**CHAPTER 3**

**DEVELOPMENT PROCESS**

* 1. **REQUIREMENT ANALYSIS**

Requirements are a feature of a system or description of something that the system is capable of doing in order to fulfil the system’s purpose. It provides the appropriate mechanism for understanding what the customer wants, analyzing the needs assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are translated into an operational system.

* + 1. **PYTHON:**

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.

For example, x=10. Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear [syntax](https://whatis.techtarget.com/definition/syntax) and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of [operating system](https://whatis.techtarget.com/definition/operating-system-OS)s, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

**Features in Python**

There are many features in Python, some of which are discussed below

* Easy to code
* Free and Open Source
* Object-Oriented Language
* GUI Programming Support
* High-Level Language
* Extensible feature
* Python is Portable language
* Python is Integrated language
* Interpreted Language
  1. **ANACONDA**

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) as well as the [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between conda and the [pip package manager](https://en.wikipedia.org/wiki/Pip_(package_manager)) is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g., the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Opensource packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

### Anaconda Navigator

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter#JupyterLab)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* QtConsole
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software))
* [Glue](https://en.wikipedia.org/wiki/Glue_(software))
* [Orange](https://en.wikipedia.org/wiki/Orange_(software))
* [RStudio](https://en.wikipedia.org/wiki/RStudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)
  + 1. **JUPYTER NOTEBOOK**

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface) (formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter [web application](https://en.wikipedia.org/wiki/Web_application), Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release[[11]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote23-11)[[12]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote20-12) (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)).

The Notebook interface was added to IPython in the 0.12 release[[14]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote012-14) (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple](https://en.wikipedia.org/wiki/Maple_(software)), [Mathematica](https://en.wikipedia.org/wiki/Mathematica), and [SageMath](https://en.wikipedia.org/wiki/SageMath), a computational interface style that originated with Mathematica in the 1980s. According to [The Atlantic](https://en.wikipedia.org/wiki/The_Atlantic), Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

* 1. **RESOURCE REQUIREMENTS:**

**SOFTWARE REQUIREMENTS**:

|  |  |
| --- | --- |
| Operating System | Windows 7or later |
| Simulation Tool | Anaconda (Jupyter notebook) |
| Documentation | Ms – Office |

**HARDWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| CPU type | Intel Pentium |
| Ram size | 4GB |
| Hard disk capacity | 80 GB |
| Keyboard type | Internet keyboard |
| Monitor type | 15 Inch colour monitor |
| CD -drive type | 52xmax |

* 1. **PROPOSED SYSTEM**
* Depression as a common mental health disorder, With the development of Internet usage, people have started to share their experiences and challenges with mental health disorders through online forums, micro-blogs or tweets.
* Their online activities inspired many researchers to introduce new forms of potential health care solutions and Our proposed methods for early depression detection systems. Using different Natural Language Processing (NLP) techniques and text classification approaches,
* The strength and effectiveness of the combined model bag of words and TF IDF are demonstrated with the NLP resulting in the top performance for depression detection reaching 91% accuracy and 0.93 F1 score.

There are a growing number of methodologies to detect depression from the posts. In our study, we incorporate a technical description of approaches applied for depression identification using the NLP and text classifying techniques. The framework in consists of data-pre-processing, feature extraction followed by the machine learning classifiers, features analysis and experimental results.

* + 1. **ADVANTAGES**
* More accuracy prediction comparing to existing model
* Increase speed
* feature extraction is more in our proposed model
  1. **SYSTEM ARCHITECTURE**



**Noisy data removing**

**Upload dataset**

**Pre-Processing**

**Server**

**EDA**

**Feature Exaction**

**Classification**

**Performance analysis Result**

**Prediction of depression**

* 1. **SYSTEM MODULES**
* Module 1: Data collection
* Module 2: Pre-Processing
* Module 3: Feature Extraction
* Module 4: Classification
* Module 5: Evaluation

**Module 1: Data collection**

First step involved in analysis of sentiment is the collection of information from the social network website like twitter. This helps in extracting the tweet message but this message also includes extra data like tweets likes, dislikes and comments.

**Module 2: Pre-Processing**

We use the NLP tools to pre-process the dataset before it is proceeded to the feature selection and training stage. First, we use tokenization to divide the posts into individual tokens. Next, we remove all the URLs, punctuations and stop words which could lead into erratic results if stay ignored. Then we apply stemming in order to reduce the words to their root form and group similar words together.

Text pre-processing is an essential a part of any NLP method and the significance of the NLP pre-processing are

* To minimize indexing (or knowledge) records dimension of the textual content records

1. Stop words bills 20-30% of total phrase counts in a special textual content record
2. Stemming may just diminish indexing size as much as forty- 50%

* To make stronger the efficiency and effectiveness of the IR method

1. Stop words aren't valuable for shopping or textual content mining
2. Stemming used for matching the similar words in a text record

**Tokenization:**

Tokenization is the process of breaking a circulate of textual content into phrases, phrases, symbols, or different significant factors called tokens .The aim of the tokenization is the exploration of the phrases in a sentence. The list of tokens turns into input for further processing akin to parsing or textual content mining. Tokenization is valuable both in linguistics (where it's a form of textual content segmentation), and in laptop science, the place it forms a part of lexical analysis. Textual knowledge is simplest a block of characters at the starting. All strategies in know-how retrieval require the words of the data set. For that reason, the requirement for a parser is a tokenization of records. This might be sound trivial because the text is already saved in computing device-readable codecs. However, some problems are nonetheless left, like the removing of punctuation marks. Different characters like brackets, hyphens, and so on require processing as well.

