## 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.

## 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

#### Artificial Intelligence 32

#### Sample of handy Machine Learning Algo Mind Map 33

#### Software Development Life Cycle 36

#### Steps of Methodology 36

#### Confusion Matrix and Formulae 39

#### SQL Table Structure 45

#### Output Diagram 54

#### Histogram Figure 55

#### Screenshot 1. 60

#### Screenshot 2. 61

**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. TESTING & IMPLEMENTATION 56
   1. Software Testing Introduction 56
   2. SDLC Models 57
   3. Types of Testing 58
2. RESULT & ANALYSIS 60
3. CONCLUSION & FUTURE SCOPE 62
4. References 63
5. Annexure 65

# 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

* + - * More accurate result.
      * Able to detect different fraudulent behavior.
      * 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.

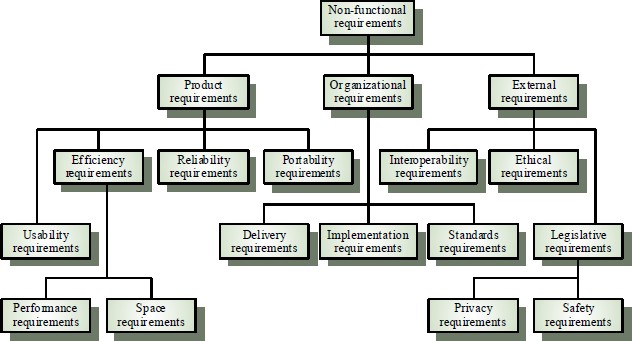


Fig.2.1 Some Non-Functional Requirements Chart

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

* + - **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.

