**TABLE OF CONTENT**

**Title Page Number**

**CHAPTER 1:- INTRODUCTION 1-3**

* 1. **Abstract**
  2. **Problem statement**
  3. **Need for the new system**
  4. **Objective**
  5. **Application**

**CHAPTER 2:- LITERATURE SURVEY 4-8**

* 1. **Work done by others**
  2. **Benefits**
  3. **Proposed solution**
  4. **Technology used**
     1. **Software requirement**
     2. **Hardware requirement**

**CHAPTER 3:- SOFTWARE ENGINEERING APPROACH 9-15**

* 1. **Software engineering paradigm applied**
     1. **Description**
     2. **Advantage and Disadvantage**
     3. **Reasons for use**
  2. **Requirement Analysis**
     1. **Software Requirement Specification**
     2. **Glossary**
     3. **Supplementary Specification**
     4. **Use Case Model**

**CHAPTER 4:- DESIGN 16-33**

* 1. **Design concept**
  2. **Design Techniques**
  3. **Modeling**
     1. **ER Model**
     2. **DFD Model Level-0 and 1**
  4. **Activity Diagram**
  5. **Class Diagram**
  6. **Object Diagram**
  7. **Sequence Diagram**
  8. **Collaboration Diagram**

**CHAPTER 5:- IMPLEMENTATION 34-37**

* 1. **Screenshots**
  2. **Coding**

**CHAPTER 6:- TESTING OBJECTIVE 38-41**

**CHAPTER 7:- LIMITATION OF THE PROJECT 42**

**CHAPTER 8:- FUTURE ENHANCEMENTS 43**

**CHAPTER 9:- CONCLUSION 44**

**CHAPTER 10:- REFERENCES 45**

**CHAPTER-1**

**INTRODUCTION**

**1.1 Abstract**

There is an increasing amount of false news and information floating about on social media. Twitter is one such micro blogging site where users can tweet about any topic within the given 280 characters limit. These tweets can be *retreated* by the followers of the user. Unfortunately, the users not only get breaking news and essential advice from Twitter, but they may also be subjected to spam’s and rumors. We present a credibility analysis system for tweet credibility. Our solution is based on first identifying extracted tweets as facts or opinions and then determining their credibility according to the supervised training model.

**1.2** **Problem Statement**

Twitter is a big platform for sharing valuable information about what is going on in society. People like to spread their knowledge via twitter. This is the good side of Twitter but as we know there are two sides of coin and Twitter has the same. Twitter plays role in spreading rumors and fake news and this is a major source of concern. Misinformation and disinformation in social media, and particularly in Twitter, has been observed during major events that include the 2010 earthquake in Chile, the Hurricane Sandy in 2012 and the Boston Marathon blasts in 2013. Fake news or rumors spread quickly on Twitter and this can adversely affect thousands of people. Not only Twitter, many social sites are plagued with misinformation and rumours that spread rapidly like wildfire.

     The objective of this project is to model a system to automatically assess the credibility of the content of tweets posted on Twitter. In future, though the trained model can be used to assess credibility of contents on other sites as well.

**1.3** **Need for the new System**

The current state of the art in content credibility is the Google chrome extension “TweetCred”. That was the inspiration for our system. We realized that the current system, though greatly automated, lacked the understanding of the user’s sentiments. It lacked to identify whether the user was stating a fact (making a claim), or simply stating his/her opinion.

For instance, a user may simply be stating that she likes mangoes. Giving such a sentiment a credibility rating is what makes the current system not very effective.

This creates the need for the new system wherein the tweets deemed to be stating some form of fact or claim are the ones that are being judged for their credibility.

The other existing systems are simply way too specific to be really applicable. They are unable to be operated in the real world for their lack of flexibility. In our system, with just a few tweaks, our machine can learn to give credibility rating on all databases. This gives us an edge over the other systems.

**1.4** **Objective**

Objective of this project is to develop a system that will be able to tell the user about the content of tweet that the tweet is credible or not with all the chart and graph comparisons that are drawn.

**1.5** **Application**

1.    This will be applicable on Twitter for checking the credibility of Tweets

2.    Applicable for facebook and other social media.

**CHAPTER-2**

**LITERATURE SURVEY**

**2.1 Work done by other**

After a thorough study, we have realised that there is only one software that provides the credibility assessment. It is based on ranking techniques to assess credibility of content posted on Twitter in real-time. It is provided to the users in the form of a chrome extension. TweetCred uses only the data available on each message, without assuming extensive historical or complete data for a user or an event.

Other works are based on off-line classification of content in a post-hoc setting where the model is trained and suited only for a specific domain or an event and does not work for the rest.

•    http://twitdigest.iiitd.edu.in/TweetCred/

**2.2 Benefits**

The existing system, though being an amazing one, still has its drawbacks.\

**•    Flexible**

The other many works have the drawbacks of being too specific. They are too focused on a specific domain of problems such that they are unusable in other scenarios. They become dependent on one event only.

Our Model has been trained in such a way that it is extremely fluid. Any dataset can be taken as Input and it will give the accurate results with just a little tweaking in the system.

**•    Accuracy**

The accuracy of our system is also exceptionally high. The model is precise and reliable and maintains this accuracy rate for any many various datasets.

The first model for claim finding has the accuracy of 94%

And the second one for credibility assessment has an accuracy of 82%.

•    **Features**

Despite the TweetCred being the only real-time credibility assessment system, it still has its problems. One of which being the lack of features. The tweets that we extracted from the Twitter's streaming API have an extensible set of features for each tweet. Each tweet has 653 features which are cleaned and wrangled and then fed into the system.

•    **Analysis**

Our system is more for analysis and research purposes than just for the user's enjoyment. As a result, the output of our system consists of a myriad number of graphs and charts. These graphical representations help our more analysis oriented users to get a deeper understanding of the data that has been collected.

•    **Sentiment**

The previous works on this topic lack the understanding of the perspective of the tweet author. They lack to understand whether the user actually is stating a claim or news, or is just stating an opinion. Our system identifies the facts and gives the credibility analysis on only facts claimed by tweets and not on all.

**2.3 Proposed Solution**

Our solution builds up on the number of existing models. It combines the approach taken by the different systems and gives us an application with high accuracy and versatility.

The various drawbacks of the existing systems, as stated above were lack of features, lack of analysing data, too specific, not for general dataset, etc. These drawbacks have been overcome in our system.

