“CREDIT CARD FRAUD DETECTION”

A

MAJOR PROJECT REPORT

Submitted for the partial fulfilment of the requirement for the award of Degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

PRACHI MORE (0101CS171074)

PRAGYA GAJBHIYE (0101CS171076)

RAJEEV KUSHRAM (0101CS171085)

Guided By:

Dr. Raju Baraskar

Prof. Priyanka Dixit

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA

BHOPAL-462033

SESSION 2017-2021

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Prachi More, Pragya Gajbhiye, Rajeev kushram** of B.E. Fourth Year, Computer Science & Engineering have completed their major Project titled **"Credit Card Fraud Detection"** during the year 2020-21 under our guidance and supervision.

We approve the project for the submission for the partial fulfillment of the requirement for the award of degree of B.Tech. in Computer Science & Engineering.

Dr. Raju Baraskar
Professor
(Project Guide)
Department of Computer
Science & Engineering

Prof. Priyanka Dixit
Professor
(Project Guide)
Department of Computer
Science & Engineering

Dr. Roopam Gupta
Professor & HoD
Department of Computer
Science & Engineering

Dr. S.S. Bhadauria
Director
UIT RGPV Bhopal

DECLARATION BY CANDIDATE

We, hereby declare that the work which is presented in the major project, entitled "Credit Card Fraud **Detection" submitted in partial fulfilment of the requirement for the award of Bachelor degree in** Computer Science and Engineering has been carried out at University Institute of Technology RGPV, Bhopal and is an authentic record of our work carried out under the guidance of Dr. Raju Baraskar (Project Guide), Prof. Priyanka Dixit (Project Guide), Department of Computer Science and Engineering, UIT RGPV, Bhopal.

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

PRACHI MORE (0101CS171074)

PRAGYA GAJBHIYE (0101CS171076)

RAJEEV KUSHRAM (0101CS171085)

ACKNOWLEDGEMENT

After the completion of major project work, words are not enough to express our gratitude to everyone helped us reach our goal, and above all we thank almighty for helping us reach this moment in life.

First and foremost, we take this opportunity to express our deepest regards and heartfelt gratitude to our project guide Dr Raju Baraskar and Prof. Priyanka Dixit of Computer Scienceand Engineering Department, UIT RGPV Bhopal for their inspiring guidance and timely suggestions in carrying out our project successfully. They have also been a constant source of inspiration for us.

We are grateful for our source of inspiration Dr. Roopam Gupta , Prof. & HoD ,DoCSE UIT-RGPV, Bhopal for his unforgettable support and inspiration towards the project. We would also like to thank all the teachers of our department for providing invaluable support and motivation. We are also grateful to our teammates, friends and colleagues for their help and cooperation throughout this work.

PRACHI MORE (0101CS171074)

PRAGYA GAJBHIYE (0101CS171076)

RAJEEV KUSHRAM (0101CS171085)

ABSTRACT

Finance fraud is a growing problem with far consequences in the financial industry and while many techniques have been discovered. Data mining has been successfully applied to finance databases to automate analysis of huge volumes of complex data. Data mining has also played a salient role in the detection of credit card fraud in online transactions. Fraud detection in credit card is a data mining problem, It becomes challenging due to two major reasons–first, the profiles of normal and fraudulent behaviours change frequently and secondly due to reason that credit card fraud data sets are highly skewed. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. Credit Card Fraud Detection is a typical sample of classification. In this process, we have focused on analysing and pre-processing data sets.

It is vital that credit card companies are able to identify fraudulent credit card transactions. This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far out-number fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1 INTRODUCTION

1.0 What is credit card fraud?

’Fraud’ in credit card transactions is unauthorized and unwanted usage of an account by someone other than the owner of that account. Necessary prevention measures can be taken to stop this abuse and the behaviour of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future.In other words, Credit Card Fraud can be defined as a case where a person uses someone else’s credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used.

Fraud detection involves monitoring the activities of populations of users in order to estimate, perceive or avoid objectionable behaviour, which consist of fraud, intrusion, and defaulting.

This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated.

This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize.

Machine learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent.

The investigators provide a feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

Fraud detection methods are continuously developed to defend criminals in adapting to their fraudulent strategies.

These frauds are classified as:

- Credit Card Frauds: Online and Offline
- Card Theft
- Account Bankruptcy
- Device Intrusion
- Application Fraud
- Counterfeit Card
- Telecommunication Fraud

Some of the currently used approaches to detection of such fraud are:

- Artificial Neural Network
- Fuzzy Logic
- Genetic Algorithm
- Logistic Regression
- Decision tree
- Support Vector Machines
- Bayesian Networks
- Hidden Markov Model
- K-Nearest Neighbour

1.1 DATA ANALYSIS WITH MACHINE LEARNING

Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning uses category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

Machine started leaning own there on from past experience and without any explicit programming is called machine learning. Machine learning is subset of Artificial intelligence. Machine learning is the process of developing, testing, and applying predictive algorithms to achieve this goal.

Example: tesla automated car, chat bot.

1.2 TYPES OF MACHINE LEARNING

1.2.1 **Supervised machine learning**:

- It consist of a target/outcome variable which is to be predicted from a given set of predictors.
- Using these set of variables, we generate a function that map inputs to desired outputs.
- The training process continue until the model achiever a desired level of accuracy on the training data.
- We give enough input to the algorithm and supervisor keep correcting result until it became to make decision by their own.
- Input have expected output associated with them.
- When algorithm is trained it will predict correct output of never seen input.

Application: hi cortana, ok google , siri etc.

Algorithm:

- Linear regression
- Random Forest
- Support Vector Machine

1.2.2 **Unsupervised machine learning**:

- In this we do not have any target or outcome variable to predict/estimate.
- It is used for clustering population in different groups, which is widely used for segmenting customers in different groups of specific intervention.
- Input don't have expected output associated with them instead it detect pattern of initial characteristic of input data.
- No previous knowledge
- Example clustering: similar data group together, if we are watching football match for first time then we make assumption on our own.

Algorithm:

- K-Mean algorithm
- Apriori algorithm
- Hierarchical clustering

1.2.3 **Reinforcement machine learning**:

- Using this, the machine is trained to make specific decisions.
- The machine is exposed to an environment where it trains itself continually using trial and error.
- This machine learn from past experience and tries to capture the best possible knowledge to make accurate business decisions.
- Automatically determine the ideal behavior
- Interaction between two elements environment and learning agent
- We give reward and penalty for the decision so that accuracy get increased.

Example: Markov decision process.

CHAPTER 2

LITERATURE SURVEY

2.1 MACHINE LEARNING ALGORITHMS

2.1.1 The Random Forests Algorithm

Select random samples from a given dataset. Construct a decision tree for each sample and get a prediction result from each decision tree. Perform a vote for each predicted result. Select the prediction result with the most votes as the final prediction.