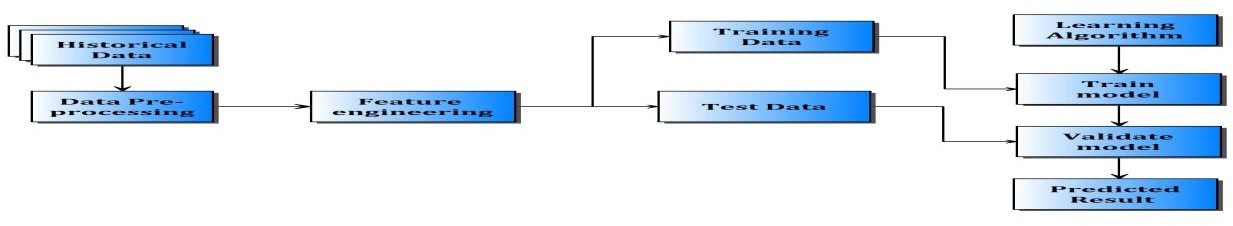


Fig.3.1 System Architecture

## 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.

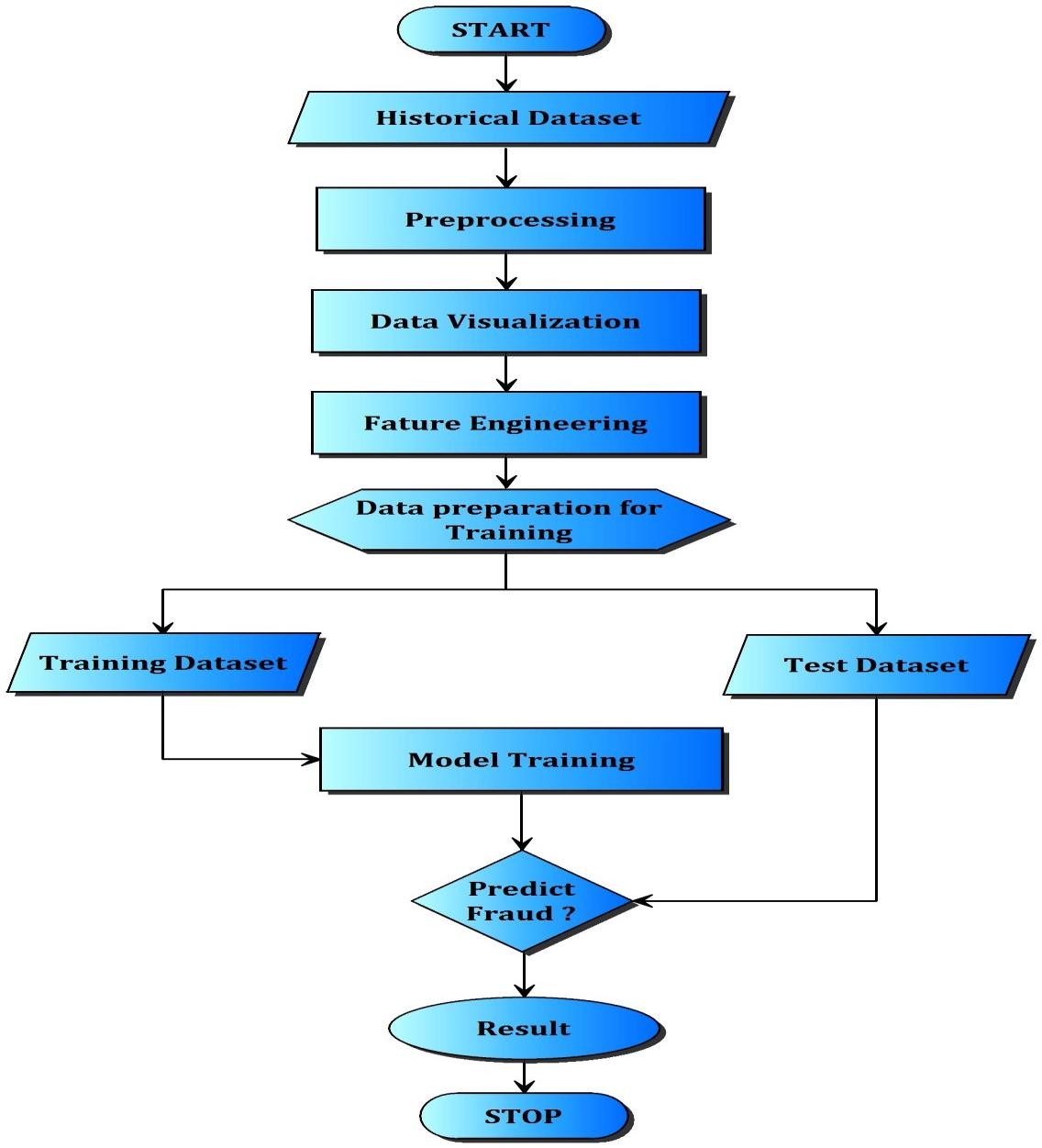


Fig.3.2 Flow Chart Diagram

## 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

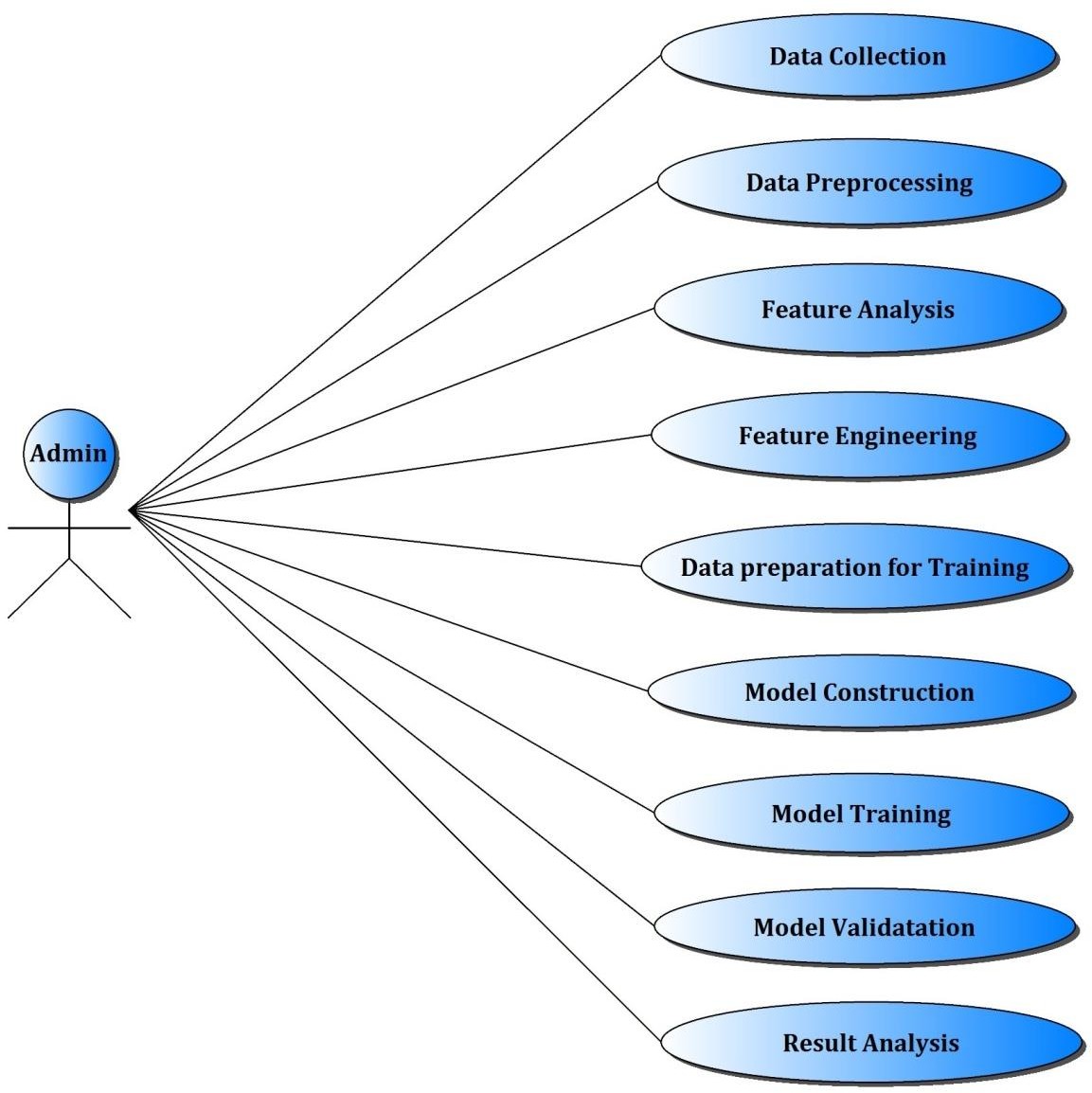


Fig.3.3 Use Case Diagram Admin

## 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

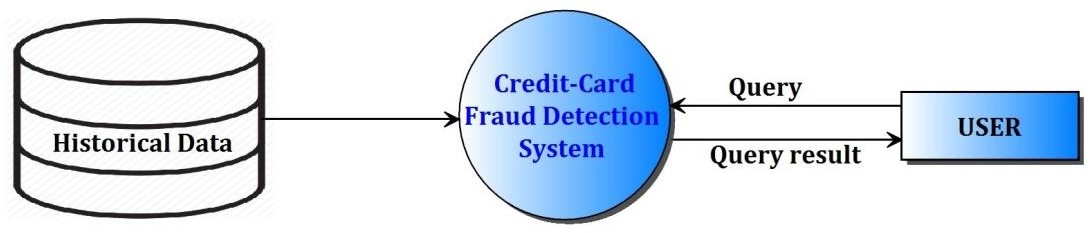


Fig.3.4 Level 0 Data Flow Diagram

