## ABSTRACT

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

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

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

**CONTENTS**

1. INTRODUCTION 8
   1. Existing System 8
      1. Disadvantages of Existing System 9
   2. Problem Statement 9
   3. Proposed System 10
      1. Advantages of Proposed System 10
2. ANALYSIS 12
   1. Functional Requirements 12
   2. Non-functional Requirements 13
3. SYSTEM DESIGN 15
   1. System Design 15
   2. Flow chart diagram 15
   3. Use case diagrams 17
      1. Use case diagram admin 17
   4. Data flow diagram 18
      1. Level 0 data flow diagram 18
      2. Level 1 data flow diagram 19
      3. Level 2 data flow diagram 20
4. SYSTEM IMPLEMENTATION 21
   1. Python Introduction 21
   2. Algorithms Used 27
      1. Decision Tree 27
      2. Random Forest 30

SDLC.........................................................................................................................................36

1. References 63

## LIST OF FIGURES

#### 2.1. Some Non-functional requirements chart 13

#### System Architecture 15

#### Flow Chart Diagram 16

#### Use Case Diagram 17

#### Level 0 Data Flow Diagram 18

#### Level 1 Data Flow Diagram 19

#### Level 2 Data Flow Diagram 20

#### Decision Tree Algorithm 28

#### Binary Decision Tree Diagram 29

#### Random Forest Algorithm 31

# INTRODUCTION

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

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

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

## EXISTING SYSTEM

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

## Disadvantages of Existing System

#### Imbalanced data:

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

#### Overlapping data:

Many transactions may be considered fraudulent, while actually they are normal (false positive) and reversely, a fraudulent transaction may also seem to be legitimate (false negative). Hence obtaining low rate of false positive and false negative is a key challenge of fraud detection systems.

#### Lack of adaptability:

Classification algorithms are usually faced with the problem of detecting new types of normal or fraudulent patterns. The supervised and unsupervised fraud detection systems are inefficient in detecting new patterns of normal and fraud behaviors, respectively.

#### Fraud detection cost:

The system should take into account both the cost of fraudulent behavior that is detected and the cost of preventing it.

#### Lack of standard metrics:

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

## Problem Statement

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

## Proposed system

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

## Advantages of Proposed System

* + - 1. More accurate result.
      2. Able to detect different fraudulent behavior.
      3. Cost and Time efficient.

# ANALYSIS

1. A Novel Approach for Credit Card Fraud Detection

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

1. Implementation of Novel Approach for Credit Card Fraud Detection

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

1. Credit card fraud detection using Machine Learning Techniques

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

rate. The results shows of optimal accuracy for naïve bayes, k-nearest neighbor and logistic regression classifiers are 97.92%, 97.69% and 54.86% respectively. The comparative results show that k-nearest neighbor performs better than naïve bayes and logistic regression techniques.

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

## FUNCTIONAL REQUIREMENTS

* + - The system should be developed using Python Programming on Anaconda Framework in Windows Operating System.
    - Once user can able to upload credit card fraud dataset in .csv file format.
    - The User can able to convert .csv file format into ARFF file Format (Attribute relation file format ) CSV – (Comma separated value).
    - This System must have processing module which has to clear & unnecessary data.
    - User can able to divide processed data into training dataset & test dataset in data preparation module.
    - From training dataset this system has to create validation model using Random Forest Algorithm.
    - Once Validation model is created this system can able to take test dataset & predict the credit card fraud.
    - In result analysis this system must show the accuracy level of our built validation module.

## NON-FUNCTIONAL REQUIREMENTS

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system’s *quality attributes*. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

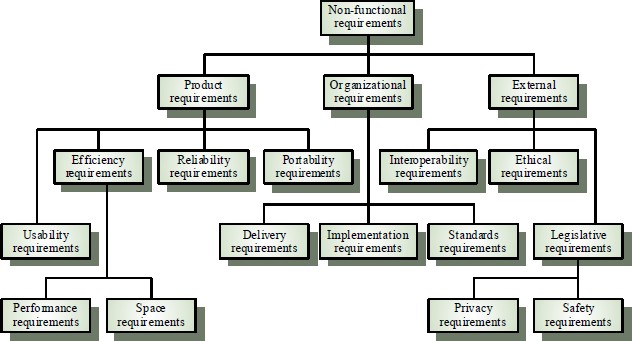


Fig.2.1 Some Non-Functional Requirements Chart

#### Some Non-Functional Requirements are as follows:

* + 1. **Reliability**

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

#### Maintainability

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

#### Performance

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

#### Portability

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

#### Scalability

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

#### Flexibility

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

# 3. SYSTEM DESIGN

System Architecture design-identifies the overall hypermedia structure for the WebApp. Architecture design is tied to the goals establish for a WebApp, the content to be presented, the users who will visit, and the navigation philosophy that has been established. Content architecture, focuses on the manner in which content objects and structured for presentation and navigation. WebApp architecture, addresses the manner in which the application is structure to manage user interaction, handle internal processing tasks, effect navigation, and present content. WebApp architecture is defined within the context of the development environment in which the application is to be implemented.