Advantages:

•Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.

•It does not suffer from the over-fitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

•The algorithm can be used in both classification and regression problems.

•Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.

•You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages:

•Random forests is slow in generating predictions because it has multiple decision trees.

•The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Finding important features :

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Random Forests vs Decision Trees

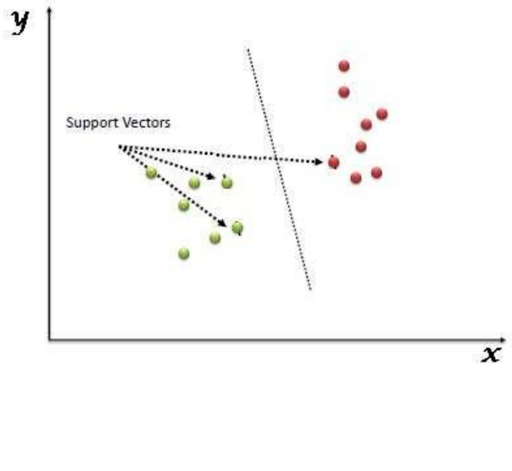
- •Random forests is a set of multiple decision trees.
- •Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- •Decision trees are computationally faster.
- •Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

2.1.2 Support Vector Machine

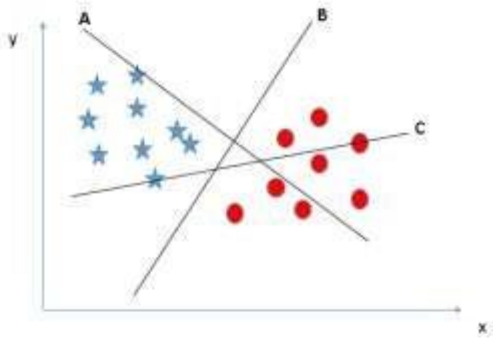
Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, perform classification by finding the hyper-plane that differentiate the two classes very well

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

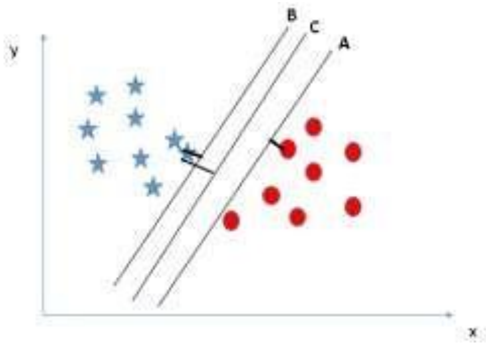


How does it work

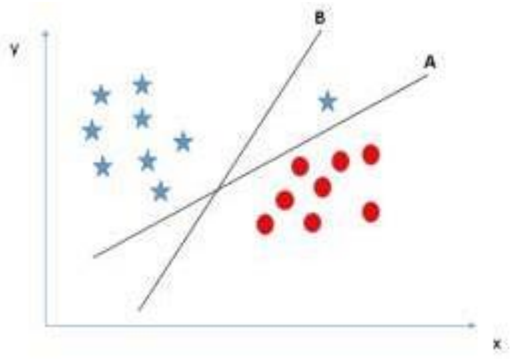
- **Identify the right hyper-plane:** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle. Select the hyper-plane which segregates the two classes better.



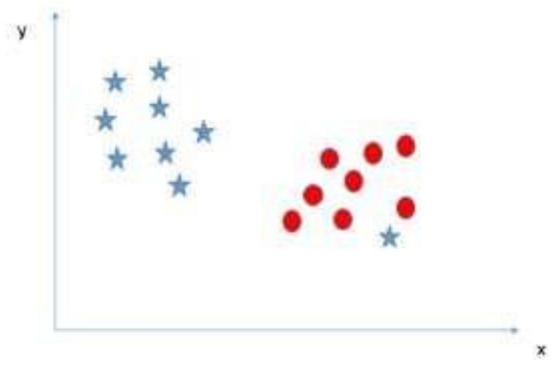
- **Identify the right hyper-plane:** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now Here, maximizing the distances between nearest data point and hyper-plane will help to decide the right hyper-plane.
This distance is called as **Margin**.



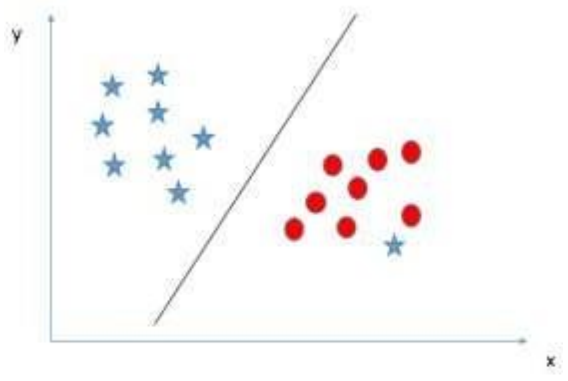
- **Identify the right hyper-plane:** Select the hyper-plane **B** as it has higher margin compared to **A.** But here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.Hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**



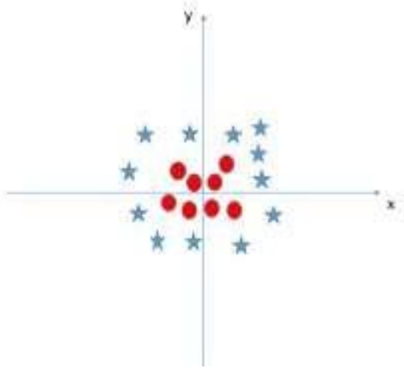
- **We classify two classes:** segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, SVM is robust to outliers.



- **Find the hyper-plane to segregate to classes:** We can’t have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linearhyper-plane.

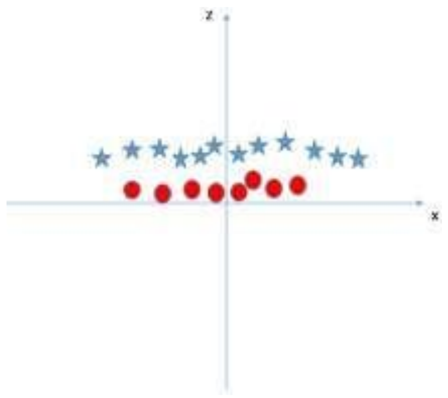


SVM can solve this problem. Easily! It solves this problem by Introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z:

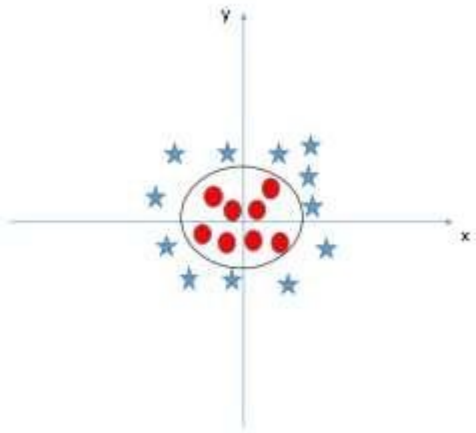


In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.



