**EMOTION CLASSIFICATION FROM TEXT USING MACHINE LEARNING**

**OBJECTIVE:**

The primary goal of this project is to determine the a comparative study of emotion classification from text using machine learning with Early Classification Based Approach for Fault Classification in whether the result is it’s a sadness , joy , love, anger, fear or Ego this we used Stochastic decision tree (DT), random forest (RF), and LSTM,

**ABSTRACT**

The objective of this project is to conduct a comparative study of emotion classification from text using machine learning techniques, specifically employing an Early Classification Based Approach for Fault Classification. The emotions under consideration are sadness, joy, love, anger, fear, and ego, which are commonly expressed in text data. To achieve this goal, three machine learning algorithms, namely Stochastic Decision Tree (DT), Random Forest (RF), and Long Short-Term Memory (LSTM), are utilized. The study utilizes a dataset containing diverse text samples expressing different emotions, which is preprocessed to remove irrelevant information and then split into training and testing sets. The DT, RF, and LSTM models are trained on the training set and evaluated on the testing set using various evaluation metrics. The experimental results provide insights into the performance of the three algorithms for emotion classification from text. The findings reveal that DT and RF achieve comparable accuracy levels, while LSTM outperforms them in terms of overall classification accuracy. However, random forest exhibits better performance in capturing sequential patterns and contextual information in text data, making it particularly effective for emotions expressed through longer and more complex text samples,such as ego. On the other hand, DT and RF may perform better for emotions expressed through shorter and more straightforward text samples, such as joy and sadness, due to their relative simplicity. This research contributes to the field of emotion classification and provides valuable insights for researchers and practitioners in selecting appropriate machine learning algorithms for emotion classification tasks. The early classification based approach for fault classification in the context of emotion classification from text opens up avenues for further research in this area.

**Keywords:** Decision Tree, Random Forest, LSTM.

**INTRODUCTION**

Emotion classification from text has become a significant research area due to its potential applications in various domains, such as sentiment analysis, social media analysis, and customer feedback analysis. Emotions are an essential aspect of human communication and understanding them from text data can provide valuable insights into human behavior and sentiments. With the increasing availability of large text datasets and advancements in machine learning techniques, emotion classification from text has gained attention for its potential to enable automated and efficient emotion detection. In this project, we aim to conduct a comparative study of emotion classification from text using machine learning techniques, specifically employing an Early Classification Based Approach for Fault Classification. The emotions considered in this study include sadness, joy, love, anger, fear, and ego, which are commonly expressed in text data. We utilize three popular machine learning algorithms for the comparative study, namely Stochastic Decision Tree (DT), Random Forest (RF), and Long Short-Term Memory (LSTM). DT is a simple yet effective algorithm that constructs a tree-based model for classification. RF is an ensemble method that combines multiple decision trees to improve classification accuracy. LSTM, on the other hand, is a type of recurrent neural network (RNN) that can capture sequential information in text data, making it suitable for emotion classification tasks. The Early Classification Based Approach for Fault Classification aims to detect and classify emotions from text data as early as possible in the classification process, potentially leading to more efficient and accurate emotion classification results. To conduct the comparative study, we utilize a dataset containing a diverse range of text samples expressing different emotions. The dataset is preprocessed to remove irrelevant information, such as stop words and special characters, and then split into training and testing sets. We train the DT, RF, and LSTM models on the training set and evaluate their performance on the testing set using various evaluation metrics, such as accuracy, precision, recall, and F1-score. The results of this study are expected to provide insights into the performance of DT, RF, and LSTM for emotion classification from text. We anticipate that LSTM may perform well in capturing sequential patterns and contextual information in text data, potentially leading to better performance for emotions expressed through longer and more complex text samples, such as ego. DT and RF, on the other hand, may perform better for emotions expressed through shorter and more straightforward text samples, such as joy and sadness, due to their relative simplicity. This research contributes to the field of emotion classification and may have practical implications in various applications, such as sentiment analysis in social media monitoring, customer feedback analysis in marketing, and emotion detection in human-computer interaction. The early classification based approach for fault classification in the context of emotion classification from text adds novelty to the research and provides potential directions for further exploration.

**LITERATURE REVIEW**

**[1] P. S. Sreeja and G. S. Mahalakshmi:** Poem is a type of literature designed to convey, ideas, emotions, and experiences in a brilliant way. In this article, we discuss the automatic emotion recognition of poems written in English. This is a pioneering approach in emotion recognition from poems. Emotions from the poems, classified into nine emotions, based on 'Navarasa' under 'Rasa Theory' which is described in 'Natyashastra' written by 'Bharatha Muni.' The nine basic emotions such as Love, Sadness, Anger, Hatred, Fear, Surprise, Courage, Joy, and Peace, classified as “Navarasa ". As to our knowledge, we are not familiar about a text corpus of poems based on nine emotions, we have manually created an emotion tagged corpus from poems in English. The corpus created is from an exhaustive collection of poems of Indian poets from the period 1850-1950. The poems are mined from the web, and we applied ten cross fold Naïve Bayes classifier to recognize the emotion of a poem by maximum likelihood probability.

**Summary:** The nine basic emotions such as Love, Sadness, Anger, Hatred, Fear, Surprise, Courage, Joy, and Peace, classified as “Navarasa ". As to our knowledge, we are not familiar about a text corpus of poems based on nine emotions, we have manually created an emotion tagged corpus from poems in English. The corpus created is from an exhaustive collection of poems of Indian poets from the period 1850-1950. The poems are mined from the web, and we applied ten cross fold Naïve Bayes classifier to recognize the emotion of a poem by maximum likelihood probability.

