

# Toward Credit card Fraud Detection With Machine Learning in Python

A MINI PROJECT REPORT

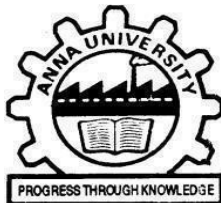
*Submitted by*

BHARATHI.J	[410718104010]
DIVYA.S	[410718104015]
GOPIKA.E	[410718104020]
POOJA.S	[410718104051]

*in partial fulfilment for the award of the degree of*  
**BACHELOR OF ENGINEERING**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**



**DHANALAKSHMI COLLEGE OF ENGINEERING**



**ANNA UNIVERSITY::CHENNAI 600 025**

**APRIL 2021**

**ANNA UNIVERSITY:: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report - **"CREDIT CARD FRAUD DETECTION WITH MACHINE LEARNING IN PYTHON"** is the bonafide work of **"BHARATHI.J (410718104010), DIVYA.S(410718104015) ,GOPIKA.E(410718104020) and POOJA.S(410718104051)"** who carried out the project work under my supervision.

**SIGNATURE**

**Dr. S Sivasubramanian, Ph.D.,  
HEAD OF THE  
DEPARTMENT,**

Computer Science and  
Engineering, Dhanalakshmi  
College of Engineering,  
Dr. VPR Nagar,  
Manimangalam, Chennai –  
601 301.

**SIGNATURE**

**Ms.R.Reni Hena Helan, M.Tech,  
PROJECT SUPERVISOR ,  
ASSISTANT PROFESSOR,**

Computer Science and  
Engineering ,Dhanalakshmi  
College of engineering,  
Dr. VPR Nagar,  
Manimangalam, Chennai -601  
301.

Submitted for the project viva-voce examination held at  
Dhanalakshmi College of Engineering on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We express the heartfelt thanks to our honorable **Founder and Chairman Dr.V.P.Ramamurthi, Ph.D.,** Dhanalakshmi College of Engineering, for guiding us and permitting us to do our project by our own.

We express our sincere gratitude and wish to thank our beloved **Principal, Dr. V. Krishnakumar, Ph.D.,** for his support and guidance.

We extend our gratitude and heartfelt thanks to **Head of the Department, Dr. S.Sivasubramanian, Ph.D** for guiding us in all aspects of our project in each stage and providing us with valuable suggestions.

We would like to thank at most to our **Project Supervisor, Ms.R.Reni Hena Helan, M.Tech, Assistant Professor** who rendered valuable guidance and full support always.

Finally, we take this opportunity to thank all the **Faculty members of the Department of Computer Science and Engineering** for their unwavering support and cooperation which made us keep our zeal and spirits high to complete this project work successfully.

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	VII
	LIST OF SYMBOLS	VIII
	LIST OF FIGURES	X
1	INTRODUCTION	1
2	SYSTEM ANALYSIS	12
2.1	Existing System	12
2.1.1	Disadvantages	12
2.2	Proposed System	13
2.2.1	Advantages	13
3	SYSTEM REQUIREMENTS	
3.1	Introduction	14
3.2.1	Hardware requirements	15
3.2.2	Software requirements	15
3.3	Technologies Used	15
3.4.1	Python IDE	15
3.4.2	Jupyter Notebook in Python Anaconda	16

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>18</b>
4.1	System Architecture	18
4.2	UML Diagrams	19
4.2.1	Use case Diagram	19
4.2.2	Class Diagram	20
4.2.3	Sequence Diagram	21
4.2.4	Activity Diagram	22
<b>5</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>23</b>
5.1	Algorithm Used	23
5.2	Modules	40
5.2.1	Data Processing	42
5.2.2	EDA	43
5.2.3	Feature Selection and Data Split	44
5.2.4	Modelling	45
5.2.5	Evaluation of the Models	46

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>6</b>	<b>SYSTEM TESTING</b>	<b>49</b>
6.1	Introduction	49
6.2	Goals of Testing	50
6.3	Testing Methodologies	50
6.3.1	Unit Testing	50
6.3.2	Integration Testing	51
6.3.3	White Box Testing	52
6.3.4	Black Box Testing	52
6.3.5	Program Testing	53
6.3.6	Security Testing	53
6.3.7	User Acceptance Testing	54
6.3.8	Validation Testing	54
<b>7</b>	<b>RESULTS</b>	<b>55</b>
<b>8</b>	<b>CONCLUSION AND FUTURE WORKS</b>	<b>56</b>
	<b>REFERENCE</b>	<b>57</b>
	<b>APPENDIX I</b>	
	<b>APPENDIX II</b>	

## **ABSTRACT**

The online shopping growing day to day. Credit cards are used for purchasing goods and services through online transactions and e-commerce platforms.

Since the card usage has been increased, credit card fraudulent has also been increasing.

Fraud detection process is a process used to identify those fraudulent transactions so that

customers are not charged for the items that they did not purchase. In this project, our goal

is to build classification models that classify or detect if the transactions are fraud or not

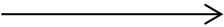
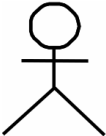

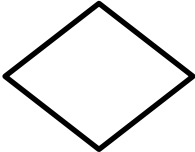


and to evaluate the accuracy of those models using various packages in python and given

dataset (information) containing the transactions between people.

Through this project we will be able to find an efficient model that will play a key role in

reducing these credit card fraudulent.

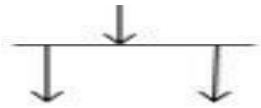
## LIST OF SYMBOLS

SYMBOL	SYMBOL NAME
	Data Flow
	Actor
 Start	
	Decision
	Process
	Input/Output

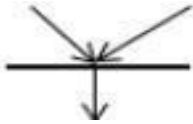




Sequence



Fork



Join



Synchronization



Node



Class



Entity



Stop

## LIST OF FIGURES

Figure No	Figure Name	Page no
4.1.1	System Architecture	18
4.2.1	Use Case Diagram	19
4.2.2	Class Diagram	20
4.2.3	Sequence Diagram	21
4.2.4	Activity Diagram	22

## CHAPTER 1

### INTRODUCTION

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.

The main contributions of this project are listed as follows:

Massimiliano Zanin conceived and elaborated the method and performed the numerical experiments. Massimiliano Zanin, Miguel Romance, Regino Criado, and Santiago Moral analyzed

the data, prepared the figures, and wrote the text of the Manuscript.

## CHAPTER 2

### SYSTEM ANALYSIS

#### 2.1 EXISTING SYSTEM

The fraud detection module will work in the following steps:

- The Incoming set of transactions and amount are treated as credit card transactions.
- The credit card transactions are given to machine learning algorithms as an input.
- The output will result in either fraud or valid transaction by analyzing the data and observing
- a pattern and using machine learning algorithms such as local outlier factor and isolation

forest to do anomaly detection. The fraud transactions are given to alarm which alerts the

user that fraud transaction has occurred and the user can block the card to prevent further

financial loss to him as well as the credit card company.

- The valid transactions are treated as genuine transactions.
- Reduction in number of fraud transactions.
- User can safely use his credit card for online transaction.
- Added layer of security.

##### 2.1.1 DISADVANTAGES OF EXISTING SYSTEM

- Machine learning algorithms work only for huge sets of data.
- For smaller amount of data the results may be not accurate.
- For large organizations, this data volume is not an issue but for others, there must

be enough data points to identify legitimate cause and effect relations.

- It takes a significant amount of data for machine learning models to become accurate.

## **2.2 PROPOSED SYSTEM**

- Assume that you are employed to help a credit card company to detect potential fraud cases.
- So that the customers are ensured that they won't be charged for the items they did not purchase.
- You are given a dataset containing the transactions between people, the information that they are fraud or not, and you are asked to differentiate between them.
- This is the case we are going to deal with. Our ultimate intent is to tackle this situation by building classification models to classify and distinguish fraud transactions.

### **2.2.1 ADVANTAGES OF PROPOSED SYSTEM:**

- This project helps major financial related or other major companies detect fraud transactions before the fraud activities turn into significant damage to their company.
- All of these models have an accuracy rate of 90 % and above.
- This project helps you figure out the best classification model (algorithm) to detect or predict fraudulent transaction.

## **CHAPTER 3**

### **System requirements**

#### **3.1 INTRODUCTION**

Design is a multi- step that focuses on data structure software architecture, procedural details, algorithm etc... and interface between modules. The design process also translate the requirements into presentation of software that can be accessed for quality before coding begins. Computer software design change continuously as new methods; better analysis and border understanding evolved. Software design is at relatively early stage in its revolution. Therefore, software design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However techniques for software designs do exist, criteria for design qualities are available and design notation can be applied.

##### **3.2.1 Hardware requirements**

- RAM: 4 GB and Higher
- Processor : Intel i3 and above
- Hard Disk : 500GB : Minimum

##### **3.2.2 Software requirements**

- Operating System : Windows or Linux
- Python IDE : python 2.7.x and above  
Setup tools and pip to be installed for 3.6 and above
- Language : Python
- Software used:  
JUPITER NOTEBOOK in ANACONDA ENVIRONMENT



- Libraries used:  
Pandas, numpy, scikit learn etc

### 3.3 TECHNOLOGIES USED

- PYTHON
- JUPITERNOTEBOOK in ANACONDA ENVIRONMENT

#### 3.4.1 PYTHON IDE:

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

#### Features:

Easy to code: Python is a high-level programming language. ...  
Free and Open Source: ...  
Object-Oriented Language: ...  
GUI Programming Support: ...  
High-Level Language: ...  
Extensible feature: ...  
Python is Portable language: ...  
Python is Integrated language:

#### USAGE:

- Web Development. Python can be used to make web-applications at a rapid rate. ...
- Game Development. ...
- Machine Learning and Artificial Intelligence. ...
- Data Science and Data Visualization. ...
- Desktop GUI. ...
- Web Scraping Applications. ...
- Business Applications. ...
- CAD Applications.