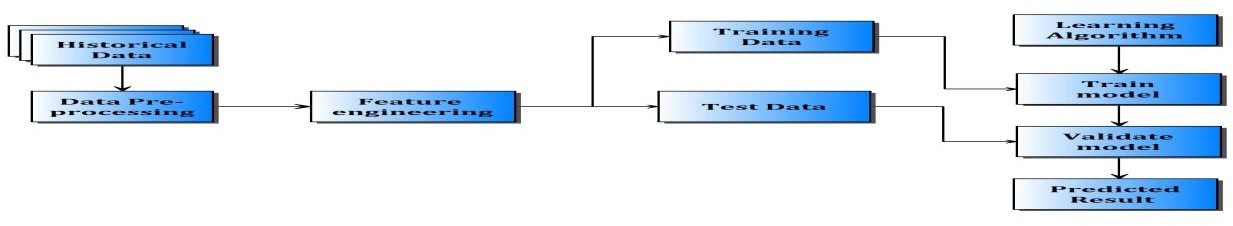


Fig.3.1 System Architecture

## 3.2. FLOW CHART DIAGRAM

It is important to complete all tasks and meet deadlines. There are many project management tools that are available to help project managers manage their tasks and schedule and one of them is the [flowchart.](https://project-management-knowledge.com/definitions/f/flowchart/)

A [flowchart](https://project-management-knowledge.com/definitions/f/flowchart/) is one of the seven basic quality tools used in project management and it displays the actions that are necessary to meet the goals of a particular task in the most practical sequence. Also called as process maps, this type of tool displays a series of steps with

branching possibilities that depict one or more inputs and transforms them to outputs. The advantage of flowcharts is that they show the activities involved in a project including the decision points, parallel paths, branching loops as well as the overall sequence of processing through mapping the operational details within the horizontal value chain. Moreover, this particular tool is very used in estimating and understanding the cost of quality for a particular process. This is done by using the branching logic of the workflow and estimating

the [expected monetary](https://project-management-knowledge.com/definitions/e/expected-monetary-value-analysis/) returns.

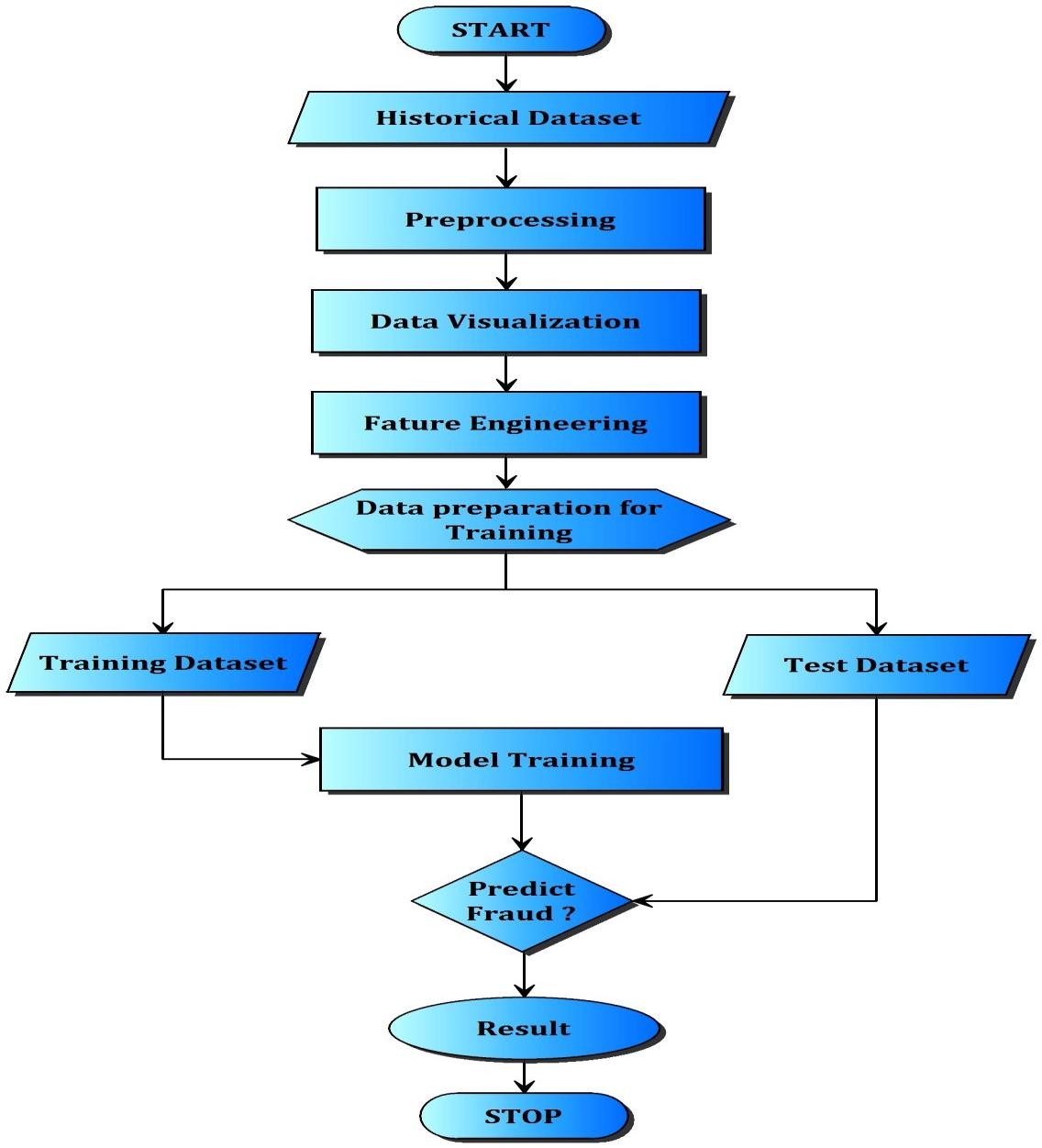


Fig.3.2 Flow Chart Diagram

## USE CASE DIAGRAMS :

A use case is a set of scenarios that describing an interaction between a source and a destination. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. shows the use case diagram.

