**Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection**

**ABSTRACT**

The goal of data analytics is to delineate hidden patterns and use them to support informed decisions in a variety of situations. Credit card fraud is escalating significantly with the advancement of the modernized technology and become an easy target for fraudulent. Credit card fraud is a severe problem in the financial service and costs billions of a dollar every year. The design of fraud detection algorithm is a challenging task with the lack of real-world transaction dataset because of confidentiality and the highly imbalanced publicly available datasets.

Apply different unsupervised machine learning algorithms to detect credit card fraudulent transaction using a real-world dataset. Furthermore, employ these algorithms to implement a super classifier using ensemble learning methods. Identify the most important variables that may lead to higher accuracy in credit card fraudulent transaction detection. Additionally, we compare and discuss the performance of various supervised machine learning algorithms exist in literature against the super classifier that can implemented.

**INTRODUCTION**

1. **INTRODUCTION**

Credit cards are most widely used in onsite payment process and card-not-present transactions such as internet or online shopping. With their rising popularity of online payment transaction, credit card transactions are getting highly vulnerable to threats called credit card frauds such as eavesdropping, phishing, intrusion, denial-of-service (DoS), database stealing, etc.

Credit card frauds are classified into three categories which are card related frauds, merchant related frauds and Internet related frauds. Card related frauds include frauds such as use of lost or stolen cards and fake or counterfeit cards, shoulder surfing [1], account takeover etc. Merchant related frauds include merchant collusion where a person at merchant’s side steals credit card information of their customers. But the most severe of all the frauds are Internet related frauds which are made during online transactions such as site-cloning, creating false merchant sites, snooping, man-in-the-middle attack [MITM], and denial-of-service [2]. Traditionally used AVS (Address Verification System), CVM (Card Verification Method) and other security systems have proven insecure against these attacks and therefore consumers cannot fully utilize the credit card transaction system.

Credit card fraud can be defined as the illegal use of any system or, criminal activity. Increase in e-commerce and the ease of online transactions and payments has led to an exponential increase in the number of people opting for online purchases. The most common method of payment for online purchase is credit card. Credit-card-based purchases can be categorized into two types: 1) physical card and 2) virtual card. When the user is present physically with the card during the transaction, it is categorized as Physical purchasing. Especially all credit card operations are performed by web payment gateways, e.g., PayPal and Alipay.

In real life, fraudulent transaction are scattered with genuine transactions and simple pattern matching Techniques are not often sufficient to detect those frauds accurately. Outlier detection is a data mining technique commonly used for fraud detection [3][4]. Anomaly based methods in which the analyzed spending pattern of each cardholder makes up the cardholder profile. Any incoming transaction that is inconsistent with the cardholder’s profile would be considered suspicious.

Every card holder having unique pattern contains information about amount of transactions, details of purchased items, merchant information, date of transaction etc. It will be the most effective method to counter fraud transaction through internet. If any deviation will be noticed from available patterns of the card holder, then it will generate an alarm to the system to stop the transaction. The aim of the fraud detection system is to detect fraud accurately and before fraud is committed. The goal is to detect least and accurate false fraud detection.

First, totally order the attributes of transaction records, and then classify the values of every attribute. Based on them, we construct a logical graph of BP (LGBP) which abstracts and covers all different transaction records. Based on LGBP, we define the path-based transition probability and diversity coefficient to characterize users’ transaction behaviors and diversity. Also define a state transition probability matrix to capture temporal features of transactions, and then construct a BP for each user. A BP-based fraud detection method is proposed to determine the legality of an incoming transaction, and it considers the concept drift problem [5].

Outliers are used to detect the fraud. While in supervised method, the models are used to differentiate between fraudulent and non-fraudulent behavior to obtain the outlier. Clustering has the application in the field of engineering and scientific disciplines like psychology, biology, medicine, computer vision, communication and remote sensing. A set of pattern is observed by abstracting underlying structure in clustering. The patterns are clustered on the basis of more similar features than other pattern of group. Various clustering algorithms have been proposed to fulfill different requirements. Clustering algorithms are based on the structure of abstraction and are classified into hierarchical and partition algorithms. Catalyst optimization algorithm is used to create physical plan. Catalyst is a tree composed of node objects.

**LITERATURE SURVEY**

**2.1. LITERATURE SURVEY**

The work developed by N. Abdelhamid, A. Ayesh, and F. Thabtah [8] Phishing is serious web security problem that involves mimicking legitimate websites to deceive online users in order to steal their sensitive information. Phishing can be seen as a typical classification problem in data mining where the classifier is constructed from large number of website’s features. There are high demands on identifying the best set of features that when mined the predictive accuracy of the classifiers is enhanced. Compare two known features selection method in order to determine the least set of features of phishing detection using data mining.

The work developed by R. Brause, T. Langsdorf, and M. Hepp, [9] The prevention of credit card fraud is an important application for prediction techniques. One major obstacle for using neural network training techniques is the high necessary diagnostic quality: Since only one financial transaction of a thousand is invalid no prediction success less than 99.9% is acceptable. Due to these credit card transaction proportions complete new concepts had to be developed and tested on real credit card data. This paper shows how advanced data mining techniques and neural network algorithm can be combined successfully to obtain a high fraud coverage combined with a low false alarm rate.

The work developed by R. C. Chen, S. T. Luo, X. Liang, and V. C. S. Lee, [10] Online shopping and banking has increased by the growth of internet and by use of credit card. Along with this number of credit card fraud is also increased. Many modern techniques based on Artificial Intelligence, Data warehousing has evolved in detecting various credit card fraudulent transactions. We proposed a system which detect fraud in credit card transaction processing using a decision tree with combination of Luhn's algorithm and Hunt's algorithm. Luhn’s algorithm is used to validate the card number. Address matching rule checks whether the Billing Address and Shipping Address match or not. This check does not guarantee whether a transaction is fraud or genuine. But if the two addresses match, the transaction can be classified as genuine with a high probability. Else, the transaction is labelled as suspect. A customer usually carries out similar types of transactions in terms of amount, which can be visualized as part of a cluster. Since a fraudster is likely to differ from the customer’s account, his transactions can be detected as exceptions to the cluster – a process known as outlier detection.