### Level1 data flow diagram

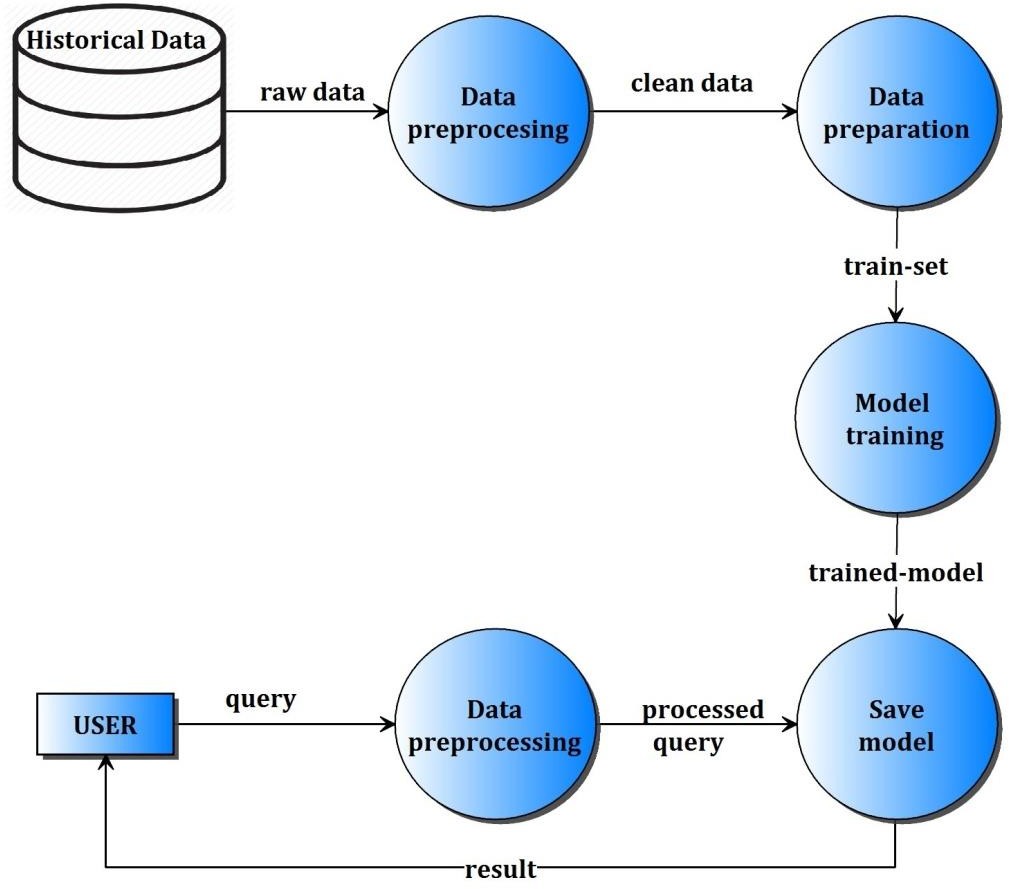


Fig.3.5 Level 1 Data Flow Diagram

## Level2 data flow diagram

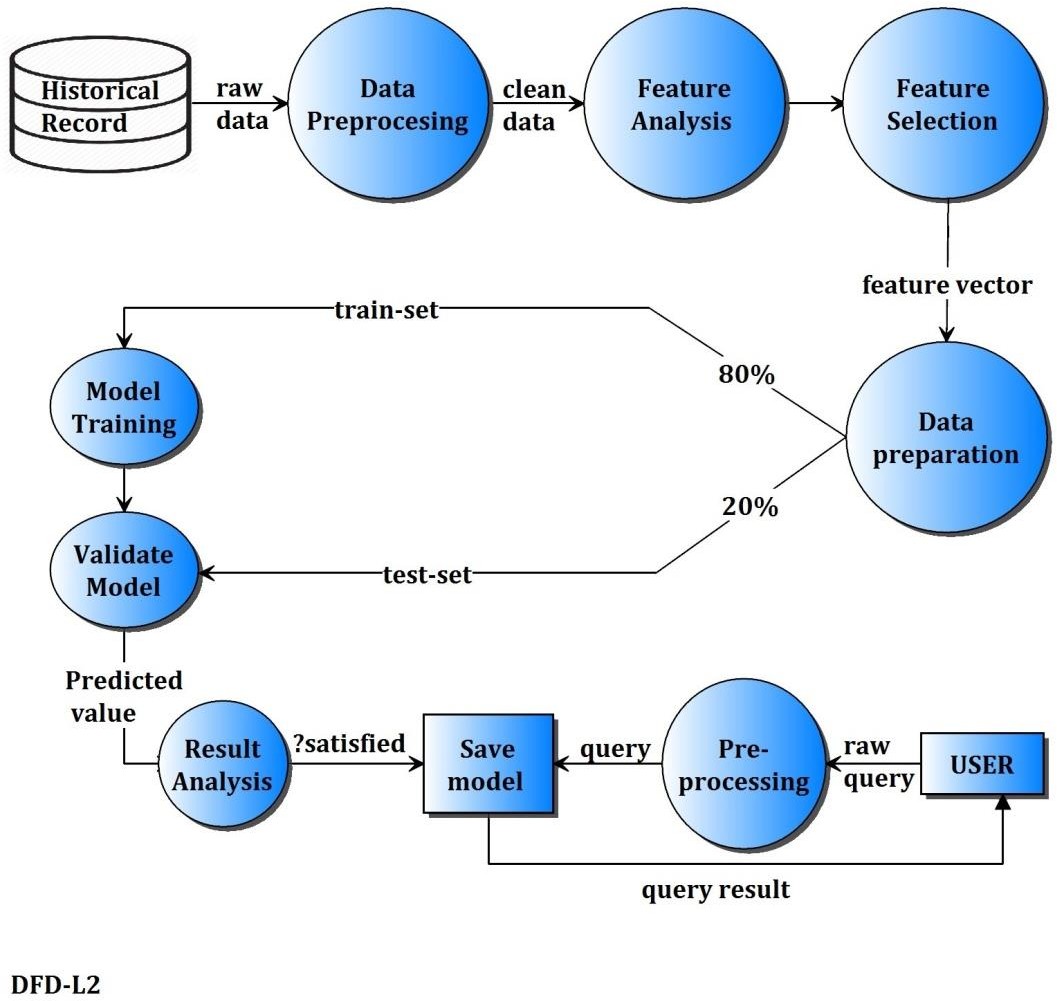


Fig.3.6 Level 2 Data Flow Diagram

# 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. Place 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.

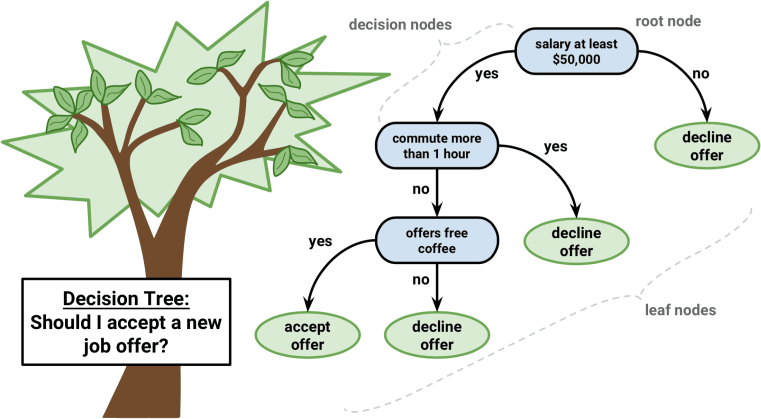


Fig.4.1 Decision Tree Algorithm Diagram

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.