### Use case diagram admin

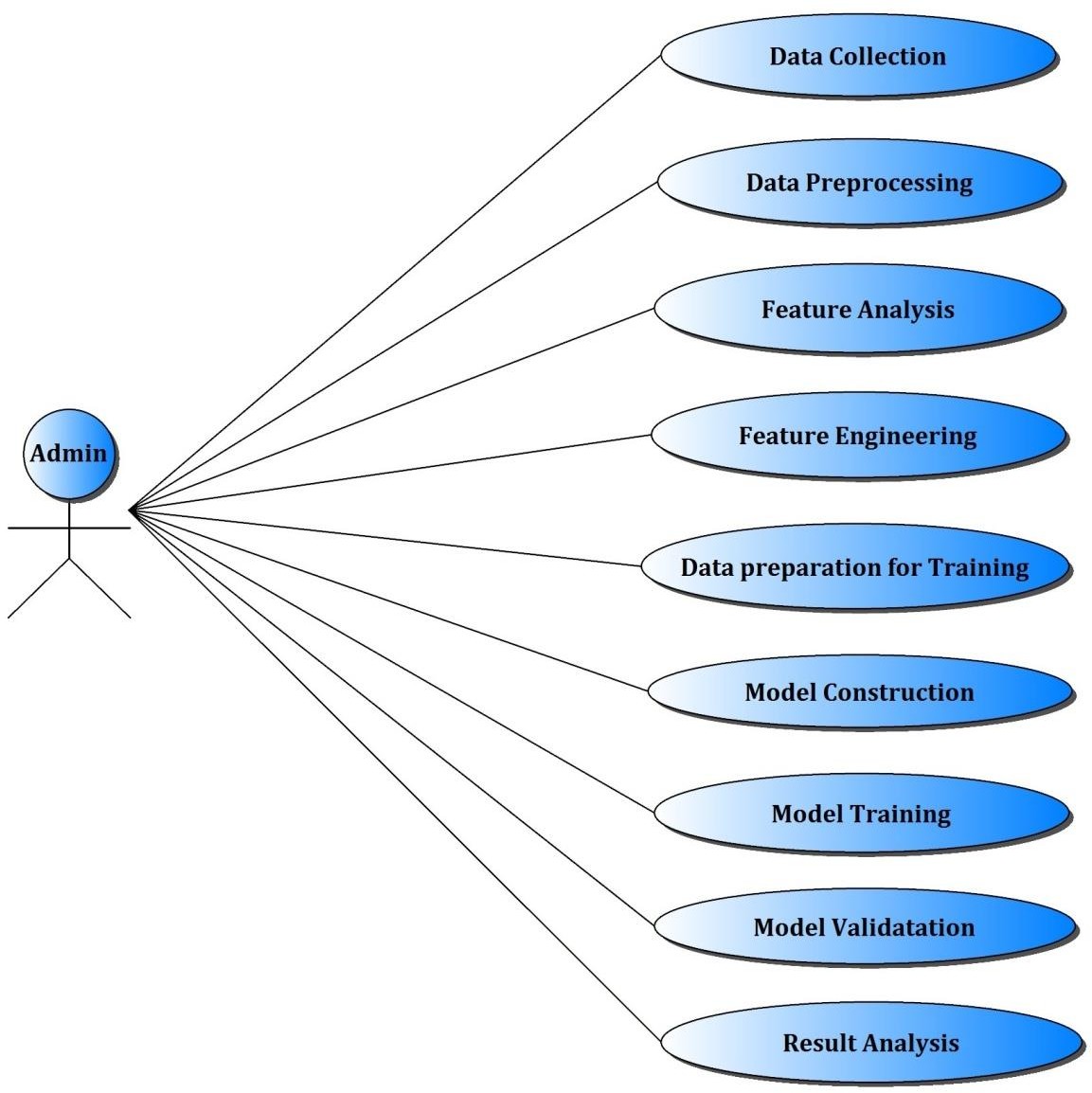


Fig.3.3 Use Case Diagram Admin

## DATA FLOW DIAGRAM :

1. A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities. DFD’s show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols:
2. Squares representing external entities, which are sources and destinations of information entering and leaving the system.
3. Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it.
4. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly.
5. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.