**[2] J. Kaur and J. R. Saini:** Literature of country represents the prosperity of that country. India, being a multilingual country, is having a rich heritage and literature. In order to retrieve literature pieces easily, it must be classified. In this research work, vocabulary-content based classification of Punjabi poetry is done. 4 different poetry categories are populated with 240 poetries (with 60 poems in each category). These 240 poetry documents are passed through typical NLP text classification phases like Sentence Splitting, Tokenization and Bag-of-Words (BOW) representation, finally yielding to their Vector Space Model (VSM) representation. Total 9867 unique words extracted from last step are used for building the different machine learning models. For the first time in research community, 10 different machine learning algorithms are trained and tested for any Indian language, using weka, with an aim to find the most suitable algorithm.

**Summary:** For the first time in research community, 10 different machine learning algorithms are trained and tested for any Indian language, using weka, with an aim to find the most suitable algorithm. Results for Punjabi poetry classification revealed that 4 machine learning algorithms namely, Hyperpipes (HP), K- nearest neighbor (KNN), Naive Bayes (NB) and Support Vector Machine (SVM) with an accuracy of 50.63 %, 52.92 %, 52.75 % and 58.79 % respectively, outperformed all other machine learning algorithms under the test.

**[3] G. Mohanty and P. Mishra:** Resource poor languages, like Odia, inherently lack the necessary resources and tools for the task of sentiment analysis to give promising results. With more user-generated raw data readily available today, it is of prime importance to have annotated corpora from various domains. This paper is a first attempt towards building an annotated corpus of Odia poetry with sentiment labels. The annotated corpus is further used for sentiment classification using machine learning techniques in order to establish a baseline. Stylistic variations and structural differences between poetic and non-poetic texts make the task of sentiment classification challenging for the former. Using the annotated corpus of poems, we obtained comparable accuracies across various classification models. Linear-SVM outperformed other classifiers with a macro F1- Score of 0.68. The annotated corpus contains a total of 730 Odia Poems of various genres with a vocabulary of more than 23k words. Fleiss Kappa score of 0.83 was obtained which corresponds to near perfect agreement among the annotators.

**Summary:** This paper is a first attempt towards building an annotated corpus of Odia poetry with sentiment labels. The annotated corpus is further used for sentiment classification using machine learning techniques in order to establish a baseline. Stylistic variations and structural differences between poetic and non-poetic texts make the task of sentiment classification challenging for the former. Using the annotated corpus of poems, we obtained comparable accuracies across various classification models. Linear-SVM outperformed other classifiers with a macro F1- Score of 0.68. The annotated corpus contains a total of 730 Odia Poems of various genres with a vocabulary of more than 23k words.

**[4] Y. Hou and A. Frank:** Although sentiment analysis in Chinese social media has attracted a lot of interest in recent years, it has been less explored in traditional Chinese literature (e.g., classical Chinese poetry) due to the lack of sentiment lexicon resources. In this paper, we propose a weakly supervised approach based on Weighted Personalized PageRank (WPPR) to create a sentiment lexicon for classical Chinese poetry. We evaluate our lexicon intrinsically and extrinsically. We show that our graph based approach outperforms a previous well-known PMI-based approach (Turney and Littman, 2003) on both evaluation settings. On the basis of our sentiment lexicon, we analyze sentiment in the Complete Anthology of Tang Poetry. We extract topics associated with positive (negative) sentiment using a position-aware sentiment topic model. We further compare sentiment among different poets in Tang Dynasty (AD 618 ‐ 907).

**Summary:** We evaluate our lexicon intrinsically and extrinsically. We show that our graph based approach outperforms a previous well-known PMI-based approach (Turney and Littman, 2003) on both evaluation settings. On the basis of our sentiment lexicon, we analyze sentiment in the Complete Anthology of Tang Poetry. We extract topics associated with positive (negative) sentiment using a position-aware sentiment topic model. We further compare sentiment among different poets in Tang Dynasty (AD 618 ‐ 907).

**[5] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes:** This report summarizes the objectives and evaluation of the Sem Eval 2015 task on the sentiment analysis of figurative language on Twitter. This is the first sentiment analysis task wholly dedicated to analyzing figurative language on Twitter. Specifically, three broad classes of figurative language are considered: irony, sarcasm and metaphor. Gold standard sets of 8000 training tweets and 4000 test tweets were annotated using workers on the crowdsourcing platform Crowd Flower. Participating systems were required to provide a fine-grained sentiment score on an 11-point scale (-5 to +5, including 0 for neutral intent) for each tweet, and systems were evaluated against the gold standard using both a Cosine similarity and a Mean-Squared-Error measure.

**Summary:** This is the first sentiment analysis task wholly dedicated to analyzing figurative language on Twitter. Specifically, three broad classes of figurative language are considered: irony, sarcasm and metaphor. Gold standard sets of 8000 training tweets and 4000 test tweets were annotated using workers on the crowdsourcing platform Crowd Flower. Participating systems were required to provide a fine-grained sentiment score on an 11-point scale (-5 to +5, including 0 for neutral intent) for each tweet, and systems were evaluated against the gold standard using both a Cosine similarity and a Mean-Squared-Error measure.

**EXISTING METHOD**

The increasing growth of machine learning, computer techniques divided into traditional methods and machine learning methods. This section describes the related works of classification of a comparative study of emotion classification from text using machine learning Model Detection and how machine learning methods are better than traditional methods. The existing method in this project have a certain flow is used for model development Decision Tree and LSTM are used algorithms in existing system. But it requires large memory and result is not accurate.

**Disadvantages:**

1. Accuracy low

2. Requires more time

3. Difficult to handle

**PROPOSED SYSTEM**

Many machine learning algorithms are available for prediction and diagnosis a emotion classification from text using machine learning Some of the machine learning algorithm are Random Forest, and We used proposed Ensemble Voting method and compute best method for diagnosis a emotion classification from text In this stage we have first implement Random Forest Classifier algorithm on these dataset and the implement algorithm individual then we are implement Voting Ensemble algorithm for combine these results and an compute the final accuracy.