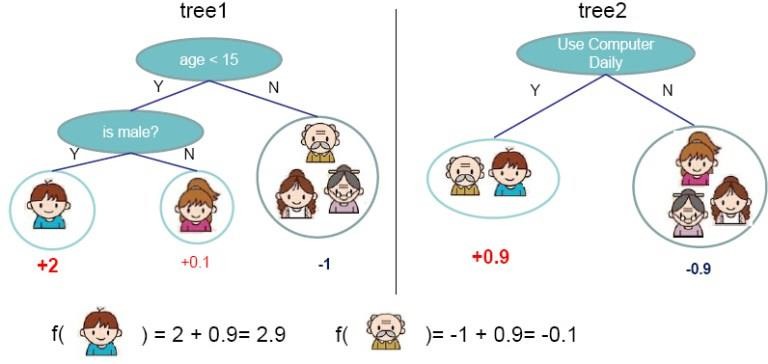


Fig.4.2 Binary Decision Tree Diagram.

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%.

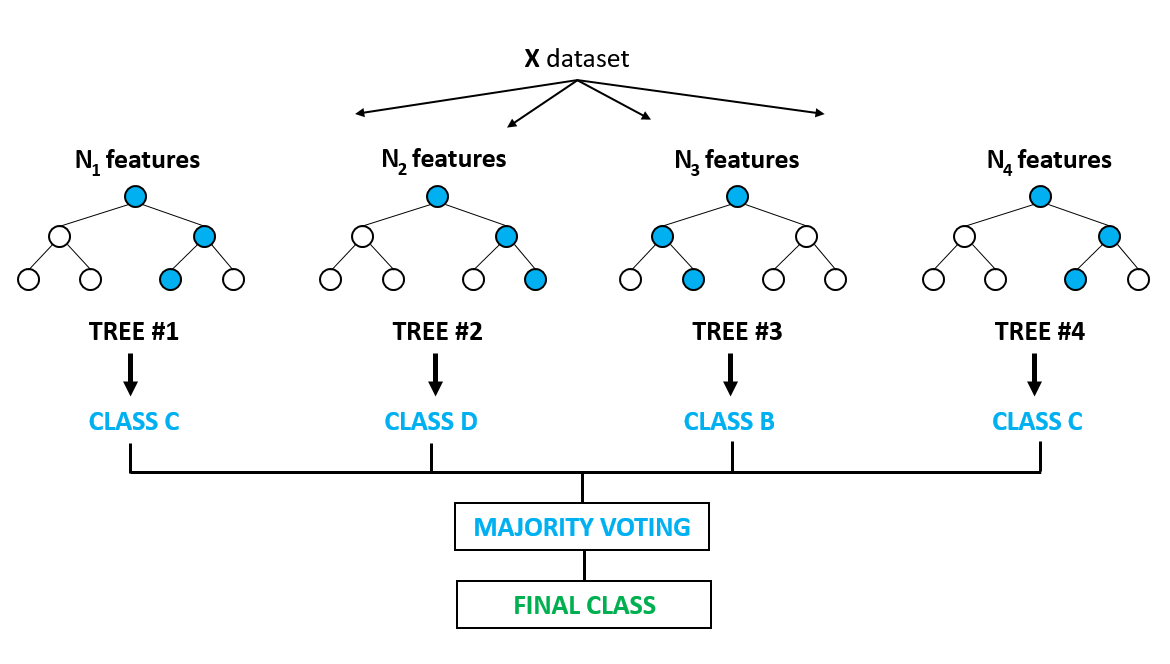
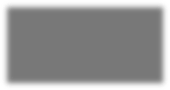


Fig.4.3 Random Forest Algorithm Diagram.

#### Artificial intelligence

Artificial Intelligence encompasses a very broad scope. You could even consider something like Dijkstra's shortest path algorithm as Artificial Intelligence. However, two categories of AI are frequently mixed up: Machine Learning and Deep Learning. Both of these refer to statistical modeling of data to extract useful information or make predictions. In this article, we will list the reasons why these two statistical modeling techniques are not the same and help you further frame your understanding of these data modeling paradigms.

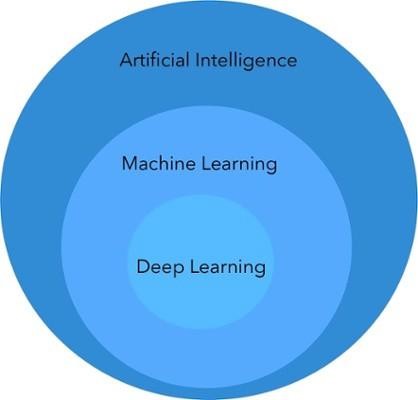


Fig.4.4 Artificial Intelligence

#### Machine learning

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

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

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

Supervised Machine Learning

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

Y = f(X)

The goal is to approximate the mapping function so well that when you have new input data

(x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct

answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Get your FREE Algorithms Mind Map

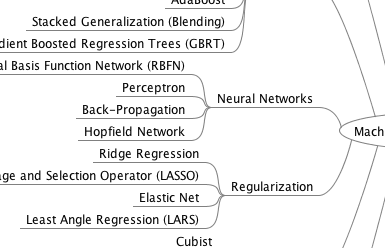


Fig.4.5 Sample of handy machine learning algorithms mind map.

Supervised learning problems can be further grouped into regression and classification problems.

* **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
* **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

* Linear regression for regression problems.
* Random forest for classification and regression problems.
* Support vector machines for classification problems.

Unsupervised Machine Learning

Unsupervised learning is where you only have input data (X) and no corresponding output

[33]

variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

* **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
* **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Some popular examples of unsupervised learning algorithms are:

* k-means for clustering problems.
* Apriori algorithm for association rule learning problems.

Semi-Supervised Machine Learning

Problems where you have a large amount of input data (X) and only some of the data is labeled

(Y) are called semi-supervised learning problems.

These problems sit in between both supervised and unsupervised learning.

A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.

to domain experts. Whereas unlabeled data is cheap and easy to collect and store.

You can use unsupervised learning techniques to discover and learn the structure in the input variables.