## Level 0 data flow diagram

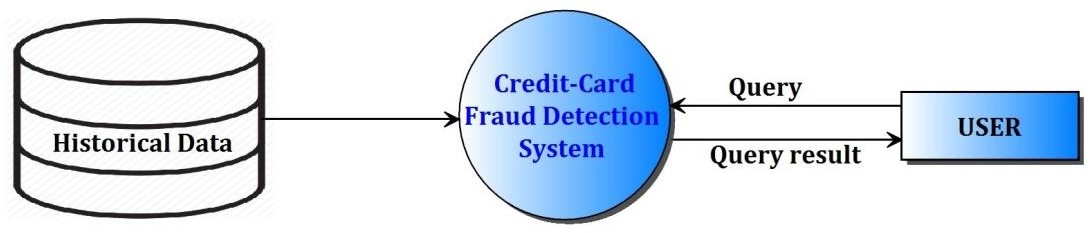


Fig.3.4 Level 0 Data Flow Diagram

### Level1 data flow diagram

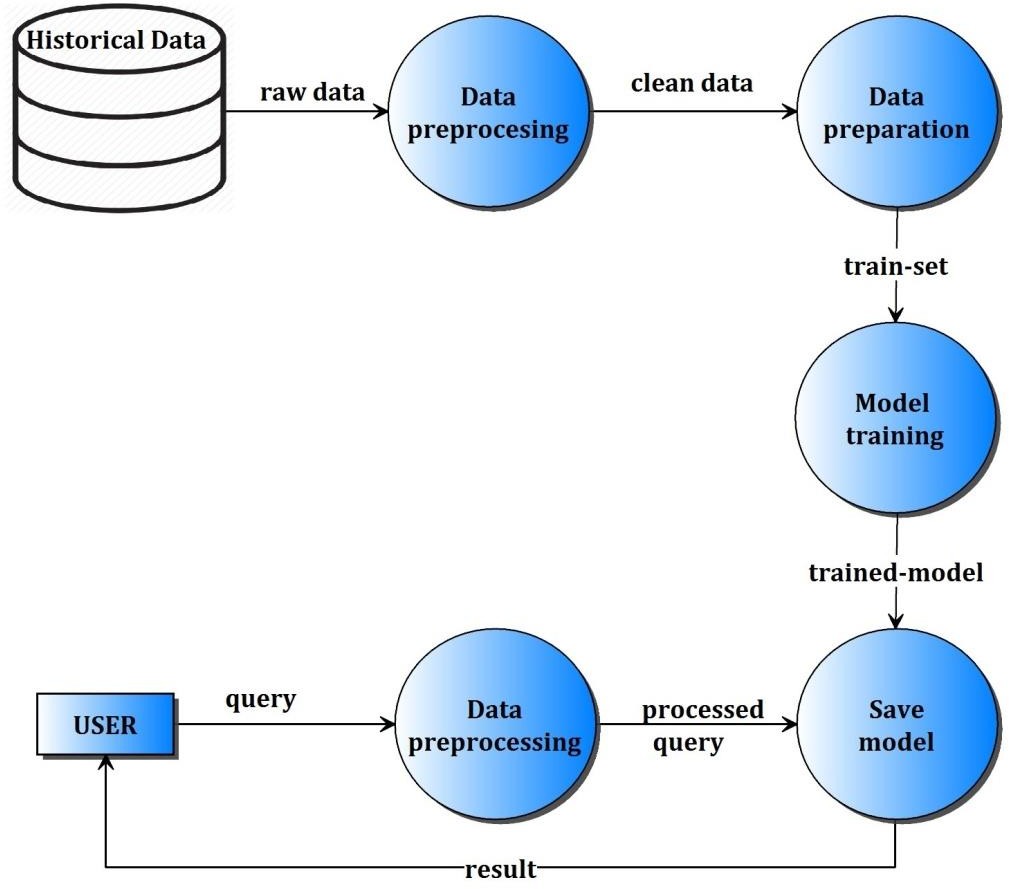


Fig.3.5 Level 1 Data Flow Diagram

## Level2 data flow diagram

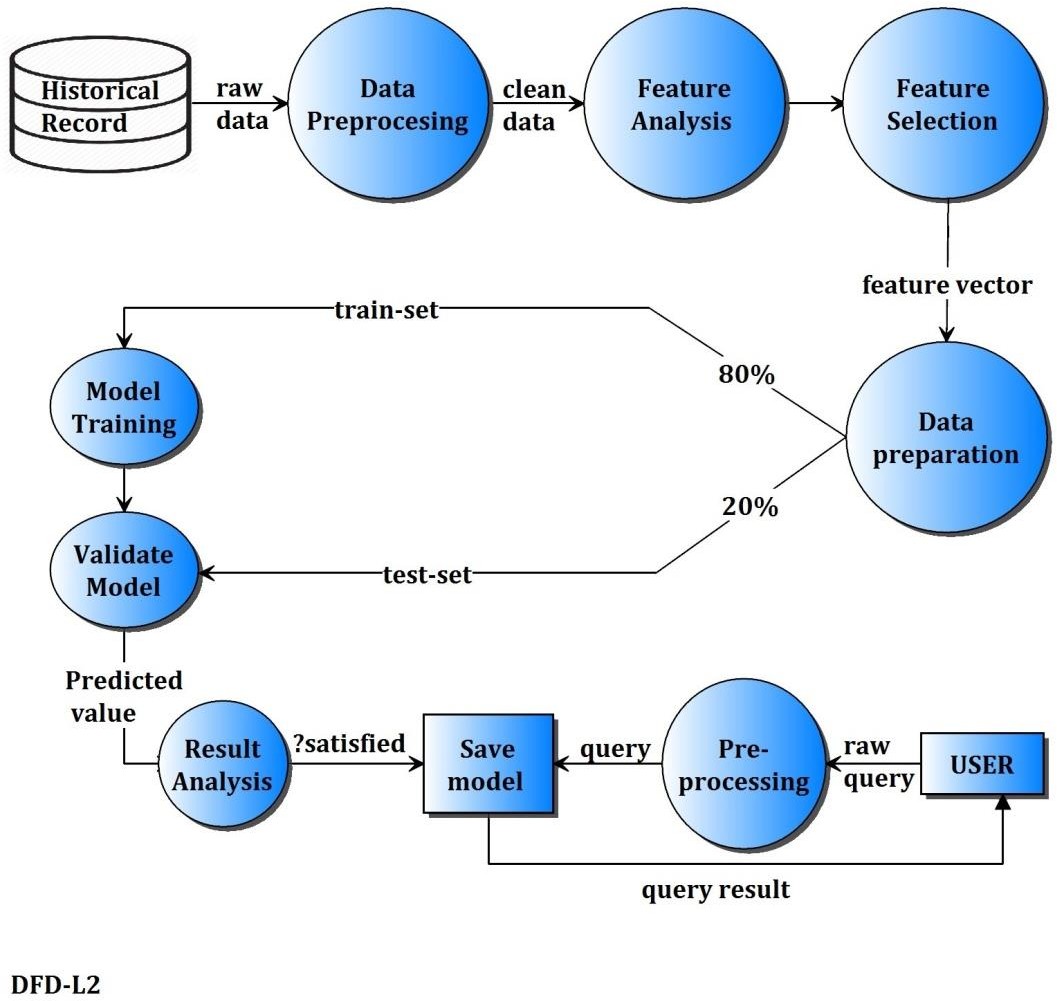


Fig.3.6 Level 2 Data Flow Diagram

# SYSTEM IMPLEMENTATION

* 1. **PYTHON INTRODUCTION**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

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

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

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

* + - **Python is Interpreted −** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
    - **Python is Interactive −** you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
    - **Python is Object-Oriented −** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
    - **Python is a Beginner's Language −** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### Features:

Python's features include – All these.

* + - **Easy-to-learn −** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
    - **Easy-to-read −** Python code is more clearly defined and visible to the eyes.
    - **Easy-to-maintain −** Python's source code is fairly easy-to-maintain.
    - **A broad standard library −** Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.
    - **Interactive Mode −** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
    - **Portable −** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
    - **Extendable −** you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
    - **Databases −** Python provides interfaces to all major commercial databases.
    - **GUI Programming −** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
    - **Scalable −** Python provides a better structure and support for large programs than shell scripting.

#### Why python emerging as a leader:

There’s battle out there happening in the minds of aspiring data scientists to choose the best data science tool. Though there are quite a number of data science tools that provide the much-needed option, the close combat narrows down between two popular languages – Python and R.

Between the two, Python is emerging as the popular language used more in data science applications.

Take the case of the tech giant Google that has created the deep learning framework called tensor flow – Python is the primary language used for creating this framework. Its footprint has continued to increase in the environment promoted by Netflix. Production engineers at Face book and Khan Academy have for long been using it as a prominent language in their environment.

Python has other advantages that speed up it’s upward swing to the top of data science tools. It integrates well with the most cloud as well as platform-as-a-service providers. In supporting multiprocessing for parallel computing, it brings the distinct advantage of ensuring large-scale performance in data science and machine learning. Python can also be extended with modules written in C/C++.