In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel **trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.When we look at the hyper-plane in original input space it looks like a circle:



Pros and Cons associated with SVM

- **Pros:**
  - It works really well with clear margin of separation
  - It is effective in high dimensional spaces.
  - It is effective in cases where number of dimensions is greater than the number of samples.
  - It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- **Cons:**
  - It doesn't perform well, when we have large data set because the required training time is higher
  - It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

2.1.3 Linear Regression Algorithm:

Linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

Linear Regression Model Representation

It is an attractive model because the representation is so simple. The representation is a linear equation that combines a specific set of input values (x) the solution to which is the predicted output for that set of input values (y). As such, both the input values (x) and the output value are numeric. The linear equation assigns one scale factor to each input value or column, called a coefficient and represented by the capital Greek letter Beta (B). One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the intercept or the bias coefficient.
For example, in a simple regression problem (a single x and a single y), the form of the model would be:

y = B0 + B1*x

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. B0 and B1 in the above example).

When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model (0 * x = 0). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

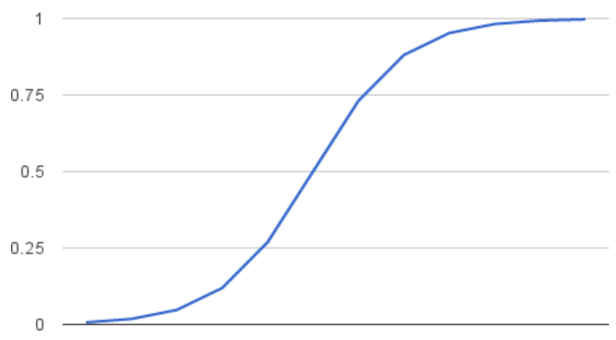
Linear Regression Learning the Model

1. Simple Linear Regression

Simple linear regression when we have a single input, we can use statistics to estimate the coefficients. This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

2. Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients. The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.



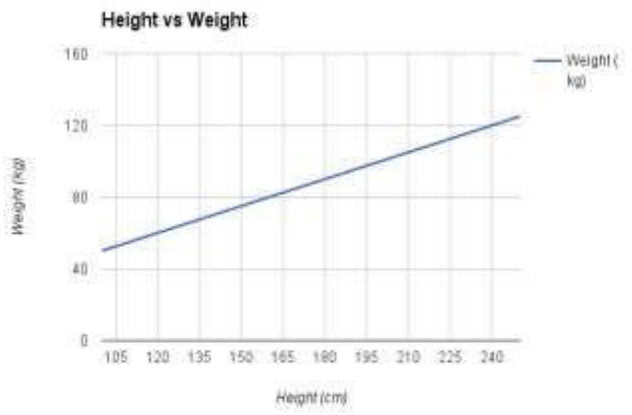
Making Predictions with Linear Regression

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs. Let's make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

y = B0 + B1 * x1 or weight =B0 +B1 * height

Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

Sample Height vs Weight Linear Regression



2.1.4 Logistic Regression Algorithm

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

Logistic Function

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)

Where e is the base of the natural logarithms and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation. Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation: $y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$ Where,

y is the predicted output

b0 is the bias or intercept term

b1 is the coefficient for the single input value (x).

Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Logistic Regression Predicts Probabilities

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modelling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

$P(sex=male|height)$

Written another way, we are modelling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

$P(X) = P(Y=1|X)$

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction.

$p(X) = e^{(b0 + b1*X)} / (1 + e^{(b0 + b1*X)})$

$\ln(p(X) / 1 – p(X)) = b0 + b1 * X$

$\ln(odds) = b0 + b1 * X$

We can move the exponent back to the right and write it as:

$odds = e^{(b0 + b1 * X)}$

The linear combination relates to the log-odds of the default class.

Learning the Logistic Regression Model

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data
The best coefficients would result in a model that would predict a value very close to 1 (e.g. male) for the default class and a value very close to 0 (e.g. female) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

Prepare Data for Logistic Regression

The assumptions made by logistic regression about the distribution and relationships in data are much the same as the assumptions made in linear regression.

- **Binary Output Variable**: The logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- **Remove Noise**: Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- **Gaussian Distribution**: Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model.
- **Remove Correlated Inputs**: Like linear regression, the model can overfit if you have multiple highly-correlated inputs.
- **Fail to Converge**: It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

2.2 MACHINE LEARNING TOOLS

The proliferation of free open source software has made machine learning easier to implement both on single machines and at scale, and in most popular programming languages. These open source tools include libraries for the likes of Python, R, C++, Java, Scala, JavaScript and Go.

2.2.1 Pandas:

It is a free library for modeling in Python. It converts CSV, JSON, and TSV data files or a SQL database into rows and columns to simplify analysis. *Pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. *Pandas* is a Num-Focus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to donate to the project. Pandas is hands down one of the best libraries of python. It supports reading and writing excel spreadsheets, CVS's and a whole lot of manipulation. It is more like a mandatory library you need to know if you're dealing with datasets from excel files and CSV files, i.e. for Machine learning and data science.

Benefits of pandas

- A lot of functionality
- Active community
- Extensive documentation
- Plays well with other packages
- Built on top of NumPy
- Works well with scikit-learn

Python has long been great for data mining and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.Combined with the excellent IPython toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.

A fast and efficient DataFrame object for data manipulation with integrated indexing Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format. Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form. Flexible reshaping and pivoting of data sets. Intelligent label-based slicing, fancy indexing, and sub setting of large data sets. Columns can be inserted and deleted from data structures for size mutability.

Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.

High performance merging and joining of data sets. Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

2.2.2 Matplotlib:

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

It is a plotting library, meaning it's used for visualization of machine learning data. It's also flexible and easily integrated with third-party tools such as ggplot, NumPy, and HoloViews.

A Matplotlib figure can be categorized into several parts as below:

- Figure: It is a whole figure which may contain one or more than one axes (plots). You can think of a Figure as a canvas which contains plots.
- Axes: It is what we generally think of as a plot. A Figure can contain many Axes. It contains two or three (in the case of 3D) Axis objects. Each Axes has a title, an x-label and a y-label.
- Axis: They are the number line like objects and take care of generating the graph limits.
- Artist: Everything which one can see on the figure is an artist like Text objects, Line2D objects, collection objects. Most Artists are tied to Axes.

Creating different types of graphs with Pyplot

1) Bar Graphs
2) Pie Charts
3) Histogram
4) Scatter Plots and 3-D plotting

2.2.3 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. It is open source, which is an added advantage of NumPy. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Features:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

NumPy fully supports an object-oriented approach, starting, once again, with *ndarray*. For example, *ndarray* is a class, possessing numerous methods and attributes. Many of its methods mirror functions in the outer-most NumPy namespace, giving the programmer complete freedom to code in whichever paradigm she prefers and/or which seems most appropriate to the task at hand.