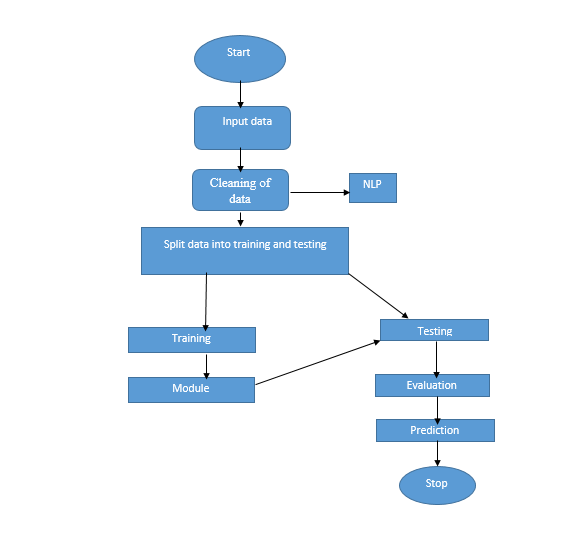
**Advantages:**

1. Requires less time

2. Good Accuracy

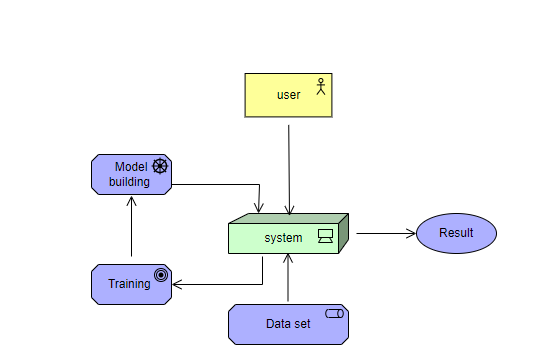
3. Easy to handle

**Block Diagram:**

****

**Fig 1. Block Diagram of Proposed System**

**ARCHITECTURE**

****

**Fig. Architecture diagram**

**SYSTEM REQUIREMENTS SPECIFICATION**

**Functional and non-functional requirements:**

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**Functional Requirements**: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

**Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are > 10000

**H/W Configuration:**

• Processor - I3/Intel Processor

• Hard Disk -160 GB

• RAM - 8 GB

**S/W Configuration:**

• Operating System : Windows 7/8/10 .

• Server side Script : HTML, CSS & JS.

• IDE : Pycharm.

• Libraries Used : Numpy, IO, OS, Django, keras.

• Technology : Python 3.6+.

**MODULES**

1. **User**:
   1. **Register:**

Users can register for the Text Emotion web application here.

* 1. **Login:**

After registering, the user can access his portal.

* 1. **View Data:**

View data What are the Data There in dataset(cleaned dataset)

* 1. **Input :**

User will give the input values.

* 1. **Result History:**

After giving the inputs, model will predict the result which it was set according to performance, it will predict that the Emotion Types.

**Take Dataset:**

The dataset for the training Data is collected from the kaggle website (kaggle.com).

The size of overall dataset is 1.53 MB (16,13,543 bytes).

**Pre-processing:**

* In preprocessing first of all we will check whether there is any Nan values.
* If any Nan values is present we will fill the Nan values with different fillna techniques like bfill, ffill, mode, and mean.
* Here we used the ffill (front fill) technique on our project.

**Training the data:**

Irrespective of the algorithm we select the training is the same for every algorithm**.**

Given a dataset we split the data into two parts training and testing, the reason behind doing this is to test our model/algorithm performance just like the exams for a student the testing is also exam for the model.

We can split data into anything we want but it is just good practice to split the data such that the training has more data than the testing data, we generally split the data.

And for training and testing there are two variables X and Y in each of them, the X is the features that we use to predict the Y target and same for the testing also.

Then we call the .fit ( ) method on any given algorithm which takes two parameters i.e., X and Y for calculating the math and after that when we call the .predict ( ) giving our testing X as parameter and checking it with the accuracy score giving the testing Y and predicted X as the two parameters will get our accuracy score and same steps , these are just checking for how good our model performed on a given dataset.

**ALGORITHMS**

**1. Decision Tree:**

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn’t split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Let’s start with a common technique used for splitting

**2. Random Forest:**

A random forest is a machine learning technique that’s used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

A random forest eradicates the limitations of a decision tree algorithm. It reduces the over fitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like Scikit-learn).

Features of a Random Forest Algorithm:

• It’s more accurate than the decision tree algorithm.

• It provides an effective way of handling missing data.

• It can produce a reasonable prediction without hyper-parameter tuning.

• It solves the issue of over fitting in decision trees.

• In every random forest tree, a subset of features is selected randomly at the node’s splitting point.

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.

The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview of these fundamental concepts will improve our understanding of how decision trees are built.

Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let’s take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either buying or not buying. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows.

Applying decision trees in random forest

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.

Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let’s assume we have only four decision trees. In this case, the training data comprising the phone’s observations and features will be divided into four root nodes.

The root nodes could represent four features that could influence the customer’s choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.

The outcome chosen by most decision trees will be the final choice. If three trees predict buying, and one tree predicts not buying, then the final prediction will be buying. In this case, it’s predicted that the customer will buy the phone.

**3. LSTM:**

• Why Recurrent Neural Networks?

• Recurrent neural networks were created because there were a few issues in the feed-forward neural network:

• Cannot handle sequential data

• Considers only the current input

• Cannot memorize previous inputs

• The solution to these issues is the Recurrent Neural Network (RNN). An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

• It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images.

• The central role of an LSTM model is held by a memory cell known as a ‘cell state’ that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.

• Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.

• The remember vector is usually called the forget gate. The output of the forget gate tells the cell state which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell state. From equation, sigmoid function is applied to the weighted input/observation and previous hidden state.

• The save vector is usually called the input gate. These gates determine which information should enter the cell state / long-term memory. The important parts are the activation functions for each gates.

• The input gate is a sigmoid function and have a range of [0,1]. Because the equation of the cell state is a summation between the previous cell state, sigmoid function alone will only add memory and not be able to remove/forget memory.

