication. Credit card fraud costs consumers and the financial company billions of dollars annually, and fraudsters continuously try to find new rules and tactics to commit illegal actions. Thus, fraud detection systems have become essential for banks and financial institution, to minimize their losses. However, there is a lack of published literature on credit card fraud detection techniques, due to the unavailable credit card transactions dataset for researchers. The most commonly techniques used fraud detection methods are Naïve Bayes (NB), Support Vector Machines (SVM), K-Nearest Neighbor algorithms (KNN). These techniques can be used alone or in collaboration using ensemble or meta-learning techniques to build classifiers. But amongst all existing method, ensemble learning methods are identified as popular and common method, not because of its quite straight forward implementation, but also due to its exceptional predictive performance on practical problems. In this paper we trained various data mining techniques used in credit card fraud detection and evaluate each methodology based on certain design criteria.

## CHAPTER 3

**SYSTEM REQUIREMENTS AND SPECIFICATIONS**

##### GENERAL DESCRIPTION OF THE SYSTEM

A proper and thorough literature survey concludes that there are various methods that can be used to detect credit card fraud detection. Some of these approaches are:

* + 1. Artificial Neural Network
    2. Bayesian Network
    3. Neural Network
    4. Hidden Markov Method
    5. Genetic Algorithm

##### Overview of Functional requirements

Preprocess Data

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of Iterative Analysis. The set of steps is known as Data Preprocessing. It includes

Data Cleaning Data Integration

Data Transformation Data Reduction

Data Preprocessing is necessary because of the presence of unformatted real-world data. Mostly real- world data is composed of –

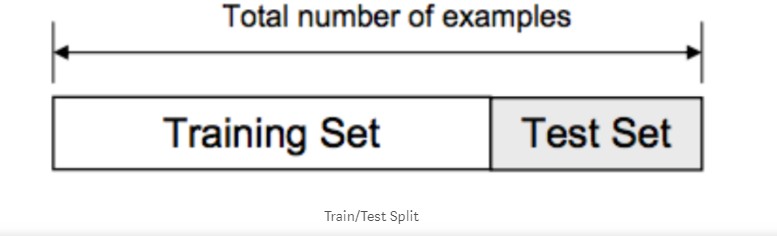
Inaccurate data (missing data) - There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics and much more.

The presence of noisy data (erroneous data and outliers) - The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and

much more.

Inconsistent data - The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more.

Train and Test Data Creation

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model’s prediction on this subset.

Model Creation

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from. The term ML model refers to the model artifact that is created by the training process.

The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

You can use the ML model to get predictions on new data for which you do not know the target. For example, let's say that you want to train an ML model to predict if an email is spam or not spam. You would provide training data that contains emails for which you know the target (that is, a label that tells whether an email is spam or not spam). Machine would train an ML model by using this data, resulting in a model that attempts to predict whether new email will be spam or not spam.

In our project we are using Random Forest Algorithm to build our Model on Credit Card Fraud Dataset. Result Analysis

In this final phase, we will test our model on our prepared dataset and also measure the Fraud detection

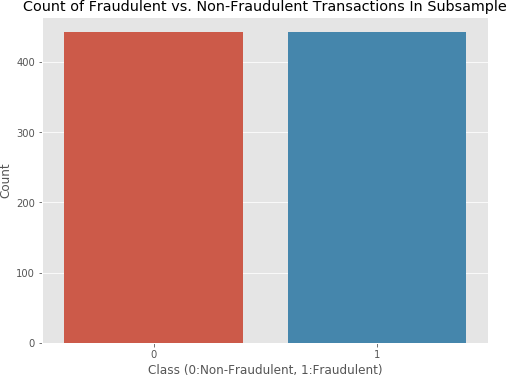
performance on our dataset. To evaluate the performance of our created classification and make it comparable to current approaches, we use Accuracy to measure the effectiveness of classifiers. We consider the Fraud class as a negative class and Non-Fraud class as a positive class.

* + 1. OVERVIEW OF DATA REQIREMENTS

Creating a training data set that will allow our algorithms to pick up the specific characteristics that make a transaction more or less likely to be fraudulent. Using the original data set would not prove to be a good idea for a very simple reason: Since over 99% of our transactions are non-fraudulent, an algorithm that always predicts that the transaction is non-fraudulent would achieve accuracy higher than 99%. Nevertheless, that is the opposite of what we want. We do not want a 99% accuracy that is achieved by never labeling a transaction as fraudulent, we want to detect fraudulent transactions and label them as such.

There are two key points to focus on to help us solve this. First, we are going to utilize **random under- sampling** to create a training dataset with a balanced class distribution that will force the algorithms to detect fraudulent transactions as such to achieve high performance. Speaking of performance, we are not going to rely on accuracy. Instead, we are going to make use of the Receiver Operating Characteristics- Area Under the Curve or ROC-AUC performance measure (I have linked further reading below this article). Essentially, the ROC-AUC outputs a value between zero and one, whereby one is a perfect score and zero the worst. If an algorithm has a ROC-AUC score of above 0.5, it is achieving a higher performance than random guessing.

To create our balanced training data set, I took all of the fraudulent transactions in our data set and counted them. Then, I randomly selected the same number of non-fraudulent transactions and concatenated the two. After shuffling this newly created data set, I decided to output the class distributions once more to visualize the difference.



* 1. Technical Requirements of the System
     1. Hardware Requirements

System Processor : Core i3 / i5

Hard Disk : 500 GB.

Ram : 4 GB.

*Any desktop / Laptop system with above configuration or higher level.*

* + 1. Software Requirements

Operating system : Windows 8 / 10 Programming Language : Python Framework : Anaconda

IDE : Jupyter Notebook

DL Libraries : Numpy, Pandas

* 1. LANGUAGE SPECIFICATION
     1. PYTHON INTRODUCTION:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic

typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, [*https://www.python.org/*,](https://www.python.org/) and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly

to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

* + 1. MACHINE LEARNING FEATURES

Machine Learning is a method of statistical learning where each instance in a dataset is described by a set of features or attributes. In contrast, the term “Deep Learning” is a method of statistical learning that extracts features or attributes from raw data. Deep Learning does this by utilizing neural networks with many hidden layers, big data, and powerful computational resources. The terms seem somewhat interchangeable, however, with Deep Learning method, The algorithm constructs representations of the data automatically. In contrast, data representations are hard-coded as a set of features in machine learning algorithms, requiring further processes such as feature selection and extraction, (such as PCA).

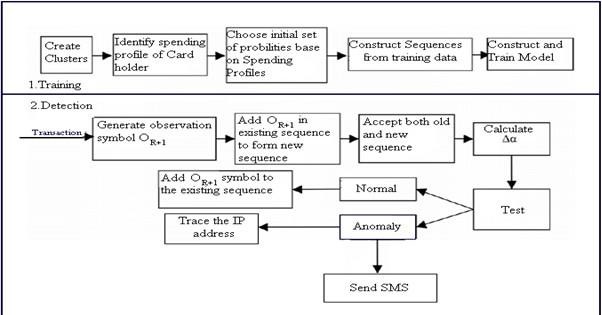
Both of these terms are in dramatic contrast with another class of classical artificial intelligence algorithms known as Rule-Based Systems where each decision is manually programmed in such a way that it resembles a statistical model.

In Machine Learning and Deep Learning, there are many different models that fall into two different categories, supervised and unsupervised. In unsupervised learning, algorithms such as k-Means, hierarchical clustering, and Gaussian mixture models attempt to learn meaningful structures in the data. Supervised learning involves an output label associated with each instance in the dataset. This output can be discrete/categorical or real-valued. Regression models estimate real-valued outputs, whereas classification models estimate discrete-valued outputs. Simple binary classification models have just two output labels, 1 (positive) and 0 (negative). Some popular supervised learning algorithms that are considered Machine Learning: are linear regression, logistic regression, decision trees, support vector machines, and neural networks, as well as non-parametric models such as k-Nearest Neighbors.