### Operating system supported:

- Mac OS X
- Source release
- Windows

### Platform:

- Linux.
- Windows Vista and newer for Python 3.7, Windows XP and newer for Python 2.7.
- FreeBSD 10 and newer.
- macOS Snow Leopard (macOS 10.6, 2008) and newer.

### Size:

- Windows: 25MB
- Linux: 25MB
- macOS: 25MB

## 3.4.2 JUPITER NOTEBOOK IN PYTHON ANANCONDA

The Jupyter Notebook application allows you to **create and edit documents** that display the input

and output of a Python or R language script. Once saved, you can share these files with others.

NOTE: Python and R language are included by default, but with customization, Notebook can run

several other kernel environments.

## Technical Specification:

- CPU: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs.
- RAM: 32 GB (or 16 GB of 1600 MHz DDR3 RAM)
- Storage: 300 GB. ...
- Internet access to download the files from Anaconda.org or a USB drive containing all of the files you need with alternate instructions for air gapped installations.

## FEATURES:

- It is free and open-source
- It has more than 1500 Python/***R data science packages***
- Anaconda simplifies package management and deployment
- It has tools to easily collect data from sources using machine learning and AI
- It creates an environment that is easily manageable for deploying any project
- Anaconda is the industry standard for developing, testing and training on a single machine
- It has good community support- you can ask your questions there.

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 System Architecture

Machine Learning Use case

## Credit Card Fraud Detection

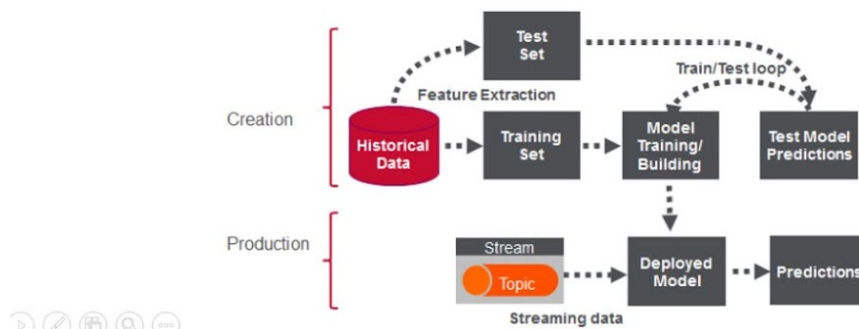


Figure 4.1.1 architecture diagram

## 4.2 UML Diagrams:

Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

### 4.2.1 Use case Diagram:

A use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. It consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system

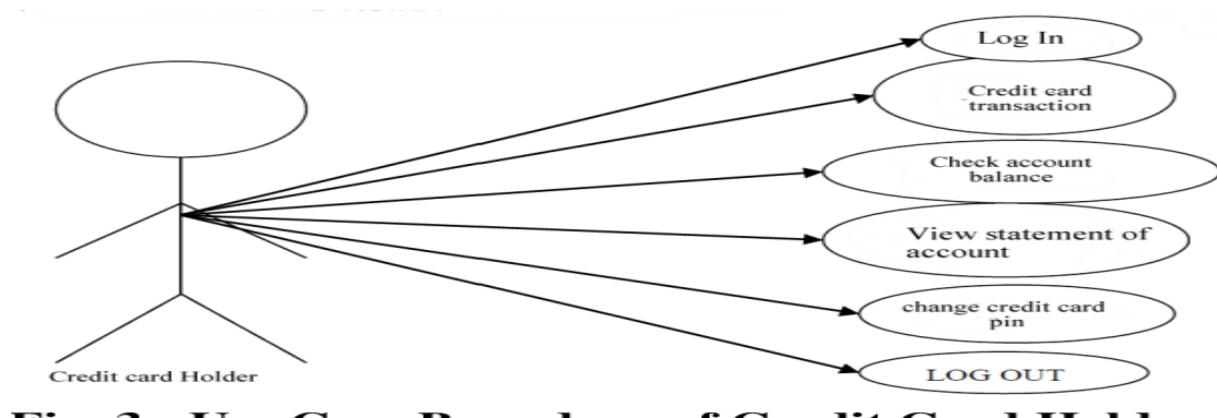


Figure 4.2.1 Use case Diagram

4.2.2Class Diagram

A Class diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

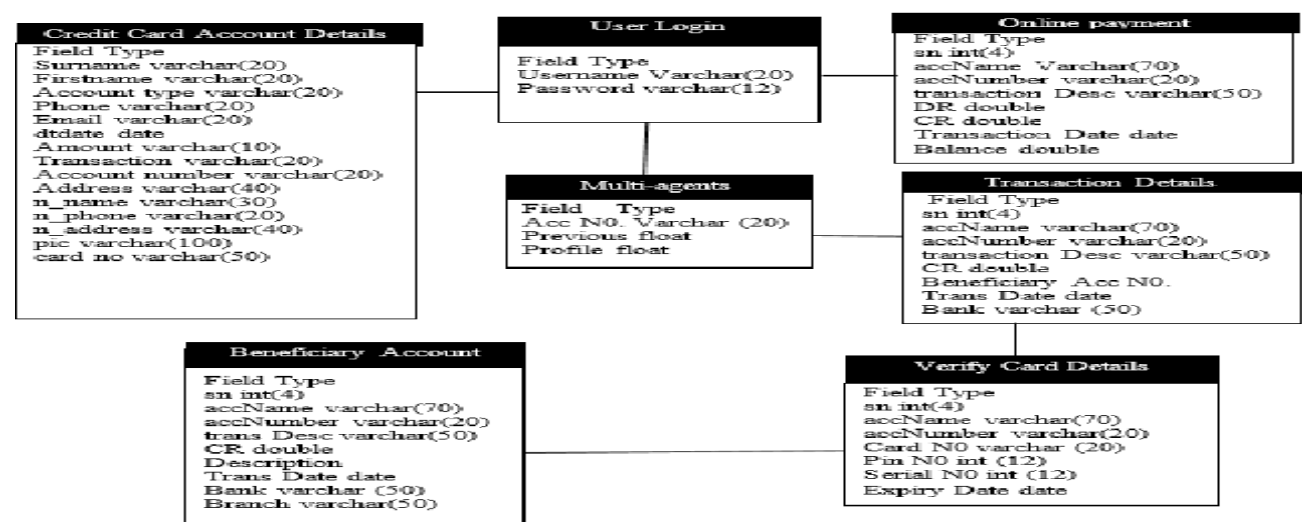
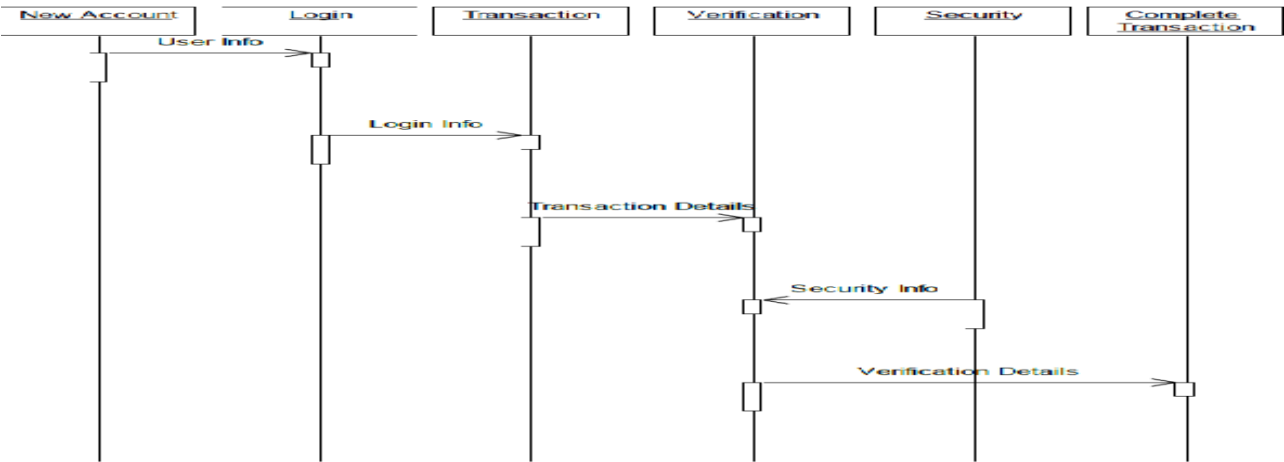


Fig. 12: Entity Relationship Diagrams of Credit Card

FIGURE 4.2.2: CLASS DIAGRAM

4.2.3 Sequence diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



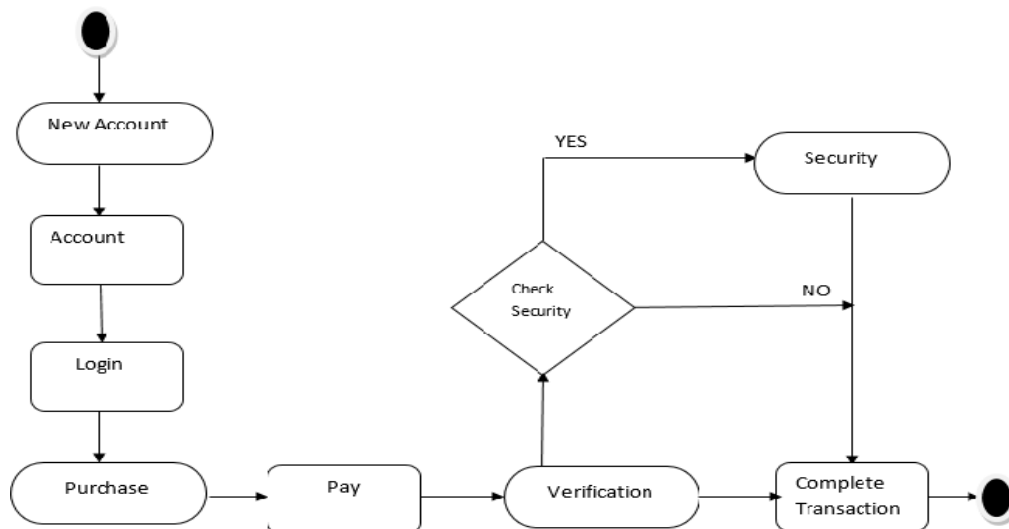
**Fig. 5: Sequence Diagram of Credit Card Transactions**