You can also use supervised learning techniques to make best guess predictions for the unlabeled data, feed that data back into the supervised learning algorithm as training data and use the model to make predictions on new unseen data.

#### Data Size:

Both Machine Learning and Deep Learning are able to handle massive dataset sizes, however, machine learning methods make much more sense with small datasets. For example, if you only have 100 data points, decision trees, k-nearest neighbors, and other machine learning models will be much more valuable to you than fitting a deep neural network on the data. This is due to the next topic of difference, Interpretability **Interpretability:**

#### Example for how interpretability works in ML & DL:

A lot of the criticism of deep learning methods and machine learning algorithms such as Support Vector Machine or (maybe, because you can at least visualize the constituent probabilities making up the output), Naive Bayes, are due to their difficulty to interpret. For example, when a Convolutional Neural Network outputs ‘cat’ in a dog vs. cat problem, nobody seems to know why it did that. In contrast, when you are modeling data such as an electronic health record or bank loan dataset with a machine learning technique, it is much easier to understand the reasoning for the model’s prediction.

One of the best examples of interpretability is decision trees where you follow logical tests down nodes of the tree until you reach a decision. Another machine learning algorithm with high interpretability is k-Nearest Neighbors. This is not a parametric learning algorithm but still falls under the category of machine learning algorithms. It is very interpretability because you easily reason about the similar instances for yourself.

* 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.

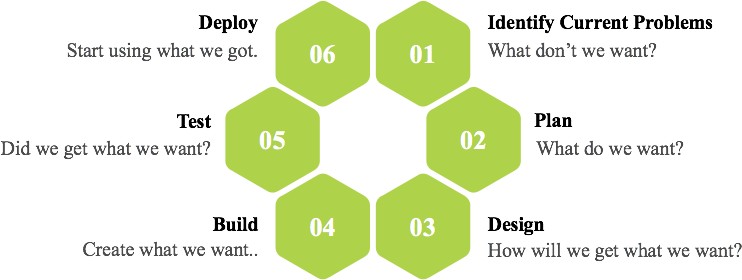
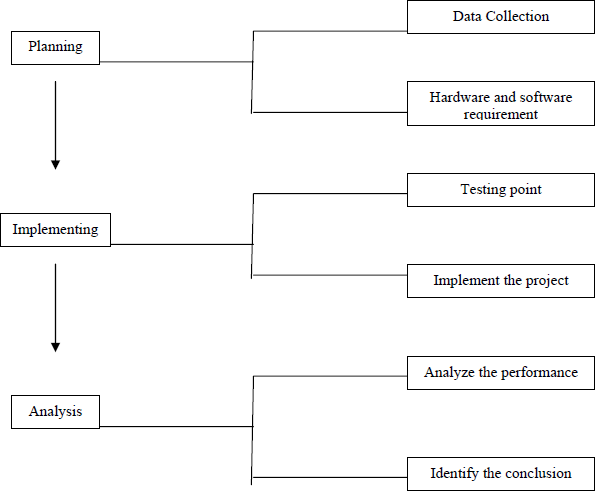


Fig 4.6 Software Development Life Cycle



**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 -

* + - Data Cleaning
    - Data Integration
    - Data Transformation
    - Data Reduction

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

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

**Implementing**

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

#### Analysis

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

After model building, knowing the power of model prediction on a new instance, is very important issue. Once a predictive model is developed using the historical data, one would be curious as to how the model will perform on the data that it has not seen during the model building process. One might even try multiple model types for the same prediction problem, and then, would like to know which model is the one to use for the real-world decision making situation, simply by comparing them on their prediction performance (e.g., accuracy). To measure the performance of a predictor, there are commonly used performance metrics, such as accuracy, recall etc. First, the most commonly used performance metrics will be described, and then some famous estimation methodologies are explained and compared to each other. "Performance Metrics for Predictive Modeling In classification problems, the primary source of performance measurements is a coincidence matrix (**classification matrix or a contingency table**)”. Above figure shows a coincidence matrix for a two-class classification problem. The equations of the most commonly used metrics that can be calculated from the coincidence matrix are also given in Fig 2.7.

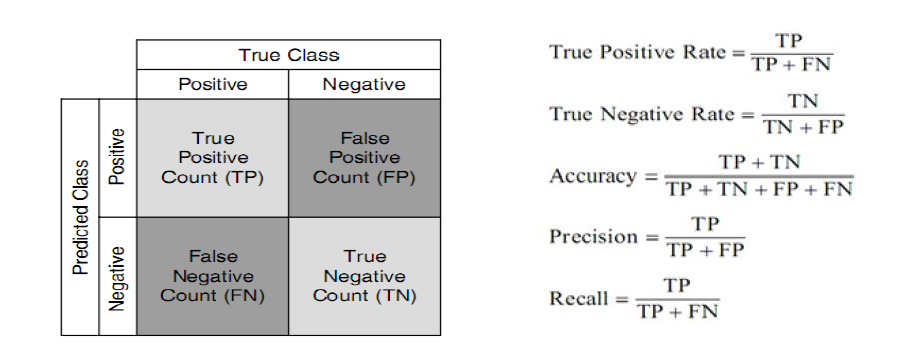


Fig 4.8 confusion matrix and formulae

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

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

#### Flexibility

Sometimes you just don’t want to use what is already there but you want to define something of your own (for example a cost function, a metric, a layer, etc.).

Although Keras 2 has been designed in such a way that you can implement almost everything you want but we all know that low-level libraries provides more flexibility. Same is the case with TF. *You can tweak* TF much more as compared to Keras.

#### Functionality

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

#### Threading and Queues

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

#### Debugger

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

#### Control

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

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

#### Numpy

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

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

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

Operations using NumPy

Using NumPy, a developer can perform the following operations −

* + - Mathematical and logical operations on arrays.
    - Fourier transforms and routines for shape manipulation.
    - Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy AReplacement for Mat Lab

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

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

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

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

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

An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows

−

numpy.array

It creates an ndarray from any object exposing array interface, or from any method that returns

an array.