#### Where Python becomes the perfect-fit:

There are tailor-made situations where it is the best data science tool for the job. It is perfect when data analysis tasks involve integration with web apps or when there is a need to incorporate statistical code into the production database. The full-fledged programming nature of Python makes it a perfect fit for implementing algorithms. Its packages rooted for specific data science jobs. Packages like Numpy, SciPy, and pandas produce good results for data analysis jobs. While there is a need for graphics, Python’s matplotlib emerges as a good package, and for machine learning tasks, scikit-learn becomes the ideal alternate.

#### Why is Python preferred over other data science tools?

It is ‘Pythonic’ when the code is written in a fluent and natural style. Apart from that, it is also known for other features that have captured the imaginations of data science community.

#### Easy to learn:

The most alluring factor of Python is that anyone aspiring to learn this language can learn it easily and quickly. When compared to other data science languages like R, Python promotes a shorter learning curve and scores over others by promoting an easy-to-understand syntax.

#### Scalability:

When compared to other languages like R, Python has established a lead by emerging as a scalable language, and it is faster than other languages like Matlab and Stata. Python’s scalability lies in the flexibility that it gives to solve problems, as in the case of YouTube that migrated to Python. Python has come good for different usages in different industries and for rapid development of applications of all kinds.

#### Choice of data science libraries:

The significant factor giving the push for Python is the variety of data science/data analytics libraries made available for the aspirants. Pandas, StatsModels, NumPy, SciPy, and Scikit-Learn, are some of the libraries well known in the data science community. Python does not stop with that as libraries have been growing over time. What you thought was a constraint

a year ago would be addressed well by Python with a robust solution addressing problems of specific nature.

#### Python community:

One of the reasons for the phenomenal rise of Python is attributed to its ecosystem. As Python extends its reach to the data science community, more and more volunteers are creating data science libraries. This, in turn, has led the way for creating the most modern tools and processing in Python.

The widespread and involved community promotes easy access for aspirants who want to find solutions to their coding problems. Whatever queries you need, it is a click or a Google search away. Enthusiasts can also find access to professionals on Code mentor and Stack Overflow to find the right answers for their queries.

#### Graphics and visualization:

Python comes with varied visualization options. Matplotlib provides the solid foundation around which other libraries like Sea born, pandas plotting, and ggplot have been built. The visualization packages help you get a good sense of data, create charts, graphical plot and create web-ready interactive plots.

Why Choose Python?

If you’re going to write programs, there are literally dozens of commonly used languages to choose from. Why choose Python? Here are some of the features that make Python an appealing choice.