**Figure 4.2.3Sequence Diagram**



#### 4.2.4 Activity Diagram:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.



**Fig. 8: Activity Diagram of Credit Card Transactions**

**FIGURE : 4.2.4 ACTIVITY DIAGRAM**

## CHAPTER 5 SYSTEM IMPLEMENTATION

### 5.1 ALGORITHM USED:

#### 1. Random Forest Model

Random forest is a [supervised learning algorithm](#). The "forest" it builds, is an ensemble of

decision trees, usually trained with the "bagging" method. The general idea of the bagging method

is that a combination of learning models increases the overall result.

**Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.**

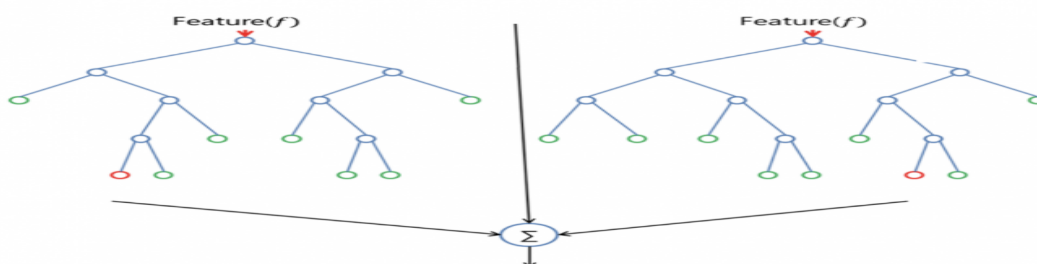
One big advantage of random forest is that it can be used for both classification and regression

problems, which form the majority of current machine learning systems. Let's look at random forest

in classification, since classification is sometimes considered the building block of machine

learning. Below you can see how a random forest would look like with two trees:

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier.



Fortunately, there's no need to combine a decision tree with a bagging classifier.

### Figure: 5.1.1

Random forest adds additional randomness to the model, while growing the trees. Instead of

searching for the most important feature while splitting a node, it searches for the best feature

among a random subset of features. This results in a wide diversity that generally results in a better

model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the

algorithm for splitting a node. You can even make trees more random by additionally using random

thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

### Difference between Decision Trees and Random Forests

While random forest is a collection of decision trees, there are some differences.

If you input a training dataset with features and labels into a decision tree, it will formulate some

set of rules, which will be used to make the predictions.

For example, to predict whether a person will click on an online advertisement, you might collect

the ads the person clicked on in the past and some features that describe his/her decision. If you put

the features and labels into a decision tree, it will generate some rules that help predict whether the

advertisement will be clicked or not. In comparison, the random forest algorithm randomly selects

observations and features to build several decision trees and then averages the results.

Another difference is "deep" decision trees might suffer from overfitting. Most of the time, random

forest prevents this by creating random subsets of the features and building smaller trees using

those subsets. Afterwards, it combines the subtrees. It's important to note this doesn't work every

time and it also makes the computation slower, depending on how many trees the random forest builds.

## 2. Decision Tree Algorithm

---

Decision Tree algorithm belongs to the family of supervised learning algorithms.

Unlike other supervised learning algorithms, the decision tree algorithm can be used

for solving **regression and classification problems** too.

The goal of using a Decision Tree is to create a training model that can use to predict

the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the **root** of the

tree. We compare the values of the root attribute with the record's attribute. On the

basis of comparison, we follow the branch corresponding to that value and jump to the next node.

## Types of Decision Trees

Types of decision trees are based on the type of target variable we have. It can be of two types:

1. **Categorical Variable Decision Tree:** Decision Tree which has a categorical target variable then it called a **Categorical variable decision tree**.
2. **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called **Continuous Variable Decision Tree**.

## Important Terminology related to Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.

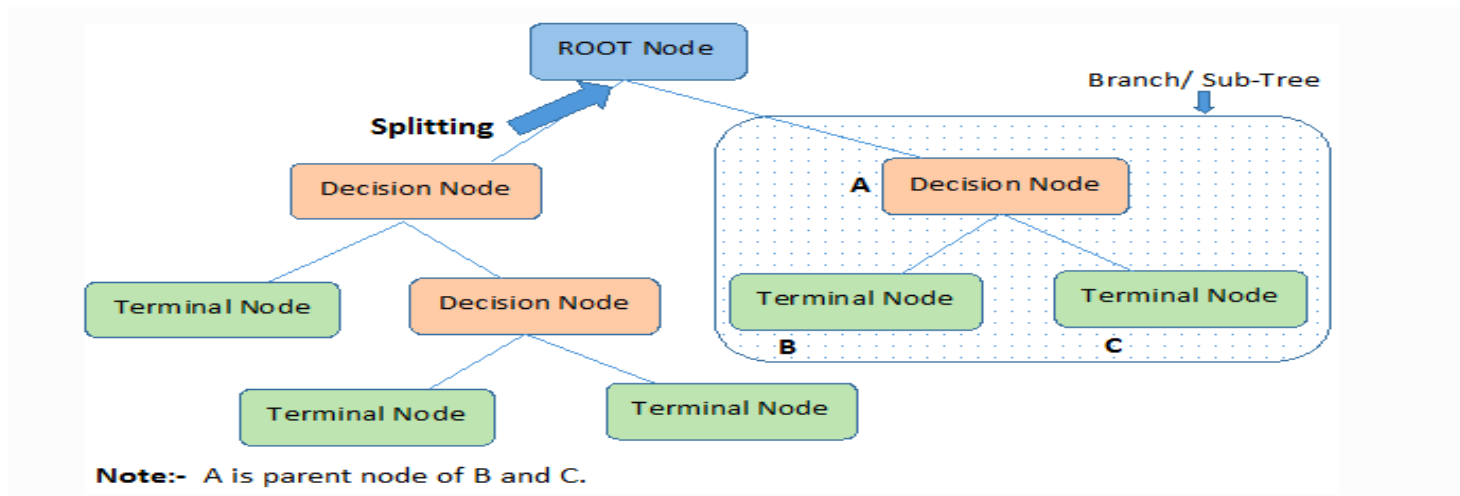


FIGURE : 5.1.2

2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a

parent node of sub-nodes whereas sub-nodes are the child of a parent node.

Decision trees classify the examples by sorting them down the tree from the root to

some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending

from the node corresponds to the possible answers to the test case. This process is

recursive in nature and is repeated for every subtree rooted at the new node.

## **How do Decision Trees work?**

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are

different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The

creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say

that the purity of the node increases with respect to the target variable. The decision tree splits the

nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

The algorithm selection is also based on the type of target variables. Let us look at some

algorithms used in Decision Trees:

**ID3** → (extension of D3)

**C4.5** → (successor of ID3)

**CART** → (Classification And Regression Tree)

**CHAID** → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)

**MARS** → (multivariate adaptive regression splines)

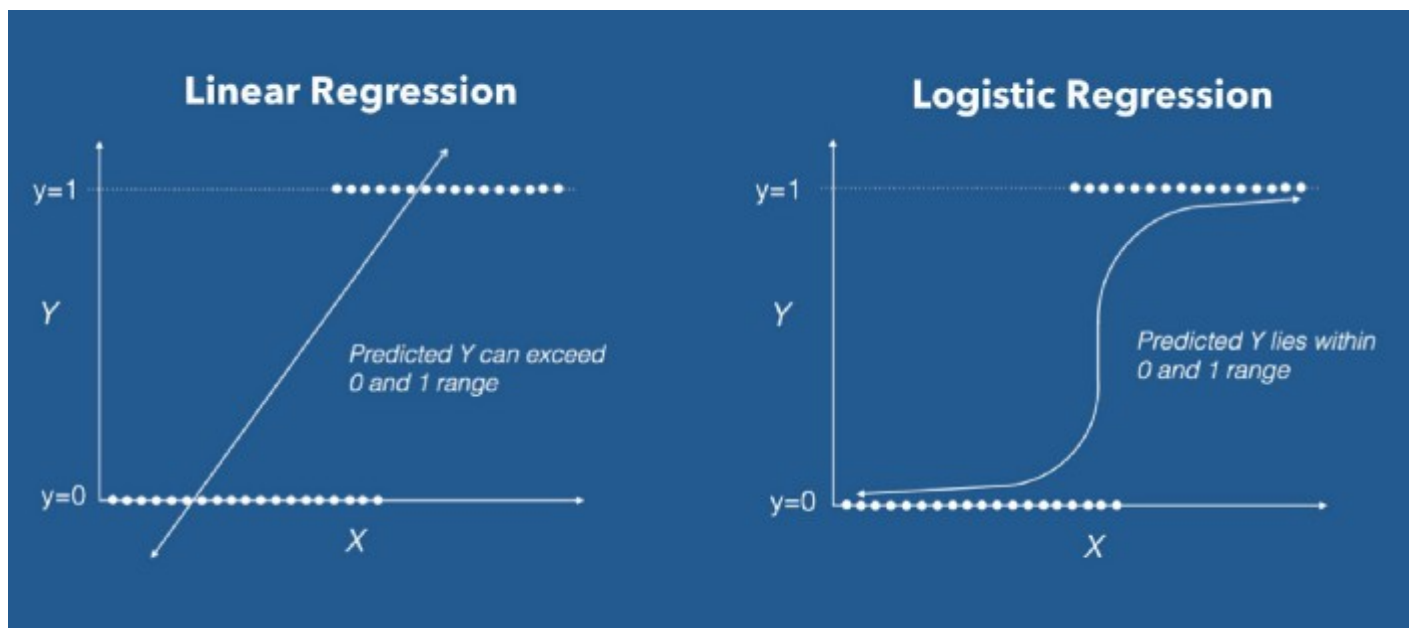
The ID3 algorithm builds decision trees using a top-down [greedy search](#) approach through the

space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always

makes the choice that seems to be the best at that moment.