The ndarray objects can be saved to and loaded from the disk files. The IO functions available are −

* + - **load()** and **save()** functions handle /numPy binary files (with **npy**extension)
    - **loadtxt()** and **savetxt()** functions handle normal text files

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

numpy.save()

The **numpy.save()** file stores the input array in a disk file with **npy**extension.

import numpy as np

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

To reconstruct array from **outfile.npy**, use **load()** function.

import numpy as np

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

It will produce the following output −

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

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

savetxt()

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

Example

import numpy as np

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

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

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

It will produce the following output −

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

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

#### Pandas:

**pandas** is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

* Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
* Ordered and unordered (not necessarily fixed-frequency) time series data.
* Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
* Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

|  |  |  |
| --- | --- | --- |
| The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2- dimensional), handle the vast majority of typical use cases in finance, statistics, social science, | | |
| and many areas of engineering. For R users, Data Frame provides everything that R’s | | data. |
|  | | |
| Frame | provides and much more. Pandas is built on top of [NumPy](http://www.numpy.org/) and is intended to integrate | |
| well within a scientific computing environment with many other 3rd party libraries.  Here are just a few of the things that pandas do well: | | |

* Easy handling of **missing data** (represented as NaN) in floating point as well as non- floating point data
* Size mutability: columns can be **inserted and deleted** from DataFrame and higher dimensional objects
* Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let *Series*, *DataFrame*, etc. automatically align the data for you in computations
* Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
* Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
* Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets
* Intuitive **merging** and **joining** data sets
* Flexible **reshaping** and pivoting of data sets
* **Hierarchical** labeling of axes (possible to have multiple labels per tick)
* Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**
* **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Features of DataFrame

* + - Potentially columns are of different types
    - Size – Mutable
    - Labeled axes (rows and columns)
    - Can Perform Arithmetic operations on rows and columns Structure

Let us assume that we are creating a data frame with student’s data.

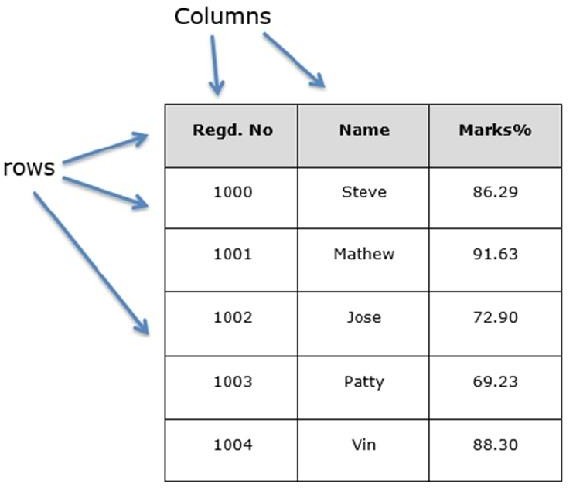


Fig 4.9 SQL Table Structure

You can think of it as an SQL table or a spreadsheet data representation.

pandas.DataFrame

A pandas DataFrame can be created using the following constructor −

pandas.DataFrame( data, index, columns, dtype, copy)

The parameters of the constructor are as follows −

|  |  |
| --- | --- |
| **Sr.No** | **Parameter & Description** |
| 1 | **data** |

|  |  |
| --- | --- |
|  | data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame. |
| 2 | **index**  For the row labels, the Index to be used for the resulting frame is Optional Default np.arrange(n) if no index is passed. |
| 3 | **columns**  For column labels, the optional default syntax is - np.arrange(n). This is only true if no index is passed. |
| 4 | **dtype**  Data type of each column. |
| 5 | **copy**  This command (or whatever it is) is used for copying of data, if the default is False. |

Create DataFrame

A pandas DataFrame can be created using various inputs like −

* + - Lists
    - dict
    - Series
    - Numpy ndarrays
    - Another DataFrame

In the subsequent sections of this chapter, we will see how to create a DataFrame using these inputs.

Create an Empty DataFrame

A basic DataFrame, which can be created is an Empty Dataframe.

Example

#import the pandas library and aliasing as pd

import pandas as pd

df = pd.DataFrame() print df

Its **output** is as follows −

Empty DataFrame Columns: [] Index: []

Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists. Example 1

import pandas as pd data = [1,2,3,4,5]

df = pd.DataFrame(data)

print df

Its **output** is as follows −

0

0 1

1 2

2 3

3 4

4 5

Example 2

import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age']) print df

Its **output** is as follows −

Name Age

1. Alex 10
2. Bob 12
3. Clarke 13

The **Pandas I/O API** is a set of top level reader functions accessed like **pd.read\_csv()** that generally return a Pandas object.

The two workhorse functions for reading text files (or the flat files) are **read\_csv()** and **read\_table()**. They both use the same parsing code to intelligently convert tabular data into a **DataFrame** object −

pandas.read\_csv(filepath\_or\_buffer, sep=',', delimiter=None, header='infer', names=None, index\_col=None, usecols=None pandas.read\_csv(filepath\_or\_buffer, sep='\t', delimiter=None, header='infer', names=None, index\_col=None, usecols=None

Here is how the **csv** file data looks like −

S.No,Name,Age,City,Salary 1,Tom,28,Toronto,20000 2,Lee,32,HongKong,3000 3,Steven,43,Bay Area,8300 4,Ram,38,Hyderabad,3900

Save this data as **temp.csv** and conduct operations on it.

S.No,Name,Age,City,Salary 1,Tom,28,Toronto,20000 2,Lee,32,HongKong,3000

3,Steven,43,Bay Area,8300 4,Ram,38,Hyderabad,3900

Save this data as **temp.csv** and conduct operations on it.

read.csv

**read.csv** reads data from the csv files and creates a DataFrame object.

import pandas as pd

df=pd.read\_csv("temp.csv") print df

Its **output** is as follows −

Salary 20000

3000

8300

3900

28 Toronto

32 HongKong

43 Bay Area

38 Hyderabad

Tom Lee

Steven

Ram

City

Name Age

S.No 1

2

3

4

0

1

2

3

custom index

This specifies a column in the csv file to customize the index using **index\_col.**

import pandas as pd

df=pd.read\_csv("temp.csv",index\_col=['S.No']) print df

Its **output** is as follows −

S.No Name Age

City Salary

1. Tom 28 Toronto 20000
2. Lee 32 HongKong 3000
3. Steven 43 Bay Area 8300
4. Ram 38 Hyderabad 3900

Converters

**dtype** of the columns can be passed as a dict.

import pandas as pd

df = pd.read\_csv("temp.csv", dtype={'Salary': np.float64}) print df.dtypes

Its **output** is as follows −

S.No int64

Name object

Age int64

City object Salary float64 dtype: object

By default, the **dtype** of the Salary column is **int**, but the result shows it as **float** because we have explicitly casted the type.