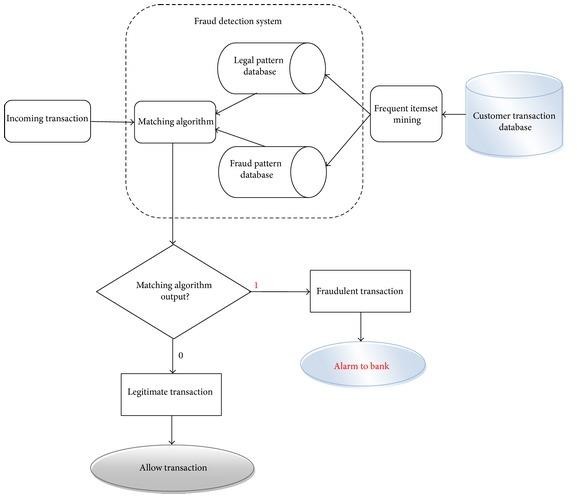
CHAPTER 4

SYSTEM DESIGN AND ANALYSIS

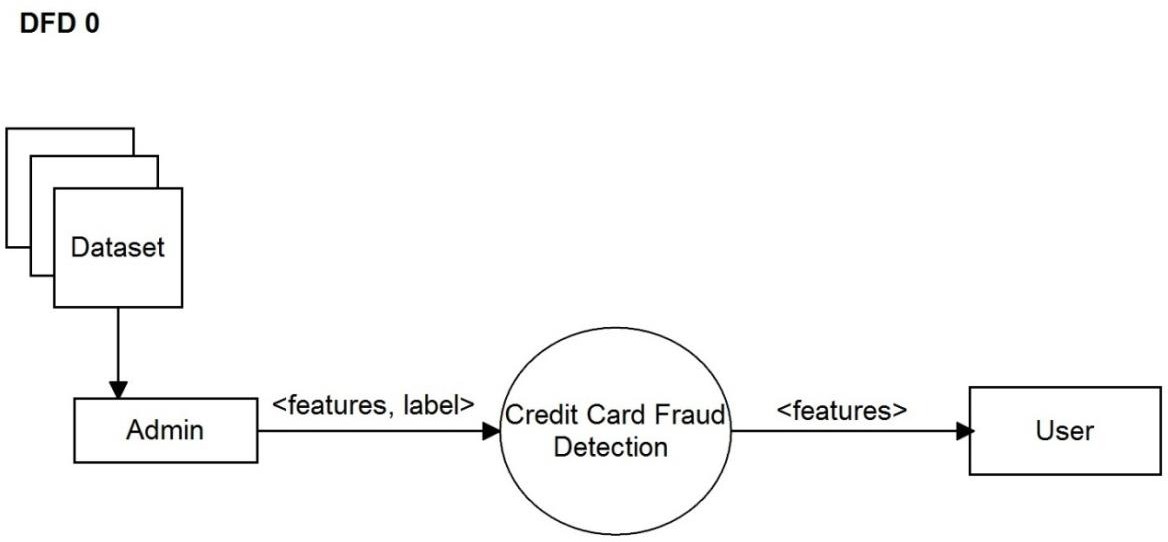
* 1. System Architecture



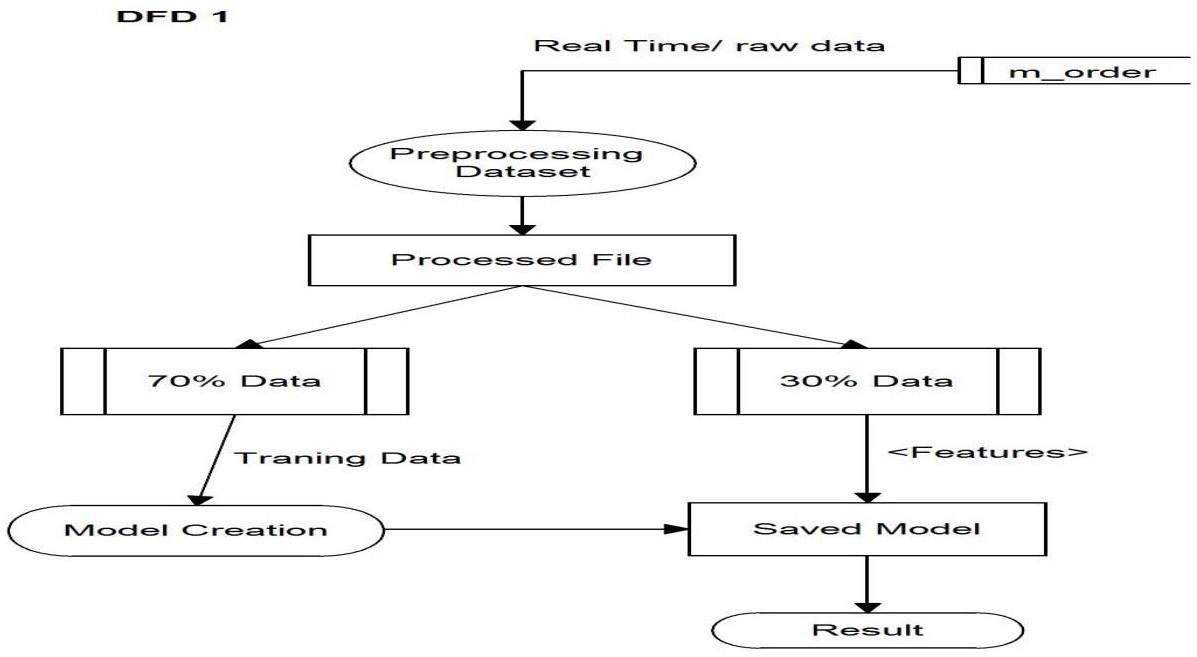
* 1. Data Flow Diagram



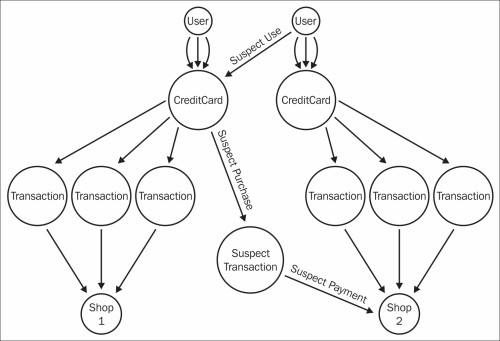
* + 1. DFD for Data Extraction



* + 1. DFD for Classification of Data



* 1. Use Case Diagram



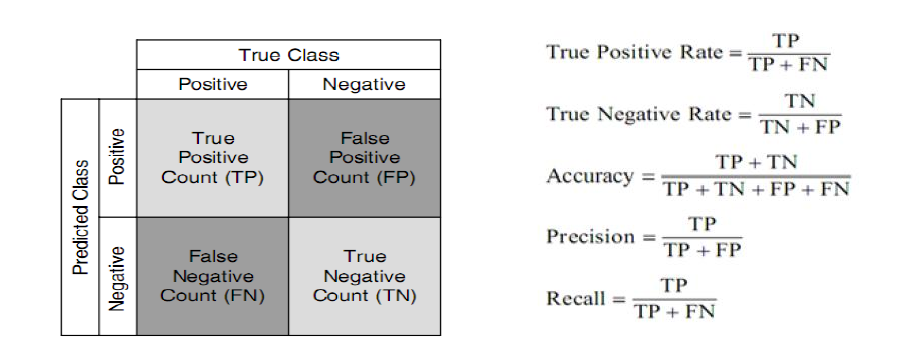
### CHAPTER 5 IMPLEMENTATION

In this work, a business intelligent model has been developed, to classify different animals, based on a specific business structure deal with Animal classification using a suitable machine learning technique. The model was evaluated by a scientific approach to measure accuracy. We are using Convolutional Neural Network (CNN) to build our model.

Analysis:

In this final phase, we will test our classification model on our prepared image dataset and also measure the performance on our dataset. To evaluate the performance of our created classification and make it comparable to current approaches, we use accuracy to measure the effectiveness of classifiers.