Python is Popular

Python has been growing in popularity over the last few years. The 2018 [Stack Overflow](https://insights.stackoverflow.com/survey/2018) [Developer Survey](https://insights.stackoverflow.com/survey/2018) ranked Python as the 7th most popular and the number one most wanted technology of the year. [World-class software development countries around the globe use](https://realpython.com/world-class-companies-using-python/) [Python every single day.](https://realpython.com/world-class-companies-using-python/)

According to [research by Dice](https://insights.dice.com/2016/02/01/whats-hot-and-not-in-tech-skills/) Python is also one of the hottest skills to have and the most popular programming language in the world based on the. [Popularity of Programming](https://pypl.github.io/PYPL.html) [Language Index](https://pypl.github.io/PYPL.html)

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you’d like to dig deeper into [Python salary](https://dbader.org/blog/why-learn-python) [statistics and job opportunities, you can do so here](https://dbader.org/blog/why-learn-python).

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes. A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smartphones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way. A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language. Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter

examples probably vary somewhat by implementation or dialect. Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It’s Not That Simple

For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, a[nd object-oriented programming.](https://realpython.com/python3-object-oriented-programming/)

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don’t: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

#### Is Python ‘the’ tool for machine learning?

When it comes to data science, machine learning is one of the significant elements used to maximize value from data. With Python as the data science tool, exploring the basics of machine learning becomes easy and effective. In a nutshell, machine learning is more about statistics, mathematical optimization, and probability. It has become the most preferred machine learning tool in the way it allows aspirants to ‘do math’ easily.

Name any math function, and you have a Python package meeting the requirement. There is Numpy for numerical linear algebra, CVXOPT for convex optimization, Scipy for general scientific computing, SymPy for symbolic algebra, PYMC3, and Statsmodel for statistical modeling.

With the grip on the basics of machine learning algorithm including logistic regression and linear regression, it makes it easy to implement machine learning systems for predictions by way of its scikit-learn library. It’s easy to customize for neutral networks and deep learning with libraries including Keras, Theano, and TensorFlow.

Data science landscape is changing rapidly, and tools used for extracting value from data science have also grown in numbers. The two most popular languages that fight for the top spot are R and Python. Both are revered by enthusiasts, and both come with their strengths and weaknesses.

# ALGORITHMS USED

* + 1. **Decision Tree**

Decision Tree algorithm belongs to the family of [supervised learning algorithms](https://dataaspirant.com/2014/09/19/supervised-and-unsupervised-learning/). Unlike other supervised learning algorithms, decision tree algorithm can be used for solving [regression and classification](https://dataaspirant.com/2014/09/27/classification-and-prediction/) problems too.

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rule inferred from prior data(trainingdata).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each **leaf node** corresponds to a class label.

**Decision Tree Algorithm Pseudocode**

* + - 1. The best attribute of the dataset at the **root** of the tree.
      2. Split the training set into **subsets**. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
      3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

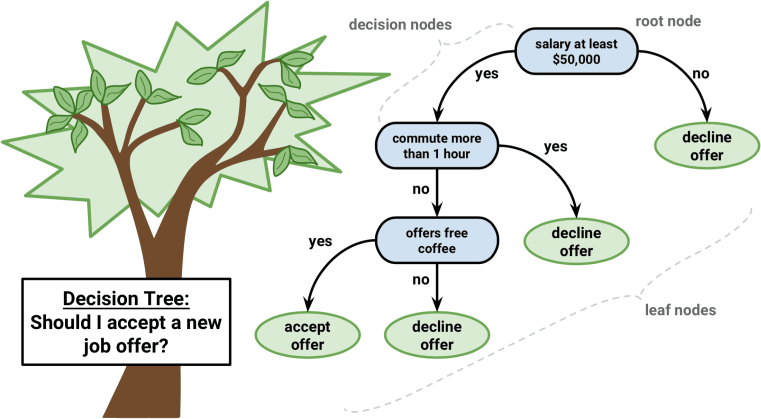


Fig.4.1 Decision Tree Algorithm Diagram

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 record’s attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

We continue comparing our record’s attribute values with other **internal nodes**of the tree until we reach **a leaf node** with predicted class value. As we know how the modeled decision tree can be used to predict the target class or the value. Now let’s understanding how we can create the decision tree model.

**Assumptions while creating Decision Tree**

The below are the some of the assumptions we make while using Decision tree:

* At the beginning, the whole training set is considered as the **root.**
* Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
* Records are distributed recursively on the basis of attribute values.
* Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

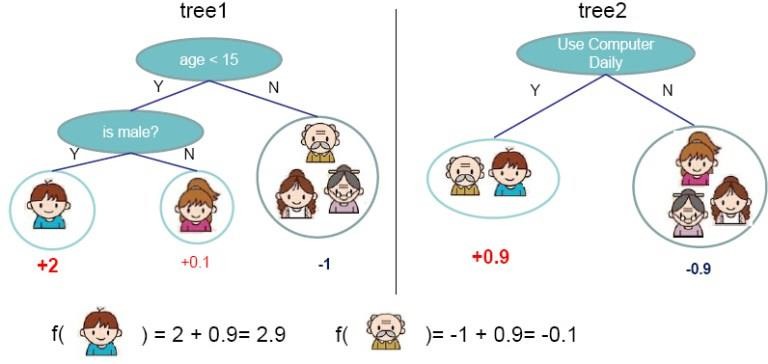


Fig.4.2 Binary Decision Tree Diagram.

Decision Trees follow Sum of Product (SOP) representation. For the above images, you can see how we can predict can we accept the new job offer? and Use computer

daily? from traversing for the root node to the leaf node.

It’s a sum of product representation. The Sum of product(SOP) is also known as Disjunctive Normal Form. For a class, every branch from the root of the tree to a leaf node having the same class is a conjunction(product) of values, different branches ending in that class form a disjunction(sum).