Thus, the data looks like float −

3000.0

8300.0

3900.0

Ram 38 Hyderabad

4

3 Steven 43 Bay Area

Tom 28 Toronto 20000.0

Lee 32 HongKong

1

2

0

1

2

3

S.No Name Age City Salary

header\_names

Specify the names of the header using the names argument.

import pandas as pd

df=pd.read\_csv("temp.csv", names=['a', 'b', 'c','d','e'])

print df

Its **output** is as follows −

20000

3000

8300

3900

Toronto HongKong Bay Area

Hyderabad

Steven 43

Ram 38

3

4

Tom 28

Lee 32

e

City Salary

Name Age

d

b c

a S.No

1

2

0

1

2

3

4

Observe, the header names are appended with the custom names, but the header in the file has not been eliminated. Now, we use the header argument to remove that.

If the header is in a row other than the first, pass the row number to header. This will skip the preceding rows.

import pandas as pd

df=pd.read\_csv("temp.csv",names=['a','b','c','d','e'],header=0) print df

Its **output** is as follows −

|  |  |
| --- | --- |
| a b c d e   1. S.No Name Age City Salary 2. 1 Tom 28 Toronto 20000 3. 2 Lee 32 HongKong 3000 4. 3 Steven 43 Bay Area 8300 5. 4 Ram 38 Hyderabad 3900 | |
| import pandas as pd |  |
| df=pd.read\_csv("temp.csv", | skiprows=2) |
| print df |  |
| Its **output** is as follows − |  |
| 2 Lee 32 HongKong | 3000 |
| 0 3 Steven 43 Bay Area | 8300 |
| 1 4 Ram 38 Hyderabad | 3900 |

#### Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

#### Installation:

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package :

python -mpip install -U matplotlib

#### Importing matplotlib:

from matplotlib import pyplot as plt

*or*

import matplotlib.pyplot as plt

**Basic plots in Matplotlib :**

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information.

#### Uses of matplotlib

* + - Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-**oriented** API for **embedding**plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.
    - MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.
    - Matplotlib supports both the categories, particularly with the following output formats:

Matplotlib is a plotting library for Python. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

Matplotlib module was first written by John D. Hunter. Since 2012, Michael Droettboom is the principal developer. Currently, Matplotlib ver. 1.5.1 is the stable version available. The package is available in binary distribution as well as in the source code form on [www.matplotlib.org.](http://www.matplotlib.org/)

Conventionally, the package is imported into the Python script by adding the following statement −

from matplotlib import pyplot as plt

Here **pyplot()** is the most important function in matplotlib library, which is used to plot 2D data. The following script plots the equation **y = 2x + 5**

Example

import numpy as np

from matplotlib import pyplot as plt x = np.arange(1,11)

y = 2 \* x + 5

plt.title("Matplotlib demo") plt.xlabel("x axis caption") plt.ylabel("y axis caption") plt.plot(x,y)

plt.show()

An ndarray object x is created from **np.arange() function** as the values on the **x axis**. The corresponding values on the **y axis** are stored in another **ndarray object y**. These values are plotted using **plot()** function of pyplot submodule of matplotlib package.

The graphical representation is displayed by **show()** function.

The above code should produce the following output −

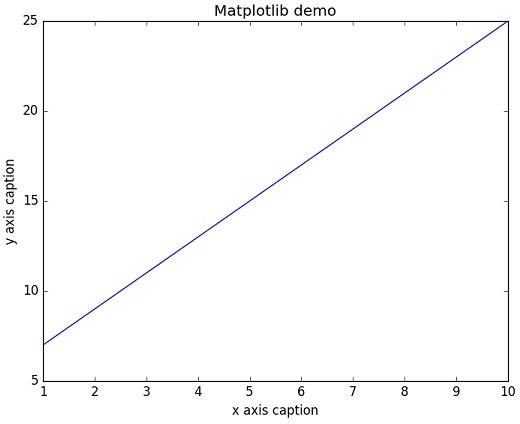


Fig 4.10 Output Diagram

Instead of the linear graph, the values can be displayed discretely by adding a format string to the **plot()** function. Following formatting characters can be used.

NumPy has a **numpy.histogram()** function that is a graphical representation of the frequency distribution of data. Rectangles of equal horizontal size corresponding to class interval called **bin** and **variable height** corresponding to frequency.

numpy.histogram()

The numpy.histogram() function takes the input array and bins as two parameters. The successive elements in bin array act as the boundary of each bin.

import numpy as np

a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

np.histogram(a,bins = [0,20,40,60,80,100])

hist,bins = np.histogram(a,bins = [0,20,40,60,80,100]) print hist

print bins

It will produce the following output −

[3 4 5 2 1]

[0 20 40 60 80 100]

plt()

Matplotlib can convert this numeric representation of histogram into a graph. The **plt() function** of pyplot submodule takes the array containing the data and bin array as parameters and converts into a histogram.

from matplotlib import pyplot as plt import numpy as np

a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

plt.hist(a, bins = [0,20,40,60,80,100]) plt.title("histogram")

plt.show()

It should produce the following output −

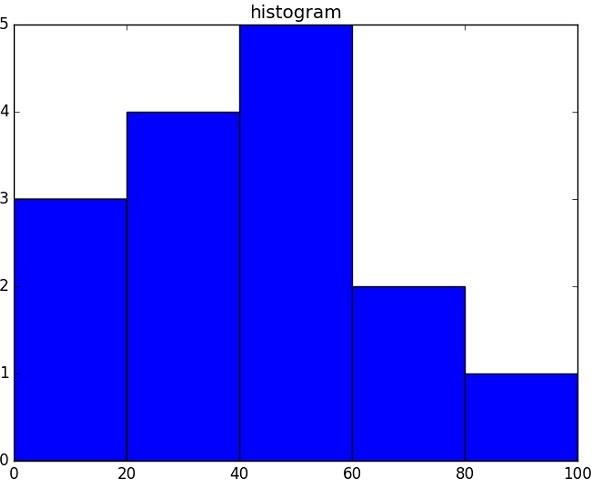


Fig 4.11 Histogram Figure

# TESTING & IMPLEMENTATION

## SOFTWARE TESTING INTRODUCTION

Software testing is a process used to help identify the correctness, completeness and quality of developed computer software. Software testing is the process used to measure the quality of developed software .Testing is the process of executing a program with the intent of finding errors. Software testing is often referred to as verification & validation

### Explanation for SDLC & STLC

**SDLC**: The software development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

### PHASES OF SOFTWARE DEVELOPMENT

* + - Requirement Analysis
    - Software design
    - Development or Coding
    - Testing
    - Maintenance

### Requirement analysis

The requirements of a desired software product are extracted. Based the business scenario the SRS (Software Requirement Specification) document is prepared in this phase.