### 3.Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



**Figure: 5.1.3** Linear Regression VS Logistic Regression Graph



We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1.

Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_{\theta}(x) \leq 1$$

Logistic regression hypothesis expectation

What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

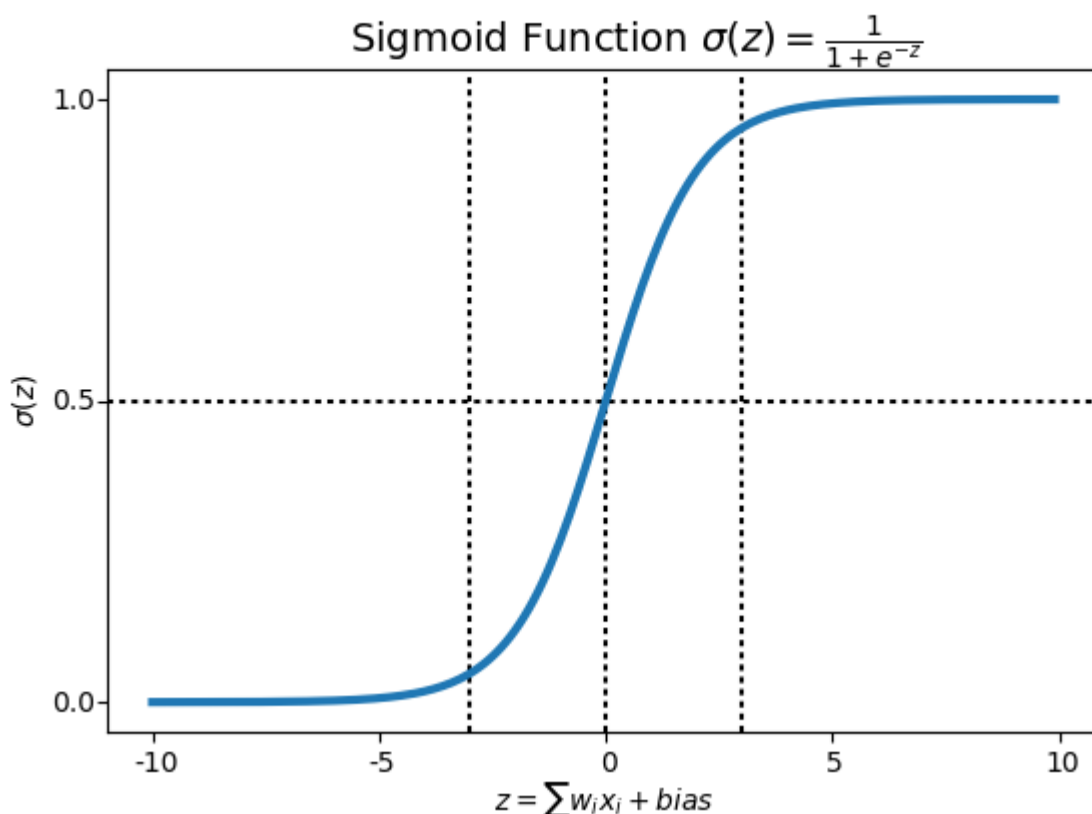


Figure: Sigmoid Function Graph

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

FIGURE: 5.1.4

Formula of a sigmoid function

### Hypothesis Representation

When using *linear regression* we used a formula of the

hypothesis i.e.

$$h\theta(x) = \beta_0 + \beta_1 X$$

For logistic regression we are going to modify it a little bit i.e.

$$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$$

We have expected that our hypothesis will give values between 0 and 1.

$$Z = \beta_0 + \beta_1 X$$

$$h\theta(x) = \text{sigmoid}(Z)$$

$$\text{i.e. } h\theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The Hypothesis of logistic regression  
Decision Boundary

We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

For Example, We have 2 classes, let's take them like cats and dogs(1 – dog , 0 – cats). We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.

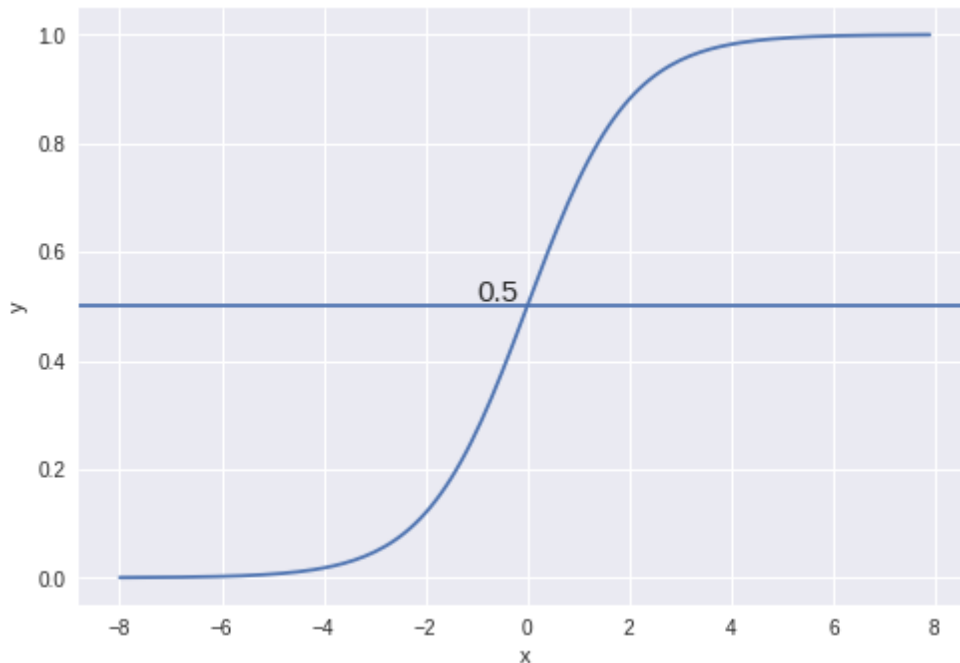


FIGURE 5.1.5

As shown in the above graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1(DOG). If our prediction returned a value of 0.2 then we would classify the observation as Class 2(CAT).

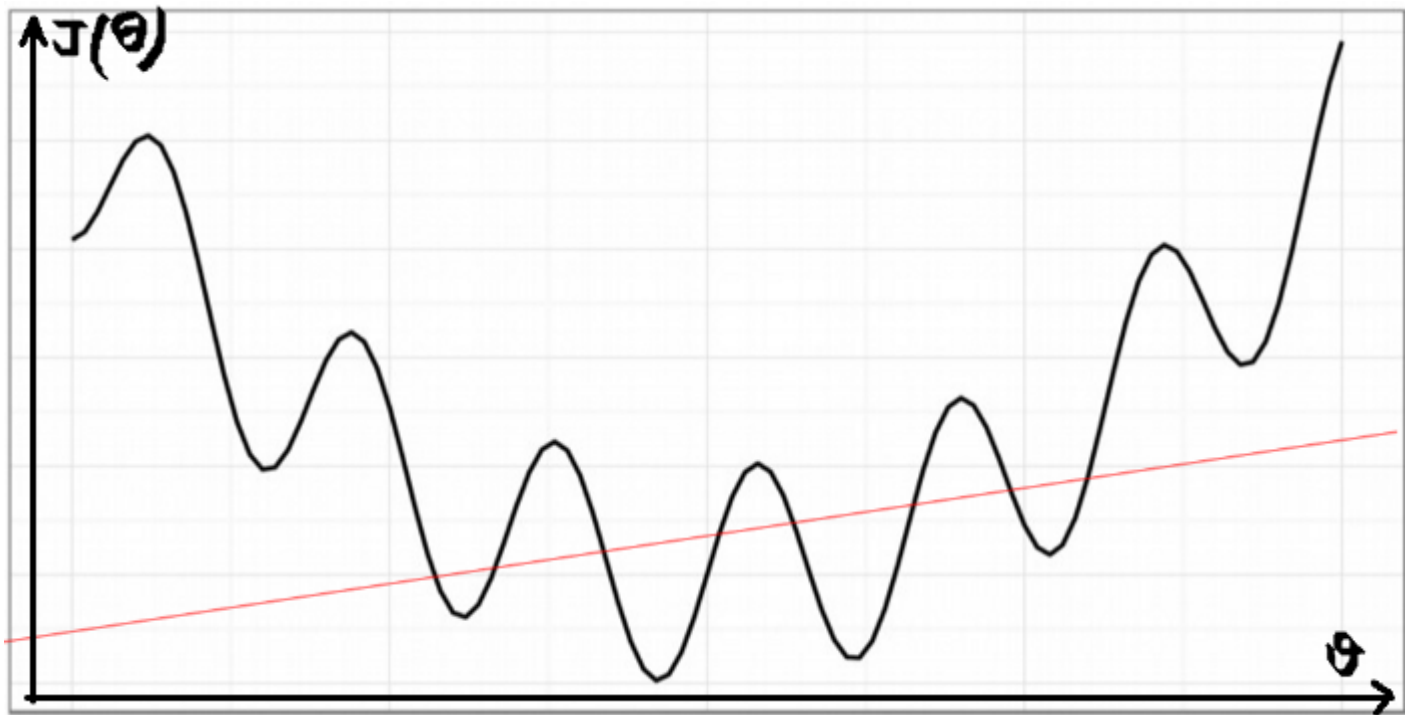
### Cost Function

We learnt about the cost function  $J(\theta)$  in the *Linear regression*, the cost function represents optimization objective i.e. we create a cost function and minimize it so that we can develop an accurate model with minimum error.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2.$$

The Cost function of Linear regression

If we try to use the cost function of the linear regression in 'Logistic Regression' then it would be of no use as it would end up being a **non-convex** function with many local minimums, in which it would be very **difficult** to **minimize the cost value** and find the global minimum.