After model building, knowing the power of model prediction on a new instance, is very important issue. Once a predictive model is developed using the historical data, one would be curious as to how the model will perform on the data that it has not seen during the model building process. One might even try multiple model types for the same prediction problem, and then, would like to know which model is the one to use for the real-world decision making situation, simply by comparing them on their prediction performance (e.g., accuracy). To measure the performance of a predictor, there are commonly used performance metrics, such as accuracy, recall etc. First, the most commonly used performance metrics will be described, and then some famous estimation methodologies are explained and compared to each other. "Performance Metrics for Predictive Modeling In classification problems, the primary source of performance measurements is a coincidence matrix (classification matrix or a contingency table)”. Above figure shows a coincidence matrix for a two-class classification problem. The equations of the most commonly used metrics that can be calculated from the coincidence matrix



The numbers along the diagonal from upper-left to lower-right represent the correct decisions made, and the numbers outside this diagonal represent the errors. "The true positive rate (also called hit rate or recall) of a classifier is estimated by dividing the correctly classified positives (the true positive count) by the total positive count. The false positive rate (also called a false alarm rate) of the classifier is estimated by dividing the incorrectly classified negatives (the false negative count) by the total negatives. The overall accuracy of a classifier is estimated by dividing the total correctly classified positives and negatives by the total number of samples.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Flexibility

Sometimes you just don’t want to use what is already there but you want to define something of your own (for example a cost function, a metric, a layer, etc.). Although Keras 2 has been designed in such a way that you can implement almost everything you want but we all know that low-level libraries provides more flexibility. Same is the case with TF. *You can tweak* TF much more as compared to Keras.

Functionality

Although Keras provides all the general purpose functionalities for building Deep learning models, it doesn’t provide as much as TF. TensorFlow offers more advanced operations as compared to Keras. This comes very handy ifyou are doing a research or developing some special kind of deep learning models. Some examples regarding high level operations are:

Threading and Queues

Queues are a powerful mechanism for computing tensors asynchronously in a graph. Similarly, you can execute multiple threads for the same Session for parallel computations and hence speed up your operations.

Debugger

Another extra power of TF. With TensorFlow, you get a specialized debugger. It provides visibility into the internal structure and states of running TensorFlow graphs. Insights from debugger can be used to facilitate debugging of various types of bugs during both training and inference.

Control

The more control you have over your network, more better understanding you have of what’s going on

with your network.

With TF, you get such a control over your network. You can control whatever you want in your network. Operations on weights or gradients can be done like a charm in TF.

Numpy

Numpy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Numpy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionality. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

Mathematical and logical operations on arrays.

Fourier transforms and routines for shape manipulation.

Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for Mat Lab

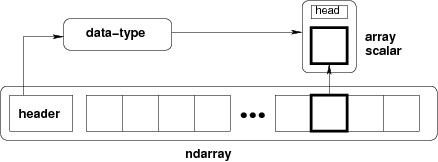
NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



An instance of ndarray class can be constructed by different array creation routines described later in the

tutorial. The basic ndarray is created using an array function in NumPy as follows −

numpy.array

It creates an ndarray from any object exposing array interface, or from any method that returns an array. The ndarray objects can be saved to and loaded from the disk files. The IO functions available are − load() and save() functions handle /numPy binary files (with npyextension)

loadtxt() and savetxt() functions handle normal text files

NumPy introduces a simple file format for ndarray objects. This .npy file stores data, shape, dtype and other information required to reconstruct the ndarray in a disk file such that the array is correctly retrieved even if the file is on another machine with different architecture.

numpy.save()

The numpy.save() file stores the input array in a disk file with npyextension. import numpy as np

a = np.array([1,2,3,4,5]) np.save('outfile',a)

To reconstruct array from outfile.npy, use load() function. import numpy as np

b = np.load('outfile.npy') print b

It will produce the following output −

array([1, 2, 3, 4, 5])

The save() and load() functions accept an additional Boolean parameter allow\_pickles. A pickle in Python is used to serialize and de-serialize objects before saving to or reading from a disk file.

savetxt()

The storage and retrieval of array data in simple text file format is done with savetxt() and loadtxt() functions.

Example

import numpy as np

a = np.array([1,2,3,4,5])

np.savetxt('out.txt',a)

b = np.loadtxt('out.txt') print b

It will produce the following output −

[ 1. 2. 3. 4. 5.]

The savetxt() and loadtxt() functions accept additional optional parameters such as header, footer, and delimiter.

CODE:

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

%matplotlib inline

card\_data = pd.read\_csv('input/GermanDataset.csv') card\_data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1000 entries, 0 to 999 Data columns (total 21 columns):

over\_draft 1000 non-null int64 credit\_usage 1000 non-null int64 credit\_history 1000 non-null int64 purpose 1000 non-null object current\_balance 1000 non-null int64 Average\_Credit\_Balance 1000 non-null int64 employment 1000 non-null int64 location 1000 non-null int64 personal\_status 1000 non-null int64 other\_parties 1000 non-null int64 residence\_since 1000 non-null int64 property\_magnitude 1000 non-null int64 cc\_age 1000 non-null int64 other\_payment\_plans 1000 non-null int64 housing 1000 non-null int64 existing\_credits 1000 non-null int64

job 1000 non-null int64 num\_dependents 1000 non-null int64 own\_telephone 1000 non-null int64 foreign\_worker 1000 non-null int64 class 1000 non-null int64

dtypes: int64(20), object(1) memory usage: 160.2+ KB card\_data.describe() plt.figure(figsize=(12,6))

card\_data[card\_data['num\_dependents']==1]['current\_balance'].hist(alpha=0.5,color='blue', bins=30,label='Credit.Policy=1')

card\_data[card\_data['num\_dependents']==2]['current\_balance'].hist(alpha=0.5,color='red', bins=30,label='Credit.Policy=2')

plt.legend()

plt.xlabel('Current Balance') plt.figure(figsize=(15,7), dpi=90)

sns.countplot(x='purpose',hue='class',data=card\_data,palette='Set1') Out[7]:<matplotlib.axes.\_subplots.AxesSubplot at 0xb784170> plt.figure(figsize=(15,7), dpi=90) sns.countplot(x='purpose',hue='class',data=card\_data,palette='Set1') Out[7]:

<matplotlib.axes.\_subplots.AxesSubplot at 0xb784170> sns.jointplot(x='credit\_usage',y='current\_balance',data=card\_data,color='purple') plt.figure(figsize=(11,7)) sns.lmplot(y='current\_balance',x='credit\_usage',data=card\_data,hue='num\_dependents',

col='class',palette='Set1') plt.figure(figsize=(11,7))

sns.lmplot(y='current\_balance',x='credit\_usage',data=card\_data,hue='num\_dependents', col='class',palette='Set1')

card\_data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1000 entries, 0 to 999 Data columns (total 21 columns):

over\_draft 1000 non-null int64 credit\_usage 1000 non-null int64 credit\_history 1000 non-null int64 purpose 1000 non-null object current\_balance 1000 non-null int64 Average\_Credit\_Balance 1000 non-null int64 employment 1000 non-null int64 location 1000 non-null int64 personal\_status 1000 non-null int64 other\_parties 1000 non-null int64 residence\_since 1000 non-null int64 property\_magnitude 1000 non-null int64 cc\_age 1000 non-null int64 other\_payment\_plans 1000 non-null int64 housing 1000 non-null int64 existing\_credits 1000 non-null int64

job 1000 non-null int64 num\_dependents 1000 non-null int64 own\_telephone 1000 non-null int64 foreign\_worker 1000 non-null int64 class 1000 non-null int64

dtypes: int64(20), object(1) memory usage: 160.2+ KB cat\_feats = ['purpose']

final\_data = pd.get\_dummies(card\_data, columns=cat\_feats, drop\_first=True) final\_data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1000 entries, 0 to 999 Data columns (total 29 columns):

over\_draft 1000 non-null int64

credit\_usage 1000 non-null int64

credit\_history 1000 non-null int64 current\_balance 1000 non-null int64 Average\_Credit\_Balance 1000 non-null int64 employment 1000 non-null int64 location 1000 non-null int64 personal\_status 1000 non-null int64 other\_parties 1000 non-null int64 residence\_since 1000 non-null int64 property\_magnitude 1000 non-null int64 cc\_age 1000 non-null int64 other\_payment\_plans 1000 non-null int64 housing 1000 non-null int64 existing\_credits 1000 non-null int64

job 1000 non-null int64 num\_dependents 1000 non-null int64

own\_telephone 1000 non-null int64

foreign\_worker 1000 non-null int64 class 1000 non-null int64 purpose\_'new car' 1000 non-null uint8 purpose\_'used car' 1000 non-null uint8 purpose\_business 1000 non-null uint8 purpose\_education 1000 non-null uint8

purpose\_furniture/equipment 1000 non-null uint8 purpose\_other 1000 non-null uint8 purpose\_radio/tv 1000 non-null uint8 purpose\_repairs 1000 non-null uint8 purpose\_retraining 1000 non-null uint8 dtypes: int64(20), uint8(9)

memory usage: 165.1 KB final\_data.head()

from sklearn.model\_selection import train\_test\_split X = final\_data.drop('class', axis=1)

y = final\_data['class']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=101) X\_test.head(10)