Design

Plans are laid out concerning the physical construction, hardware, operating systems, programming, communications, and security issues for the software. Design phase is concerned with making sure the software system will meet the requirements of the product. There are 2 stages in design,

HLD – High Level Design LLD – Low Level Design

**HLD** – gives the architecture of the software product to be developed and is done by architects and senior developers.

**LLD** – done by senior developers. It describes how each and every feature in the product should work and how every component should work. Here, only the design will be there and not the code.

Testing

Testing is evaluating the software to check for the user requirements. Here the software is evaluated with intent of finding defects.

Maintenance

Once the new system is up and running for a while, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times. Users of the system should be kept up- to-date concerning the latest modifications and procedures

## SDLC Models

Water fall model

It will be executing one by one of the SDLC process. The design Starts after completing the requirements analysis coding begins after design. It is a traditional model It is a sequential design process, often used in SDLC, in which the progress is seen as flowing steadily downwards ( like a waterfall ), through the different phases.

Prototype model

Developed from the sample after getting good feedback from the customer. This is the Valuable mechanism for gaining better understanding of the customer needs

Rapid application development model(RAD):

This mechanism will develop from already existing one. If The New requirement **is** matching in already existing requirement, will develop from that.

Spiral model

This mechanism is update the application version by version. All the SDLC process will update version by version**.**

V-MODELV:

V model is a process where the development and testing phases can do parallely. For every development phase there is a testing phase. Development phases are called as verification whereas testing phases are called as validation

STLC (Software Testing Life Cycle): Testing itself has many phases i.e. is called as STLC. STLC is part of SDLC

* + Test Plan
  + Test Development
  + Test Execution
  + Analyze Results
  + Defect Tracking
  + Summaries Report

TEST PLAN

It is a document which describes the testing environment, purpose, scope, objectives, test strategy, schedules, mile stones, testing tool, roles and responsibilities, risks, training, staffing and who is going to test the application, what type of tests should be performed and how it will track the defects.

TEST DEVELOPMENT

Preparing test cases, test data, Preparing test procedure, Preparing test scenario, Writing test script

TEST EXECUTION

In this phase we execute the documents those are prepared in test development phase Once executed documents will get results either pass or fail. we need to analyze the results

during this phase. DEFECT TRACKING:

Whenever we get defect on the application we need to prepare the bug report file and forwards to Test Team Lead and Dev Team. The Dev Team will fix the bug. Again we have to test the application. This cycle repeats till we get the software without defects.

## .TYPES OF TESTING:

White Box Testing Black Box Testing Grey box testing WHITEBOXTESTING

[White box testing](http://www.buzzle.com/editorials/4-10-2005-68350.asp) as the name suggests gives the internal view of the software. This type of testing is also known as structural testing or glass box testing as well, as the interest lies in what lies inside the box.

BLACK BOX TESTING

Its also called as behavioral testing. It focuses on the functional requirements of the software. Testing either functional or non functional without reference to the internal structure of the component or system is called black box testing.

GREY BOX TESTING

Grey box testing is the combination n of black box and white box testing. Intention of this testing is to find out defects related to bad design or bad implementation of the system.

# RESULT & ANALYSIS

A result is the final consequence of actions or events expressed qualitatively or quantitatively. Performance analysis is an operational analysis, is a set of basic quantitative relationship between the performance quantities. On Fetching the performance of the Machine Learning Model over given test data we found nearly 77% accuracy with increased number of datasets by which the accuracy of the model and performance can be increased to a greater level. When compared between model performance between decision tree and random forest algorithms, it is found that random forest has more accuracy than model developed based on decision tree algorithm. As Random Forest Algorithm takes the aggregate of random decision tree outputs, it has a greater chance of perfect predictions than single tree.

## 6.1 Screen Shots.

Fig 6.1 Screenshot1

Fig 6.2 Screenshot2

# CONCLUSION & FUTURE SCOPE

We designed a system to detect fraud in Credit Card transactions. This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions. As technology changes, it becomes difficult to track the behavior and pattern of fraudulent transactions. We have just detected the fraudulent activity but we have not prevented. Preventing known and unknown fraud in real time is not easy but it is feasible. The proposed architecture is basically designed to detect credit card fraud in online payments, and emphasis is made to provide a fraud prevention system to verify a transaction as fraudulent or legitimate.

For implementation purposes it is assumed that issuer and acquirer bank is connected to each other. If this system is to be implemented in real time scenario then exchange of best practices and raising consumer awareness among people can be very helpful in reducing the losses caused by fraudulent transactions. Further enhancement can be done by making this system secure with the use of certificates for both merchant and customer and as technology changes new checks can be added to understand the pattern of fraudulent transactions and to alert the respective card holders and bankers when fraud activity is identified. The dataset available on day to day processing may become outdated, it is necessary to have updated data for effective fraud behavior identification. To this extent, the incremental approach is necessary in making the system to learn from past as well as present data and capable of handling.

Fraudster uses different new techniques that are instantaneously growing along with new technology makes it difficult for detection. Also the nature of access pattern may vary from one geographical location 161 to another (such as urban and rural areas) that may result in a false positive detection. In such a case a future enhancement may be based on new multiple models with varying access pattern needs attention to improve the effectiveness. Privacy preserving techniques applied in distributed environment resolves the security related issues preventing private data access.

# 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.

**APPENDIX**

#### Hardware Requirements

* + System Processor : Core i3 / i5
  + Hard Disk : 500 GB.
  + Ram : 4 GB.
* *Any desktop / Laptop system with above configuration or higher level.*

#### Software Requirements

* + **Operating system** : Windows 8 / 10
  + **Programming Language** : Python
  + **Framework** : Anaconda
  + **IDE** : Jupyter Notebook
  + **DL Libraries** : Numpy, Pandas