The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is know the attributes selection. We have different attributes selection measure to identify the attribute which can be considered as the root note at each level.

**The popular attribute selection measures:**

* Information gain
* Gini index

**Attributes Selection**

If dataset consists of **“n”** attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can’t solve the issue. If we follow a random approach, it may give us bad results with low accuracy.

For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criterion* like **information gain, gini index,** etc. These criterions will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root.

While using information Gain as a criterion, we assume attributes to be categorical, and for gini index, attributes are assumed to be continuous.

**Information Gain**

By using information gain as a criterion, we try to estimate the information contained by each attribute. We are going to use some points deducted from[information theory](https://en.wikipedia.org/wiki/Information_theory).

To measure the randomness or uncertainty of a random variable X is defined by**Entropy**. For a binary classification problem with only two classes, positive and negative class.

* If all examples are positive or all are negative then entropy will be zero i.e, low.
* If half of the records are of positive class and half are of negative class then entropy is one i.e, high.

By calculating entropy measure of each attribute we can calculate their information gain.

Information Gain calculates the expected reduction in entropy due to sorting on the attribute. Information gain can be calculated.

* + 1. **Random Forest Algorithm**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. Random Forest Classifier being ensemble algorithm tends to give more accurate result. This is because it works on principle, Number of weak estimators when combined forms strong estimator.

Even if one or few decision trees are prone to a noise, overall result would tend to be correct. Even with small number of estimators = 30 it gave us high accuracy as 97%.

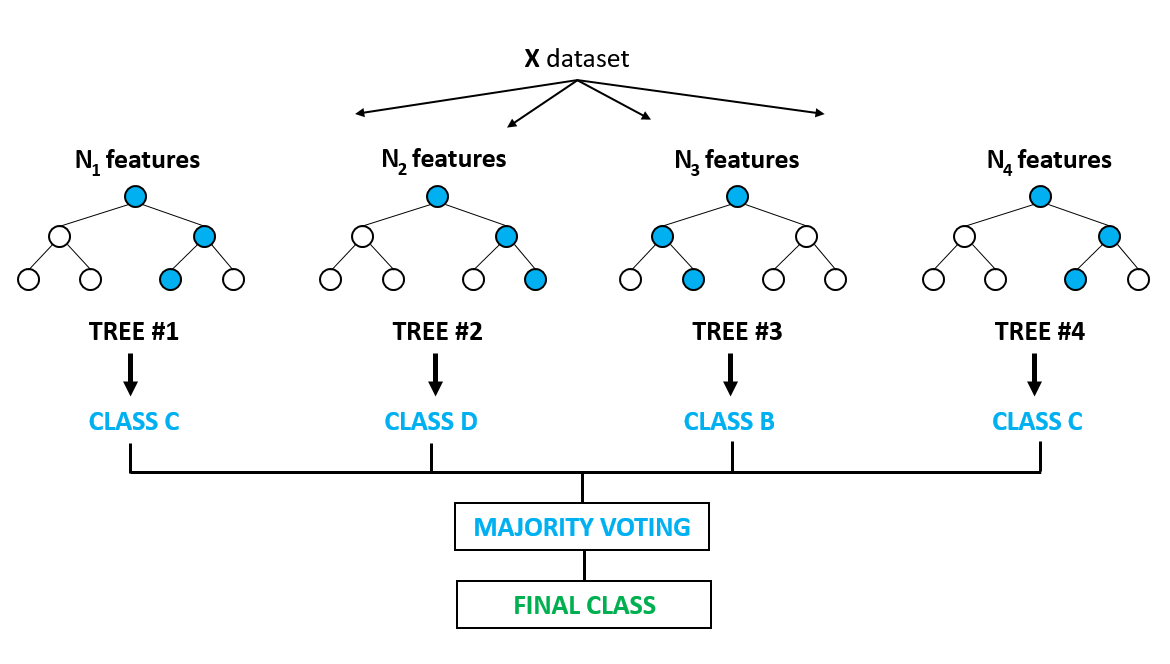
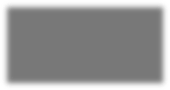


Fig.4.3 Random Forest Algorithm Diagram.

* 1. **SDLC**

It will cover the details explanation of methodology that is being used to make this project complete and working well. Many methodology or findings from this field mainly generated into journal for others to take advantages and improve as upcoming studies. The method is use to achieve the objective of the project that will accomplish a perfect result. In order to evaluate this project, the methodology based on System Development Life Cycle (SDLC), generally three major step, which is planning, implementing and analysis.

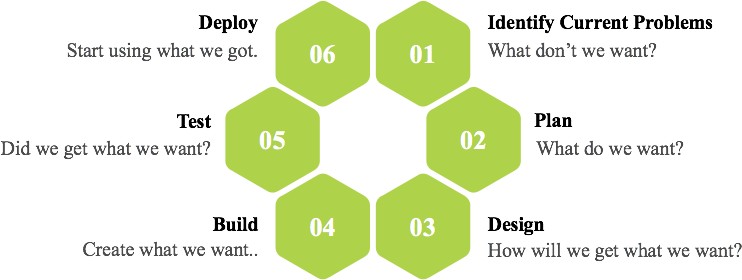
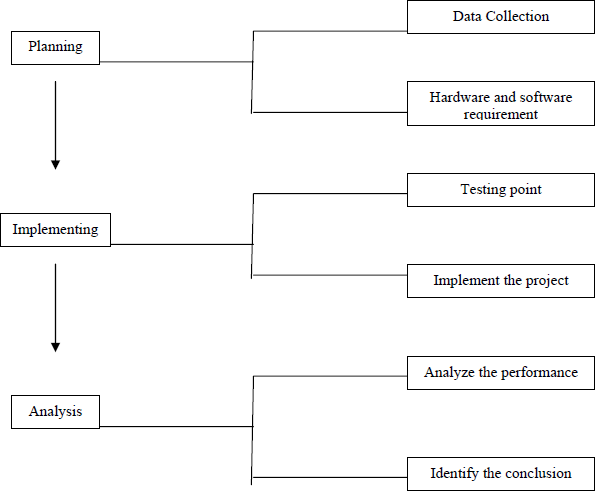


Fig 4.6 Software Development Life Cycle



**Planning:**

Fig 4.7 Steps of Methodology

To identify all the information and requirement such as hardware and software, planning must be done in the proper manner. The planning phase has two main elements namely data collection and the requirements of hardware and software.

#### Data collection:

Machine learning needs two things to work, data (lots of it) and models. When acquiring the data, be sure to have enough features (aspect of data that can help for a prediction, like the surface of the house to predict its price) populated to train correctly your learning model. In general, the more data you have the better so make to come with enough rows.

The primary data collected from the online sources remains in the raw form of statements, digits and qualitative terms. The raw data contains error, omissions and inconsistencies. It requires corrections after careful scrutinizing the completed questionnaires. The following steps are involved in the processing of primary data. A huge volume of raw data collected through field survey needs to be grouped for similar details of individual responses.

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

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

* + 1. Data Cleaning
    2. Data Integration
    3. Data Transformation
    4. Data Reduction

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

* + 1. **Inaccurate data (missing data) -** There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics and much more.
    2. **The presence of noisy data (erroneous data and outliers) -** The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.
    3. **Inconsistent data -** The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more.

# REFERENCES

1. S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decis. Support Syst.*, vol. 50, no. 3, pp. 602–613, 2011.
2. K. Chaudhary, J. Yadav, and B. Mallick, “A review of Fraud Detection Techniques: Credit Card,” *Int. J. Comput. Appl.*, vol. 45, no. 1, pp. 975–8887, 2012.
3. “Mining of Massive Datasets Second Edition.”
4. F. N. Ogwueleka, “Data Mining Application in Credit Card Fraud Detection System,” vol. 6, no. 3, pp. 311–322, 2011.
5. H. Nordberg, K. Bhatia, K. Wang, and Z. Wang, “BioPig: a Hadoop-based analytic toolkit for large-scale sequence data,”*Bioinformatics*, vol. 29, no. 23, pp. 3014– 3019, Dec. 2013.
6. M. Hegazy, A. Madian, and M. Ragaie, “Enhanced Fraud Miner: Credit Card Fraud Detection using Clustering Data Mining Techniques,” *Egypt. Comput. Sci.*, no. 03, pp. 72–81, 2016.
7. M. Zareapoor and P. Shamsolmoali, “Application of credit card fraud detection: Based on bagging ensemble classifier,” *Procedia Comput. Sci.*, vol. 48, no. C, pp. 679–686, 2015.
8. O. S. Yee, S. Sagadevan, N. Hashimah, and A. Hassain, “Credit Card Fraud Detection Using Machine Learning As Data Mining Technique,” vol. 10, no. 1, pp. 23–27.
9. K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, “Credit Card Fraud Detection Using AdaBoost and Majority Voting,” *IEEE Access*, vol. 6, pp. 14277– 14284, 2018.
10. N. Mahmoudi and E. Duman, “Detecting credit card fraud by Modified Fisher Discriminant Analysis,” *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2510–2516, 2015.
11. Anderson M. (2008). ‗From Subprime Mortgages to Subprime Credit Cards‘. Communities and Banking, Federal Reserve Bank of Boston, pp. 21-23.
12. Anwer et al. (2009-2010). ‗Online Credit Card Fraud Prevention System for Developing Countries‘, International Journal of Reviews in Computing, ISSN: 2076- 3328, Vol. 2, pp. 62-70.
13. Arias, J.C. & Miller R. (2009). ‗Market Analysis of Student about Credit Cards‘. Business Intelligence Journal, Vol. 3, No. 1, pp. 23-36.
14. Bhatla T.P. et al. (2003). ‗Understanding Credit Card Frauds‘. Cards Business Review, 01, pp. 01-15.
15. Bhusari V. & Patil S. (2011). ‗Study of Hidden Markov Model in Credit Card Fraudulent Detection‘. International Journal of Computer Applications, Vol. 20, No. 5, pp. 33-36.
16. Calem P. & Mester L. (1995). ‗Consumer Behavior and the Stickiness of Credit Card Interest Rates‘. The American Economic Review, Vol. 85, No. 5, pp. 1327-1333.
17. Chakravorti S. (2003). ‗Theory of Credit Card Networks: A Survey of the Literature‘, Review of Network Economics, Vol. 2, Issue 2, pp. 50-68.
18. Chan P.K. et al (1999). ‗Distributed Data Mining in Credit Card Fraud Detection‘, IEEE Intelligent Systems, pp. 67-74.
19. Chang C. & Chang S. (2010). ‗The Design of E-Traveler‘s Check with efficiency and Mutual Authentication‘. Journal of Networks, Vol. 5, No. 3, pp. 275-282.
20. Delamaire et al. (2009) ‘Credit Card Fraud Detection Techniques: A Review’, Banks and banks Systems, Vol. 4, Issue 2, pp. 57-68.