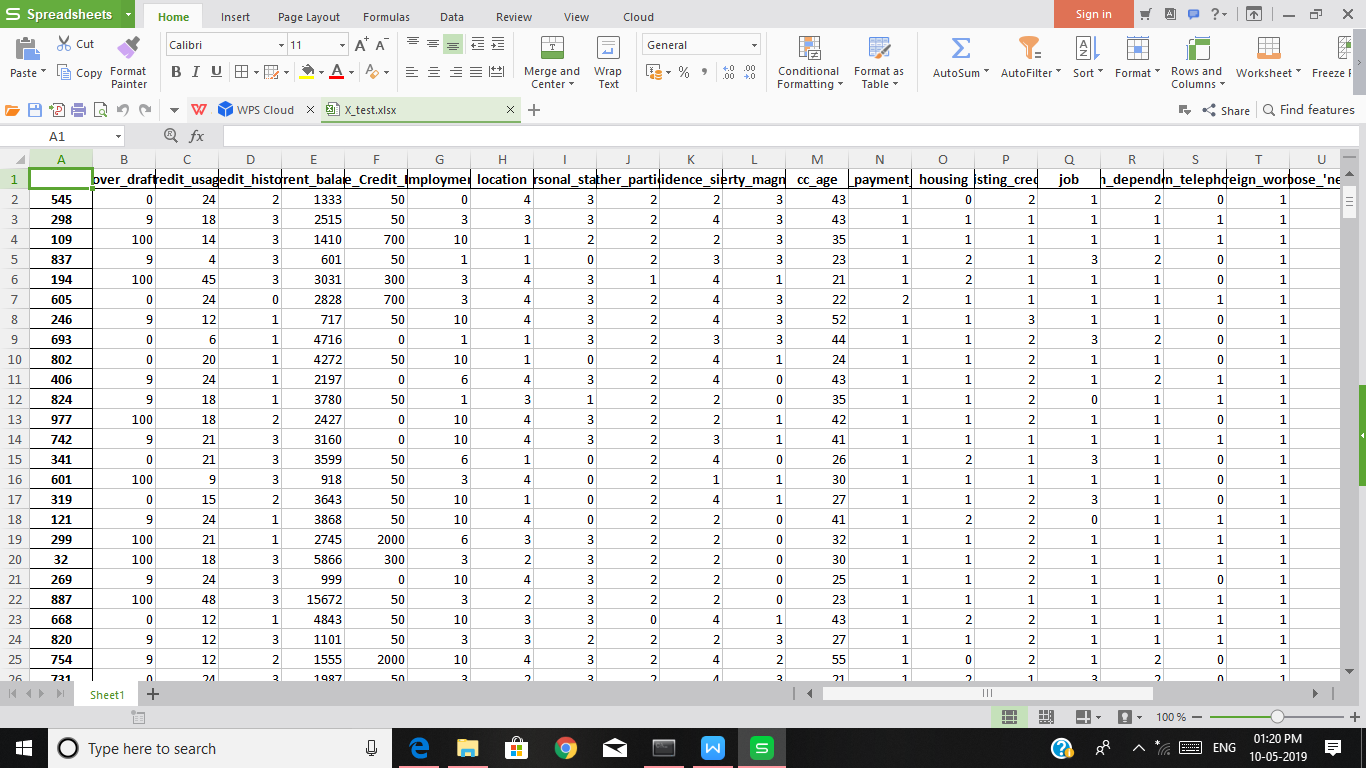
X\_train.to\_excel('Traning\_Testing/X\_train.xlsx') X\_test.to\_excel('Traning\_Testing/X\_test.xlsx') y\_train.to\_excel('Traning\_Testing/y\_train.xlsx') y\_test.to\_excel('Traning\_Testing/y\_test.xlsx') from sklearn.tree import DecisionTreeClassifier dtree = DecisionTreeClassifier() dtree.fit(X\_train,y\_train)

predictions = dtree.predict(X\_test)

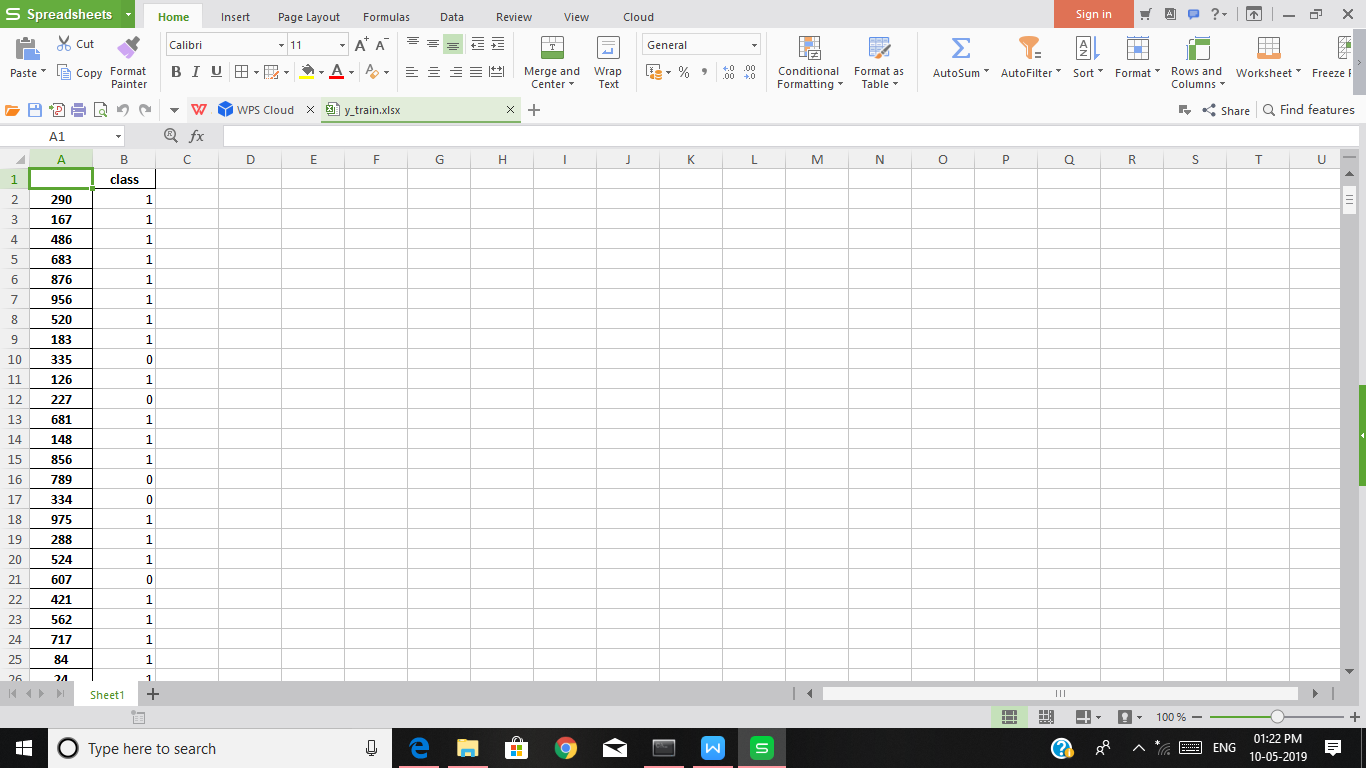
from sklearn.metrics import classification\_report,confusion\_matrix print(classification\_report(y\_test,predictions))

* 1. Different modules of the project
     + Training Testing
     + Credit card
     + Credit card test
     + German Dataset