**Figure: 5.1.6** Non-convex function

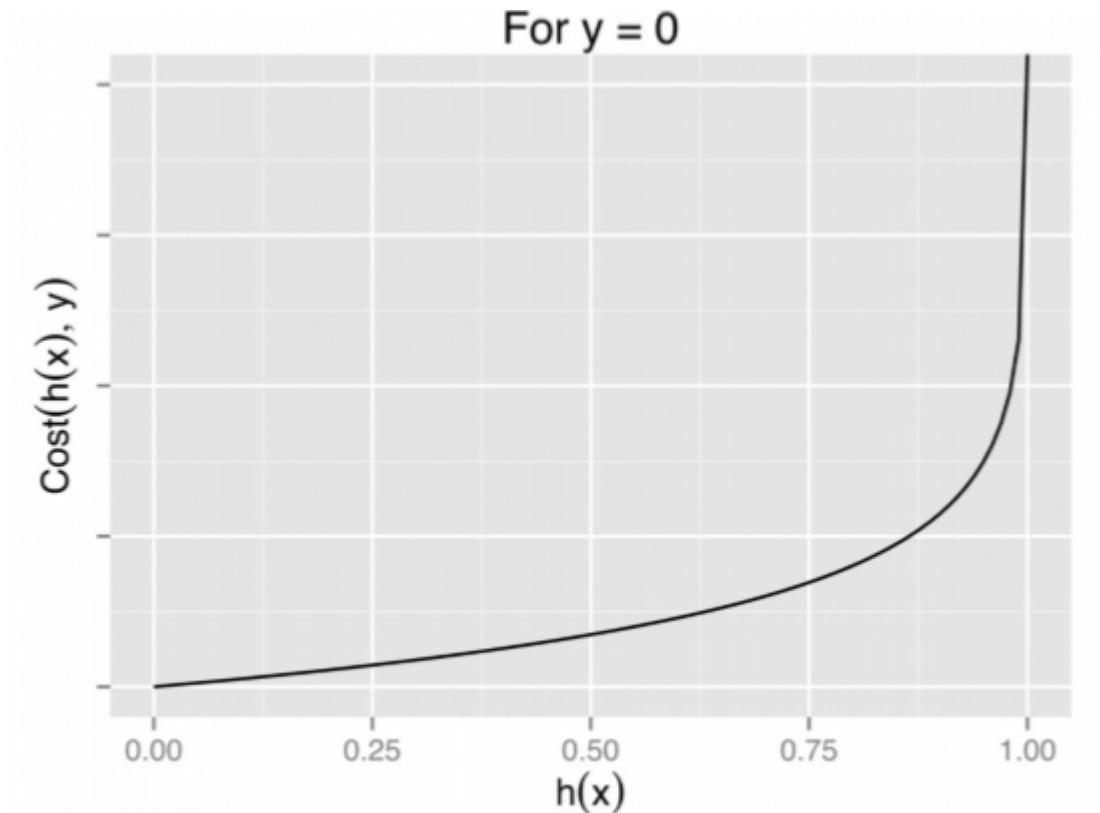
For logistic regression, the Cost function is defined as:

$-\log(h\theta(x))$  if  $y = 1$

$-\log(1-h\theta(x))$  if  $y = 0$

$$Cost(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

Cost function of Logistic Regression



**FIGURE 5.1.7** Graph of logistic regression

The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m} \sum \left[ y^{(i)} \log(h\theta(x(i))) + (1 - y^{(i)}) \log(1 - h\theta(x(i))) \right]$$

Above functions compressed into one cost function

### Gradient Descent

Now the question arises, how do we reduce the cost value. Well, this can be done by using **Gradient Descent**. The main goal of Gradient descent is to **minimize the cost value**. i.e.  $\min J(\theta)$ .

Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Objective: To minimize the cost function we have to run the gradient descent function on each parameter

Want  $\min_{\theta} J(\theta)$ :

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

} (simultaneously update all  $\theta_j$ )

### Gradient Descent Simplified

Gradient descent has an analogy in which we have to imagine ourselves at the top of a mountain valley and left stranded and blindfolded, our objective is to reach the bottom of the hill. Feeling the slope of the terrain around you is what everyone would do. Well, this action is analogous to calculating the gradient descent, and taking a step is analogous to one iteration of the update to the parameters.

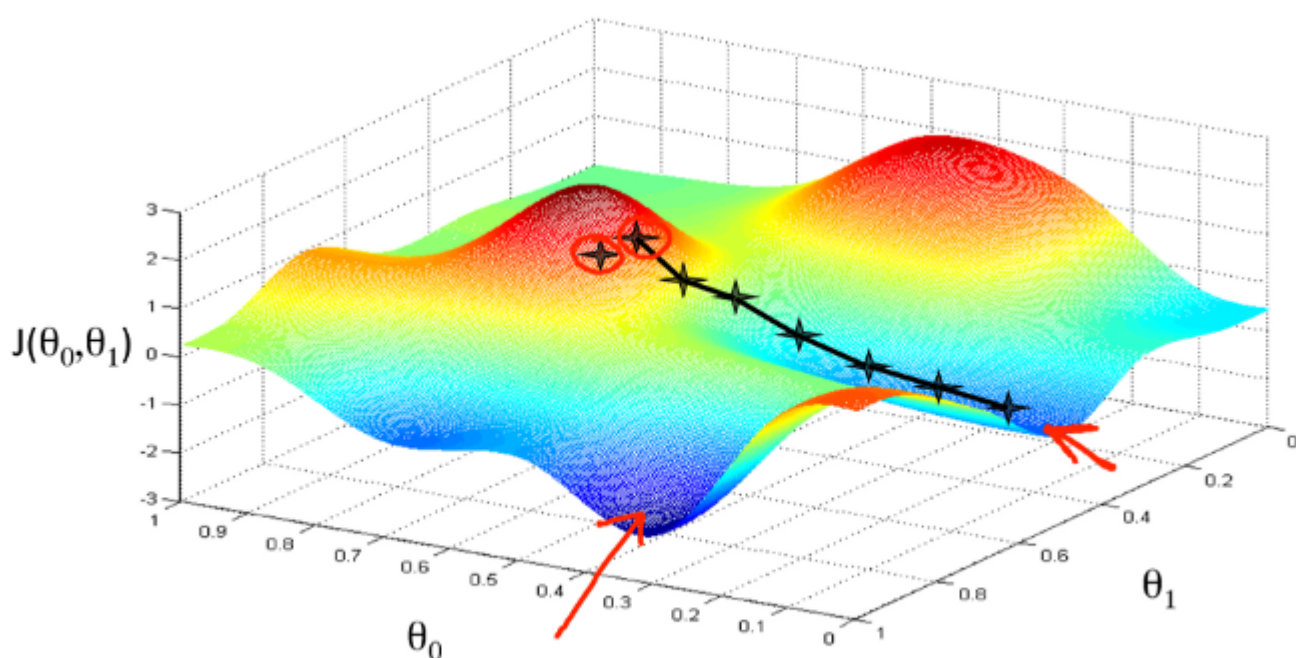


FIGURE :5.1.8

## EVALUATION:

### 1.Accuracy score

`sklearn.metrics.accuracy_score(y_t`

*rue, y\_pred, \*, normalize=True,*

*sample\_weight=None)*

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in `y_true`.