NumPy gives us the best of both worlds: element-by-element operations are the "default mode" when an ndarray is involved, but the element-by-element operation is speedily executed by pre-compiled C code. In NumPy

```
c = a * b
```

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

2.3 MACHINE LEARNING FRAMEWORK

2.3.1 Scikit-learn:

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction. Scikit-learn comes loaded with a lot of features. scikit-learn is used to build models. It should not be used for reading the data, manipulating and summarizing it.

Components of scikit-learn:

- Supervised learning algorithms: Think of any supervised learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn.
- Cross-validation: There are various methods to check the accuracy of supervised models on unseen data
- Unsupervised learning algorithms: Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- Various toy datasets: This came in handy while learning scikit-learn. I had learnt SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- Feature extraction: Useful for extracting features from images and text (e.g. Bag of words)

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

Implementation: Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

2.3.2 Anaconda:

Anaconda is a package manager, an environment manager, a Python/R data science distribution, and a collection of over 1,500+ open source packages. Anaconda is free and easy to install, and it offers free community support.

Download anaconda from the official website and Install it. After installing Anaconda or Miniconda, for a desktop graphical user interface (GUI) use Navigator. For Anaconda prompt (or Terminal on Linux or macOS), use that and conda.

Packages available in Anaconda

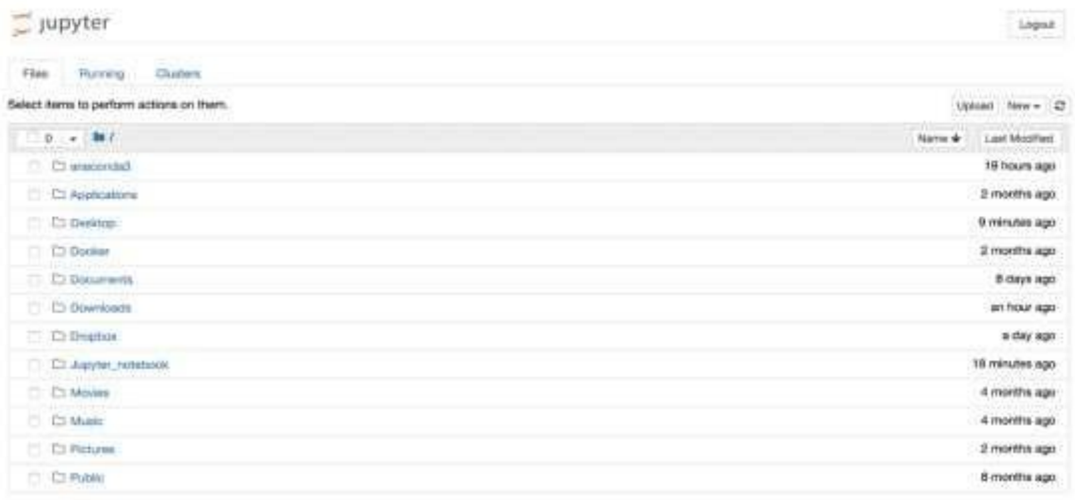
- Over 200 packages are automatically installed with Anaconda.
- Over 2000 additional open source packages (including R) can be individually installed from the Anaconda repository with the condainstall command.
- Thousands of other packages are available from Anaconda Cloud.
- Other packages can be downloaded using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.

2.3.3 Jupyter notebook:

Project Jupyter is a comprehensive software suite for interactive computing, that includes various packages such as Jupyter Notebook, QtConsole, nbviewer, JupyterLab. This tutorial gives you an exhaustive knowledge on Project Jupyter. By the end of this tutorial, you will be able to apply its concepts into your software coding.

"Jupyter" is a loose acronym meaning Julia, Python, and R.

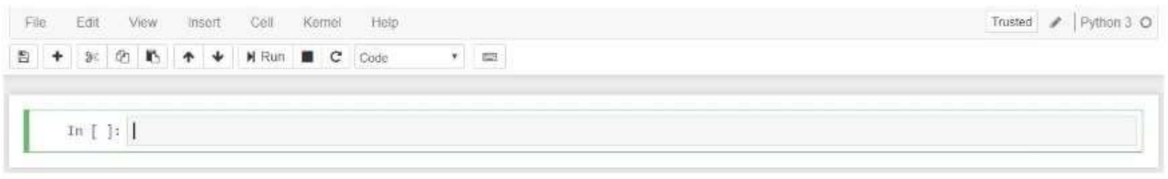
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



Select items to perform actions on them.

anaconda3	19 hours ago
Applications	2 months ago
Desktop	9 minutes ago
Docker	2 months ago
Documents	8 days ago
Downloads	an hour ago
Dropbox	a day ago
Jupyter_notebook	18 minutes ago
Movies	4 months ago
Music	4 months ago
Pictures	2 months ago
Public	8 months ago

The notebook interface



2.3.4 Spyder:

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.n Furthermore, Spyder offers built-in integration with many popular scientific packages, including NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy, and more.n Beyond its many built-in features, Spyder can be extended even further via third-party plugins.n Spyder can also be used as a PyQt5 extension library, allowing you to build upon its functionality and embed its components, such as the interactive console or advanced editor, in your own software.

conda install -c anaconda spyder

Core components

1. Editor: Work efficiently in a multi-language editor with a function/class browser, real-time code analysis tools (pyflakes, pylint, and pycodestyle), automatic code completion (jedi and rope), horizontal/vertical splitting, and go-to-definition.
2. Interactive console: Harness the power of as many IPython consoles as you like with full workspace and debugging support, all within the flexibility of a full GUI interface. Instantly run your code by line, cell, or file, and render plots right inline with the output or in interactive windows.
3. Documentation viewer: Render documentation in real-time with Sphinx for any class or function, whether external or user-created, from either the Editor or a Console.
4. Variable explorer: Inspect any variables, functions or objects created during your session. Editing and interaction is supported with many common types, including numeric/strings/bools, Python lists/tuples/dictionaries, dates/timedeltas, Numpy arrays, Pandas index/series/dataframes, PIL/Pillow images, and more.
5. Development tools: Examine your code with the static analyzer, trace its execution with the interactive debugger, and unleash its performance with the profiler. Keep things organized with project support and a built-in file explorer, and use find in files to search across entire projects with full regex support.

2.4 DATA

It is useful to characterize learning problems according to the type of data they use. This is a great help when encountering new challenges, since quite often problems on similar data types can be solved with very similar techniques. For instance natural language processing and bioinformatics use very similar tools for strings of natural language text and for DNA sequences. Vectors constitute the most basic entity we might encounter in our work. For instance, a life insurance company might be interesting in obtaining the vector of variables (blood pressure, heart rate, height, weight, cholesterol level, smoker, gender) to infer the life expectancy of a potential customer. A farmer might be interested in determining the ripeness of fruit based on (size, weight, spectral data). An engineer might want to find dependencies in (voltage, current) pairs. Likewise one might want to represent documents by a vector of counts which describe the occurrence of words. The latter is commonly referred to as bag of words features.

2.4.1 TYPES OF DATA

One of the challenges in dealing with vectors is that the scales and units of different coordinates may vary widely.

Lists: In some cases the vectors we obtain may contain a variable number of features. For instance, a physician might not necessarily decide to perform a full battery of diagnostic tests if the patient appears to be healthy.

Sets: Sets may appear in learning problems whenever there is a large number of potential causes of an effect, which are not well determined. Consequently we need to infer the properties of an object given a set of features, whose composition and number may vary considerably.

Matrices: Matrices are a convenient means of representing pairwise relationships. For instance, in collaborative filtering applications the rows of the matrix may represent users whereas the columns correspond to products. Only in some cases we will have knowledge about a given (user, product) combination, such as the rating of the product by a user.

Images: Images could be thought of as two dimensional arrays of numbers, that is, matrices. This representation is very crude, though, since they exhibit spatial coherence (lines, shapes) and (natural images exhibit) a multiresolution structure.

Video: Video adds a temporal dimension to images. Again, we could represent them as a three dimensional array.

Trees and Graphs: Trees and Graphs are often used to describe relations between collections of objects. In the case of gene ontology the relationships form a directed acyclic graph.

Strings: Strings occur frequently, mainly in the area of bioinformatics and natural language processing. They may be the input to our estimation problems, e.g. when classifying an e-mail as spam, when attempting to locate all names of persons and organizations in a text, or when modeling the topic structure of a document.

Compound structures: Compound structures are the most commonly occurring object. That is, in most situations we will have a structured mix of different data types. For instance, a webpage might contain images, text, tables, which in turn contain numbers, and lists, all of which might constitute nodes on a graph of webpages linked among each other.

CHAPTER-3
PROBLEM DESCRIPTION

3.1 PROBLEM STATEMENT

Our main objective in this project is to design a detection model to analyze the data set to find out the fraudulent transactions and applying various machine learning tools to find best probability of the data set. In order to save costs and time, it is important to filter the dataset to keep a certain success rate.

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 are fraudulent.
- The systems should be able to adapt themselves to new kinds 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.
- 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.

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.

3.2 RELATED WORK

Several studies have been delved into by many researchers and data miners in the detection of frauds related to Credit Cards. Various modern techniques based on Sequence Alignment, Machine learning, Artificial Intelligence, Genetic Programming, Data mining etc. has been evolved and is still evolving to detect fraudulent transactions in credit card. A sound and clear understanding on all these approaches is needed that will certainly lead to an efficient credit card fraud detection system. Survey of various techniques used in credit card fraud detection mechanisms has been done along with evaluation of each methodology based on certain design criteria. Analysis on Credit Card Fraud Detection Methods has been done as well other data driven approaches were extensively explored. Many works including decision trees and SVMs to decrease the risk of the banks. They have suggested Artificial Neural networks and Logistic Regression classification models are more helpful to improve the performance in detecting the frauds. Here the training data sets distribution became more biased and the distribution of the training data sets became more biased and the efficiency of all models decreased in catching the fraudulent transactions. This work is done to detect the credit card fraud in the dataset obtained by applying Logistic regression, Decision tree, SVM, Random Forest and to evaluate their Accuracy, sensitivity, specificity, precision using different models and compare and collate them to state the best possible model to solve the credit card fraud detection problem. The three classification models i.e. decision tree, neural network and logistic regression, among the three models neural network and logistic regression outperforms than the decision tree. Artificial Neural networks and Logistic Regression classification models are more helpful.

CHAPTER 4

PROPOSED WORK

4.1 OVERVIEW

We are preparing a detection model for data set analysis.

That data set is related to the credit card transactions of the customers. The goal is to predict whether the transaction is fraudulent or not.

In the data set based on some parameter the model will predict probabilities of the transaction of the credit card to be fraudulent.

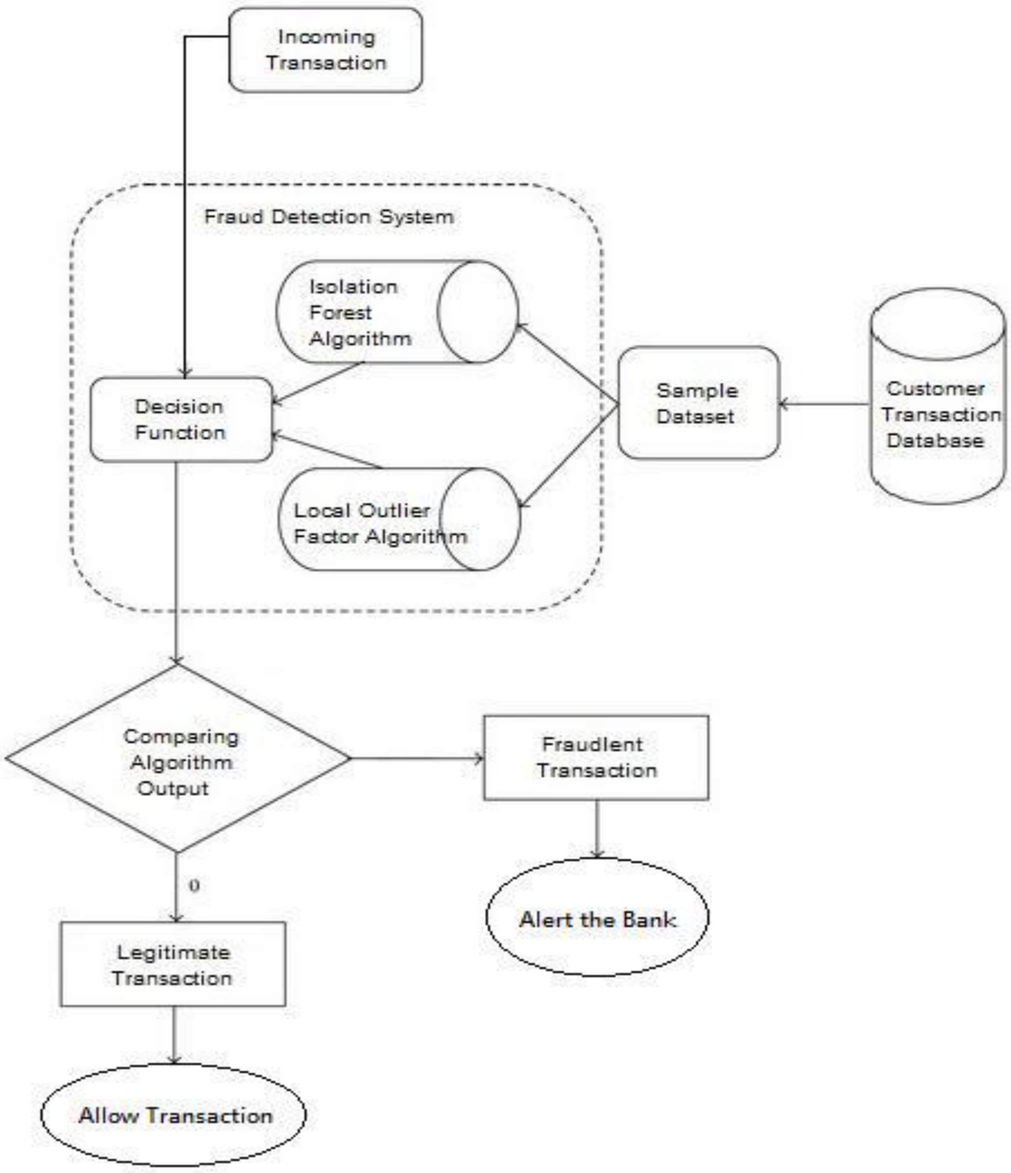
We have applied a machine learning algorithm to prepare a detection model and shown a comparison between machine learning algorithms.