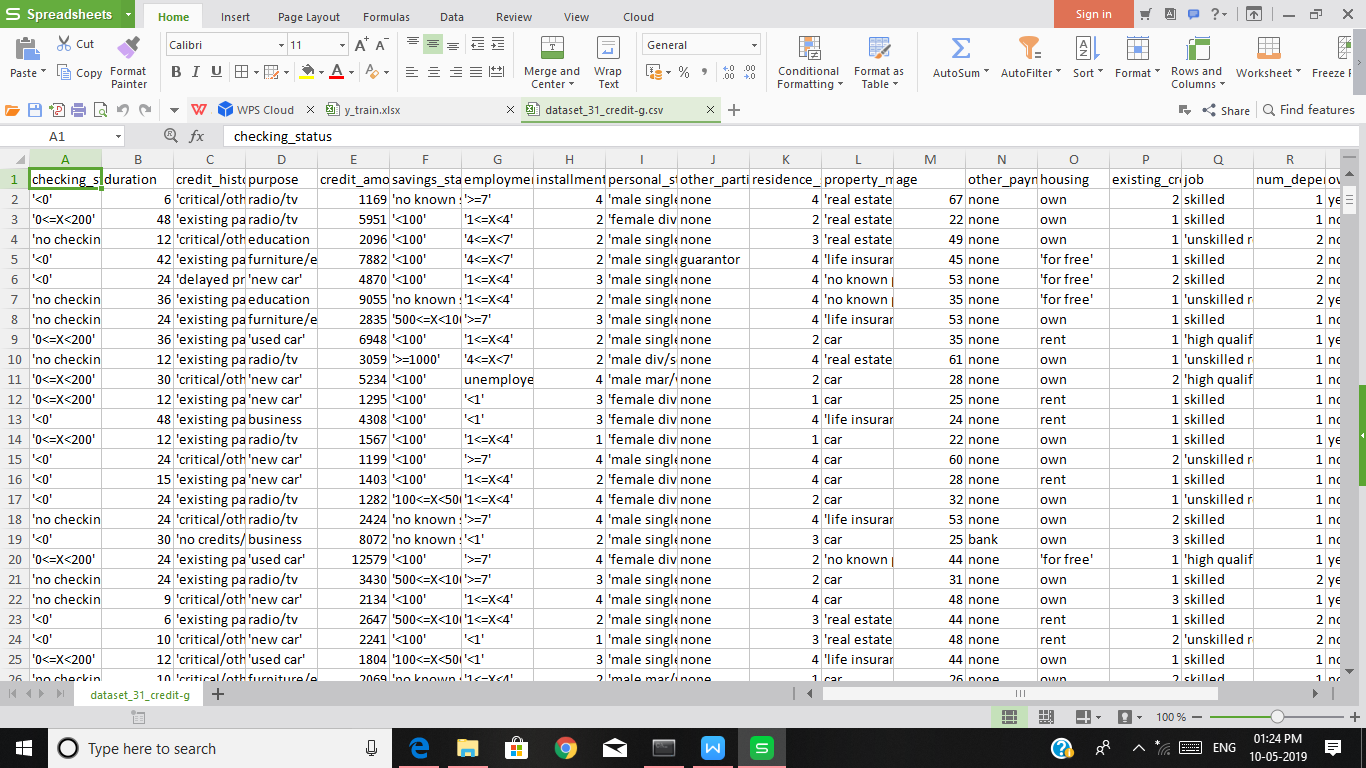
Training Testing: X-test



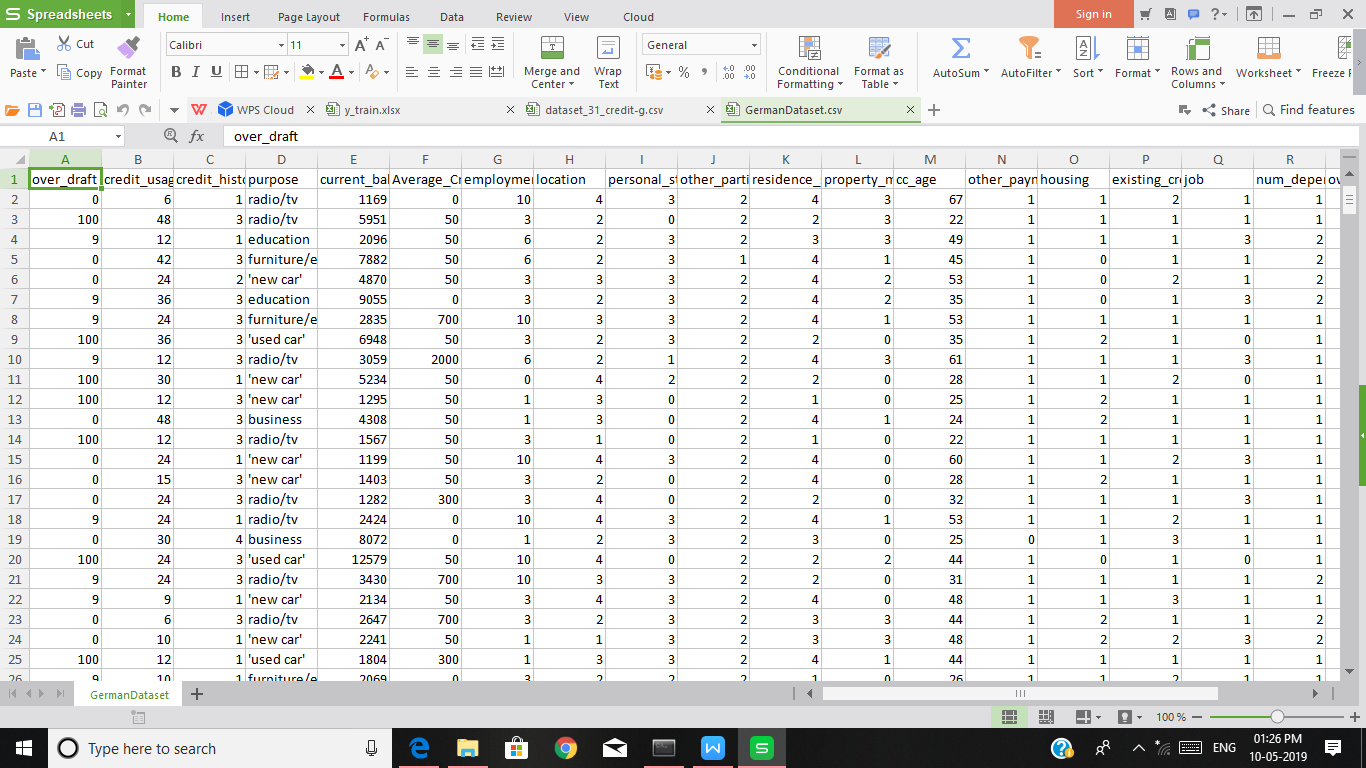
Y-train



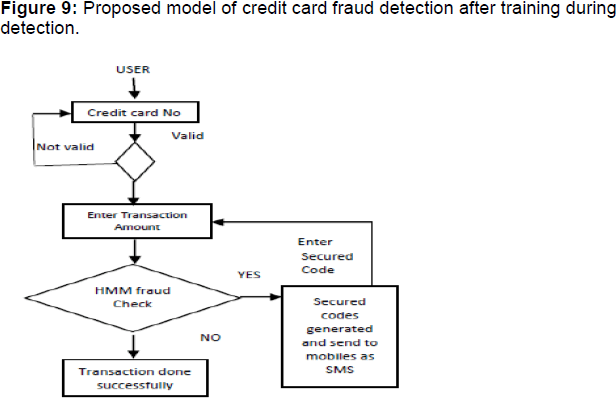
Credit card:



German Dataset



* 1. Flow Chart Of The Proposed System

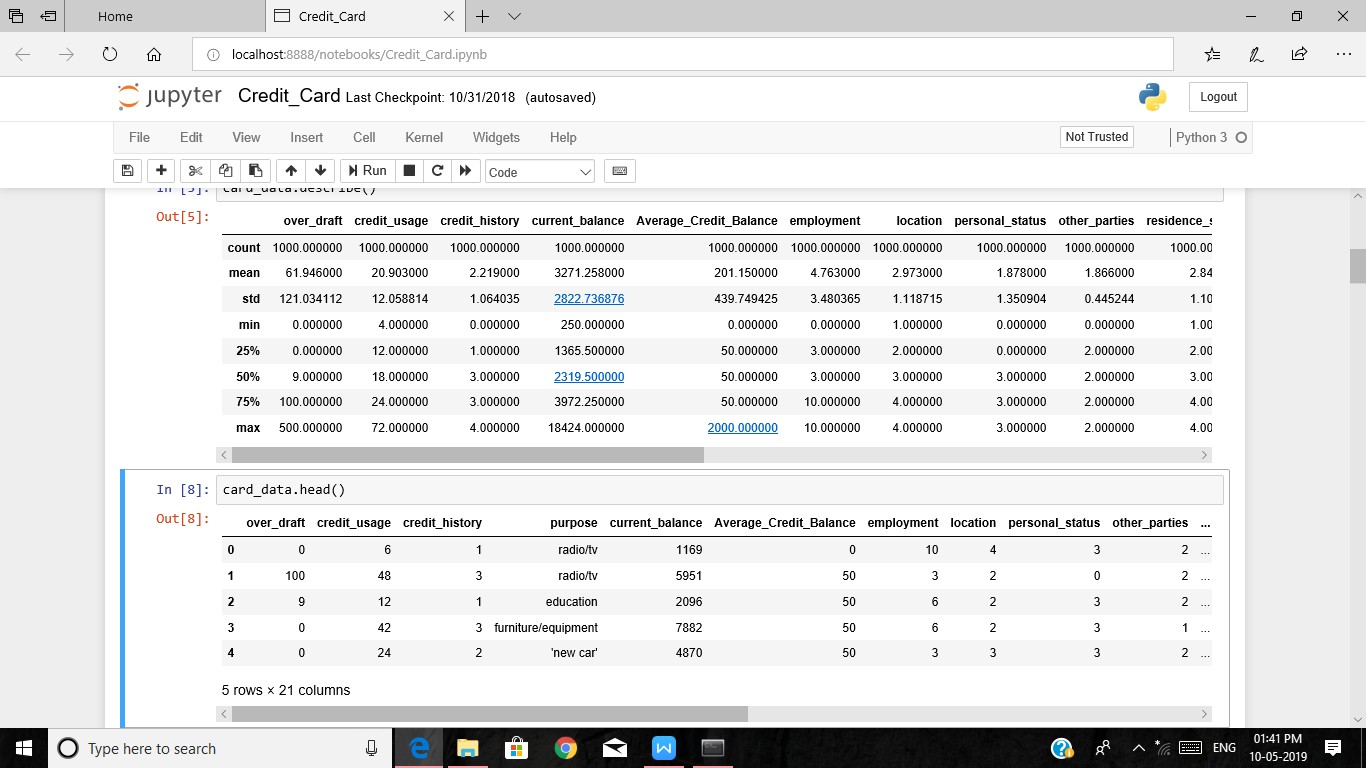


### CHAPTER 6 EXPERIMENTAL RESULTS

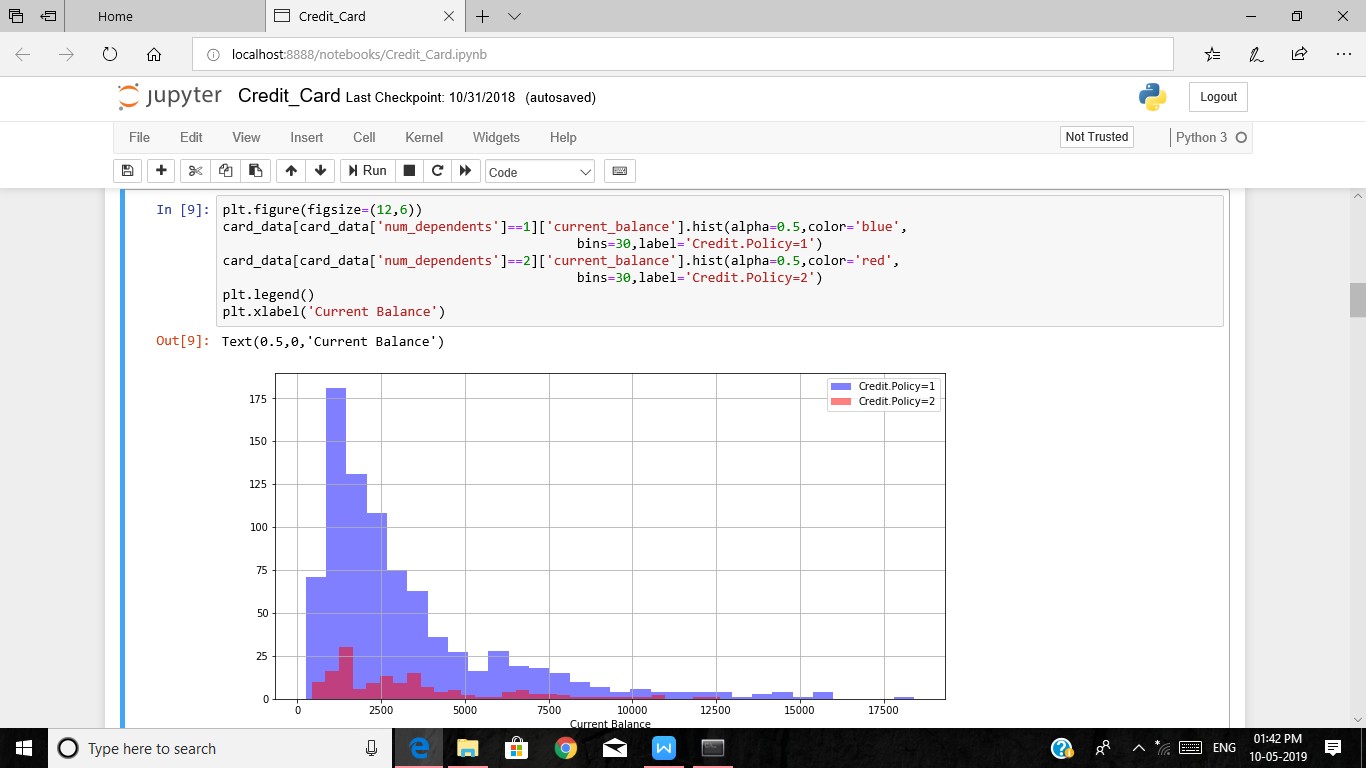
6.1 Outcomes of the Proposed System

In this work, a business intelligent model has been developed, to classify different animals, based on a specific business structure deal with Animal classification using a suitable machine learning technique. The model was evaluated by a scientific approach to measure accuracy. We are using Convectional Neural Network (CNN) to build our model.