The work developed by C. Cortes and D. Pregibon [11] Data mining technology is applied to fraud detection to establish the fraud detection model, describe the process of creating the fraud detection model, then establish data model with ID3 decision tree, and establish example of fraud detection model by using this model. As e-commerce sales continue to grow, the associated online fraud remains an attractive source of revenue for fraudsters. These fraudulent activities impose a considerable financial loss to merchants, making online fraud detection a necessity. The problem of fraud detection is concerned with not only capturing the fraudulent activities, but also capturing them as quickly as possible. This timeliness is crucial to decrease financial losses.

V. Dheepa and R. Dhanapal [12] Along with the great increase of internet and e-commerce, the use of credit card is an unavoidable one. Due to the increase of credit card usage, the frauds associated with this have also increased. There are a lot of approaches used to detect the frauds. In this paper, behavior based classification approach using Support Vector Machines are employed and efficient feature extraction method also adopted. If any discrepancies occur in the behaviors transaction pattern then it is predicted as suspicious and taken for further consideration to find the frauds. Generally credit card fraud detection problem suffers from a large amount of data, which is rectified by the proposed method. Achieving finest accuracy, high fraud catching rate and low false alarms are the main tasks of this approach.

The work developed by S. Gupta and R. Johari, [13] 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.

The work developed by [14] Although credit card fraud detection has been studied for many years, these detection models cannot effectively help financial experts handle fraud alerts since they only predict a risk degree for a transaction but are unable to provide any information to explain why the transaction is of this risk degree. This paper presents a Kernel-based Supervised Hashing (KSH) model to detect credit card fraud. KSH is based on the idea of approximate nearest neighbor that can provide the most similar existing fraud samples for a transaction when the transaction is predicted to be fraud. These similar samples can help experts analyze the transaction, improve the detection accuracy and lower the disturbing rate. Additionally, KSH is very suitable for the large and high-dimension dataset. To the best of our knowledge, it is the first time that KSH is used to detect credit card fraud, and our experiments on a real large transaction dataset illustrate its advantages and effectiveness.

**SYSTEM ANALYSIS**

**3.1. OBJECTIVES**

The main objective of Credit Card Fraud Detection project is to identify and detect the wrong use of credit card transaction accurately. If there is a fraudulent usage then the system denies to proceed the transaction further.

**3.2. EXISTING SYSTEM**

* The volume of the electronic transaction has raised significantly in recent years due to the popularization of online shopping (e.g., Amazon, eBay).
* A physical card is not required in the scenario of online shopping and only the information of the card is enough for a transaction. Therefore, it is much easier for a fraudster to make a fraud.
* This kind of approach usually has to know the previous cases of fraud in order to obtain the different fraud patterns. Various supervised learning methods like neural networks, decision trees, logistic regression, and support vector machine are often used to obtain the fraud patterns.
* Fraud Detecting Software are existing but they become unstable and require a lot of data to be processed
* There are ways in which the existing system built over the supervised learning can be trained in wrong ways.
* The system can be trained that a fraud Credit Card holder as a true holder.
* That’s where the Spark tool of splitting the Data Frames and processing it becomes easy over the existing one.

**3.3. PROPOSED SYSTEM**

* The popularization of online shopping, transaction fraud is growing seriously.
* First, we totally order the attributes of transaction records, and then classify the values of every attribute.
* Behavior Profiles based on their transaction records, which is used to detect transaction fraud in the online shopping scenario.
* **Logical graph of** Behavior Profiles which is a total order-based model to represent the logical relation attributes of transaction records.
* We compare OM with other two anomaly detection methods: Bayesian learning-based fraud detection and self-organizing maps-based fraud detection. The two methods are called phase Modulation (PM).
* Transactions labeled as fraudulent. True negative (TN)is the total number of legal transactions labeled as legal. False negative (FN) is the total number of fraud transactions which are not detected.
* OM is automatically classifies the values of transaction attributes so that our model can characterize the user’s personalized behavior more precisely
* The working of the Fraud Detection includes the proposed system.

**SYSTEM REQUIREMENTS**

**4.1. SYSTEM REQUIREMENTS**

The system requirement is a main part in the analyzing phase of the project. The analyzer of the project has to properly analyze the hardware and the software requirements, otherwise in future the project designer will face more trouble with the hardware and software required. Below specified are the project hardware and software requirements.

**4.2. HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

System : Pentium dual core

Hard disk : 120 GB

Monitor : 15” LCD

Input device : Keyboard, Mouse

RAM : 1GB

**4.3. SOFTWARE REQUIREMENTS**

The software requirements is the software specification of the system. It should include both a definition and a specification of a requirements. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team’s progress throughout the development activity.

Operating system : Ubuntu

Programming Language : Python

Tool : Anaconda

Database : Tensorflow,Keras

**4.4. SYSTEM ANALYSIS**

**PYTHON**

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library). Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open-source_software)software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) .Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). It also features dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages)), which binds method and variable names during program execution.Python's design offers some support for [functional programming](https://en.wikipedia.org/wiki/Functional_programming) in the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition.It has filter(), map(), and reduce() functions; [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), and sets; and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions. The standard library has two modules that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).

**KERAS**

Kerasisan OpenSource Neural Network library written in Python hatrunson top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet ,a Google engineer.Keras doesn’t handle low-level computation.Instead, it uses an other library to do it, called the "Backend. So Kerasis high-level API wrapper for the low-level API ,capable of running on top of TensorFlow, CNTK,orTheano.

Keras High-LevelAPI handles the way we make models ,defining layers ,or setup multipleinput-outputmodels.In this level Kerasal so compiles our model with loss and optimizer functions training process with fit function.Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because I th as been handled by the"backend"engine.