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

BENEFIT OF PROPOSED WORK:-

- To reduce number of fraud transaction.
- Reduces time complexity.
- To add layer of security.

4.2 FLOWCHART



CHAPTER-5

DESIGN AND DEVELOPMENT

5.1TOOL DESCRIPTION

5.1.1 HARDWARE REQUIREMENTS

- CPU (Central Processing Unit) –Intel Core i3
- GPU (Graphics Processing Unit)
- RAM -8Gb

5.1.2 SOFTWARE REQUIREMENTS

- Python
- Anaconda
- Jupyter notebook
- Spyder

5.2 PRELIMINARY INSTALLATION

5.2.1 INSTALLING DEPENDENCIES

Following things are needed to execute the code we will be writing.

- Python3

  Download Python 3.7.1 from official website (https://www.python.org/downloads/) and install it.
- Anaconda Distribution

  Download Anaconda Distribution with Python 3.7.1 version from the official website (https://www.anaconda.com/distribution/) and Install it. After installing Anaconda, for a desktop graphical user interface (GUI) use Navigator.
- Jupyter notebook

  First install Python and Anaconda Distribution. When these are installed Jupyter notebook is also installed. To run the notebook:

```
jupyter notebook
```

CHAPTER 6

IMPLEMENTATION

6.1 DATASET

The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. Furthermore, there were no missing values in the data set. Equipped with this basic description of the data, let's jump into some exploratory data analysis.

6.2 DATA PRE-PROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Whenever the data is gathered from different online sources it is collected in raw format which is not suitable for the analysis. For achieving better results from the applied model in Machine Learning the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Steps in data preprocessing are:

- First the raw data is collected from the online data repositories for our project data set is downloaded from https://archive.ics.uci.edu/ml/datasets
- The downloaded data is in raw format so for making the data more usable first convert the dataset into comma separated values, for this purpose Microsoft Excel has text to column option available which can change the delimiter of a data set.
- Some cells which contain missing data field have to be either removed or replaced with the respective columns mean values.
- For analysis purpose the data has to be in numeric form so the string value has to be replaced with suitable numeric data .Thus the data columns which have yes and no value can be exchanged with 1 and 0.
- Another problem associated with raw data is that sometimes it has some values which appear to be numeric but are actually mistakenly printed text then that whole , those values also needs to be carefully examined and removed because if there is any value other that numeric the model will not work.

6.3 ALGORITHM

6.3.1 Logistic Function

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

Logistic Function

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)

Where e is the base of the natural logarithms and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation. Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation: y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)) Where,

y is the predicted output

b0 is the bias or intercept term

b1 is the coefficient for the single input value (x).

Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Logistic Regression Predicts Probabilities

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modelling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

Written another way, we are modelling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

P(X) = P(Y=1|X)

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction.

p(X) = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))

ln(p(X) / 1 – p(X)) = b0 + b1 * X

ln(odds) = b0 + b1 * X

We can move the exponent back to the right and write it as:

odds = e^(b0 + b1 * X)

The linear combination relates to the log-odds of the default class.

6.4 CODE AND CORRESPONDING RESULTS

Code : Importing all the necessary Libraries

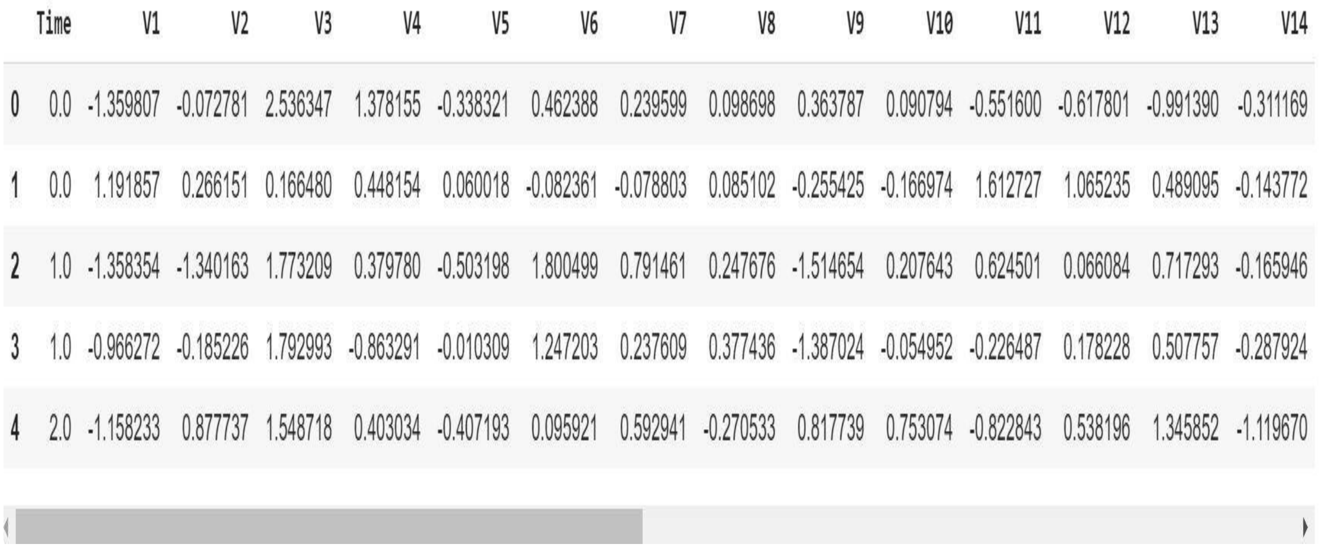
```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
```

Code : Loading the Data

```
# Load the dataset from the csv file using pandas
# best way is to mount the drive on colab and
# copy the path for the csv file
data = pd.read_csv("credit.csv")
```

Code : Understanding the Data

```
# Grab a peek at the data
data.head()
```



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

Code : Describing the Data

```
# Print the shape of the data
# data = data.sample(frac = 0.1, random_state = 48)
print(data.shape)
print(data.describe())
```

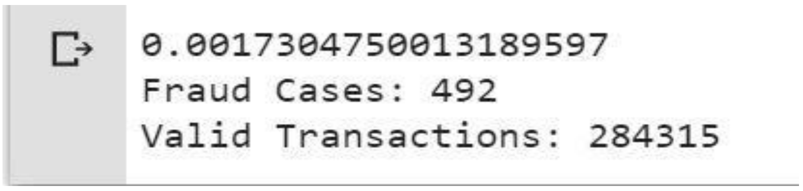
Output :

```
(284807, 31)
                Time            V1  ...         Amount          Class
count  284807.000000  2.848070e+05  ...  284807.000000  284807.000000
mean    94813.859575  3.919560e-15  ...      88.349619       0.001727
std     47488.145955  1.958696e+00  ...     250.120109       0.041527
min         0.000000 -5.640751e+01  ...       0.000000       0.000000
25%     54201.500000 -9.203734e-01  ...       5.600000       0.000000
50%     84692.000000  1.810880e-02  ...      22.000000       0.000000
75%    139320.500000  1.315642e+00  ...      77.165000       0.000000
max    172792.000000  2.454930e+00  ...   25691.160000       1.000000

[8 rows x 31 columns]
```

Code : Imbalance in the data

```
# Determine number of fraud cases in dataset
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
```

```
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```



```
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
```

Only *0.17%* are fraudulent transaction out all the transactions. The data is highly unbalanced. Let’s first apply our models without balancing it and if we don’t get a good accuracy then we can find a way to balance this dataset. But first, let’s implement the model without it and will balance the data only if needed.

Code : Print the amount details for Fraudulent Transaction

```
print(“Amount details of the fraudulent transaction”)
fraud.Amount.describe()
```

Output :

```
Amount details of the fraudulent transaction

count     492.000000

mean      122.211321

std       256.683288

min         0.000000

25%         1.000000

50%         9.250000

75%       105.890000

max      2125.870000

Name: Amount, dtype: float64
```

Code : Print the amount details for Normal Transaction

```
print(“details of valid transaction”)
valid.Amount.describe()
```

Output :

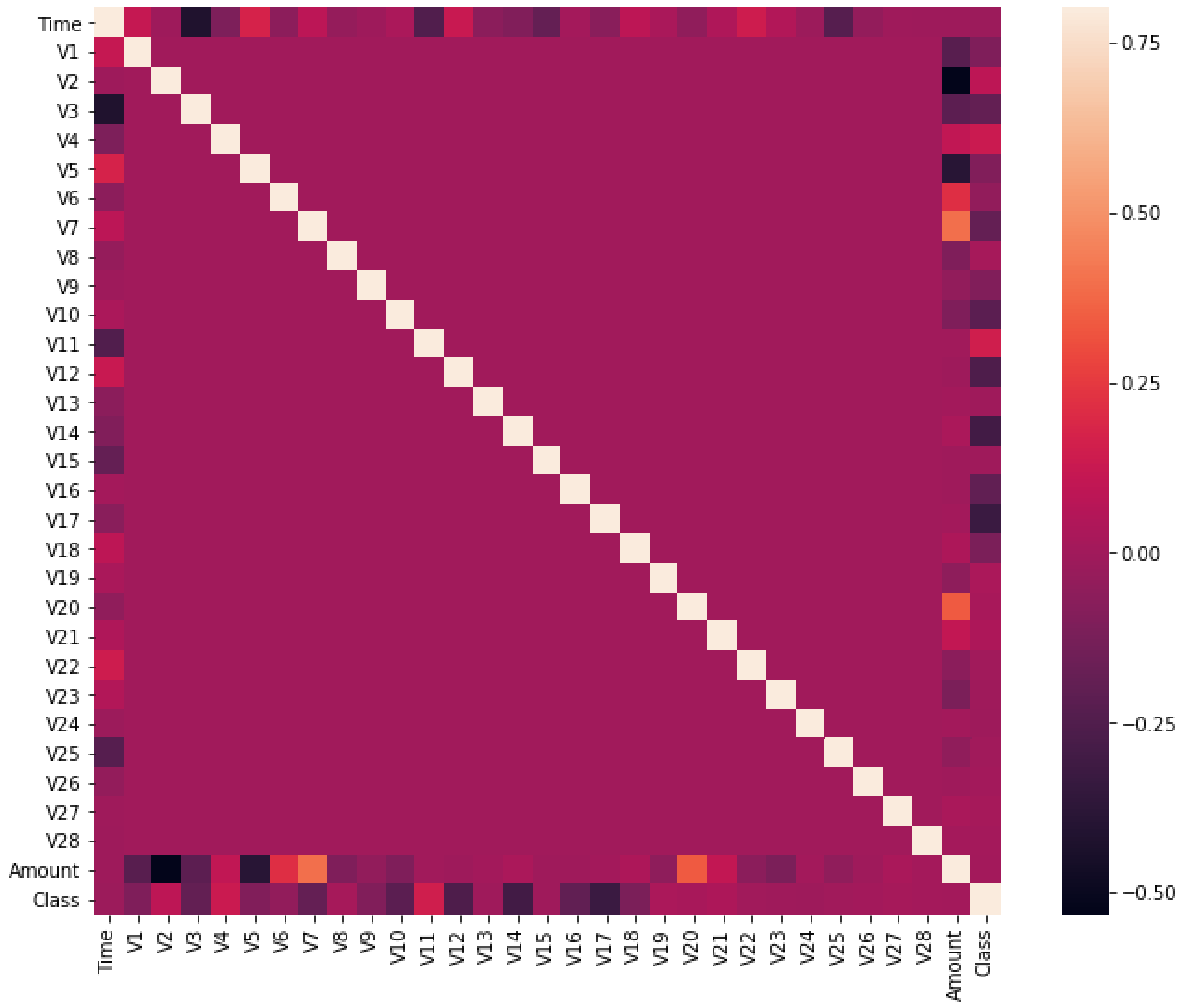
```
Amount details of valid transaction
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

As we can clearly notice from this, the average Money transaction for the fraudulent ones is more. This makes this problem crucial to deal with.

Code : Plotting the Correlation Matrix

The correlation matrix graphically gives us an idea of how features correlate with each other and can help us to predict what are the features that are most relevant for the prediction.

```
# Correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



In the Heat-Map we can clearly see that most of the features do not correlate to other features but there are some features that either has a positive or a negative correlation with each other. For example, *V2* and *V5* are highly negatively correlated with the feature called *Amount*. We also see some correlation with *V20* and *Amount*. This gives us a deeper understanding of the Data available to us.

Code : Separating the X and the Y values

Dividing the data into inputs parameters and outputs value format

```
# dividing the X and the Y from the dataset
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
print(X.shape)
print(Y.shape)
# getting just the values for the sake of processing
# (its a numpy array with no columns)
xData = X.values
yData = Y.values
```

Output :

```
(284807, 30)
(284807, )
```

Training and Testing Data Bifurcation

We will be dividing the dataset into two main groups. One for training the model and the other for Testing our trained model’s performance.