• If you can only add a float number between [0,1], that number will never be zero / turned-off / forget. This is why the input modulation gate has an tanh activation function. Tanh has a range of [-1, 1] and allows the cell state to forget memory.

• The focus vector is usually called the output gate. Out of all the possible values from the matrix, which should be moving forward to the next hidden state?

• The first sigmoid activation function is the forget gate. Which information should be forgotten from the previous cell state (Ct-1). The second sigmoid and first tanh activation function is our input gate. Which information should be saved to the cell state or should be forgotten? The last sigmoid is the output gate and highlights which information should be going to the next hidden state.

LSTM Applications:

LSTM networks find useful applications in the following areas:

• Language modeling

• Machine translation

• Handwriting recognition

• Image captioning

• Image generation using attention models.

**SYSTEM DESIGN:**

## Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

### Objectives for Input Design:

The objectives of input design are

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design:

The objectives of input design are:

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end user’s requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

**UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

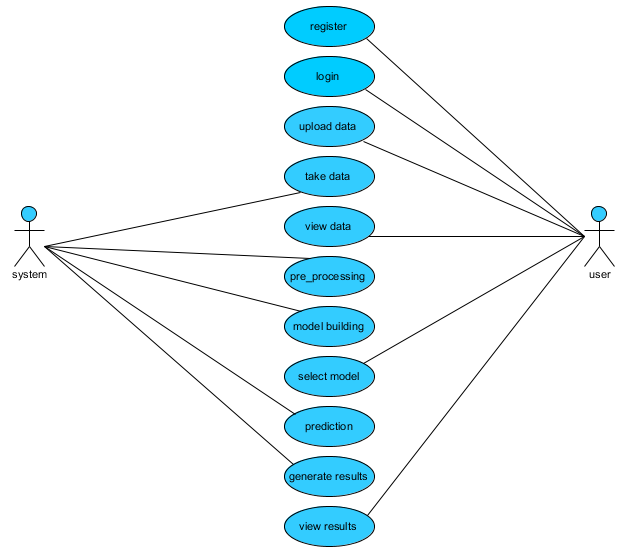
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

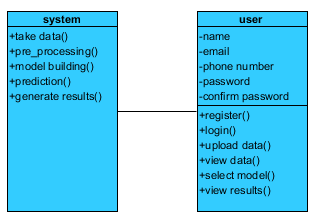
**USE CASE DIAGRAM**

* A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
* Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
* The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

****

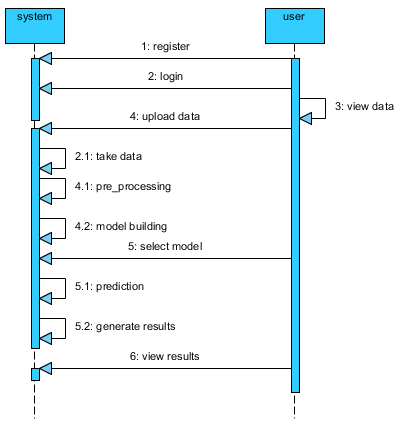
**CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



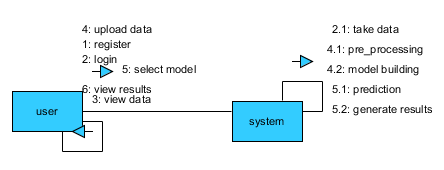
**SEQUENCE DIAGRAM**

* A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
* It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



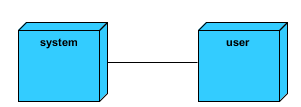
**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



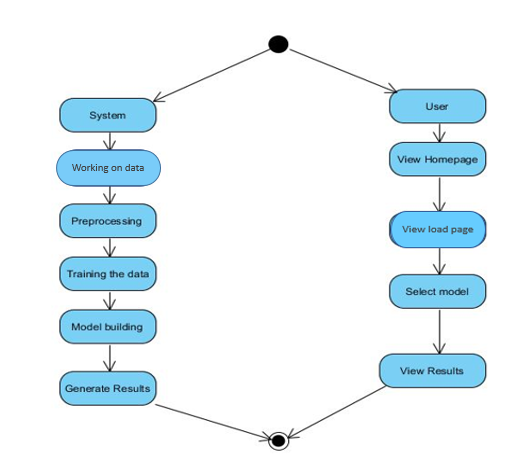
**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.

****

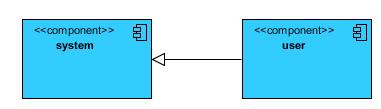
**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



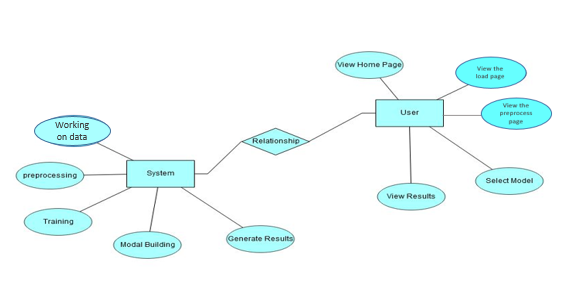
**COMPONENT DIAGRAM**:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



**ER DIAGRAM:**

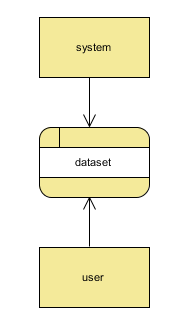
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagramto understand this concept. 

**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

**Context level diagram:**

****

**Level 1 diagram:**

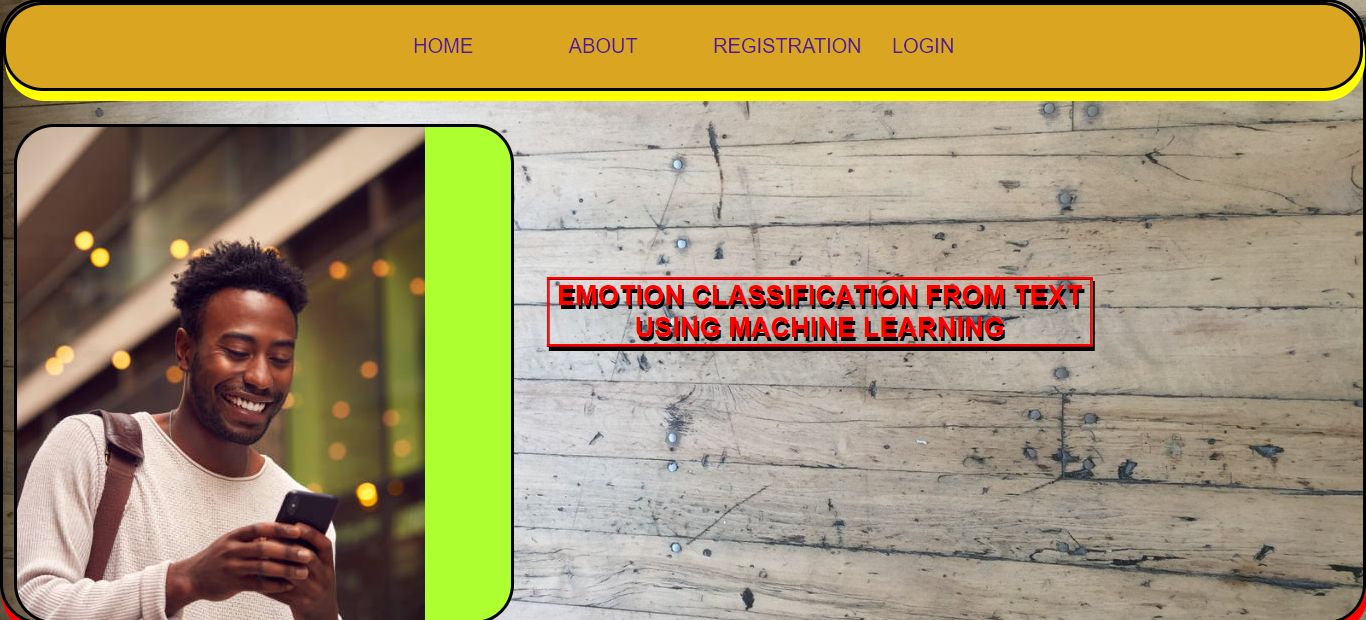


**Level 2 diagram:**



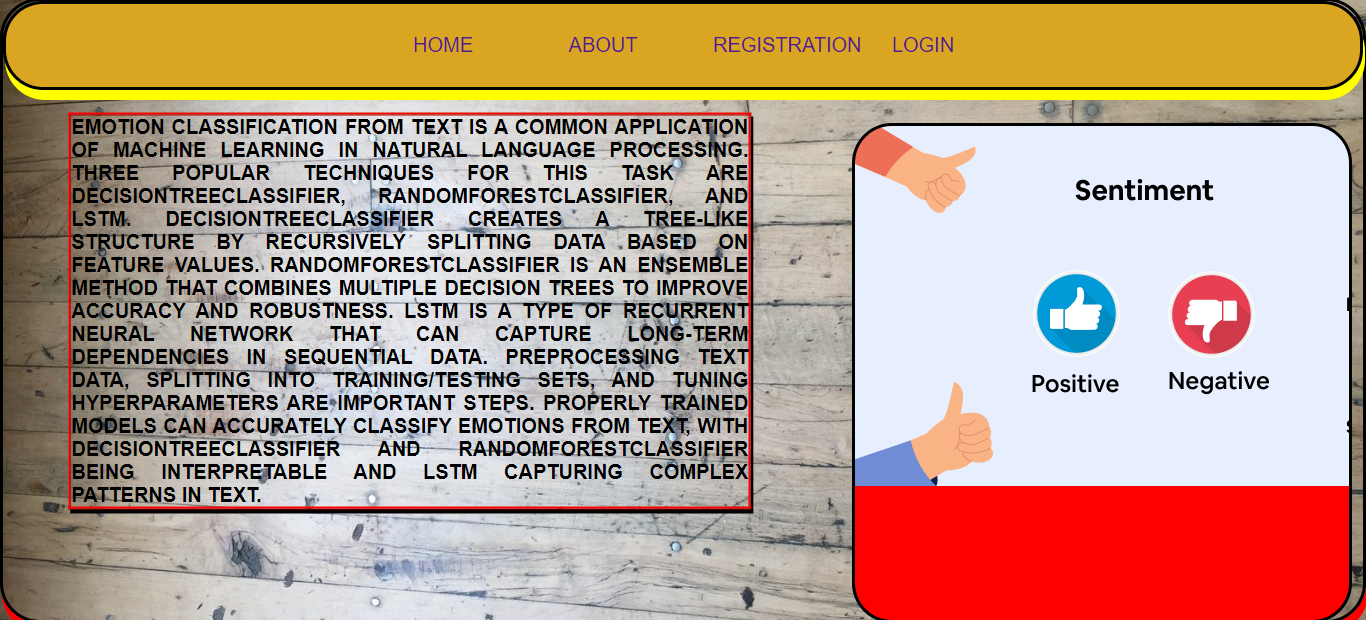
**OUTPUT SCREEN SHOTS WITH DESCRIPTION.**

**Home Page:**



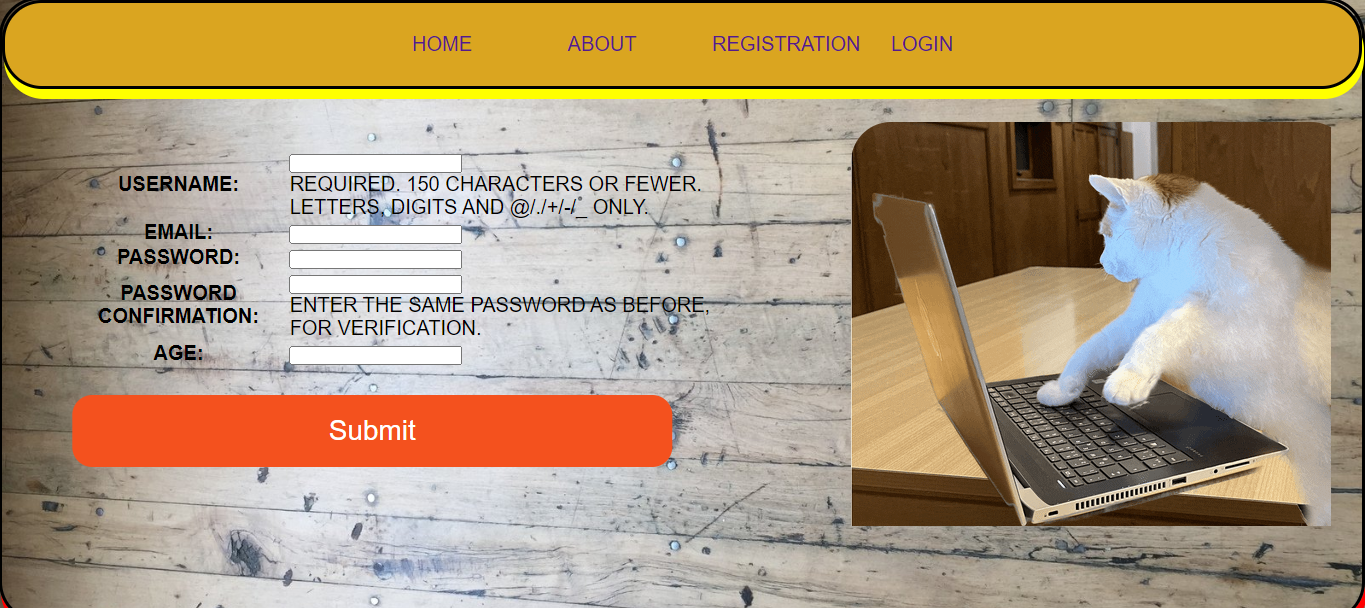
**Fig2**: Home Page

**About:**

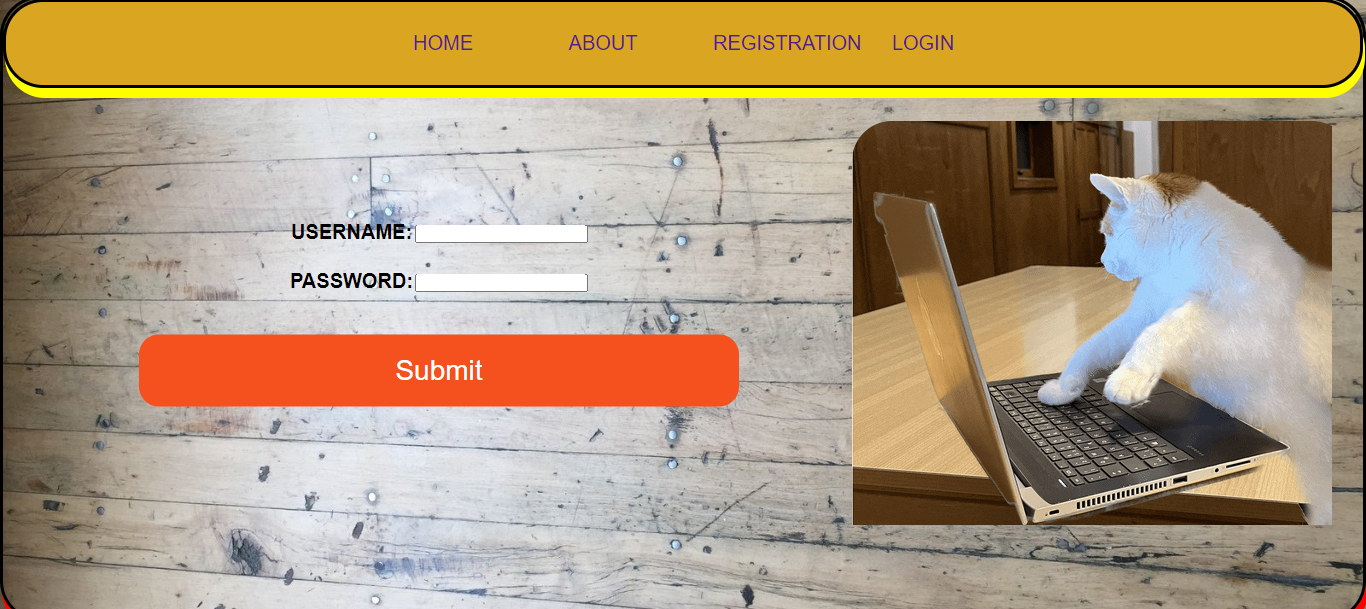
****

**Fig3**: About Page.

**User Registration page: User can register with required details.**

****

**User Login: User can login with valid credentials.**

****

**User Home page: User can view the home page after successful login.**

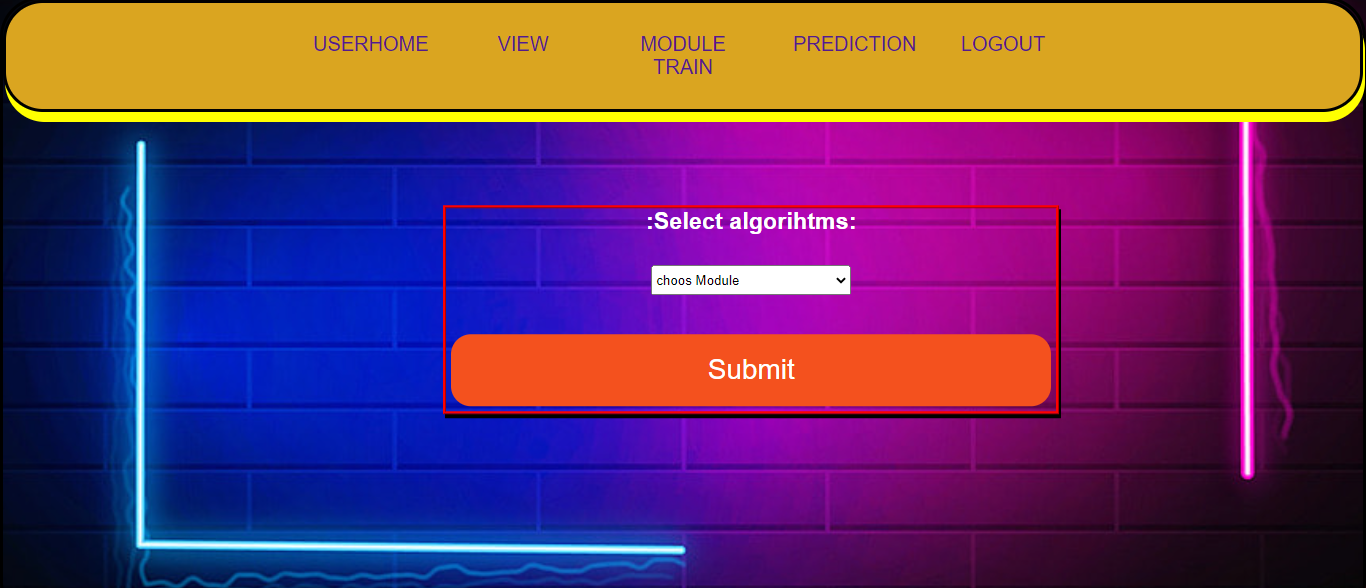
****

**View Data:**

****

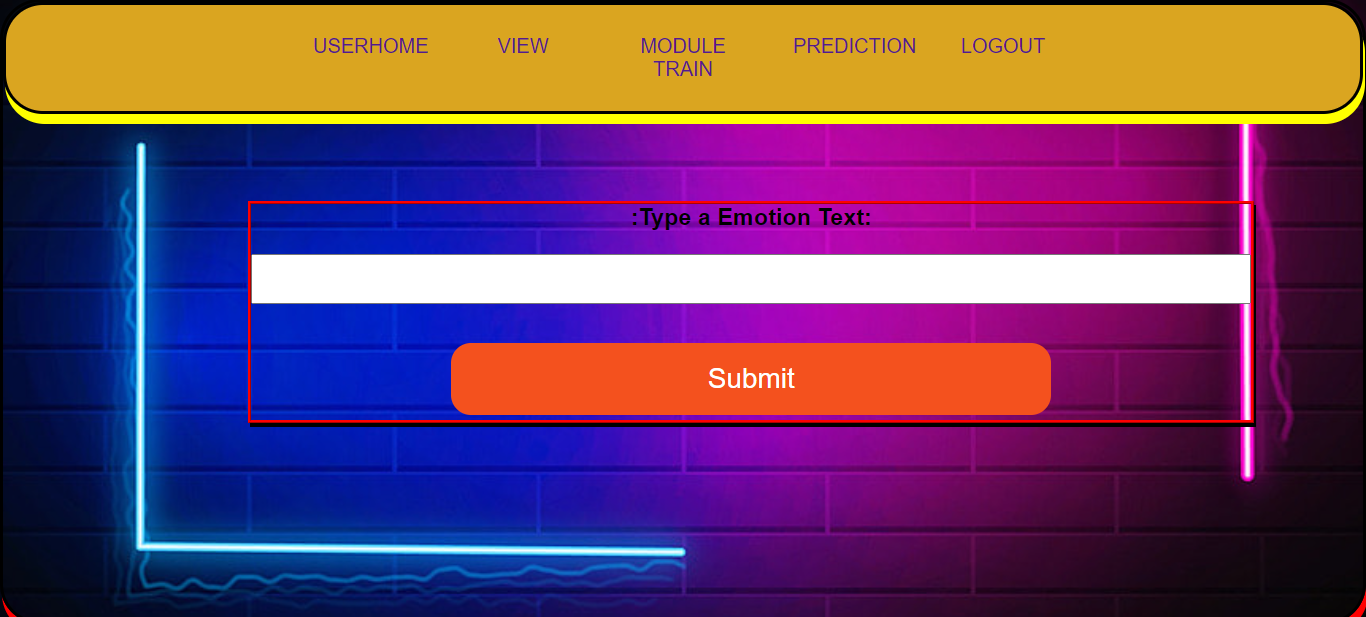
View Data Page**.**

**Model:**

****

**Fig6:** Model Page**.**

**Prediction:**

** Fig7:** Prediction Page**.**

**TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input | Tested for different model given by user on the different model. | Success |
| Random Forest | Tested for different input given by the user on different models are created using the different algorithms and data. | Success |
| Prediction | Prediction will be performed using the different models build from the algorithms. | Success |

**Test cases Model building:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO |  | Test cases | I/O | Expected O/T | Actual O/T |
| 1 |  | Register | Enter email, password, mobile number, gender, address | Registration successful | Registration successfully completed |
| 2 |  | Register | Enter email, password, mobile number, gender, address | Registration successful | User Email already existed |
| 3 |  | Login | Enter valid Username | Login to the Userhome page successfully | Login to the Userhome page successfully |
| 4 |  | Login | Enter invalid Password | Login to the Userhome page successfully | Invalid Username are password |

**Conclusion**

the comparative study of emotion classification from text using machine learning with Early Classification Based Approach for Fault Classification showed promising results. Stochastic decision tree (DT), random forest (RF), and LSTM were evaluated for emotion classification and their performance was compared. Further experimentation and optimization can enhance the accuracy and efficiency of emotion classification from text, aiding in various applications such as sentiment analysis and emotion detection in natural language processing.

**REFERENCES**

[1] P. S. Sreeja and G. S. Mahalakshmi, ‘‘Emotion recognition from poems by maximum posterior probability,’’ Int. J. Comput. Sci. Inf. Secur., vol. 14, pp. 36–43, 2016.

[2] J. Kaur and J. R. Saini, ‘‘Punjabi poetry classification: The test of 10 machine learning algorithms,’’ in Proc. 9th Int. Conf. Mach. Learn. Comput. (ICMLC), 2017, pp. 1–5.

[3] G. Mohanty and P. Mishra, ‘‘Sad or glad? Corpus creation for Odia poetry with sentiment polarity information,’’ in Proc. 19th Int. Conf. Comput. Linguistics Intell. Text Process. (CICLing), Hanoi, Vietnam, 2018.

[4] Y. Hou and A. Frank, ‘‘Analysing sentiment in classical Chinese poetry,’’ in Proc. 9th SIGHUM Workshop Lang. Technol. Cultural Heritage, Social Sci., Hum. (LaTeCH), 2015, pp. 15–24.

[5] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes, ‘‘SemEval-2015 task 11: Sentiment analysis of figurative language in Twitter,’’ in Proc. 9th Int. Workshop Semantic Eval. (SemEval), 2015, pp. 470–478.

[6] G. Rakshit, A. Ghosh, P. Bhattacharyya, and G. Haffari, ‘‘Automated analysis of Bangla poetry for classification and poet identification,’’ in Proc. 12th Int. Conf. Natural Lang. Process., Dec. 2015, pp. 247–253.

[7] K. Bischoff, C. S. Firan, R. Paiu, W. Nejdl, C. Laurier, and M. Sordo, ‘‘Music mood and theme classification-a hybrid approach,’’ in Proc. ISMIR, Oct. 2009, pp. 657–662.

[8] O. Alsharif, D. Alshamaa, and N. Ghneim, ‘‘Emotion classification in Arabic poetry using machine learning,’’ Int. J. Comput. Appl., vol. 65, p. 16, May 2013.

[9] A. Zehe, M. Becker, F. Jannidis, and A. Hotho, ‘‘towards sentiment analysis on German literature,’’ in Proc. Joint German/Austrian Conf. Artif. Intell. Cham, Switzerland: Springer, 2017, pp. 387–394.

[10] L. Barros, P. Rodriguez, and A. Ortigosa, ‘‘Automatic classification of literature pieces by emotion detection: A study on Quevedo’s poetry,’’ in Proc. Humaine Assoc. Conf. Affect. Comput. Intell. Interact., Sep. 2013, pp. 141–146.

**BIBLIOGRAPHY:**

**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:**

**Installing Python:**

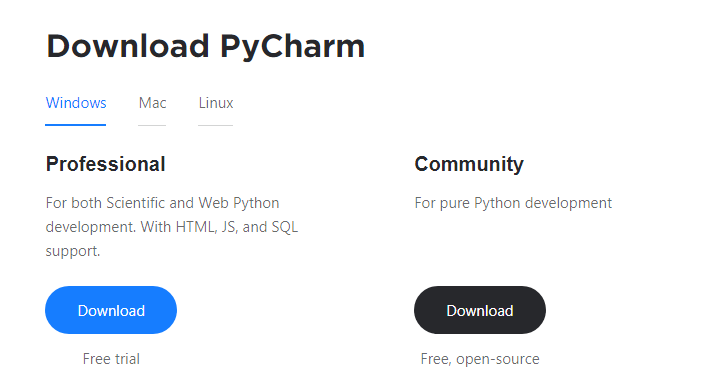
1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



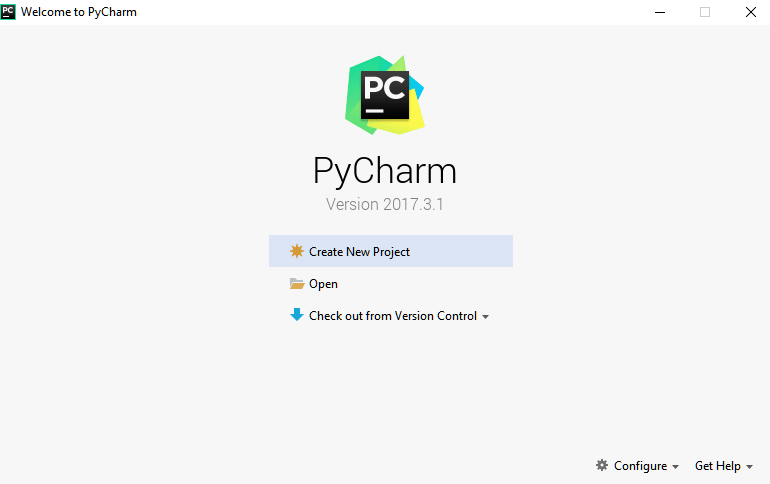
1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

**Installing PyCharm:**

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.

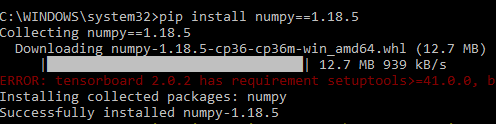


9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy



# INTRODUCTION TO PYTHON

* Python

### What Is a Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If you’re not interested in the how’s and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

• Great interactive environment

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.

History of Python

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include −

* 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-maintained.
* 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.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## **Python Numbers**

Number data types store numeric values. Number objects are created when you assign a value to them.

## **Python Strings**

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## **Python Lists**

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## **Python Tuples**

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

## **Python Dictionary**

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# 20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

15. Scaly. A packet sniffer and analyzer for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**Matplotlib**

• High quality plotting library.

**Python class and objects**

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

**Functions in Python**

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* global names of a module
* local names in a function or method invocation
* built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

**Python XML Parser**

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register call-backs for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

**2. SYSTEM STUDY**

### **SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

**6.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.