<b>Parameters:</b>	<p><b>y_true : 1d array-like, or label indicator array / sparse matrix</b> Ground truth (correct) labels.</p> <p><b>y_pred : 1d array-like, or label indicator array / sparse matrix</b> Predicted labels, as returned by a classifier.</p> <p><b>normalize : bool, default=True</b> If <code>False</code>, return the number of correctly classified samples. Otherwise, return the fraction of correctly classified samples.</p> <p><b>sample_weight : array-like of shape (n_samples,), default=None</b> Sample weights.</p>
<b>Returns:</b>	<p><b>score : float</b> If <code>normalize == True</code>, return the fraction of correctly classified samples (float), else returns the number of correctly classified samples (int).</p> <p>The best performance is 1 with <code>normalize == True</code> and the number of samples with <code>normalize == False</code>.</p>

## 2:F1 SCORE:

`.f1_score` `sklearn.metrics.f1_score(y_true, y_pred, *, labels=None, pos_label=1, average='binary', sample_weight=None, zero_division='warn')`



<b>Parameters:</b>	<div data-bbox="236 210 1045 297"><b>y_true : 1d array-like, or label indicator array / sparse matrix</b> Ground truth (correct) target values.</div> <div data-bbox="236 322 1054 409"><b>y_pred : 1d array-like, or label indicator array / sparse matrix</b> Estimated targets as returned by a classifier.</div> <div data-bbox="236 434 1596 757"><b>labels : array-like, default=None</b> The set of labels to include when <code>average != 'binary'</code>, and their order if <code>average is None</code>. Labels present in the data can be excluded, for example to calculate a multiclass average ignoring a majority negative class, while labels not present in the data will result in 0 components in a macro average. For multilabel targets, labels are column indices. By default, all labels in <code>y_true</code> and <code>y_pred</code> are used in sorted order.<div data-bbox="284 689 1244 757">Changed in version 0.17: Parameter <code>labels</code> improved for multiclass problem.</div></div> <div data-bbox="236 781 1596 904"><b>pos_label : str or int, default=1</b> The class to report if <code>average='binary'</code> and the data is binary. If the data are multiclass or multilabel, this will be ignored; setting <code>labels=[pos_label]</code> and <code>average != 'binary'</code> will report scores for that label only.</div> <div data-bbox="236 929 1596 1854"><b>average : {'micro', 'macro', 'samples', 'weighted', 'binary'} or None, default='binary'</b> This parameter is required for multiclass/multilabel targets. If <code>None</code>, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:<div data-bbox="271 1086 1596 1220"><b>'binary':</b> Only report results for the class specified by <code>pos_label</code>. This is applicable only if targets (<code>y_{true,pred}</code>) are binary.</div><div data-bbox="271 1245 1481 1332"><b>'micro':</b> Calculate metrics globally by counting the total true positives, false negatives and false positives.</div><div data-bbox="271 1357 1576 1491"><b>'macro':</b> Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.</div><div data-bbox="271 1516 1596 1695"><b>'weighted':</b> Calculate metrics for each label, and find their average weighted by support (the number of true instances of each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.</div><div data-bbox="271 1720 1596 1854"><b>'samples':</b> Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from <code>accuracy_score</code>).</div></div> <div data-bbox="236 1879 1077 1966"><b>sample_weight : array-like of shape (n_samples,), default=None</b> Sample weights.</div> <div data-bbox="236 1991 1596 2148"><b>zero_division : "warn", 0 or 1, default="warn"</b> Sets the value to return when there is a zero division, i.e. when all predictions and labels are negative. If set to <code>"warn"</code>, this acts as 0, but warnings are also raised.</div>
<b>Returns:</b>	<div data-bbox="236 2150 1596 2148"><b>f1_score : float or array of float, shape = [n_unique_labels]</b> F1 score of the positive class in binary classification or weighted average of the F1 scores of each class for the multiclass case.</div>

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the average parameter.

Notes: When true positive + false positive == 0, precision is undefined. When true positive + false negative == 0, recall is undefined. In such cases, by default the metric will be set to 0, as will f-score, and Undefined Metric Warning will be raised. This behaviour can be modified with `zero_division`.

### 3. Confusion Matrix

```
sklearn.metrics.confusion_matrix(y_true, y_pred, *, labels=None,  
                                  sample_weight=None, normalize=None)
```

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix  $C$  is such that  $C_{i,j}$  is equal to the number of observations known to be in group  $i$  and predicted to be in group  $j$ .

Thus in binary classification, the count of true negatives is  $C_{0,0}$ , false negatives is  $C_{1,0}$ , true positives is  $C_{1,1}$  and false positives is  $C_{0,1}$ .

<b>Parameters:</b>	<p><b>y_true : array-like of shape (n_samples,)</b> Ground truth (correct) target values.</p> <p><b>y_pred : array-like of shape (n_samples,)</b> Estimated targets as returned by a classifier.</p> <p><b>labels : array-like of shape (n_classes), default=None</b> List of labels to index the matrix. This may be used to reorder or select a subset of labels. If <code>None</code> is given, labels that appear at least once in <code>y_true</code> or <code>y_pred</code> are used in sorted order.</p> <p><b>sample_weight : array-like of shape (n_samples,), default=None</b> Sample weights.  <i>New in version 0.18.</i></p> <p><b>normalize : {'true', 'pred', 'all'}, default=None</b> Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If not specified, the confusion matrix will not be normalized.</p>
<b>Returns:</b>	<p><b>C : ndarray of shape (n_classes, n_classes)</b> Confusion matrix whose i-th row and j-th column entry indicates the number of samples with true label i and predicted label being j-th class.</p>

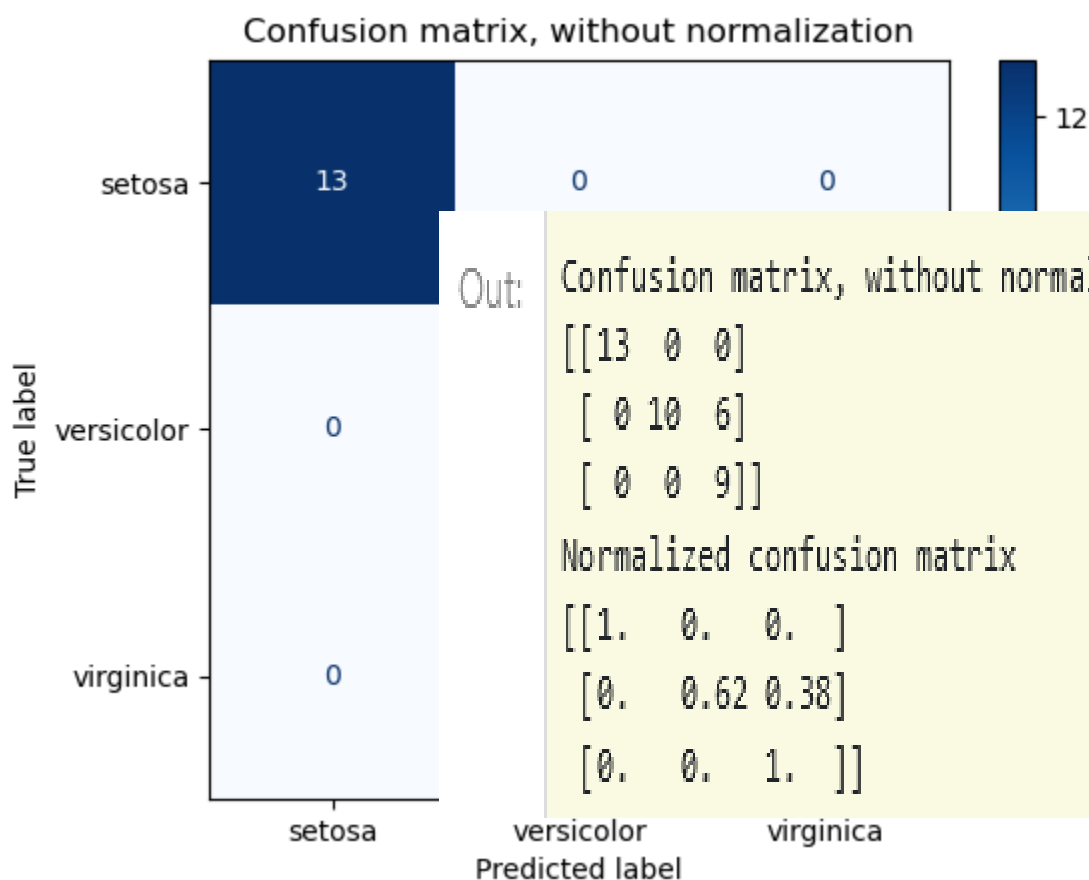


FIGURE:5.1.9

## 5.2 MODULE DESCRIPTION

### PROJECT MODULES:

- Data Processing
- EDA(Exploration Data Analysis)
- Feature Selection and DataSplit

- **Modelling (Building the classification models)**
- **Evaluation of the Models**

## 5.2.1 :DATA PROCESSING

```
#data processing
cases = len(df)
nonfraud_count = len(df[df.Class == 0])
fraud_count = len(df[df.Class == 1])
fraud_percentage = round(fraud_count/nonfraud_count*100, 2)

print(cl('CASE COUNT', attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
print(cl('Total number of cases are {}'.format(cases), attrs = ['bold']))
print(cl('Number of Non-fraud cases are {}'.format(nonfraud_count), attrs = ['bold']))
print(cl('Number of fraud cases are {}'.format(fraud_count), attrs = ['bold']))
print(cl('Percentage of fraud cases is {}'.format(fraud_percentage), attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
```

OUTPUT:

```
CASE COUNT
```

```
-----  
Total number of cases are 284807  
Number of Non-fraud cases are 284315  
Number of fraud cases are 492  
Percentage of fraud cases is 0.17  
-----
```

5.2.2:EDA(EVALUATION OF

DATA ANALYSIS)

```

#EDA

nonfraud_cases = df[df.Class == 0]
fraud_cases = df[df.Class == 1]

print(c1('CASE AMOUNT STATISTICS', attrs = ['bold']))
print(c1('-----', attrs = ['bold']))
print(c1('NON-FRAUD CASE AMOUNT STATS', attrs = ['bold']))
print(nonfraud_cases.Amount.describe())
print(c1('-----', attrs = ['bold']))
print(c1('FRAUD CASE AMOUNT STATS', attrs = ['bold']))
print(fraud_cases.Amount.describe())
print(c1('-----', attrs = ['bold']))

sc = StandardScaler()
amount = df['Amount'].values

df['Amount'] = sc.fit_transform(amount.reshape(-1, 1))

print(c1(df['Amount'].head(10), attrs = ['bold']))