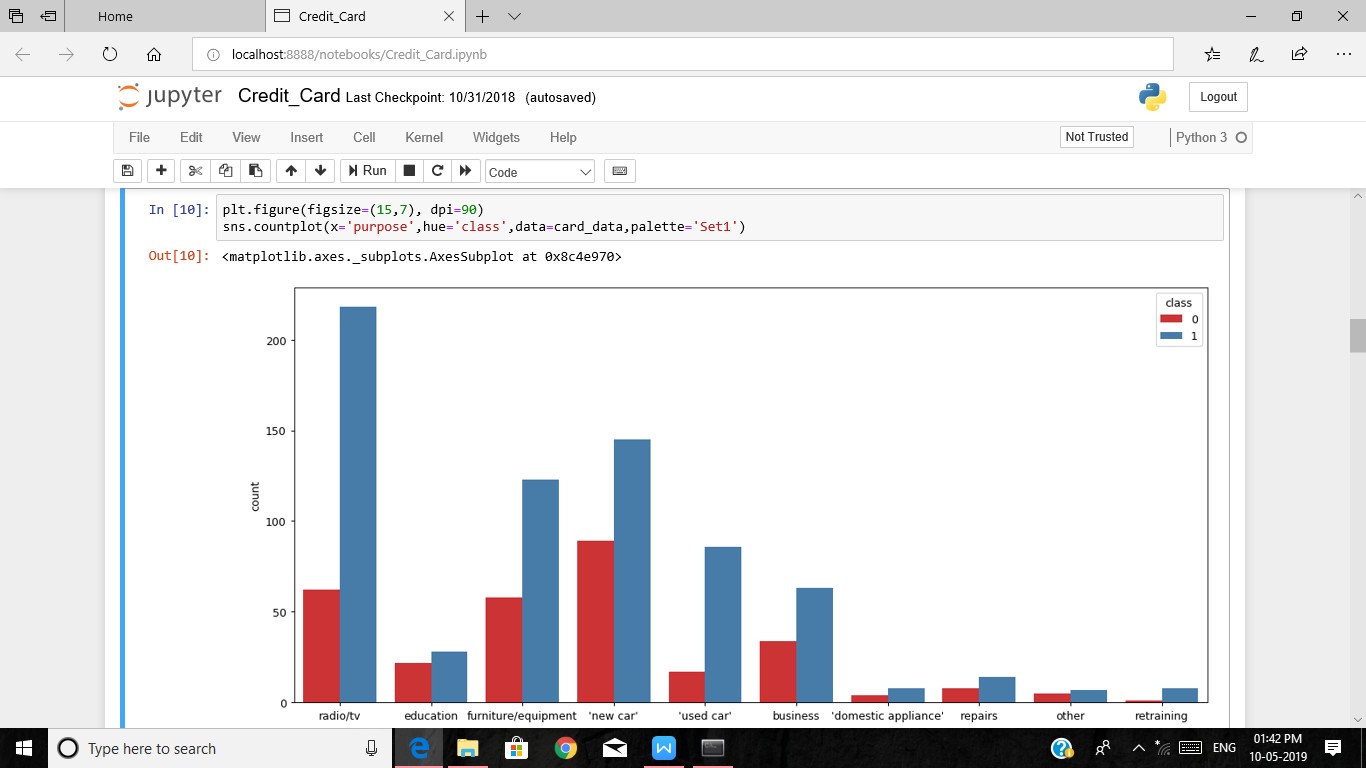
Snapshot 1



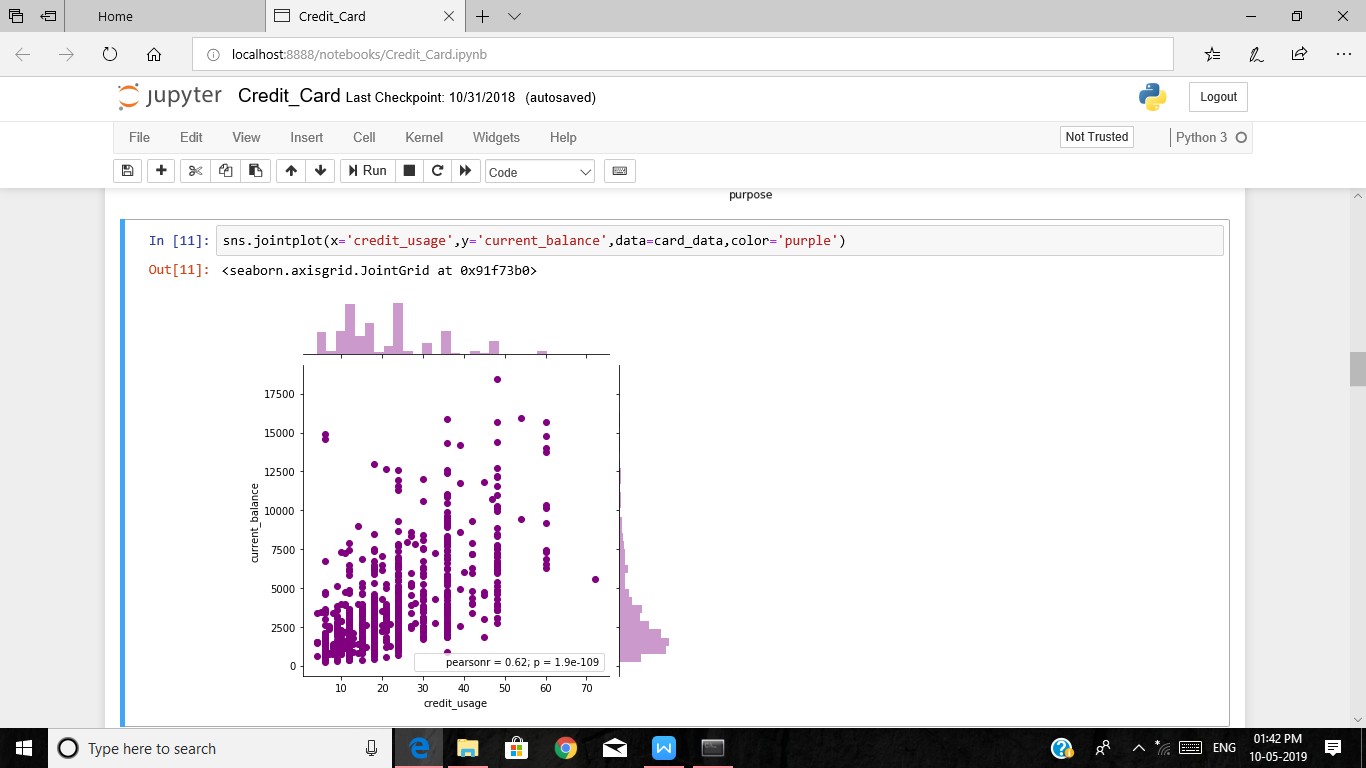
Snapshot 2



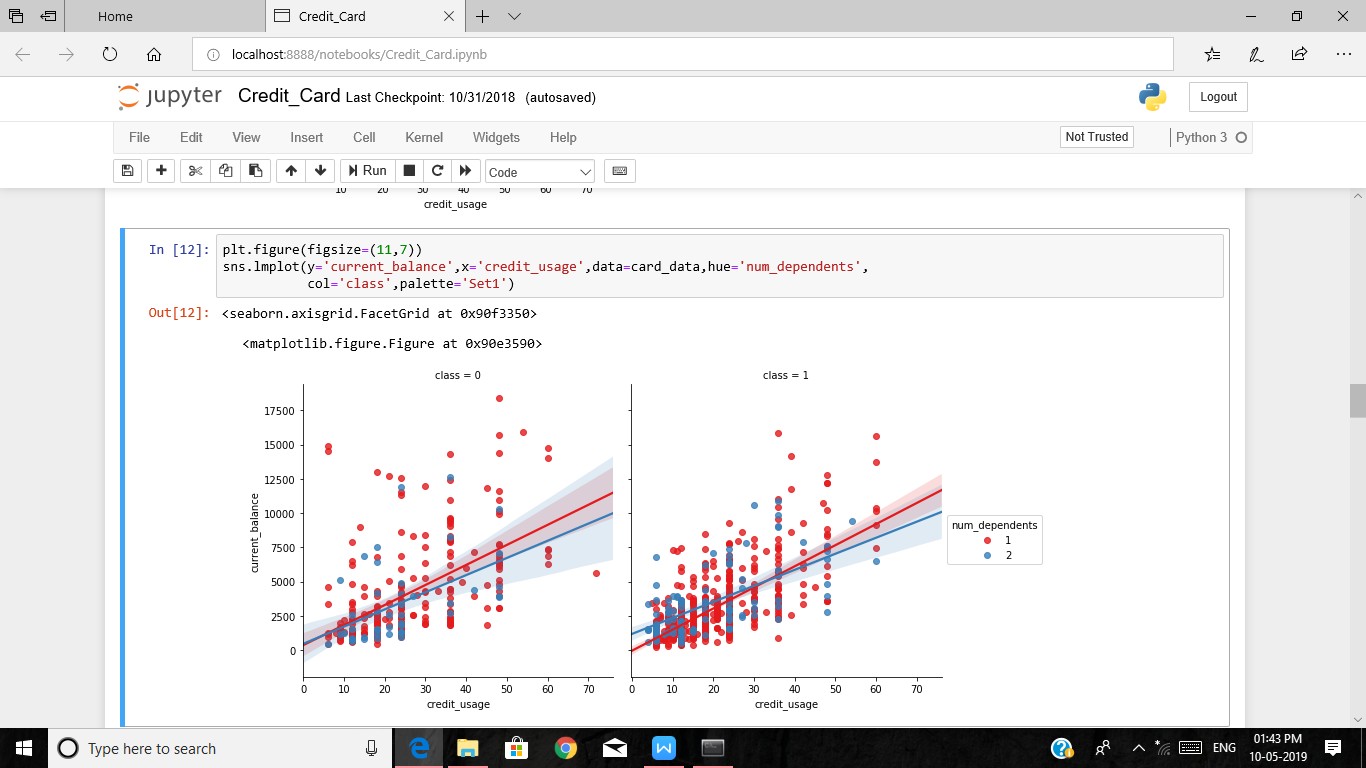
Snapshot 3



Snapshot 4



Snapshot 5



### CHAPTER 7 TESTING

* 1. Testing and Validations

Validation is a complex process with many possible variations and options, so specifics vary from database to database, but the general outline is:

Requirement Gathering

The Sponsor decides what the database is required to do based on regulations, company needs, and any other important factors.