**ANACONDA**

**Anaconda** is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager)[conda](https://en.wikipedia.org/wiki/Conda_(package_manager)). The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda is a Python-based data processing and scientific computing platform. It has built in many very useful third-party libraries. Installing Anaconda is equivalent to automatically installing Python and some commonly used libraries such as Numpy, Pandas, Scrip, and Matplotlib, so it makes the installation so much easier than regular Python installation. It is painful and you need to consider compatibility, thus it is highly recommended to directly install Anaconda.

**4.5. MODULES**

* Dataset creation using the Schemas
* Extracting the Data from the Dataset
* Training the model
* Streaming Data
* Prediction

**DATASET CREATION USING THE SCHEMAS**

In this module, we actually collect the Schema names and different data from various servers and combine them to create a new Dataframe consisting of Credit Holders Behaviour . This Dataframe has many RDDs inside which is held together using the hashing .Deploy dataset to a database for user account. Categorize the dataset using a data types whatever we deploy. And we have to predict the number of attributes in a data set. We are investigating outlier detection for categorical data sets

**EXTRACTING THE DATA FROM THE DATASET:**

Not all the RDDs are required for detection only those attributes which are the Key features determine the output . We need to classify those features from the entires RDDs

This Process is done so the RDDs from the entire Dataframe are ready for determining the output

**TRAINING THE MODEL**

Logical graph of behavior profiles which is a total order-based model to represent the logical relation of attributes of transaction records. Based on behavior profiles and users’ transaction records, we can compute a path-based transition probability from an attribute to another one. At the same time, we define an information entropy-based diversity coefficient in order to characterize the diversity of transaction behaviors of a user.

* The transaction records are classified into three categories based on transaction amount: low amount (LA), medium amount (MA), and high amount (HA).
* **Raw Trans**: raw credit card transaction events.
* **Enriched**: credit card transaction events enriched with card holder features, which were predicted to be not fraud.
* **Fraud Alert**: credit card transaction events enriched with card holder features which were predicted to be fraud.

**STREAMING DATA:**

A transaction process system and transaction processing are often contrasted with a batch process system and batch processing, where many requests are all executed at one time.

* The former requires the interaction of a user, whereas batch processing does not require user involvement. In batch processing the results of each transaction are not immediately available. Additionally, there is a delay while the many requests are being organized, stored and eventually executed.
* In transaction processing there is no delay and the results of each transaction are immediately available

**PREDICTION**

Thus, the Fraud users are detected from the modified RDDs which are fetched from the Dataframe. Thus, the Fraud Holders are detected.

**TESTING PROCESS**

**5.1. TESTING PROCESS**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**5.2. TYPES OF TESTS:**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional testing**

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:

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 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.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**5.3. Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.
* Features to be tested
* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results**

All the test cases mentioned above passed successfully. No defects encountered.

**5.4. System Implementation**

System Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the user that it will work efficiently and effectively.

The existing system was long time process. The proposed system was developed using Java Swing. The existing system caused long time transmission process but the system developed now has a very good user-friendly tool, which has a menu-based interface, graphical interface for the end user.

After coding and testing, the project is to be installed on the necessary system. The executable file is to be created and loaded in the system. Again the code is tested in the installed system. Installing the developed code in system in the form of executable file is implementation.

**Unit Testing**

Unit testing comprises the set of test performed by an individual programmer prior to integration of the unit in to a larger system. A program unit is usually small enough that the programmer who developed it can test it in great details and certainly is greater detail than will be possible when the unit is integrated in to an evolving software product.

**Unit testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id** | **Test Case** | **Expected Result** | **Actual result** | **Success/**  **failure** |
| UTC\_test1 | Registration | The user gives their details, login id, and login password to server. | The user details stored in server system. That cannot view any other user. if any fault means error report display | Success |
| UTC\_test2 | Login | Using server login id, login password user move to next step. | If it is correct login details the message will display. Otherwise the user cannot login to next step. | Success |
| UTC\_test3 | Upload File | The client creates the fileand uploaded in the cloud. | The file is being uploaded and saved in the Google App Engine. | Success |

**Validation Testing**

**Validation testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test condition** | **Test Input** | **Expected output** | **Actual output** | **Result** |
| Checking for login process | Correct username and password | Perspective main pages should appear | Main page should appeared | Success |
| Correct username and wrong password | Main page should appear | Display the message that the password is incorrect | Fail |
| Wrong username and correct password | Main page should appear | Display the message that the username is incorrect | Fail |
| Wrong username and password | Main page should appear | Display the message that the username and password are incorrect | Fail |
| File upload activity | The file is browsed and uploaded | The file is uploaded. | The file is uploaded | Success |
| File name not entered | Prompt user to enter the file name | File got stored in the persistent storage. | Fail |
| File Download Activity | File download if correct user | File download successfully | File Download | Success |
|  | File download failure | If not in member list | File Download Not Successful | Fail. |

**5.5. SYSTEM STUDY**

**Feasibility Study**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

TECHNICAL FEASIBILITY

SOCIAL FEASIBILITY

**Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

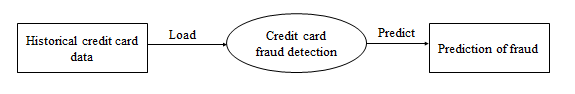
**DATA FLOW DIAGRAM**

**6.1. Data Flow Diagram**

DFDs are used to Specify Functions of the Information System and how data flow from function to function. It’s a collection of function that manipulates data. On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel.

**Data Flow Diagram Level 0 For Credit Card Fraud Detection**

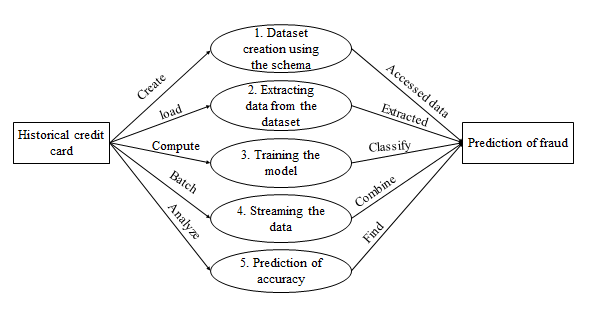
DFD level 0 for credit card fraud detection explains the overall functionality of the entire project. The fig 4.1 below shows the data flow diagram level 0 for credit card fraud detection.



**Level 0 DFD for credit card fraud detection**

In this project collect the historical credit card data which is loaded for finding the prediction of fraud. The collected data is loaded to the data base and finally, the accuracy of fraud is predicted.

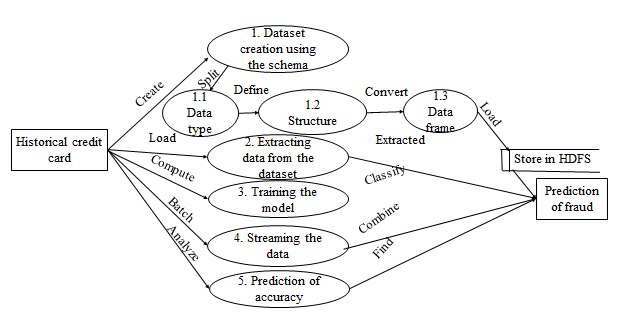
**Data Flow Diagram Level 1 For Credit Card Fraud Detection**



**Level 1 DFD for credit card fraud detection**

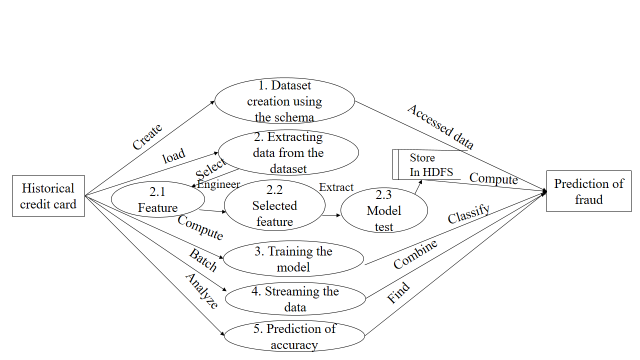
From the above diagram shows the overall representation of each module and their functionality in this project.

**Data Flow Diagram Level 2 For Credit Card Fraud Detection Data Set Creation Using Schema**

**Data Flow Diagram Level 2 for data set creation using schema**

In this module, first split the dataset based on the data type and convert them as a data frames.

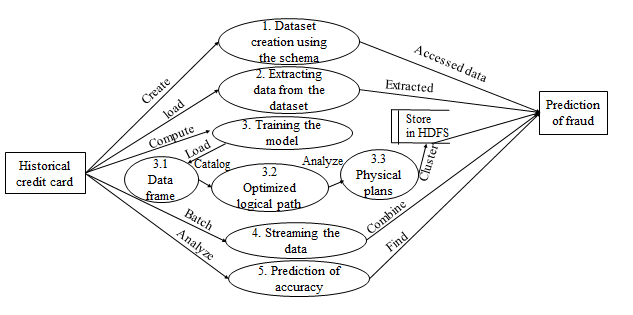
**Data Flow Diagram Level 2 For ExtractingThe Data From The Data Set**



**Data Flow Diagram Level 2 for extracting the data from the data set**

In this module extract the features which are needed to create model test.

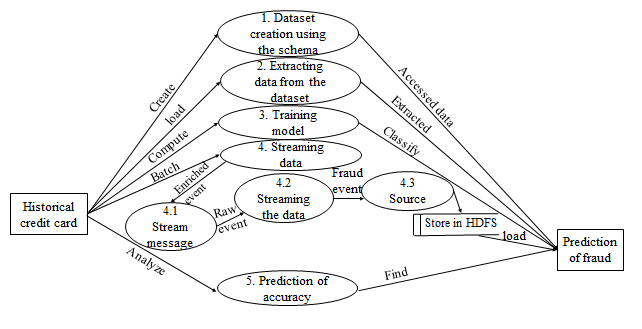
**Data Flow Diagram Level 2 ForTrainingThe Model**



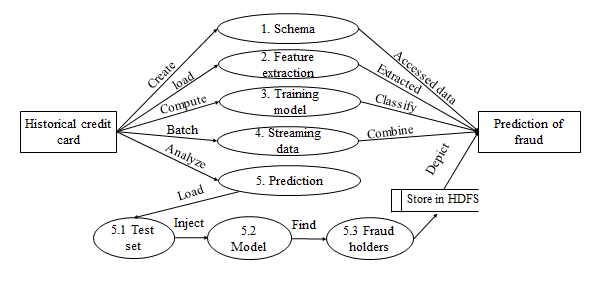
**Data Flow Diagram Level 2 for training the model**

In this module using catalyst optimization algorithm create data frame. Then catalog the optimized logical path. Analyze the physical plans and store the data to the data base.

**Data Flow Diagram Level 2 ForStreaming Data**



**Data Flow Diagram Level 2 For PredictionOf Fraud**

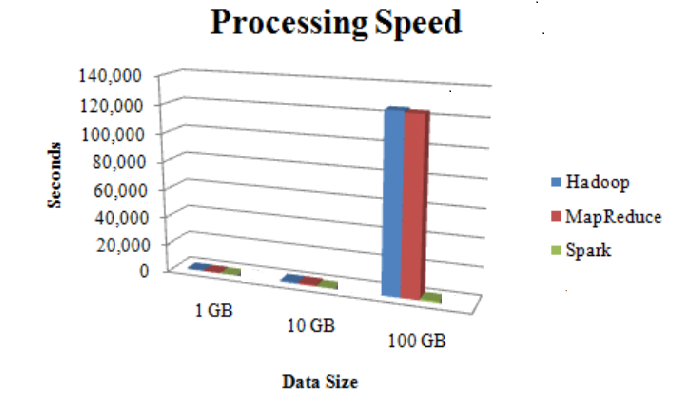


**Data Flow Diagram Level 2 for prediction of fraud**

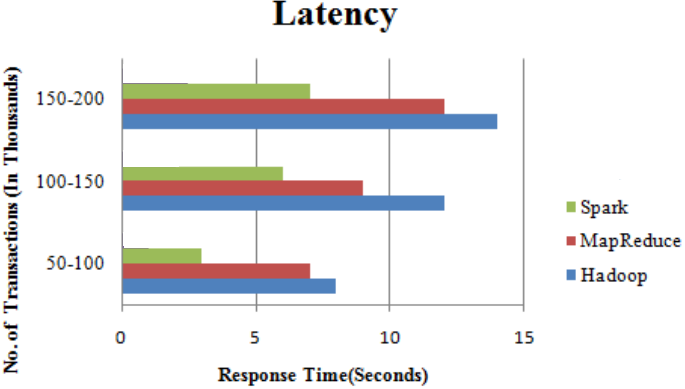
**PERFORMANCE ANALYSIS**

**7.1. PERFORMANCE ANALYSIS**

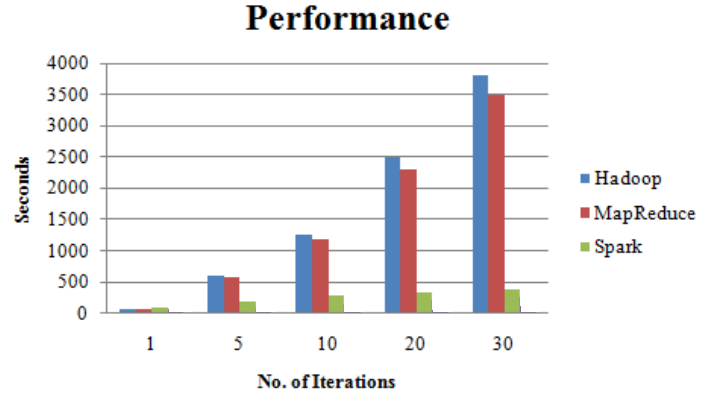
This section extends the analysis of the techniques proposed previously for the big data forensic investigation. There are different factors that can influence the selection and performance of the forensic techniques. These factors should be analysed closely before carrying out the implementation. The following table has been populated based on the available resources and the sensitivity of the data to be used for investigation.

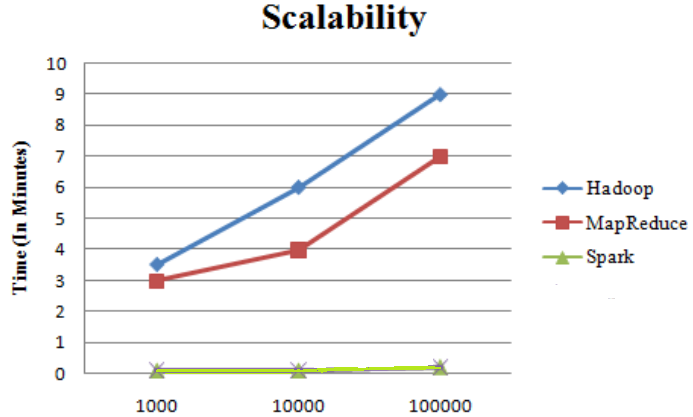
**Fig 6.1 Processing Speed**

In Figure 6.1 shows that the processing speed of Apache Spark lower even though the data size increases. But Hadoop and MapReduce increase its processing time if the data size is too big to handle.

**Fig 6.2 Latency**

The response time should be at minimum while executing the data/transactions. Among the three techniques, Apache spark has low latency while processing Big Data. While analyzing the Fault tolerance factor, Spark system replicates the input data in memory which is a most useful solution for handling faults in between the execution of transactions. Data lost due to failure can be recomputed from replicated input data.

**Fig 6.3 Performance**



**Fig 6.4 Scalability**

In Fig 6.3, shows that Spark has small execution time when compared to other techniques.

In Fig 6.4 shows the scalability factor, Spark processes the data smoothly if there is need to increase nodes for processing Big Data. Although the number of node increases, Spark performs better with Big Data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Techniques | Processing Speed | Latency | Fault Tolerance | Performance | Scalability |
| Hadoop | Medium | High | High | Slow | Medium |
| MapReduce | Slow | High | High | Slow | Medium |
| Spark | Fast | Low | High | Fast | High |

**Table 6.1 : Analysis of Techniques**

Table 6.1 shows the overall comparison of other techniques.

**CODING**

**AND**

**SCREENSHOT**

**8. CODINGS**

import numpy as np

import pandas as pd

import sys

import matplotlib.pyplot as plt

import seaborn as sns

import scipy

import sklearn

data = pd.read\_csv('D:\\CreditCard\\creditcard.csv')

print(data.columns)

data.shape

data = data.sample(frac = 0.2, random\_state = 1)

print(data.shape)

# plot the histogram of each parameter

data.hist(figsize = (20, 20))

plt.show()

# determine the number of fraud cases

fraud = data[data['Class'] == 1]

valid = data[data['Class'] == 0]

outlier\_fraction = len(fraud) / float(len(valid))

print(outlier\_fraction)

print('Fraud Cases: {}'.format(len(fraud)))

print('Valid Cases: {}'.format(len(valid)))

# correlation matrix

corrmat = data.corr()

fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)

plt.show()

# get the columns from the dataframe

columns = data.columns.tolist()

# filter the columns to remove the data we do not want

columns = [c for c in columns if c not in ['Class']]

# store the variable we will be predicting on which is class

target = 'Class'

# X includes everything except our class column

X = data[columns]

# Y includes all the class labels for each sample

# this is also one-dimensional

Y = data[target]

# print the shapes of X and Y

print(X.shape)

print(Y.shape)

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

# define a random state

state = 1

# define the outlier detection methods

classifiers = {

# contamination is the number of outliers we think there are

'Isolation Forest': IsolationForest(max\_samples = len(X),

contamination = outlier\_fraction,

random\_state = state),

# number of neighbors to consider, the higher the percentage of outliers the higher you want

to make this number

'Local Outlier Factor': LocalOutlierFactor(

n\_neighbors = 20,

contamination = outlier\_fraction)

}

n\_outliers = len(fraud)

for i, (clf\_name, clf) in enumerate(classifiers.items()):

# fit the data and tag outliers

if clf\_name == 'Local Outlier Factor':

y\_pred = clf.fit\_predict(X)

scores\_pred = clf.negative\_outlier\_factor\_

else:

clf.fit(X)

scores\_pred = clf.decision\_function(X)

y\_pred = clf.predict(X)

# reshape the prediction values to 0 for valid and 1 for fraud

y\_pred[y\_pred == 1] = 0

y\_pred[y\_pred == -1] = 1

# calculate the number of errors

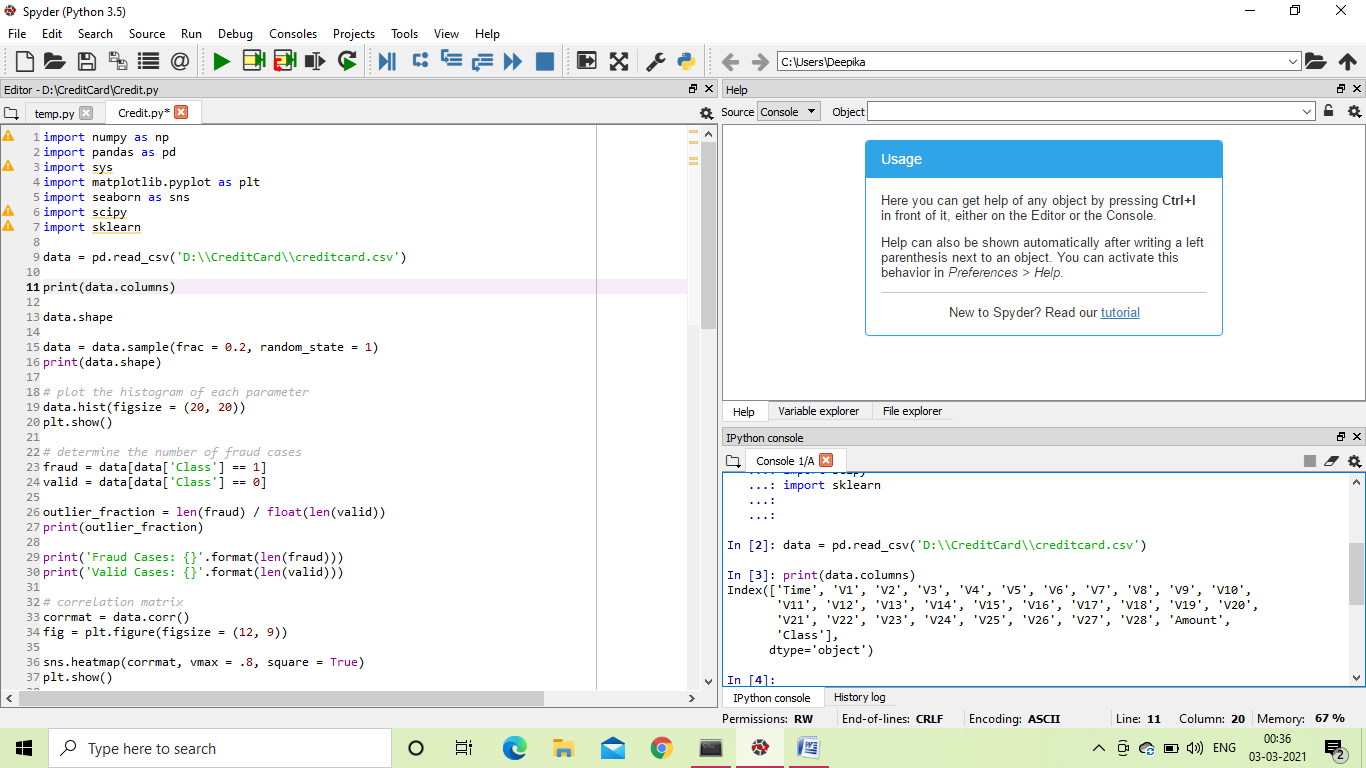
n\_errors = (y\_pred != Y).sum()

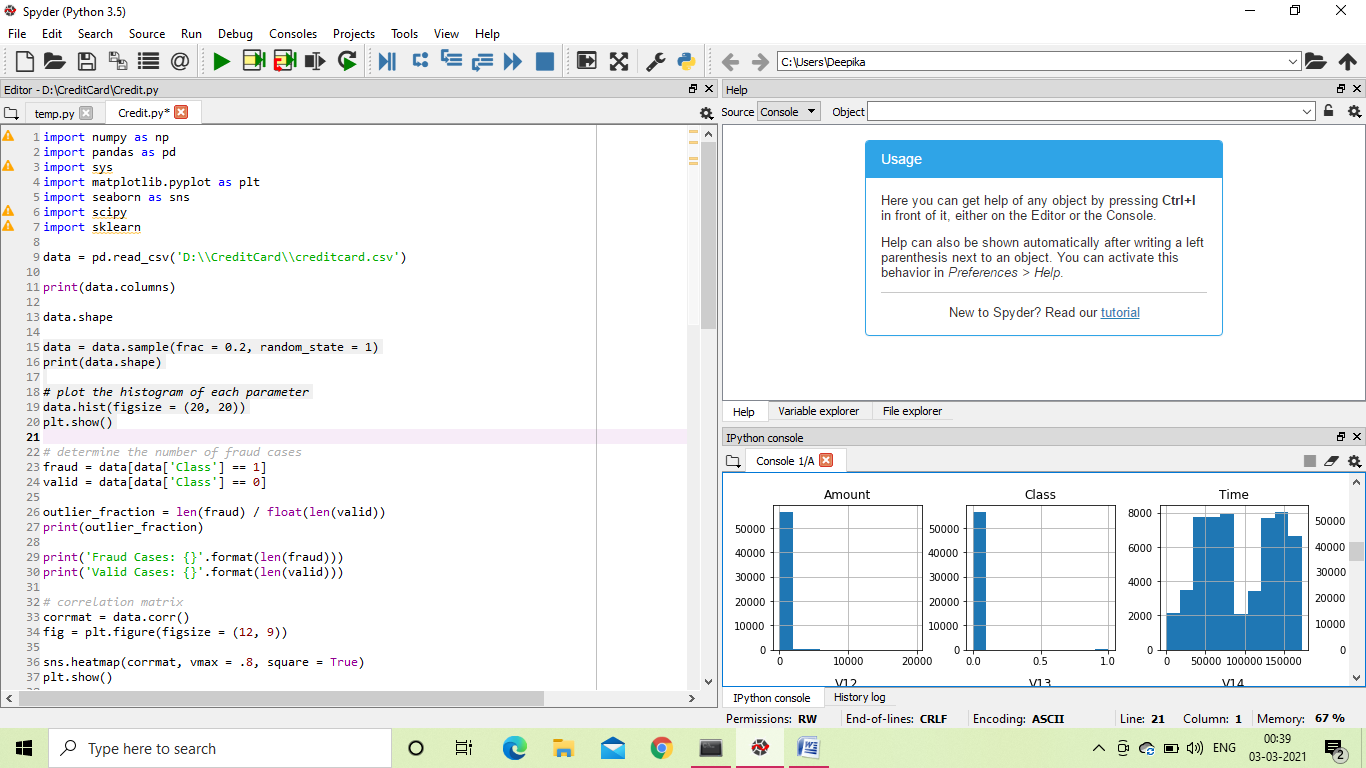
# classification matrix

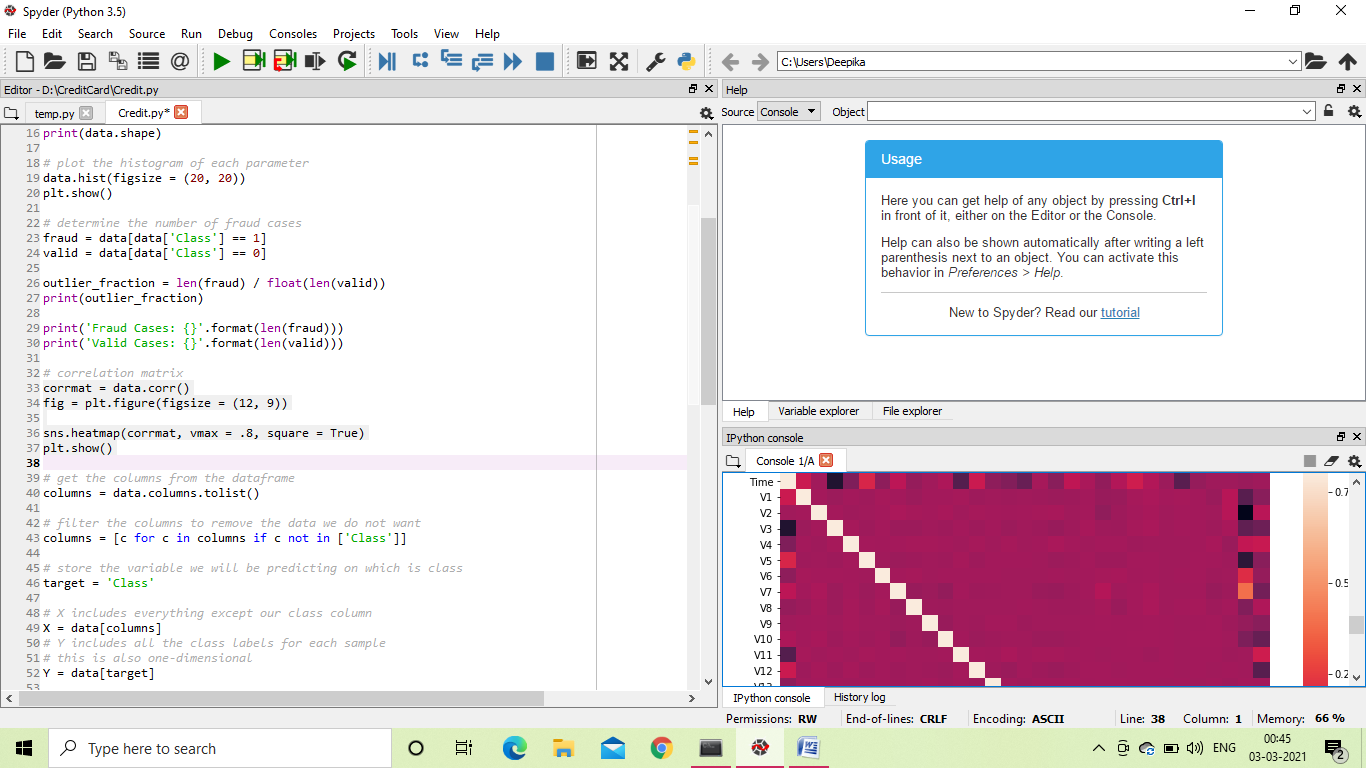
print('{}: {}'.format(clf\_name, n\_errors))

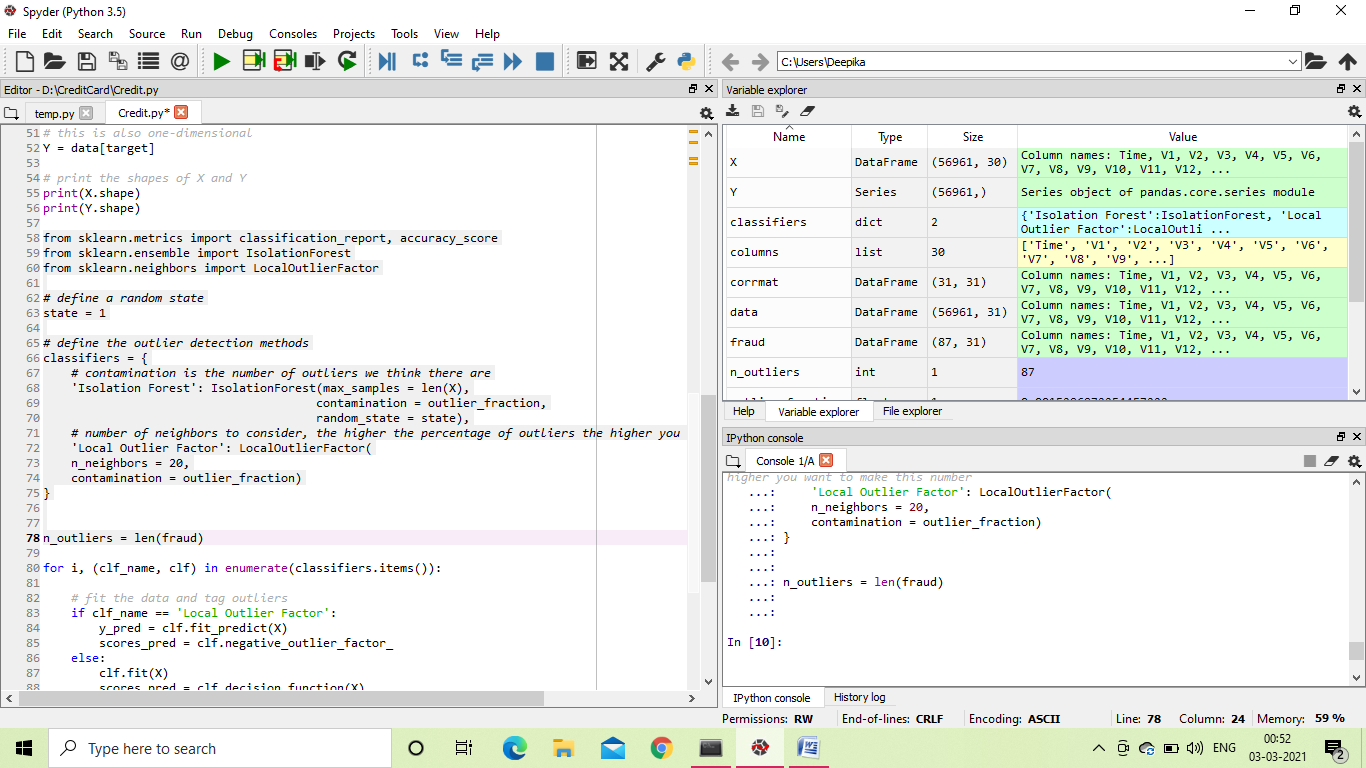
print(accuracy\_score(Y, y\_pred))

print(classification\_report(Y, y\_pred))



****

****

****

**CONCLUSION**

**9. CONCLUSION**

In this project the Data frames are manually made from the frames and schemas separately. The Data frames thus created are further divided into many RDDs. Here the Catalyst Optimizer is used to make Physical Plans and stored. Whenever required further Bayes is applied to detect the Fraud Holders. This module is better than the existing previous modules. The Spark makes efficient use of data. This gives us the idea of how to reduce the Fraudulence.

**LIMITATIONS**

**10. LIMITATIONS**

The limitations faced is the formation of data frame sometimes goes wrong if the Schema and data are different. This creates issue in making RDDs from the Data frame. The Catalogue Optimizer works perfectly and stores the Planning in accessible ways, but they are inaccessible if the code demands further processing.

**FUTURE ENHANCEMENT**

**11. FUTURE ENHANCEMENT**

The future work focuses on some machine-learning methods to automatically classify the values of transaction attributes so that our model can characterize the user’s personalized behavior more precisely. In addition, we plan to extend BP by considering other data such as user’s comments. Credit card transaction events can be delivered through the Streams messaging system, which supports the Kafka .09 API.The events can be processed and checked for Fraud by Spark Streaming using Spark Machine Learning with the deployed model.MapR-FS, which supports the posix NFS API and HDFS API, can be used for storing event data. DB a NoSql database which supports the HBase API, can be used for storing and providing fast access to credit card holder profile data.

**REFERENCES**

**12. REFERENCES**

[1]. Li Yingjiu and Xinwen Zhang, "A security-enhanced one-time payment scheme for credit card", in Proc. of 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications, 2004, Page(s): 40 - 47.

[2] W. Stallings, Cryptography and network security: principles and practices, 3rd edition.

[3] Leila Seyedhossein, mahmoudrezahashemi “Mining Information from Credit Card Time Series for Timelier Fraud Detection”. IEEE-5th International Symposium on Telecommunications. (2010).

[4] E.W.T. Ngai, Yong Hu, Y.H. Wong, Yijun Chen, Xin Sun “The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature”. Elsevier-Decision Support Systems(2011). 50; (559–569).

[5] A. Abdallah, M. A. Maarof, and A. Zainal, “Fraud detection system: A survey,” J. Netw. Comput. Appl., vol. 68, pp. 90–113, Jun. 2016

[6] Abhinav Srivastava, Amlan Kundu, Shamik Sural and Arun K. Majumdar, "CreditCard Fraud Detection Using Hidden Markov Model" IEEE, Transactions On Dependable And Secure Computing, Vol. 5, No 1. , January-March 2008.

[7] V. Bhusari, and S. Patil, “Study of Hidden Markov Model in Credit Card Fraudulent Detection”, International Journal of Computer Applications (0975 – 8887) Volume 20– No.5, April 2011.

[8] N. Abdelhamid, A. Ayesh, and F. Thabtah, “Phishing detection based associative classification data mining,” Expert Syst. Appl., vol. 41, no. 13, pp. 5948–5959, 2014.

[9] R. Brause, T. Langsdorf, and M. Hepp, “Neural data mining for credit card fraud detection,” in Proc. IEEE Int. Conf. Tools Artif. Intell., 1999, pp. 103–106.

[10] R. C. Chen, S. T. Luo, X. Liang, and V. C. S. Lee, “Personalized approach based on SVM and ANN for detecting credit card fraud,” in Proc. Int. Conf. Neural Netw. Brain, Oct. 2005, pp. 810–815

[11] C. Cortes and D. Pregibon, “Signature-based methods for data streams,” Data Mining Knowl. Discovery, vol. 5, no. 3, pp. 167–182, 2001.