```

OUTPUT:



```

CASE AMOUNT STATISTICS
-----
NON-FRAUD CASE AMOUNT STATS
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
-----
FRAUD CASE AMOUNT STATS
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
-----
0      0.244964
1     -0.342475
2      1.160686
3      0.140534
4     -0.073403
5     -0.338556
6     -0.333279
7     -0.190107
8      0.019392
9     -0.338516
Name: Amount, dtype: float64

```

### 5.2.3: Feature selection and

#### Data split:

```

#data split

X = df.drop('Class', axis = 1).values
y = df['Class'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

print(cl('X_train samples : ', attrs = ['bold']), X_train[:1])
print(cl('X_test samples : ', attrs = ['bold']), X_test[0:1])
print(cl('y_train samples : ', attrs = ['bold']), y_train[0:20])
print(cl('y_test samples : ', attrs = ['bold']), y_test[0:20])

```

#### OUTPUT:

```

X_train samples : [[-1.11504743  1.03558276  0.80071244 -1.06039825  0.03262117  0.85342216
-0.61424348 -3.23116112  1.53994798 -0.81690879 -1.30559201  0.1081772
-0.85960958 -0.07193421  0.90665563 -1.72092961  0.79785322 -0.0067594
1.95677806 -0.64489556  3.02038533 -0.53961798  0.03315649 -0.77494577
0.10586781 -0.43085348  0.22973694 -0.0705913  -0.30145418]]
X_test samples : [[-0.32333357  1.05745525 -0.04834115 -0.60720431  1.25982115 -0.09176072
1.1591015  -0.12433461 -0.17463954 -1.64440065 -1.11886302  0.20264731
1.14596495 -1.80235956 -0.24717793 -0.06094535  0.84660574  0.37945439
0.84726224  0.18640942 -0.20709827 -0.43389027 -0.26161328 -0.04665061
0.2115123  0.00829721  0.10849443  0.16113917 -0.19330595]]
y_train samples : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
y_test samples : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

```

## 5.2.4:Modelling:

```

#modelling

#Decision Tree

tree_model = DecisionTreeClassifier(max_depth = 4, criterion = 'entropy')
tree_model.fit(X_train, y_train)
tree_yhat = tree_model.predict(X_test)

#Logistic Regression

lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_yhat = lr.predict(X_test)

#Random Forest Tree

rf = RandomForestClassifier(max_depth = 4)
rf.fit(X_train, y_train)
rf_yhat = rf.predict(X_test)

```

## 5.2.5: EVALUATION OF THE MODELS:

## 1.ACCURACY SCORE:

# 1. Accuracy score

```
print(cl('ACCURACY SCORE', attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
print(cl('Accuracy score of the Decision Tree model is {}'.format(accuracy_score(y_test, tree_yhat)), attrs = ['bold']))
print(cl('-----', attrs = ['bold']))

print(cl('Accuracy score of the Logistic Regression model is {}'.format(accuracy_score(y_test, lr_yhat)), attrs = ['bold', color = 'red']))
print(cl('-----', attrs = ['bold']))

print(cl('Accuracy score of the Random Forest Tree model is {}'.format(accuracy_score(y_test, rf_yhat)), attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
```

OUTPUT:

```
ACCURACY SCORE
-----
Accuracy score of the Decision Tree model is 0.9993679997191109
convert/html/Desktop/project/output.ipynb?download=false
```

output

```
-----
Accuracy score of the Logistic Regression model is 0.9991924440855307
-----
Accuracy score of the Random Forest Tree model is 0.9992977774656788
-----
```

## 2:F1 SCORE:

# 2. F1 score

```
print(cl('F1 SCORE', attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
print(cl('F1 score of the Decision Tree model is {}'.format(f1_score(y_test, tree_yhat)), attrs = ['bold']))
print(cl('-----', attrs = ['bold']))

print(cl('F1 score of the Logistic Regression model is {}'.format(f1_score(y_test, lr_yhat)), attrs = ['bold', color = 'red']))
print(cl('-----', attrs = ['bold']))

print(cl('F1 score of the Random Forest Tree model is {}'.format(f1_score(y_test, rf_yhat)), attrs = ['bold']))
print(cl('-----', attrs = ['bold']))
|
```

OUTPUT:

## F1 SCORE


-----  
F1 score of the Decision Tree model is 0.8105263157894738  
-----

F1 score of the Logistic Regression model is 0.7356321839080459  
-----

F1 score of the Random Forest Tree model is 0.7752808988764045  
-----

## 3:CONFUSION MATRIX:

```
# 3. Confusion Matrix

# defining the plot function  Rectangular Snip

def plot_confusion_matrix(cm, classes, title, normalize = False, cmap = plt.cm.Blues):
    title = 'Confusion Matrix of {}'.format(title)
    if normalize:
        cm = cm.astype(float) / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment = 'center',
                 color = 'white' if cm[i, j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
# Compute confusion matrix for the models
```

```
tree_matrix = confusion_matrix(y_test, tree_yhat, labels = [0, 1]) # Decision Tree
```

```
lr_matrix = confusion_matrix(y_test, lr_yhat, labels = [0, 1]) # Logistic Regression
```

```
rf_matrix = confusion_matrix(y_test, rf_yhat, labels = [0, 1]) # Random Forest Tree
```

convert/html/Desktop/project/output.ipynb?download=false

output

```
# Plot the confusion matrix
```

```
plt.rcParams['figure.figsize'] = (6, 6)
```

## **CHAPTER 6**

### **SYSTEM TESTING**

#### **6.1 INTRODUCTION**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as- yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error.

A good test case design is one that as a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet

undiscovered error

## 6.2 GOALS OF TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behaviour of the source code by executing the program on the test data.

## 6.3 TESTING METHODOLOGIES

### 6.3.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.



### 6.3.2 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested.

This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications.

The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

### **6.3.3 WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving testcases
- Graph matricesControl

### **6.3.4 BLACK BOX TESTING**

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

The steps involved in black box test case design are:

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

### **6.3.5 PROGRAM TESTING**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer.

A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

### **6.3.6 SECURITY TESTING**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetratesystem.

### **6.3.7 USER ACCEPTANCE TESTING**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screendesign.
- Output screendesign.

### **6.3.8 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

## CHAPTER 7

### RESULTS

Let's take the confusion matrix of the module. Look at the first row. The first row is for transactions whose actual fraud value in the test set is 0. As you can calculate, the fraud value of 56861 of them is 0. And out of these 56861 non-fraud transactions, the classifier correctly predicted 56854 of them as 0 and 7 of them as 1. It means, for 56854 non-fraud transactions, the actual churn value was 0 in the test set, and the classifier also correctly predicted those as 0. We can say that our model has classified the non-fraud transactions pretty well.

Let's look at the second row. It looks like there were 101 transactions whose fraud value was 1. The classifier correctly predicted 79 of them as 1, and 22 of them wrongly as 0. The wrongly predicted values can be considered as the error of the model.

So we can conclude that the most appropriate model which can be used for our case is the model which can be neglected is the Logistic regression model.

## CHAPTER 8

### CONCLUSION AND FUTURE WORKS

After that, we have evaluated each of the models using the evaluation metrics and chose which model is most suitable for the given case. And feel happy about this as we have come across and successfully finished one of the most famous financial data science projects.

In this project, many more models to explore. Also, we have built the models feasibly in python but, there are more and more math and statistics behind each of the models. With that, thank you for reading this article and if you forgot to follow any of the coding parts, don't worry, I have provided the full code at the end of this article.

## References:

<https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>  
<https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>  
<https://builtin.com/data-science/random-forest-algorithm>  
<https://www.geeksforgeeks.org/understanding-logistic-regression/>  
<https://medium.com/codex/credit-card-fraud-detection-with-machine-learning-in-python-ac7281991d87>  
[Credit Card Fraud Detection Using Machine Learning – IJERT](#)

## APPENDIX 1:

### CODING:

```
#importing packages
import pandas as pd # data processing
import numpy as np # working with arrays
import matplotlib.pyplot as plt # visualization
from termcolor import colored as cl # text customization
import itertools # advanced tools

from sklearn.preprocessing import StandardScaler # data normalization
from sklearn.model_selection import train_test_split # data split
from sklearn.tree import DecisionTreeClassifier # Decision tree algorithm
from sklearn.linear_model import LogisticRegression # Logistic regression algorithm
from sklearn.ensemble import RandomForestClassifier # Random forest tree algorithm

from sklearn.metrics import confusion_matrix # evaluation metric
from sklearn.metrics import accuracy_score # evaluation metric
from sklearn.metrics import f1_score # evaluation metric
```

```
#importing dataset
```

```
df = pd.read_csv('creditcard.csv')  
df.drop('Time', axis = 1, inplace = True)
```

```
print(df.head())
```

```
#data processing
```

```
cases = len(df)  
nonfraud_count = len(df[df.Class == 0])  
fraud_count = len(df[df.Class == 1])  
fraud_percentage = round(fraud_count/nonfraud_count*100, 2)
```

```
print(cl('CASE COUNT', attrs = ['bold']))  
print(cl('—————', attrs = ['bold']))  
print(cl('Total number of cases are {}'.format(cases), attrs = ['bold']))  
print(cl('Number of Non-fraud cases are {}'.format(nonfraud_count), attrs =  
['bold']))  
print(cl('Number of fraud cases are {}'.format(fraud_count), attrs = ['bold']))  
print(cl('Percentage of fraud cases is {}'.format(fraud_percentage), attrs =  
['bold']))  
print(cl('—————', attrs = ['bold']))
```

```
#EDA
```

```
nonfraud_cases = df[df.Class == 0]  
fraud_cases = df[df.Class == 1]
```

```
print(cl('CASE AMOUNT STATISTICS', attrs = ['bold']))  
print(cl('—————', attrs = ['bold']))  
print(cl('NON-FRAUD CASE AMOUNT STATS', attrs = ['bold']))
```



```

print(nonfraud_cases.Amount.describe())
print(cl('—————', attrs = ['bold']))
print(cl('FRAUD CASE AMOUNT STATS', attrs = ['bold']))
print(fraud_cases.Amount.describe())
print(cl('—————', attrs = ['bold']))