The requirements are documented and approved. System Testing

Procedures to test the requirements are created and documented. The version of the database that will be used for validation is set up. The Sponsor approves the test procedures.

The tests are performed and documented.

Any needed changes are made. This may require another, shorter round of testing and documentation. System Release

The validation documentation is finalized. The database is put into production.

###### Testing Levels

* + 1. Functional Testing:

This type of testing is done against the functional requirements of the project. Types:

Unit testing: Each unit /module of the project is individually tested to check for bugs. If any bugs found by the testing team, it is reported to the developer for fixing.

Integration testing: All the units are now integrated as one single unit and checked for bugs. This also checks if all the modules are working properly with each other.

System testing: This testing checks for operating system compatibility. It includes both functional and non functional requirements.

Sanity testing: It ensures change in the code doesn’t affect the working of the project.

Smoke testing: this type of testing is a set of small tests designed for each build.

Interface testing: Testing of the interface and its proper functioning.

Regression testing: Testing the software repetitively when a new requirement is added, when bug fixed etc.

Beta/Acceptance testing: User level testing to obtain user feedback on the product.

* + 1. Non-Functional Testing:

This type of testing is mainly concerned with the non-functional requirements such as performance of the system under various scenarios.

Performance testing: Checks for speed, stability and reliability of the software, hardware or even the network of the system under test.

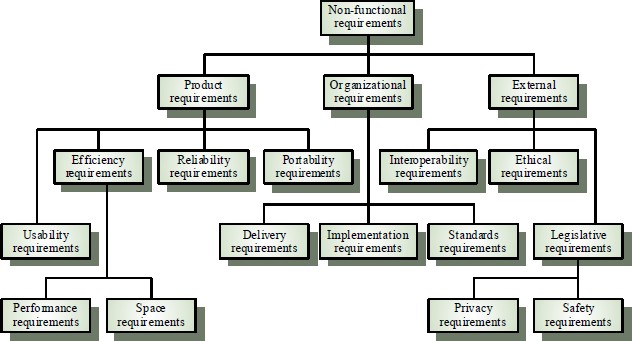
Compatibility testing: This type of testing checks for compatibility of the system with different operating systems, different networks etc.

Localization testing: This check for the localized version of the product mainly concerned with UI. Security testing: Checks if the software has vulnerabilities and if any, fix them.

Reliability testing: Checks for the reliability of the software

Stress testing: This testing checks the performance of the system when it is exposed to different stress levels.

Usability testing: Type of testing checks the easily the software is being used by the customers. Compliance testing: Type of testing to determine the compliance of a system with internal or external standards.



Reliability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

Maintainability

The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

Performance

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

Portability

The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

Scalability

The framework should be sufficiently adaptable to include new functionalities at a later stage. There should be a run of the mill channel, which can oblige the new functionalities.

Flexibility

Flexibility is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure.

* 1. White Box Testing

White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing,

Code-Based testing, and Glass Box testing. It is usually performed by developers.

It is one of two parts of the **"Box Testing" approach** to software testing. Its counterpart, **Black box testing**, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing.

The term "White Box" was used because of the see-through box concept. The clear box or White Box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "[Black Box Testing](https://www.guru99.com/black-box-testing.html)" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

Internal security holes

Broken or poorly structured paths in the coding processes The flow of specific inputs through the code

Expected output

The functionality of conditional loops

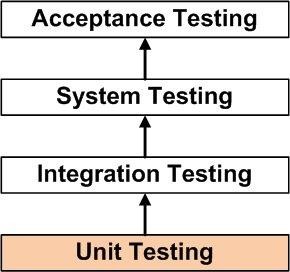
Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of white box testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

##### Different Stages of Testing

* + 1. Unit Testing

**UNIT TESTING** is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.



Unit Test Plan:

Unit Test Plan Prepare Review Rework Baseline

Unit Test Cases/Scripts Prepare

Review Rework Baseline Unit Test Perform

Benefits

Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.

Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

Development is faster. How? If you do not have unit testing in place, you write your code and perform

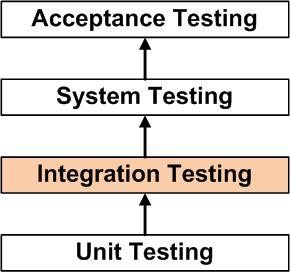
that fuzzy ‘developer test’ (You set some breakpoints, fire up the GUI, provide a few inputs that hopefully hit your code and hope that you are all set.) But, if you have unit testing in place, you write the test, write the code and run the test. Writing tests takes time but the time is compensated by the less amount of time it takes to run the tests; You need not fire up the GUI and provide all those inputs. And, of course, unit tests are more reliable than ‘developer tests’. Development is faster in the long run too. How? The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.

The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the software is live.

Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

* + 1. Integration Testing

**INTEGRATION TESTING** is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.



Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also c*omponent* integration

testing, system integration testing.

Component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components.

System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Tasks

Integration Test Plan Prepare

Review Rework Baseline

Integration Test Cases/Scripts Prepare

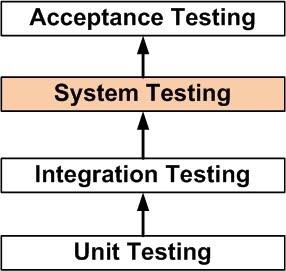
Review Rework Baseline

Integration Test

* + 1. System Testing

SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The

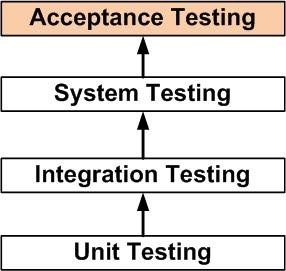
purpose of this test is to evaluate the system’s compliance with the specified requirements.



System testing: The process of testing an integrated system to verify that it meets specified requirements.

* + 1. Acceptance Testing

ACCEPTANCE TESTING is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery.



acceptance testing: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

### CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

##### Conclusion

This survey has explored almost all published fraud detection studies. It defines the adversary, the types and subtypes of fraud, the technical nature of data, performance metrics, and the methods and techniques. After identifying the limitations in methods and techniques of fraud detection, this paper shows that this field can benefit from other related fields. Specifically, unsupervised approaches from counterterrorism work, actual monitoring systems and text mining from law enforcement, and semi supervised and game-theoretic approaches from intrusion and spam detection communities can contribute to future fraud detection research. However, Fawcett and Provost (1999) show that there are no guarantees when they successfully applied their fraud detection method to news story monitoring but unsuccessfully to intrusion detection.

##### Future Enhancement

The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Improvements can be appended by changing the existing modules or adding new modules. One important development that can be added to the project in future is file level backup, which is presently done for folder level.

# REFERENCES

1. Anderson M. (2008). ‗From Subprime Mortgages to Subprime Credit Cards‘. Communities and

Banking, Federal Reserve Bank of Boston, pp. 21-23.

1. Anwer et al. (2009-2010). ‗Online Credit Card Fraud Prevention System for Developing Countries‘,

International Journal of Reviews in Computing, ISSN: 2076-3328, Vol. 2, pp. 62-70. 3. Arias, J.C. & Miller

R. (2009). ‗Market Analysis of Student about Credit Cards‘. Business Intelligence Journal, Vol.

3, No. 1, pp. 23-36. 4. Bhatla T.P. et al. (2003). ‗Understanding Credit Card Frauds‘. Cards Business

Review, 01, pp. 01-15.