**Stop word Removal:**

Stop phrases are very more often than not used fashioned phrases like ‘and’, ‘are’, ‘this’ etc. They don't seem to be useful in classification of records. So they must be removed. However, the development of such stop phrases record is problematic and inconsistent between textual sources. This process also reduces the text knowledge and improves the approach performance. Each textual content report offers with these phrases which are not vital for text mining applications.

**Stemming and Lemmatization:**

The aim of both stemming as well as lemmatization is to scale down inflectional types & mostly derivationally associated varieties of a phrase to a fashioned base kind.

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of accomplishing this goal accurately more often than not, and quite often involves the removal of derivational affixes.

Lemmatization often refers to doing matters competently with the usage of a vocabulary and morphological analysis of phrases, in most cases aiming to eliminate inflectional endings only and to come back the base or dictionary type of a word, which is often called the lemma.

**Module 3: Feature Extraction**

After data pre-processing, we feed our models with the features that reflect users’ language habits in Reddit forums. To explore the users’ linguistic usage in the posts, we employ the LIWC dictionary, LDA topics, and N-gram features. These text encoding methods are applied to encode the words to be proceeded by different classifiers. N-gram modeling is used to examine the features from the posts. It is widely used in text mining and NLP as a feature for depression detection to calculate the probability of co-occurrence of each input sentence as a unigram and bigram.

For n-gram modelling we use the Term frequency-inverse document frequency (TF-IDF) as a numeric statistic where the importance of a word with respect to each document in corpora is highlighted. The main goal of its usage is to scale down the impact of empirically less informative tokens, which occur frequently to give a space for the more informative words occurring in a small fraction. The word is ranked with greater TF-IDF value if it is present in a particular post and absent in other post. In our study, we use TF-IDF vectorizer from the scikit-learn Python librar to extract 194,613 unigrams and bigrams. We remove all the stop words from the dataset and restrict the

term-document matrix to 3000 most frequent unigrams and bigrams. In addition, we used Point wise mutual information (PMI) to filter infrequent bigrams.

LIWC, or the Linguistic Inquiry and Word Count dictionary, is widely used in computational linguistics as a source of features for psychological and psycholinguistic analysis. It works as a baseline measure with a set of words and a behavioural link. It is often presented in several mental health projects. Table 3 describes different types of approaches to text encoding methods. To accomplish our experiment, we extract 68 among 95 different features in view of psycholinguistic measures and change every depressive and non-depressive post into numerical values. This way we obtain the scores for three higher level categories considering standard linguistic dimensions, psychological processes and personal concerns.

The standard linguistic processes are one of the largest parts of the LIWC psycholinguistic vocabulary package. It was intended to quantify the words’ usage in mentally significant classifications as well as for recognizing the connection between individuals in social co-operation. In our study, we first choose 9 linguistic features (articles, auxiliary verbs, adverbs, conjunctions, impersonal and personal pronouns, negations, prepositions and verbs) to characterize the users’ text. Then we divide the Psychological processes into subcategories from which we used effective processes (anxiety, sadness, positive or negative emotion), biological processes (sexual, body, ingestion and health), social processes (family, friend, male, female), cognitive processes (cause, always, never, because), personal concerns (job, cook, cash, bury, kill), and time orientations (present, past, season). To examine the users’ linguistic usage, we implement LIWC2015 dictionary as the pre-defined category to measure all the textual content submitted by the users to extract lexico-syntactic features. We evaluate the correlation using the Pearson correlation coefficient r and also Benjamini-Hochberg selection method used in.

Based on our results, LF IDF model works best on the validation set when it is limited to 70 topics. For the topic selection we consider only the words that appear at least in more tha 10 posts. We include every post as a single document that must be further tokenized and stemmed. This way allows us to compute the topics over the collection of documents to annotate them according to detected topics. Before we start the topic modelling process, all the stop words are removed. NLP implementation is provided by the Mallet toolkit

**Module 4: Classification**

To estimate the presence of depression, we employ classifying approaches to estimate the likelihood of depression within the users. The proposed framework is developed by using Logistic Regression, Support Vector Machine, Random Forest, Adaptive Boosting and Multilayer Perceptron classifier. Logistic Regression (LR) is a linear classification approach used to estimate the probability occurrence of binary response based on one or more predictors and features. Support Vector Machine (SVM) model is a representation of the examples as points in a highly dimensional space utilized for classification, where the points of the separate categories are widely divided. New examples are then mapped into the same space and predicted to belong to a category based on which side of the gap they fall.

Random Forest (RF) is an ensemble of decision tree classifiers trained with the bagging method where a combination of learning models increases the overall result. Adaptive Boosting (AdaBoost) is an ensemble technique that can combine many weak classifiers into one strong classifier. It is widely used for binary class classification problems.

Multilayer Perceptron (MLP) is a special case of the artificial neural network often used for modelling complex relationships between the input and output layers. Due to its multiple layers and non-linear activation, it can distinguish the data that is not only non-linearly separable. In our study, we applied the MLP method and two hidden layers with 4 and 16 perceptron’s to fix for all the features in order to ensure a comparison consistency.