In our system, the aim is to develop a model for classifying tweets by their credibility. We adopt a supervised Multi-class classification approach. First, we perform feature extraction from the tweets. Second, we compare the speed and accuracy of different machine learning schemes, using the training labels obtained in the previous section.

**Feature Extraction**

Generating feature vectors from the tweets is a key step that impacts the accuracy of any statistical model built from this data. A tweet as downloaded from Twitter's API contains a series of fields in addition to the text of the message. For instance, it includes meta-data such as posting date, and information about its author at the time of posting (e.g. his/her number of followers). In total, we used 653 features.

**Training**

Then the training started. We tested and evaluated multiple algorithms to classify tweets by credibility. We experimented with various methods :

\* Random Forest

\* Neural Network

\* Logistic Regression

\* SVM

We finally used the one with the highest accuracy percentage and completed the classification of tweets based on their credibility.

**Implementation**

After the model was built, we created the output of our training datasets. It includes many graphs and pie-charts that provides the user with an arsenal of insights into the content on Twitter.

**2.4 Technology used**

2.4.1 Software requirement

* OS: Windows 7, Windows Vista or higher
* Twitter Streaming API
* Jupyter Notebook: - Jupyter Notebook provides with an easy-to-use, interactive data science environment across many programming languages that doesn’t only work as an IDE, but also as a presentation or education tool. It’s perfect for those who are just starting out with data science!
* Features: The Jupyter Notebook supports markdowns, allowing to add HTML components from images to videos. Thanks to Jupyter, you can easily see and edit your code in order to create compelling presentations. For instance, you can use data visualization libraries like Matplotlib and Seaborn and show your graphs in the same document where your code is. Besides all of this, you can export your final work to PDF and HTML files, or you can just export it as a .py file. In addition, you can also create blogs and presentations from your notebooks.
* Microsoft Excel:- Excel is a commercial spreadsheet application produced and distributed by Microsoft for Microsoft Windows and Mac OS. It features the ability to perform basic calculations, use graphing tools, create pivot tables and create macros. Excel has the same basic features as all spreadsheet applications, which use a collection of cells arranged into rows and columns to organize and manipulate data. They can also display data as charts, histograms and line graphs. We have used Microsoft Excel to view and edit the csv file.
* Notepad:- Notepad is a generic text editor included with all versions of Microsoft Windows that allows you to create, open, and read plaintext files. We have used notepad to save the Tweets came from twitter.

2.4.2 Hardware requirement

USER

* RAM: minimum 512mb
* MEMORY: minimum 50mb

DEVELOPER

* Microsoft Windows 7/8/10 (32- or64-bit)
* GB RAM minimum, 8 GB RAM recommended
* GB of available disk space minimum,
* GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution

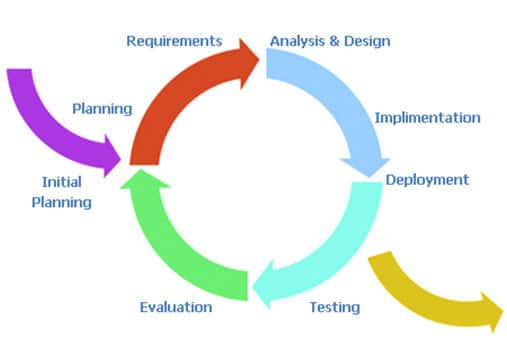
**CHAPTER-3**

**SOFTWARE ENGINEERING APPROACH**

**3.1 Software Engineering Paradigm Applied**

3.1.1 Description

In this project to develop a system we are using Incremental Process Model. Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Each iteration passes through the **requirements, design, coding and testing phases**. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.



*Fig: Basic steps of Software Development*

The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilisation of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

[](https://www.guru99.com/images/6-2015/052615_1049_WhatisIncre2.png)

*Fig: Incremental model*

3.1.2 Advantage & Disadvantage

**Advantages**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than waterfall.

3.1.3 Reasons for use

* Incremental model is used because we are dividing the task into modules.
* We can test our system in each iteration when the module get completed.
* Error detection is easy in this model.
* This model can be used when the requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some details can evolve with time.
* There is a need to get a product to the market early.
* A new technology is being used
* Resources with needed skill set are not available
* There are some high risk features and goals.

**3.2 Requirement Analysis**

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore requirements analysis means to analyze, document, validate and manage software or system requirements. High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

**3.2.1 Software Requirement Specification**

* **Feasibility Study**

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological, legal and scheduling factors. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it.

* **Technical Feasibility**

Our application is technically feasible because this application do not require any extra software’s. this application is easy for modification, we can add extra feature to this application without any risk.

* **Economical Feasibility**

The android application will be built economically. Because Twitter is available free of cost for everyone so it is economical for developer’s end as well as it doesn’t require any additional cost and any other requirement for making the application. The cost of resources will be less. Our application also economically feasible to users because our users do not has to pay for the access and for use of our application. So they can easily access our application and use almost all the features of application. So the application is also economical feasible from the user’s end.

* **Operational Feasibility**

The operational feasibility will comprise of the features that are included in the proposed system. The main aim of our application is to make easier for users to get the knowledge of rumours what they are listening is real or not.

**3.2.2 Glossary**

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure. The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

**3.2.3 Supplementary Specifications**

* **A twitter Account**: The supplementary specification for our project is to for the client to have a twitter account to be able to use  the application extensively.
* **Analyst:** The user must have a basic understanding of pie-charts and graphs that are the output of our application.

**3.2.4 Use Case Model**

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. Use cases specify the expected behaviour, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation (such as UML). A key concept of use case modelling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.

A use case diagram is usually simple. It does not show the detail of the use cases:

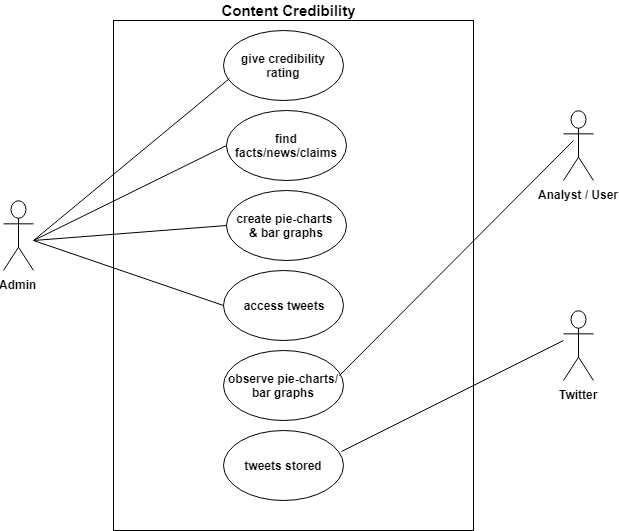
• It only summarizes some of the relationships between use cases, actors, and systems.

• It does not show the order in which steps are performed to achieve the goals of each use case. As said, a use case diagram should be simple and contains only a few shapes. If yours contain more than 20 use cases, you are probably mis-using use case diagram.

The figure below shows the UML diagram hierarchy and the positioning of UML Use Case Diagram. As you can see, use case diagrams belong to the family of behavioural diagrams.

• There are many different UML diagrams that serve different purposes (as you can see from the UML diagram tree above). You can describe those details in other UML diagram types and documents, and have them be linked from use cases.

• Use cases represent only functional requirements of a system. Other requirements such as business rules, quality of service requirements, and implementation constraints must be represented separately, again, with other UML diagrams.

****

*Fig: Use case model*

**CHAPTER-4**

**DESIGN**

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer‘s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

**4.1 Design Concept:**

The set of fundamental software design concepts are as follows:

**1. Abstraction**

• A solution is stated in large terms using the language of the problem environment at the highest level abstraction.

• The lower level of abstraction provides a more detail description of the solution.

• A sequence of instruction that contain a specific and limited function refers in a procedural abstraction.

• A collection of data that describes a data object is a data abstraction.

**2. Architecture**

• The complete structure of the software is known as software architecture.

• Structure provides conceptual integrity for a system in a number of ways.

• The architecture is the structure of program modules where they interact with each other in a specialized way.

• The components use the structure of data.

• The aim of the software design is to obtain an architectural framework of a system.

• The more detailed design activities are conducted from the framework.

**3. Patterns**

A design pattern describes a design structure and that structure solves a particular design problem in a specified content.

**4. Modularity**

• A software is separately divided into name and addressable components. Sometime they are called as modules which integrate to satisfy the problem requirements.

• Modularity is the single attribute of a software that permits a program to be managed easily.

**5. Information hiding**

Modules must be specified and designed so that the information like algorithm and data presented in a module is not accessible for other modules not requiring that information.

**6. Functional independence**

• The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.

• The functional independence is accessed using two criteria i.e Cohesion and coupling.

Cohesion

• Cohesion is an extension of the information hiding concept.

• A cohesive module performs a single task and it requires a small interaction with the other components in other parts of the program.

**7. Refinement**

• Refinement is a top-down design approach.

• It is a process of elaboration.

• A program is established for refining levels of procedural details.

• A hierarchy is established by decomposing a statement of  function in a stepwise manner till the programming language statement are reached.

**8. Refactoring**

• It is a reorganization technique which simplifies the design of components without changing its function behaviour.

• Refactoring is the process of changing the software system in a way that it does not change the external behaviour of the code still improves its internal structure.

**9. Design classes**

• The model of software is defined as a set of design classes.

• Every class describes the elements of problem domain and that focus on features of the problem which are user visible

**4.2 Design Techniques:**

The most creative and challenging face of the system development is System Design. It provides the understanding and procedural details necessary for the logical and physical stages of development. In designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfil. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing.

Design of the system can be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Thus system design is a solution to “how to” approach to the creation of a new system. This important phase provides the understanding and the procedural details necessary for implementing the system recommended in the feasibility study. The design step provides a data design, architectural design, and a procedural design.

For Example, In the Student Section we can consider student registration Students will register for the site in first semester and update the account in B.E. first semester. Online updating facility will be provided after registration. After registration record will get verified by administrator. We can also consider the departments for identifying the student in which department. Also we need to consider the additional fields like Training, Project done, Hobbies, Extracurricular activities, Technologies of by known student.

**4.3 Modelling:**

It is very important to distinguish between the UML models. Different diagrams are used for different types of UML modelling. There are three important types of UML modelling.

Structural Modelling

Structural modelling captures the static features of a system. They consist of the following −

• Classes diagrams

• Objects diagrams

• Deployment diagrams

• Package diagrams

• Composite structure diagram

• Component diagram

Structural model represents the framework for the system and this framework is the place where all other components exist. Hence, the class diagram, component diagram and deployment diagrams are part of structural modelling. They all represent the elements and the mechanism to assemble them.

The structural model never describes the dynamic behaviour of the system. Class diagram is the most widely used structural diagram.

**Behavioural Modelling**

Behavioural model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioural modelling shows the dynamic nature of the system. They consist of the following −

• Activity diagrams

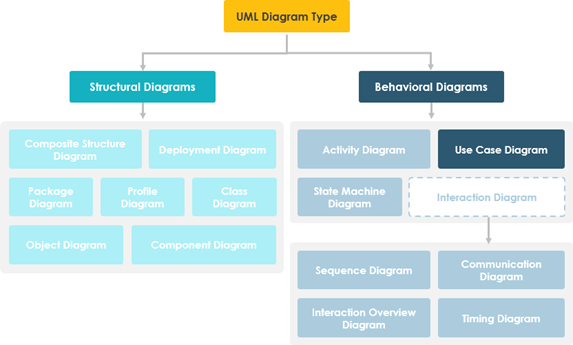
• Interaction diagrams

• Use case diagrams

All the above show the dynamic sequence of flow in a system.

**Architectural Modelling**

Architectural model represents the overall framework of the system. It contains both structural and behavioural elements of the system. Architectural model can be defined as the blueprint of the entire system. Package diagram comes under architectural modelling



*Fig: Architectural Modelling*

**4.3.1 ER Model**

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.

ER modelling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modelling before implementing your database.

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

## Components of the ER Diagram

This model is based on three basic concepts:

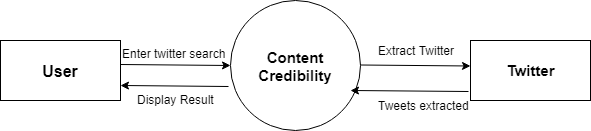
* Entities
* Attributes
* Relationships

**4.3.2 DFD Model**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

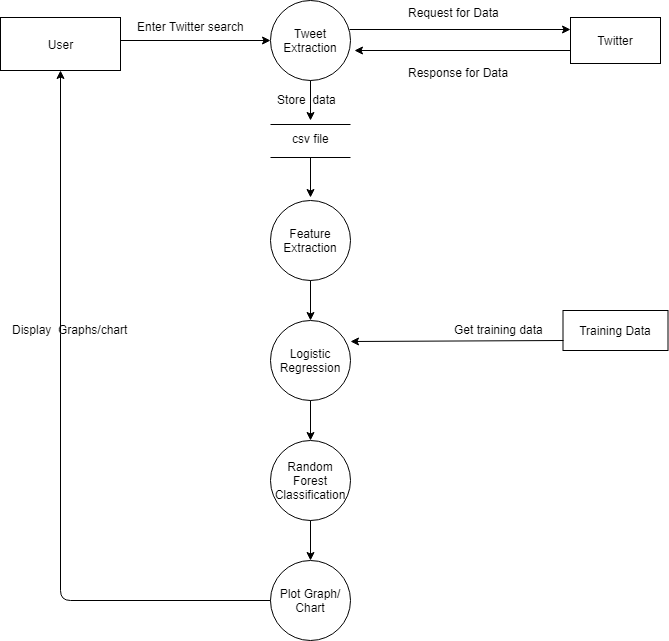
|  |  |
| --- | --- |
| **Process Notations** | https://lh6.googleusercontent.com/KvKTSHFJeCLfLe6u9vRskG0ZgZZwTS2Bv3zgvE6WEqXSwAKVVl1mx5elfuhkFFiSGJ9_PujHEUCeGVjmn38xoBCcTo5HJc3KG0OYp2EA6dc0E1Vlsiu4gPzB_aBa-yqyk5T2AtAx |
| **Datastore Notations** | https://lh5.googleusercontent.com/qsGyHos9fVgQYcpkBOi3Mehn7nM5SBd5ovLQ94aNK5bwcPHEmgg36wB3XWZ-FfxvOC_2usxLmcaYpQwK9RL-pQN-tVIOGXmmHh1YktVy7nEj1CvugQkMIETrArWd7n-jZwE-52ap |
| **Dataflow Notations** | https://lh5.googleusercontent.com/1B_MYUpPtoiVbME2sgzFRBsb_Ily1Edjp36nl9VlWLyidGm5qub7lfU8eFEd2KxQVL1PN1lGtz9MaV40W6J-HMgeMT1lqLrRD1gv2NGHRgCJBeIRB9TeFm8G4_qMsum-IKR2ZUQ4 |
| **External Entity Notations** | https://lh5.googleusercontent.com/J-bmwU5Z7ooZl2AiINQ32tF-1J4AH85kCgb87kc7m6M6pOuyvk61ElZvchc55S9yk7HmwX0dmkvL5DzatvPmLTAStoRH9NoH5ED3MmbkyAsQtUXj99s9xZ5gWz7yKL9Lh8sY0UI- |

1. **DFD level - 0**



*Fig: DFD level 0*

1. **DFD -1**



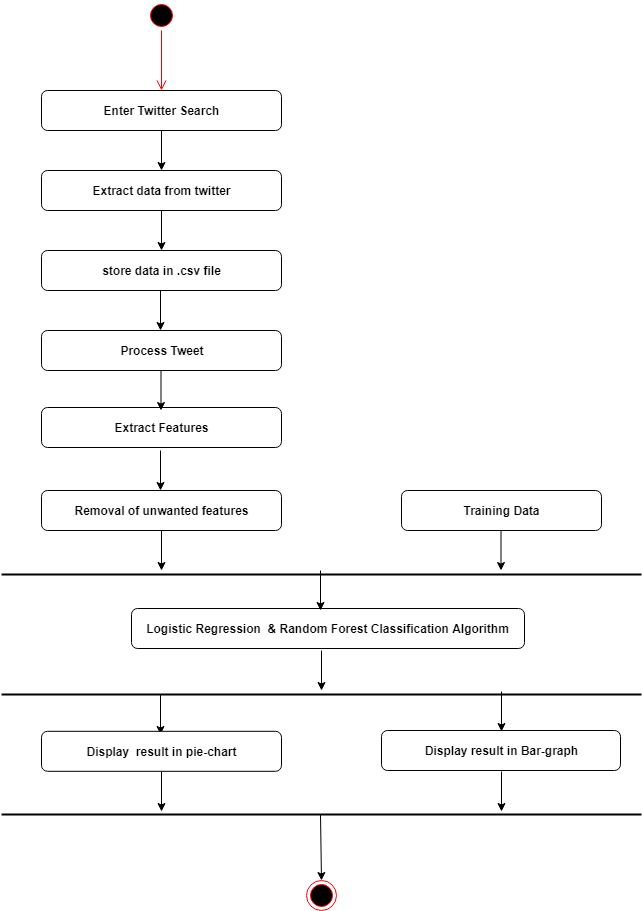
*Fig: DFD level 1*

**4.4 Activity Diagram**

We use **Activity Diagrams** to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.

|  |  |
| --- | --- |
| **Initial State** | https://lh6.googleusercontent.com/1sZUfNhwixyc99mHb1CggcRruIfBJ9w1NRWR3MFaIF-JbA6epPqG3q6CoYKP3ltIAM3xbdTlOuO6Cv1XsfCbeV2vF8XV4RAIkRuEMyXcFHe4PfvoMHd6zRWp8f7mOiUhVVP7vgrk |
| **Action or Activity State** | https://lh4.googleusercontent.com/n4h3ZYEHCB6vhT7RclufIGw5zyz1bxNx0O_6_vGfb56gKWciXMxkBnWgwENicVI3jVSHNbd4wHVoh-V3jMiiM3xdSiZCjSffiGv_gC_eIE3fZcdFJFZkEBXE02L2vuYETNhc7fHz |
| **Action Flow or Control flows** | https://lh3.googleusercontent.com/3UEJZKK8WV5dar2GTiXnWgeS1BbwoygXpEBGc_amlHI2OR75m0AKGQi9PRtJIKXq20tdR-3GqbbTvNOWyaEhSgQ5Jo5rs-NGLxgn7nvcTKOqj91m5abZUGMuOj4DNqlrirAFrjz4 |
| **Decision node and Branching** | https://lh6.googleusercontent.com/uzq2cGw7x2ktCbAgEggdWdhj3J3x4vfG1y-ndVr-lJVIFmhfENBmdvUgCuy5S7U4gwemY0DIsZ41HKGCPiX5TMcYdQpXlCKswz4dt2mKJYGh0a7iNWYI86h-BAvUc351M5or60VS |
| **Fork** | https://lh3.googleusercontent.com/VH3K23K-h-Nd2hXdmjJsfJIGD371KlnYYSL6ZMhbAP3RJ_UrLQxY6VAra1xP-YWXJTk8lMJ52mziZPW-vUhQYHJ1BdD5SRDSdQfq584CZ8dFbGqxHK-SeG6wvjukhxeurZyQK9mt |
| **Join** | https://lh3.googleusercontent.com/Iy9h2EqSZYqjRtN-xEhK7-wPw7XqbbQZFEBuDbhXCdugD81SdgL4IDZReOPVpMGK0bflGNSZdr4Egmbk9XR9jISZSuHgeEvQVwrWB5-0AGS64AtrkTTrCp6sve35YAT5cyhWg_uj |
| **Merge or Merge Event** | https://lh3.googleusercontent.com/WMDctpp_By3zXWRkGmcltV4UfJqWxVjvJL-hfxD_aoSfYym8vnHrNjpghsM8KuMIFMCYMqGfP2_LqYVhDHpLnMTZ1begu1nGW0gRcSbBNeYbe2ucx3o7x7UhDDARKqh5vODMOXWq |
| **Final State or End State** | https://lh5.googleusercontent.com/CEPP4NUes_IXkI79S_Dx1RfRBbPdPyM_IUPgnK0ye3-P1feBfUZKmaHnhTwDsitGM5EiEc8b-owcxHOraf7-ti7JHH4cOVuDCroeGc9xNxedtuG3BTDLrGySLdwh4QZpBUe4sdcw |



**Logistic Regression and Neural Network Classification algorithm**

*Fig: Activity Diagram*

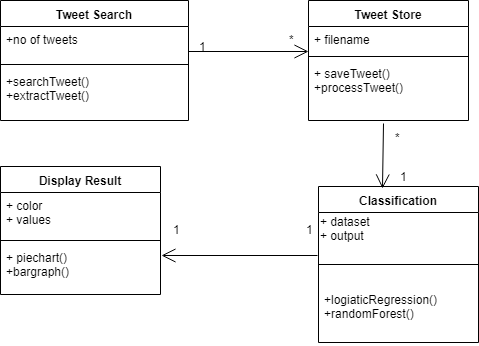
**4.5 Class Diagram**

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.

The main purpose to use class diagrams are:

* This is the only UML which can appropriately depict various aspects of OOPs concept.
* Proper design and analysis of application can be faster and efficient.
* It is base for deployment and component diagram.

|  |  |
| --- | --- |
| Class Name | https://lh6.googleusercontent.com/2lKfioa8FGCHy1tXPmcPDzqncmm45oCbfbxpYWm-NUgkfVtRjDXybONyoEr9YqneHeftUiNae7FsVK8CPCOjhbuvCnscoMua3-tsQzPwvc-y1L5_Q6fQ1dd8C-1G4lip30ldbncF |
| Class Attributes |
| Class Operations |
| Inheritance (or Generalization) | https://lh6.googleusercontent.com/9bi2lg1WHQTEZXa2pSccnxpm2puyyGb-mimhAghn8nforim7ZtE5w7xghgQEqxfOPO00lRLEPt6GQV2QcXpqUPnhbWDxjtaUbMiAQOpCZgC6tTizbFLM2TLbL7VbJQGCR41T4ZTH |
| Simple Association | https://lh3.googleusercontent.com/-LY_gINC6WCx1yktLxTT9kI4SdiNJa2ADtekTTfKdRNrWMd0ptun5qZsGbOrry5y65A4CiiazPLjJqQD_cCPPoNDro8IieBi6XpN3aoyVoVRG2f7VHsyB2-Vx5IYRxComW7Wvhwa |
| Aggregation | https://lh4.googleusercontent.com/ROJHnbnYzQiZcQ24Ts4HAjmWChwVULGNLUaxaC4or_lfk0FvIWhgoVrjpDbmjID6t6YgRiEx7FA-kvkeslgw6U3aoK69LJ5pO4MtGX9vh3-VGIkASt7nVAjrecA5BghePSCIjJ8z |
| Composition | https://lh4.googleusercontent.com/QuGLtbteDf_48sGC2gqD37GW06uDgAC7jRkRT65j2bTaQjKMLnfk_v69PHH_Y42cOdaOIcUigtEktBGgBAn-zlc5xSQDEH-4qkJkuzRztU8SsO1IVZXm2MFlzajS1ruMibObPiCp |
| Dependency | https://lh5.googleusercontent.com/paO82ZnjDbXVcKSm5V1SnSmkdghRm_nosvWBRx17hQFP9ODPpx7l0ONSc7O9gOAhqSYiWv0ibdLPqLbNW08mHMpL2Bhh_xT0pE5qVF9HFYgBOtsSo9iEKynzQOBxaI10y-x01zA9 |



+logisticRegression() +NeuralNetwork()

*Fig: Class Diagram*

**4.6 Object Diagram**

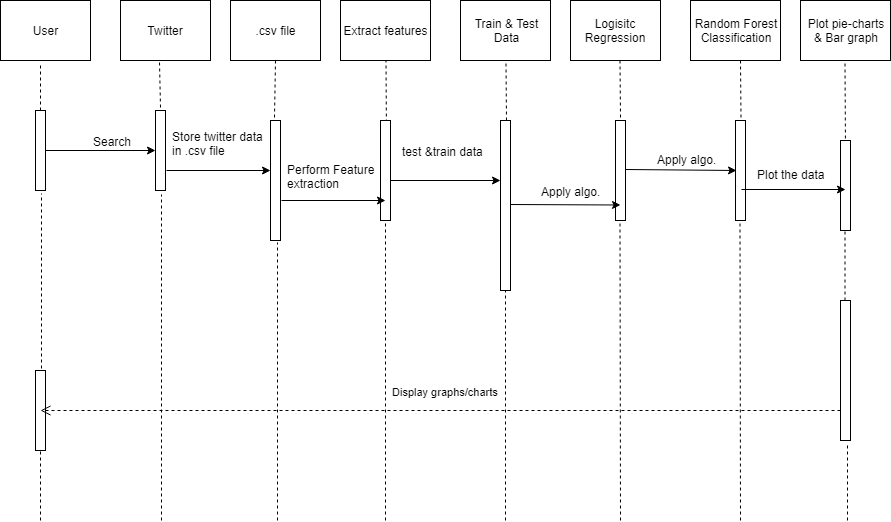
An **Object Diagram** can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. Object diagrams are vital to portray and understand functional requirements of a system. In other words, “An object diagram in the Unified Modelling Language (UML), is a diagram that shows a **complete or partial view** of the structure of a modelled system **at a specific time.**”

|  |  |
| --- | --- |
| **Objects or Instance specifications** | https://lh4.googleusercontent.com/8bmTX2-WphE3XJbChydlFG4mVwVdT3OSWuKcMEq8XSaA5bPcJS6AgNoGPjUTBG605VTTQI0ijM1kbxbElCFjGpH0FE1xEqpH3FSCtRqnsntOo606m8VSNOuFVmZkmak-RMCkSeOJ |
| **Links** | https://lh3.googleusercontent.com/8crDE3St5oV_9esHUb6XMRgFGVZXHUdICGAS82ssqlyAs38mGDSocxfsx1ec9p-NFCAvD8vFG-hZQGxt2OHfdUvnDx59tDi60eeVOz2XNg20n9Dk13enhnGeVKwAoa1FoY7LEozA |
| **Dependency Relationships** | https://lh5.googleusercontent.com/FOfPW3M4-A9liElMLXEBT9Iefhh5O2-S7eqXVAG1LBY5ruoBRh9uU3G_h8oeEGp-Ed_JsaspxK4roFWLZvDk_nAzSi9PhrG8gWVY6xT_kVaK18mDXqexrgOlJD2hUIrpjk6TBWHZ |
| **Association** | https://lh3.googleusercontent.com/6yKvQTQ3qQkLpcZo8qI66cc7aGrd-cVaxo4Vi9ztYuwCEWulwYw42YxptjjZ0iRgu6N6Lg4NnR9fajQxyY9zse8FqIAkCnWwQjcph3oAThZWOUcAmy-iyENF9oKtQ0tHRi0ibiqw |
| **Aggregation** | https://lh5.googleusercontent.com/QymWIGJZB5aBNwsswtaB5pULjhaAnkrxEDVnYFqJq6kWNX1CoFTwJCBLAuMsnA9fAvsmlWPzjeqI1sAb4UdgHszucKVg2LUaQVHlw56SKvpKK-8duzti3Tj90xQMvR-kClgqpWim |
| **Composition** | https://lh4.googleusercontent.com/REPDU2wBAhGKhQgPmo7L-3eBQksR9DRSG8awWaTTbr3FM7GIq0yPc7cdukISHUS3jGzr9jOyF5fv83CCh_6dUJJP0thalvr0eGAtKb4SKGGVAVXfns6NOR5-xfoS2n29Pz7NyYuV |

**4.7 Sequence Diagram**

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

|  |  |
| --- | --- |
| Synchronous Message | https://lh3.googleusercontent.com/5SkKw3MbmHMNlgzthpA3QwhMdemwKk4VqUcI1NOggYArf6zm-lhIPWSp0f9SGAzYsMDTuiUquXlGw7cPFHgv13nvHJoMLftf1wpFmV7URDfzTlwRzfc2Ggh5k7hcWFRA_UKxGdmV |
| Asynchronous Message | https://lh5.googleusercontent.com/MgL4LVNyJxPqdQyipxWlfciPKgGeZFDRO3ZloJMbD96ykHuBBk00r4ChJ0L856dpyCA0Fv9kQ2ZKwQ5ZUyLrmaTpWdoSpCJ36iby0kTGteb9miZhayAqb68AX03xdRJ98iVXPscA |
| Reply or Return Message | https://lh3.googleusercontent.com/oof9DMXyXen4MFlKnM6evpLpNTZjPtQfVKDEwrno46oSHuDIRkMqEMom4QAa77BNG9pR4_LlBC8in5XcE2BoQ1A9hQBKDzaraUoSCOUQdbxni0WgNYnnnAxdSI8oHEOkbYDdfVxF |
| Self Message | https://lh5.googleusercontent.com/em-jq8mv1E2runqz13q3Vvy2TR6fLFwLvZfW_071LXr-srtKJUXYYrKSM2EcYlDiBRBtHZpUcdOGhsr5S78bvbNjXfQJmTQf_M3QUC0vqxj7UPkXz8CkJVMC5gD8Zbji2lcyMQjB |
| Create Message | https://lh5.googleusercontent.com/AyeOJktQL-giknUu65bscFbgnsda27Fiv8s-pPp5hxUS1SamBCwA_l-fgXrVsA_BohY4I6eZ0fspQdu0bu_gFmFtg5bB1crrfvgBnX4mkF2q_TffCN2WiAUmYxTcCiKVpudW6y9Q |
| Delete Message | https://lh4.googleusercontent.com/oOGJWNKwhYjye4rZcG_J_cBB75YJJKp4g2jueoBMC3Xo-5HG2i_qGm-1iTp9hPO8P9vg3PRprrXiHqCL6S_d8nbHQpWVulOirYmT9lhBl53dZHibhhXavBxIAjPD3_F1iWh9K5Go |
| Found Message | https://lh3.googleusercontent.com/OsMcrCH4sllAkbdzO9_4FdGaUbWCmo3tTh2P07pYcQUe31UQmc1_gOg4mzYjBnM8sZU4opuhQqKlIW7CEOTDeEngn0Tz1mULYXZwVRGZAuwyvndY_NpxKbgUCLGQkXFIQBBwfaFR |
| Lost Message | https://lh3.googleusercontent.com/jZdJAlOImaaKDOktpMPMGkYzL_lBZrUoak2dHyzHep_t20ab5aggsyxiUX6kdwFG7QGRQ0YSPP6GCF7gkKnGaDdD4b0y-WTPwXwJWp9d_kGwis2QjxscnbYRpVOjT3-2038sVEyX |



**Neural Network**

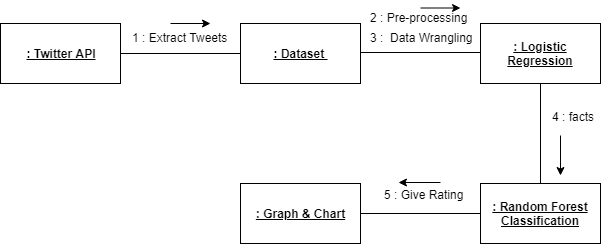
*Fig: Sequence Diagram*

**4.8 Collaboration Diagram**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). The concept is more than a decade old although it has been refined as modelling paradigms have evolved.

A collaboration diagram resembles a flowchartthat portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

|  |  |
| --- | --- |
| **Objects** | https://lh6.googleusercontent.com/AJ0x_9ICnoz4hCJhdFarIUPo1w1gU9ZlZZA7P-QJGqAEdhvoQmm01ENW7ZDv_j3GlGECo6qTve2ZYYRFAPMvji71pZh123Lc-O1Fw90g3HHaklLbWcA8pO8vRBTVwDTSwbv5YqcE |
| **Multi-object** | https://lh3.googleusercontent.com/Sxi9c86sJhOYvm2ClBmtfEpp1UkNmkU1NYJK21tTZvqlkcI8ixLW-BfndwVxY7zTSqMccB8_fKLu8kdJxVBzXnRRzxXlFz9gGD9XlvzcOiWYpHO8Hw3b-bIorXnIP8G9okfzfmLJ |
| **Association role** | https://lh5.googleusercontent.com/EjlpEwcG0OMVFz3-BYH8KjXV2PGDyA8H68jeaPKPahWuhuw0X9DzvMoUy3_JwZfzLWKUBzHn-IhVZpQ4PSnyRONDuT2b4f-22OjHEPjvh0S29EyjdeDWW8Egi1wShJAozsE36WnV |
| **Delegation** | https://lh3.googleusercontent.com/UMyaa_hpnOXxkVVwruB5W99GvBbY5h5ozNJiKr-rTQRrBDBeVikiuTkybX0FRvxUrFBVSpoMXzfWSGwl-o6Gxaay9cXzMjfqIPCd8FVN_0K6QKz0iFdok6k5PxArhqXB09Q8okkd |
| **Link to self** | https://lh4.googleusercontent.com/5JhUGbTtxU6ktcXACNq3EGy-aGGNfVEe4chEyGA3PzU-e0aKLYs34Si7PNa51PXOOVG-bS3nmIkC0o9dtHGbQxJxj4MahwKTQ74TLQ-6RsOFhsiEjr_Dkw5EQv6jmBJU0sjEjv__ |
| **Constraint** | https://lh6.googleusercontent.com/tAIpoFRO7cZxOzfIoSCKUfTvX549bmpQaJxXRBIBvxTXQhDOCeJkO_YqXZuv2fFi5tTJVVaAD6o2RYx8RroTdE1i5wM7dHMSAuXRH7CJOzGboUHgvn9yA7SFgXY4acwgA0GtBAW1 |



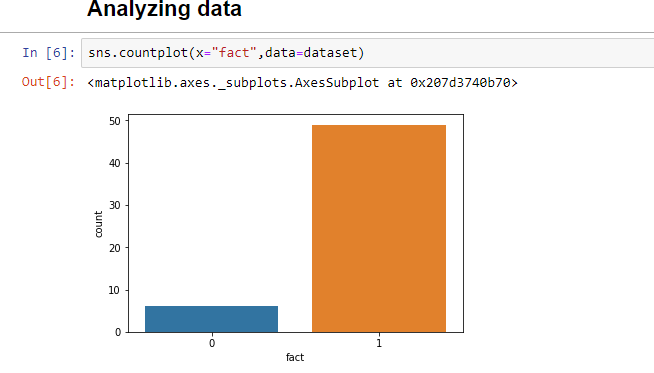
**:NeuralNetwork**

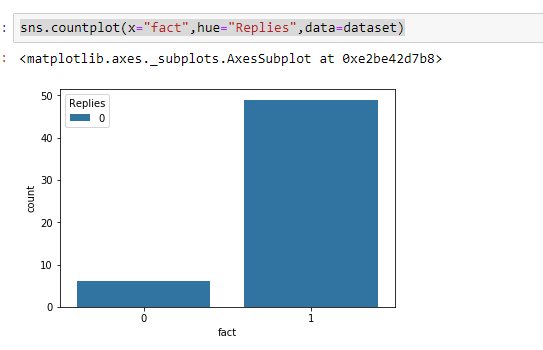
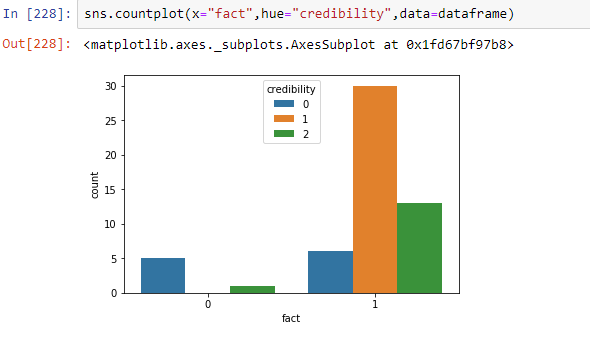
*Fig: Collaboration Diagram*

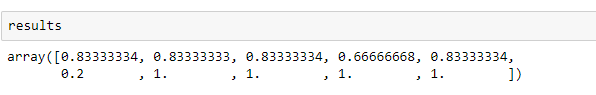
**CHAPTER-5**

**IMPLEMENTATION**

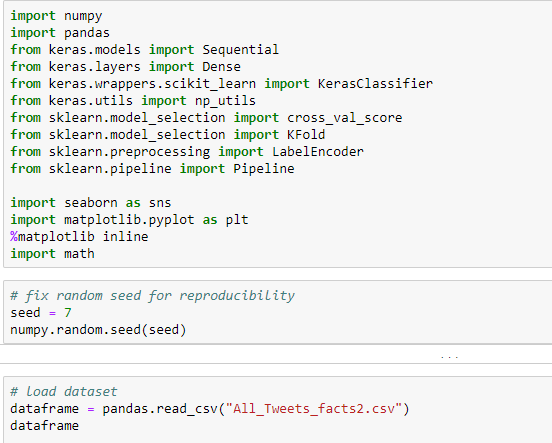
**5.1 Screenshots**



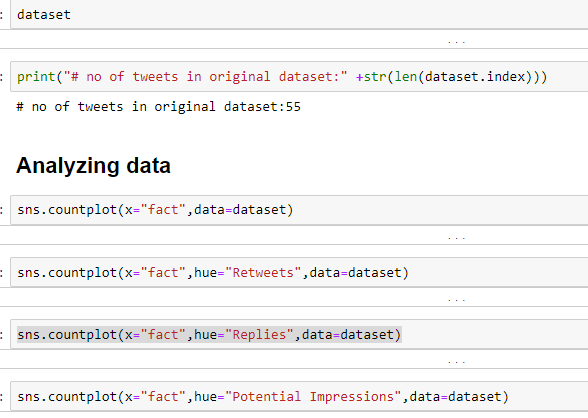




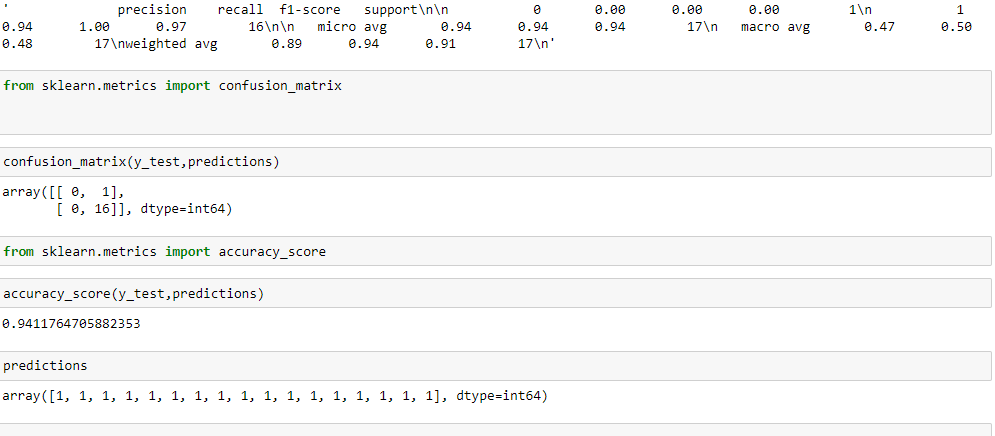
**5.2 Code**











**CHAPTER-6**

**TESTING OBJECTIVE**

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

**Types of Testing**

Testing of the online classified system was performed in three stages which are as follows:-

* Unit Testing
* Integration Testing
* System Testing

**Unit Testing:**

Unit testing is undertaken when a module has been coded and successfully reviewed.

This can be done by two methods:

**a) Black Box testing**

Test cases are designed from an examination of the input/output values only and no knowledge of designing or coding is required the following are the two main approaches of designing black-box test cases.

**b) Equivalence Class Partitioning**

The domain of input values to a program is partitioned into a set of equivalence classes. This partitioning is done on such way that the behaviour of the program is similar to every boundary value analysis. Boundary value analysis leads to selection of the test cases at the boundaries of different equivalence classes.

**Testing done by: Team Member**

In our project particularly, first we created extraction program. By running the program we conclude & tested that whether it runs properly or not. So such a way we perform the Unit Testing.

**Integration Testing:**

During integration testing different modules of the system are integrated using integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system.

Purpose:

- To test whether the module performs its intended task.

- Once all the modules have been integrated and tested, system testing can start.

In this project the Extraction modules, the data cleaning and wrangling module, and the classification modules were created independently and tested and then they were integrated together incrementally. Thus with the following way we performed Integration Testing.

**System Testing:**

System tests are designed to validate a fully developed system with a view to assuring that it meets its requirements. There are three types of system testing which are as follows:-

**Alpha Testing**:

- The initial testing of a computer program or system under actual usage conditions, it can be done in-house by the vendor, or outside by a customer or third party teaser.

- Acceptance Testing performed by the customer in a controlled environment at the developer‘s site. The software used by the customer in a setting approximating the target environment with the developer observing and recording errors and usage problems.

**Beta Testing:**

Beta Testing is done after alpha testing. The main purpose of Beta Testing are as follows:-

- Testing done by the potential or existing users, customers and end users at the external site without developers involvement is known as beta testing.

- It is operation testing i.e. it tests if the software satisfies the business or operational needs of the customers and end users.

- Beta Testing is done for external acceptance testing of COTS(Commercial off the Shelf) software.

**Type of testing approach used in the project:**

**White Box Testing**:

White-box testing is a methodology used to ensure and validate the internal framework, mechanisms, objects and components of a software application. Whitebox testing verifies code according to design specifications and uncovers application vulnerabilities.

White-box testing is also known as transparent box testing, clear box testing, structural testing and glass box testing. Glass box and clear box indicate that internal mechanisms are visible to a software engineering team.

White-box testing advantages include:

- Enables test case reusability and delivers greater stability

- Facilitates code optimization

- Facilitates finding of the locations of hidden errors in early phases of development

- Facilitates effective application testing

- Removes unnecessary lines of code

**CHAPTER-7**

**LIMITATION OF THE PROJECT**

1. This project is not real time application
2. This application is not automated.

**CHAPTER-8**

**FUTURE ENHANCEMENTS**

In the future we would like to

* Real time

We would like to make the system real time, wherein the tweets extraction and rating would all happen in real-time, live on the twitter website, perhaps as an add-on to the site

* Automated

The system could be a little more automated and less dependent on the human efforts as it is now.

* Other Applications

Instead of just being limited to twitter, we would like to expand to other applications that spread fake news as well. Like WhatsApp and Facebook.

**CHAPTER-9**

**CONCLUSIONS**

In this work, we explored the possibility of detecting credible content and text from Twitter feeds using credibility analysis. The system provides a credibility rating from low credibility high credibility for each tweet on a user’s Twitter timeline. The score is computed using a supervised algorithm, trained on human labels that determine credibility of a tweet based extracted features. All features can be computed for a single tweet, and they include the tweets content, characteristics of its author, and information about external URLs.

**CHAPTER-10**

**REFERENCES**

* <https://www.youtube.com/watch?v=VCJdg7YBbAQ>
* <https://www.youtube.com/watch?v=aircAruvnKk&t=216s>
* <https://stackoverflow.com/questions/14192741/understanding-pandas-dataframe-indexing>
* <http://adilmoujahid.com/posts/2014/07/twitter-analytics/>
* https://www.draw.io/
* A Little Bird Told Me, So I Didn't Believe It: Twitter, Credibility, and Issue Perceptions Mike Schmierbach & Anne Oeldorf-Hirsch
* TweetCred: Real-Time Credibility Assessment of Content on Twitter