```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
xTrain, xTest, yTrain, yTest = train_test_split(  xData, yData, test_size = 0.2, random_state =
42)
```

Code : Building a Random Forest Model using skicit learn

```
# Building the Random Forest Classifier (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
# predictions
yPred = rfc.predict(xTest)
```

Code : Building all kinds of evaluating parameters

```
# Evaluating the classifier
# printing every score of the classifier
# scoring in anything
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))

rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))

MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```

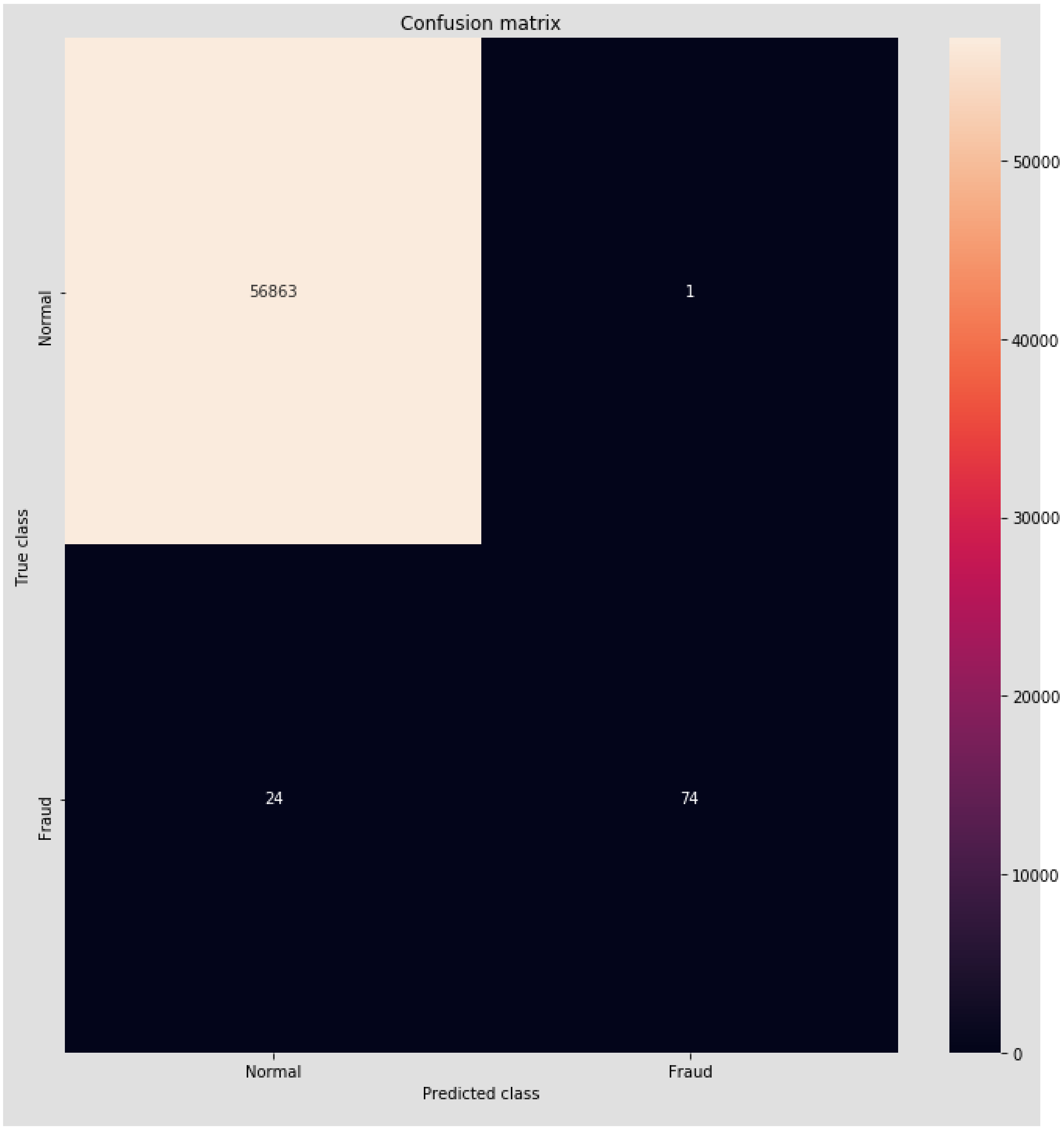
Output :

```
The model used is Random Forest classifier

The accuracy is  0.9995611109160493

The precision is 0.9866666666666667

The recall is 0.7551020408163265

The F1-Score is 0.8554913294797689

The Matthews correlation coefficient is0.8629589216367891
```

Code : Visulalizing the Confusion Matrix

```
# printing the confusion matrix
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize =(12, 12))
sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS, annot = True, fmt
="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

Output :


Confusion matrix
56863
1
24
74
Normal
Fraud
True class
Normal
Fraud
Predicted class
50000
40000
30000
20000
10000
0

CHAPTER-7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Credit card fraud is without a doubt an act of criminal dishonesty. This project explains', how machine learning can be applied to get better results in fraud detection along with the algorithm, pseudo-code, explanation its implementation and experimentation results.

From the experiments the result that has been concluded is that Decision tree shows accuracy of 82.5% but the best results are obtained by Random forest with a precise accuracy of 89.8%. The results obtained thus conclude that Random forest shows the most precise and high accuracy of 89.8% in problem of credit card fraud detection with the dataset.

The Random forest algorithm will perform better with a larger number of training data, but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more pre-processing to give better results at the results shown by SVM is great but it could have been better if more pre-processing have been done on the data.

7.2 FUTURE SCOPE

The project can be further improved by using more data entries, recent data that is being generated can also we added to the existing data set to continuously increase the accuracy of the model.

As of now the accuracy is 89.8% by training the model against much larger data set with more number of entries the accuracy will increase. The accuracy of outcome might also increase by using other algorithms such as logistic regression, support vector machine. The future scope of this work is to detect real time fraud transaction for high streaming real time data.

BIBLIOGRAPHY AND REFRENCES

- https://www.udemy.com/machinelearning/
- https://archive.ics.uci.edu/ml/index.php
- https://www.geeksforgeeks.org/machine-learning/
- https://machinelearning-blog.com
- https://stackoverflow.com/search?q=machine+learning
- https://www.anaconda.com/distribution/
- https://docs.anaconda.com/anaconda/
- https://scikit-learn.org/stable/