```

```

sc = StandardScaler()
amount = df['Amount'].values

```

```

df['Amount'] = sc.fit_transform(amount.reshape(-1, 1))

```

```

print(cl(df['Amount'].head(10), attrs = ['bold']))

```

**#data split**

```

X = df.drop('Class', axis = 1).values
y = df['Class'].values

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

```

```

print(cl('X_train samples : ', attrs = ['bold']), X_train[:1])
print(cl('X_test samples : ', attrs = ['bold']), X_test[0:1])
print(cl('y_train samples : ', attrs = ['bold']), y_train[0:20])
print(cl('y_test samples : ', attrs = ['bold']), y_test[0:20])

```

**#modelling**

**#Decision Tree**

```
tree_model = DecisionTreeClassifier(max_depth = 4, criterion = 'entropy')
tree_model.fit(X_train, y_train)
tree_yhat = tree_model.predict(X_test)
```

**#Logistic Regression**

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_yhat = lr.predict(X_test)
```

**#Random Forest Tree**

```
rf = RandomForestClassifier(max_depth = 4)
rf.fit(X_train, y_train)
rf_yhat = rf.predict(X_test)
```

**#evaluation**

**# 1. Accuracy score**

```
print(cl('ACCURACY SCORE', attrs = ['bold']))
print(cl('—————', attrs = ['bold']))
print(cl('Accuracy score of the Decision Tree model is
{}'.format(accuracy_score(y_test, tree_yhat)), attrs = ['bold']))
print(cl('—————', attrs = ['bold']))
```

```
print(cl('Accuracy score of the Logistic Regression model is
{}'.format(accuracy_score(y_test, lr_yhat)), attrs = ['bold', color = 'red']))
print(cl('—————', attrs = ['bold']))
```

```
print(cl('Accuracy score of the Random Forest Tree model is
{}'.format(accuracy_score(y_test, rf_yhat)), attrs = ['bold']))
print(cl('—————', attrs = ['bold']))
```

## # 2. F1 score

```
print(cl('F1 SCORE', attrs = ['bold']))
print(cl('—————', attrs = ['bold']))
print(cl('F1 score of the Decision Tree model is {}'.format(f1_score(y_test,
tree_yhat)), attrs = ['bold']))
print(cl('—————', attrs = ['bold']))

print(cl('F1 score of the Logistic Regression model is
{}'.format(f1_score(y_test, lr_yhat)), attrs = ['bold', color = 'red']))
print(cl('—————', attrs = ['bold']))

print(cl('F1 score of the Random Forest Tree model is
{}'.format(f1_score(y_test, rf_yhat)), attrs = ['bold']))
print(cl('—————', attrs = ['bold']))
```

## # 3. Confusion Matrix

### # defining the plot function

```
def plot_confusion_matrix(cm, classes, title, normalize = False, cmap =
plt.cm.Blues):
    title = 'Confusion Matrix of {}'.format(title)
    if normalize:
        cm = cm.astype(float) / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
```

```

tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation = 45)
plt.yticks(tick_marks, classes)

fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment = 'center',
             color = 'white' if cm[i, j] > thresh else 'black')

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

```

**# Compute confusion matrix for the models**

```

tree_matrix = confusion_matrix(y_test, tree_yhat, labels = [0, 1]) # Decision
Tree

```

```

lr_matrix = confusion_matrix(y_test, lr_yhat, labels = [0, 1]) # Logistic
Regression

```

```

rf_matrix = confusion_matrix(y_test, rf_yhat, labels = [0, 1]) # Random Forest
Tree

```

**# Plot the confusion matrix**

```

plt.rcParams['figure.figsize'] = (6, 6)

```

**# 1. Decision tree**

```

tree_cm_plot = plot_confusion_matrix(tree_matrix,

```

```
        classes = ['Non-Default(0)','Default(1)'],  
        normalize = False, title = 'Decision Tree')  
plt.savefig('tree_cm_plot.png')  
plt.show()
```

## # 2. Logistic regression

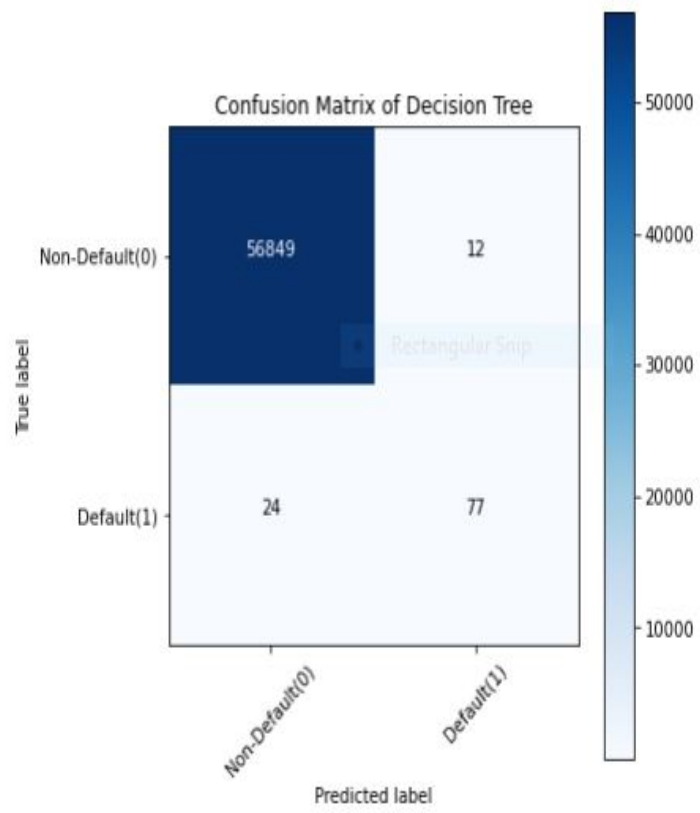
```
lr_cm_plot = plot_confusion_matrix(lr_matrix,  
        classes = ['Non-Default(0)','Default(1)'],  
        normalize = False, title = 'Logistic Regression')  
plt.savefig('lr_cm_plot.png')  
plt.show()
```

## # 3. Random forest tree

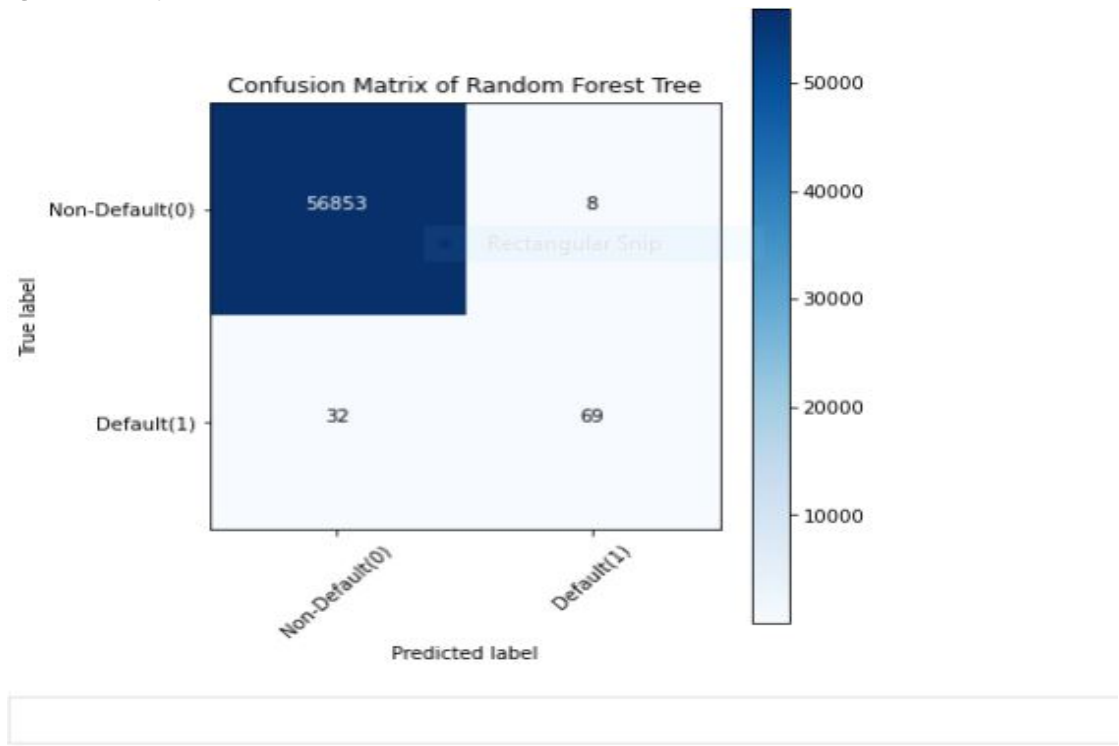
```
rf_cm_plot = plot_confusion_matrix(rf_matrix,  
        classes = ['Non-Default(0)','Default(1)'],  
        normalize = False, title = 'Random Forest Tree')  
plt.savefig('rf_cm_plot.png')  
plt.show()
```

APPENDIX 2:

DECISION TREE:



RANDOM FOREST TREE:



LOGISTIC REGRESSION:

