## CHAPTER 1

**INTRODUCTION**

**1.1 Background**

Plant disease can be defined as the sum total of abnormal changes in the physiological processes brought about by any biotic or abiotic factor(s) that ultimately threatens the normal growth and reproduction of a plant which has a great impact on the agricultural yield. Hence, there is a need to identify the plant diseases at a very early stage and take precautionary measures to increase the yield and agricultural productivity. The most widely used method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases are done. For doing so, a large team of experts as well as continuous monitoring of experts is required, which costs very high when farms are large. At the same time, in some countries, farmers don't have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. In such a condition, the suggested technique proves to be beneficial in monitoring large fields of crops. And automatic detection of the diseases by just seeing the symptoms on the plant leaves makes it easier as well as cheaper. Plant disease identification by the visual way is a more laborious task and at the same time less accurate and can be done only in limited areas. Whereas if automatic detection technique is used it will take fewer efforts, less time and more accurately. In plants, some general diseases are bacterial, black spotted, and others are Rust, viral and Red cotton Leaf. Image processing is the technique which is used for measuring the affected area of disease, and to determine the difference in the color of the affected are. Image segmentation is the process of separating or grouping an image into different parts. There are currently many different ways of performing image segmentation, ranging from the simple thresholding method to advanced color image segmentation methods. The segmentation process is based on various features found in the image. This might be color information, boundaries or segment of an image. Based on features extracted from the processed image the plant disease is identified and classified.

I

* 1. **Overview of Present Work**

Many studies and research have been conducted on plant disease recognition system , a majority of which lack accuracy, and the usage of varied techniques. In [1], B SRAVYA REDDY, R DEEPA, S SHALINI, P BHAGYA DIVYA, in their research, ‘Novel Machine Learning Based Approach For Detection And Classification Of Sugarcane Plant Disease By Using DWT’, disease detection is done using discrete wave length transform algorithm and decision tree approach to classify the disease. Infected plant images are stored in the database and are compared with the input image and classification is done using decision tree. But results are unstable, meaning that a small change in data can lead to large change in the structure of the optimal decision tree.

In [2], presented by TISEN HUANG , RUI YANG, WENSHAN HUANG , YIQI HUANG , XIQIAO.“Detecting Sugarcane Borer Diseases Using Support Vector Machine”, Sugarcane borer disease is detected using greyscale conversion for image processing , and SVM algorithm for classification. SVM is applicable only for linear dataset. The proposed system is confined to only detect stem related disease.

In [3] by MOSBAH EL SGHAIR, RAKA JOVANOVIC, MILAN TUBA “An Algorithm for Plant Diseases Detection Based on Color Features”, In this , Four different color models were tested and compared: RGB, YCbCr, HSI color model. Median filter is employed for image smoothing (Noise reduction). Kapur’s thresholding for plant diseases detection was proposed. But the system faces Disturbance due to vein is present in RGB models. Calculative dimensions of disease spot is not considered.

In [4] S SATHIAMOORTHY, R PONNUSAMY,M NATARAJAN.“ Sugarcane Disease Detection Using Data Mining Techniques” this system uses R Datasets are used for experimental studies and Techniques like k means , MLP are used to predict sugarcane leaf disease .But R Dataset considers only limited attributes and thus cannot be generalized.

In [5] TRIMI NEHA TETE, SUSHMA KAMLU “Plant Disease Detection Using Different

* Algorithms”, paper illustrates Two different segmentation techniques: Thresholding and K-means clustering algorithm and classification technique such as Artificial neural network (feed forward back propagation). But thresholding cannot be applied on images with low variation. And it is difficult to predict the k value manually.

**Problem Statement**

To design and develop a system for detection and prevention of major sugarcane diseases.

* 1. **Objectives**

1. Design an efficient system which identifies the disease in sugarcane crop
2. Use image processing, classify the disease, and provide accurate remedy.
3. Help the farmers to enhance their crop yield, by reducing the crop loss due to preventable diseases..
   1. **Organization of Project Report**

This report deals with the implementation of the project ‘Emotion Recognition from Voice”. This report is organized as 7 chapters, namely, introduction, Literature Review, System Requirements, System Designs / Methodology, implementation, Results and Discussion and lastly conclusion and future work.

Chapter 1 gives a brief introduction about the study of emotions and how various physiological effects observed by the virtue of emotional state enable the inference of the emotion causing it. It consists of description of background, overview of the present work, problem statement and objectives.

Chapter 2 deals with the detailed analysis of previous studies and research conducted. It consists of summary of prior works, outcome of the review- problems identified, and proposed works.

Chapter 3 specifies the System Requirements. It consists of specification of all functional, and non-functional requirements along with the software/ hardware that has been used in the project.

Chapter 4 specifies the design details. Design is the process of establishing a system that will satisfy the previously identified functional and non- functional requirements. It also consists of specification of the architecture of the system and the major algorithm incorporated.

Chapter 5 includes the implementation part. Implementation is the process of converting the system design into an operational one. This phase starts after the completion of the development phase and must be carefully planned and controlled as it is a key stage. It includes a list of main packages, some of the user- defined functions and some sample code.

Chapter 6 includes the description and the discussion of the results obtained. It consists of Testing and Results. Testing includes the Testing part which is an investigation conducted to provide stakeholders with information of the quality of the product or service under test. It also gives a business an opportunity to understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs.

Chapter 7 mentions the conclusion and future enhancements for the project. IT contains the summary of the work carried out, contributions and utility. Also mentioned are the glossary, acronyms, and the bibliography.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 Summary of prior works**

[1] describes disease detection is done by using Discrete wavelength transform (DWT) algorithm. There are some set of diseases in the database which check with the already stored input images. Digital camera or similar devices are used to take the pictures and stored in the data set which are different types of leaf images and those are used to identify the affected areas in a leaf. There are different types of techniques used to process those images to get different and useful features needed for the purpose of analysing later.

The methodology discussed in this paper is given below in step by step approach for the proposed image recognition and segmentation process.

1.1 Image Acquisition

In this section, disease affected leaf image is considered as an input image from the dataset of disease affected leaves.

1.2 Image Pre-Processing

After insertion of image, image is pre-processed. Preprocessing is performed to decrease the noise rate and improve the contrast of the image using filters. Spatial filters is a operation where each pixel value is changed by function of intensity of pixel of the neighbourhood image.

1.3 Image Segmentation

Enhanced image is segmented using edge detection method. To highlight the affected part and mask the green pixels and compares with the part which is turned in to another color.

1.4 Feature Extraction

Here, diseases affected Region of Interest is selected from the segmented images. Then, Convert the RGB color (ROI) image into grey scale image and maintain the color Co Occurrence Matrices (CCMs).

1.5 Classification:

Here ,the diseases are classified based on the type of the fungus, bacteria, pathogen or virus the plant is affected.

1.6 Disease detection:

Here the detection of disease is the final step, based on the image the affected part is detected and disease is identified . Obtain the useful segments to classify the leaf diseases. Segment the components using DWT algorithm.

In this paper, approach based on image processing to first detect and then classify leaves according to diseases is used. Here, image acquisition is performed by considering RGB colour disease affected leaf image. Pre-processing of an image is done to enhance the image using filtering. Image segmentation is performed by making use of threshold value. Image feature extraction is performed to obtain the features of leaf disease symptoms. Image classification is performed using Decision tree (DT).

[2] This paper addresses a specific disease called Stem Borer disease that infects a sugarcane plant, where the larvae feeds on the stem. From the images, the sugarcane borer diseases are characterized by different sizes of approximate elliptic wormhole and theirs color are black.

Considering that there was a sag in the wormhole, which was not sensitive to the light reflection, so the grey value of wormhole is lower and closer to black, while other parts are higher and closer to white or cinereous because of the high exposure.

Preprocessing of image is done as follows:

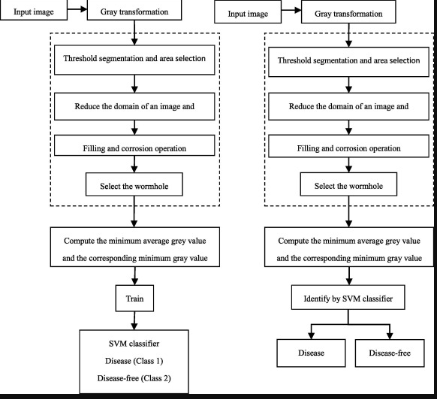
1) The target image and part area of the target were obtained by segmentation which the threshold was 150, and the connected domain was divided into disconnected area, then the largest area which was selected by area method was the target area of sugarcane seed. This step mainly reduced the influence of the background on the segmentation result and achieved the sugarcane target region which includes some noises and the wormhole target.

2) Now select the wormhole. Removed the region which area of 1 and selected the region corresponding to the minimum average grey value for the wormhole after calculating the area and average grey value of each non-connected region. As a result, the region corresponding to the minimum average grey value was the result that we segmented.

Classification is done using SVM.

The selection of the SVM has been carried out taking into account the computational resources was required by it for making a decision. But, the SVM approach has a good classification effect.

The minimum average grey value and the corresponding minimum grey value can effectively avoid miscalculation of pseudo diseases for the wax and leaf scar. Compared to RBF kernel function and Polynomial kernel function, the recognition rate is more stable and reliable when RBF kernel function as inner product function. This shows that the RBF kernel function can be used as the kernel function of the support vector machine to solve the problem of sugarcane borer disease detection.



[3] An Algorithm for Plant Diseases Detection Based on Color Features.By Mosbah El Sghair, Raka Jovanovic, Milan Tuba published at IJAS Vol.2 in 2017.

In this paper four different color models were tested and compared: RGB, YCbCr, HSI color model. Median filter is employed for image smoothing (Noise reduction). Kapur’s thresholding for plant diseases detection was proposed.

In plants, leaf vein is totally different in intensity and disease spot is different in color compared to plant leaf. Therefore if Kapur’s method is applied on grayscale image, vein will be detected in binary image with the disease spot. However the region of interest is simply disease spots, not vein. For minimize the effect of presence of vein, RGB color model is not suitable for segmentation. Thresholding method are often applied on color element to discover disease spot accurately.

The next step in proposed algorithm is image smoothing. During image assortment, some noise is also introduced due to camera flash. This noise might have an effect on the detection of disease. To remove unneeded spots, image smoothing technique is required. In this paper adjusted median filter is employed for this purpose.

After image smoothing, a method to detect and isolate the disease spot is required. It is necessary to find a threshold value that will differentiate the disease spots from plant leaf. One of the most used method for thresholding is Kapur’s method that is based on the entropy. This method maximize the amount of information between the two parts of a intensity histogram that are separated by concrete threshold value or better to say maximize the entropy measure of the part of the histogram in order to each part has a more centralized distribution.

In this paper a method based on different color models and Kapur’s thresholding for plant diseases detection was proposed. Four different color models were tested and compared: RGB, YCbCr, HSI and CIELAB color model. The best results were obtained when HSI color model was used. Component H was used for image segmentation where diseases were separated from the leaf. Median filter was applied to color transformed image. At the end, disease spots area are determined

by applying Kapur’s threshold on different color components. Experimental result shows that noise that is introduced due to background, vein and camera flash makes the least problem for HSI color model. Following this technique totally different disease spots are detected accurately and results do not seem to be laid low with background, sort of leaf, type of disease spot and camera.

[4]Sugarcane Disease Detection Using Data Mining Techniques. Published by S Sathiamoorthy, R Ponnusamy, M Natarajan in the year 2018

In this paper, the R-Dataset is used as a benchmark database to perform experimental study to find out the diseases which affects the sugarcane leaf. Classification algorithms like J48, pruned tree and Multilayer perceptron were analysed and compared with K-means clustering algorithm. These algorithms were implemented in WEKA tool forclassification and clustering.

J48 pruned trees algorithm generates the rules for the prediction of the target variable. With the help of tree classification algorithm the critical distribution of the data is easily understandable. The WEKA tool provides a number of options associated with tree pruning. In case of potential over fitting pruning can be used as a tool for précising. In other algorithms the classification is performed recursively till every single leaf is pure, that is the classification of the data should be as perfect as possible. This algorithm it generates the rules from which particular identity of that data is generated. The objective is progressively generalization of a decision tree until it gains equilibrium of flexibility and accuracy.

Basic Steps in the Algorithm:

i. In case the instances belong to the same class the tree represents a leaf so the leaf is returned by labelling with the same class.

ii. The potential information is calculated for every attribute, given by a test on the attribute. Then the gain in information is calculated that would result from a test on

the attribute.

iii. Then the best attribute is found on the basis of the present selection criterion and that attribute selected for branching.

[5]An Identification Of Crop Disease Using Image Segmentation published by K. Vinoth Kumar and T. Jayasankar in the year 2018

In this paper the methodology is broken down into two segments:

(a) Image Processing Segment: Where the properties of the leaf image will be enhanced segmented from the background.

(b) Pattern Recognition Segment: Where the required features will be extracted and this information will be matched with the predefined knowledge about the plant diseases for detecting which disease has actually affected the plant.

Pattern recognition further includes:

Feature Extraction: It is the procedure of outlining a set of necessary features, or image characteristics that form the core element which when represented in an efficient or meaningful manner give the required information that is

important for analysis and classification purpose.

Disease Identification: It is the process of understanding the meaning of the feature extracted from the image and matching the extracted information with the predefined set of rules and thus coming to a conclusion. The result of the computation is obtained in this step



**2.2 Outcome of the review- problems identified**

There are many methods in automated or computer vision plant disease detection and classification process, but still, this research field is lacking. In addition, there are still no commercial solutions on the market, which are easily accessible by farmers to recognize the diseases affecting their yield.

In the above discussed papers various algorithms like SVM and decision tree have been used. SVM can give a greater accuracy for classification only when the dataset is linear. Classification done using decision tree yield unstable results even a small change in data can lead to large change in the structure of the optimal decision tree. R data set has been considered to train the model for classification, which addresses the disease that are specific to only certain region on the globe(Brazilian region)and cannot be used to identify the disease globally.

**2.3 Proposed Work**

Considering the performance issues as mentioned above, and the lack of representative systems, a system that is capable of achieving a higher performance in classifying the 4 sugarcane diseases eye spot, red rot, wheat rust, yellow leaf, is developed employing several machine learning algorithms along with the associated analysis.

The dataset is created by clicking the images of the diseased plants from the fields. With limited dataset the prediction for disease classification will not give higher accuracy, so to overcome this issue and increase the size of dataset image augmentation technique is deployed. The images in the dataset are then pre-processed to resize and remove noise.

The Pre-processed images are then fed into the pre-trained MobileNetV2 model which is retrained according to the new customized dataset to identify and classify the sugarcane disease into the four major diseases addressed. The Trained model is then obtained as the saved model which is further fed into the TensorFlowLite convertor to obtain .tflite file which can then be deployed in android applications.

New disease suspected plant images are then clicked and uploaded to the application where the trained model executes to identify if the plant is diseased and classified accordingly. The application also helps farmers by suggesting the remedial measures to be taken to address the particular disease at the early stage thereby decreasing the crop loss.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 Functional Requirements**

Requirement analysis is very critical process that enables the success of a system or software project to be assessed which are split into two types: Functional and Non-Functional requirements. Functional Requirements define the internal working of the software, i.e., the calculations, technical details, data manipulation and processing and other specific functionalities that show how the cases are to be satisfied and how they are supported by non-functional requirements, which impose constraints on the design or the implementation. Behavioural requirements describe all the cases where the system uses the functional requirements; these are captured in use cases. It defines a system’s reaction to particular inputs at a component level.

Mobile devices which can also be called handheld devices like mobile phones, tablets, personal digital assistants (PDAs), or enterprise digital assistants need mobile applications for optimal usefulness and service delivery. Mobile applications are to be taken as utility software packages that enhance full functionality and the mobile devices’ ability to meet diverse users’ needs. The purpose of a well-done requirement analysis phase is to archive a user-friendly system by being a user, to acknowledge the appropriateness of the takes. ­­­­

The following are the Functional requirements of our system:

1. Image acquisition for dataset preparation.
2. Image augmentation and pre-processing.
3. Training the model for disease identification and classification.
4. Deployment of trained model into android application.
5. Identification and Classification of diseased plants with suggested remedies.

**3.2 Non-functional Requirements**

A non-functional requirement places constraint on “How should the software system fulfil the functional requirements?”. It defines the system’s properties and constraints applied to it as a whole. It describes system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

The following are the Functional requirements of our system:

1. Performance
2. Dependability
3. Capacity
4. Operability
5. Interoperability

**3.3 Software/Hardware used**

**HARDWARE REQUIREMENTS**:

• Operating System: Windows 7 or above

• Processor: Intel i3 or higher

• RAM: 2 GB or more

• Hard Drive: 10 GB or more

• Android 6.0(or above) smartphone

**SOFTWARE REQUIREMENTS:**

• Languages Used: Python, JAVA

• Frameworks : Tensorflow 2.0

•Jupyter Notebook

• Android Studio

**Languages/Frameworks Used**

# Python

**History of Python**

Python is an interpreted, high-level, general-purpose programming language created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages Python and CPython.

Python 3.0, a major, backwards-incompatible release, was released on December 3, 2008 after a long period of testing. Many of its major features have also been backported to the backwards-compatible Python 2.6 and 2.7.

In February 1991, van Rossum published the code (labelled version 0.9.0) to alt.sources. Already present at this stage in development were classes with inheritance, exception handling, functions, and the core datatypes of list, dict, str and so on. Also in this initial release was a module system borrowed from Modula-3; Van Rossum describes the module as "one of Python's major programming units". Python's exception model also resembles Modula-3's, with the addition of an else clause. In 1994 comp.lang.python, the primary discussion forum for Python, was formed, marking a milestone in the growth of Python's userbase.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

The latest version of Python is 3.8.3; however the latest stable release of Python is version 3.8.2, released on 24 February 2020.

# TensorFlow 2.0

TensorFlow started as an open-source [deep learning](https://courses.analyticsvidhya.com/courses/fundamentals-of-deep-learning?utm_source=blog&utm_source=tensorflow-2-tutorial-deep-learning) library and has today evolved into an end to end machine learning platform that includes tools, libraries and resources for the research community to push the state of the art in deep learning and developers in the industry to build ML & DL powered applications.

TensorFlow had its first public release back in 2015 by the Google Brain team. At the time, the evolving deep learning landscape for developers & researchers was occupied by Caffe and Theano. In a short time, TensorFlow emerged as the most popular library for deep learning.

TensorFlow is fast with backend written in C++ and has interfaces in Python, Java, Swift, and Android.

TensorFlow 2.0 makes development of ML applications much easier. With tight integration of Keras into TensorFlow, eager execution by default, and Pythonic function execution.

The standardized [SavedModel](https://www.tensorflow.org/guide/saved_model) file format can be used to run models on a variety of runtimes, including the cloud, web, browser, Node.js, mobile and embedded systems.

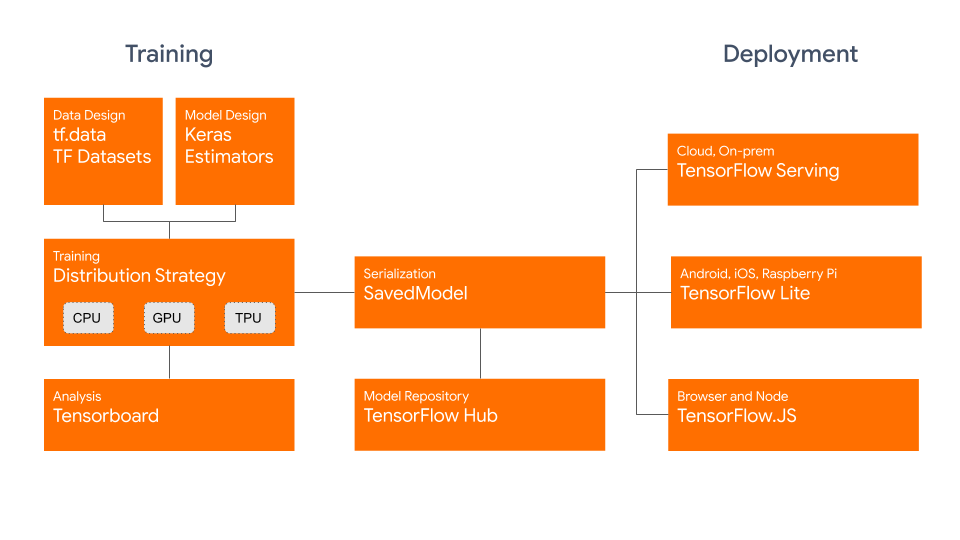


Fig:Tensorflow 2.0 Architecture

## Keras

tf.keras is TensorFlow's implementation of the Keras API specification. This is a high-level API to build and train models that includes first-class support for TensorFlow-specific functionality, such as eager execution, data pipelines, and Estimators.  keras makes TensorFlow easier to use without sacrificing flexibility and performance.

In Keras, you assemble *layers* to build *models*. A model is (usually) a graph of layers. The most common type of model is a stack of layers: the [tf.keras.Sequential](https://www.tensorflow.org/api_docs/python/tf/keras/Sequential) model.

## TensorFlow Lite

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

* The **TensorFlow Lite interpreter** , which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
* The **TensorFlow Lite converter** , which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

TensorFlow Lite is designed to make it easy to perform machine learning on devices, "at the edge" of the network, instead of sending data back and forth from a server. For developers, performing machine learning on-device can help improve:

* *Latency:* there's no round-trip to a server
* *Privacy:* no data needs to leave the device
* *Connectivity:* an Internet connection isn't required
* *Power consumption:* network connections are power hungry

TensorFlow Lite works with a huge range of devices, from tiny microcontrollers to powerful mobile phones.

Key features

* [*Interpreter*](https://www.tensorflow.org/lite/guide/inference) *tuned for on-device ML*, supporting a set of core operators that are optimized for on-device applications, and with a small binary size.
* *Diverse platform support*, covering [Android](https://www.tensorflow.org/lite/guide/android) and [iOS](https://www.tensorflow.org/lite/guide/ios) devices, embedded Linux, and microcontrollers, making use of platform APIs for accelerated inference.
* *APIs for multiple languages* including Java, Swift, Objective-C, C++, and Python.
* *High performance*, with [hardware acceleration](https://www.tensorflow.org/lite/performance/gpu) on supported devices, device-optimized kernels, and [pre-fused activations and biases](https://www.tensorflow.org/lite/guide/ops_compatibility).
* *Model optimization tools*, including [quantization](https://www.tensorflow.org/lite/performance/post_training_quantization), that can reduce size and increase performance of models without sacrificing accuracy.
* *Efficient model format*, using a [FlatBuffer](https://www.tensorflow.org/lite/convert/index) that is optimized for small size and portability.
* Pre-trained models for common machine learning tasks that can be customized to your application.

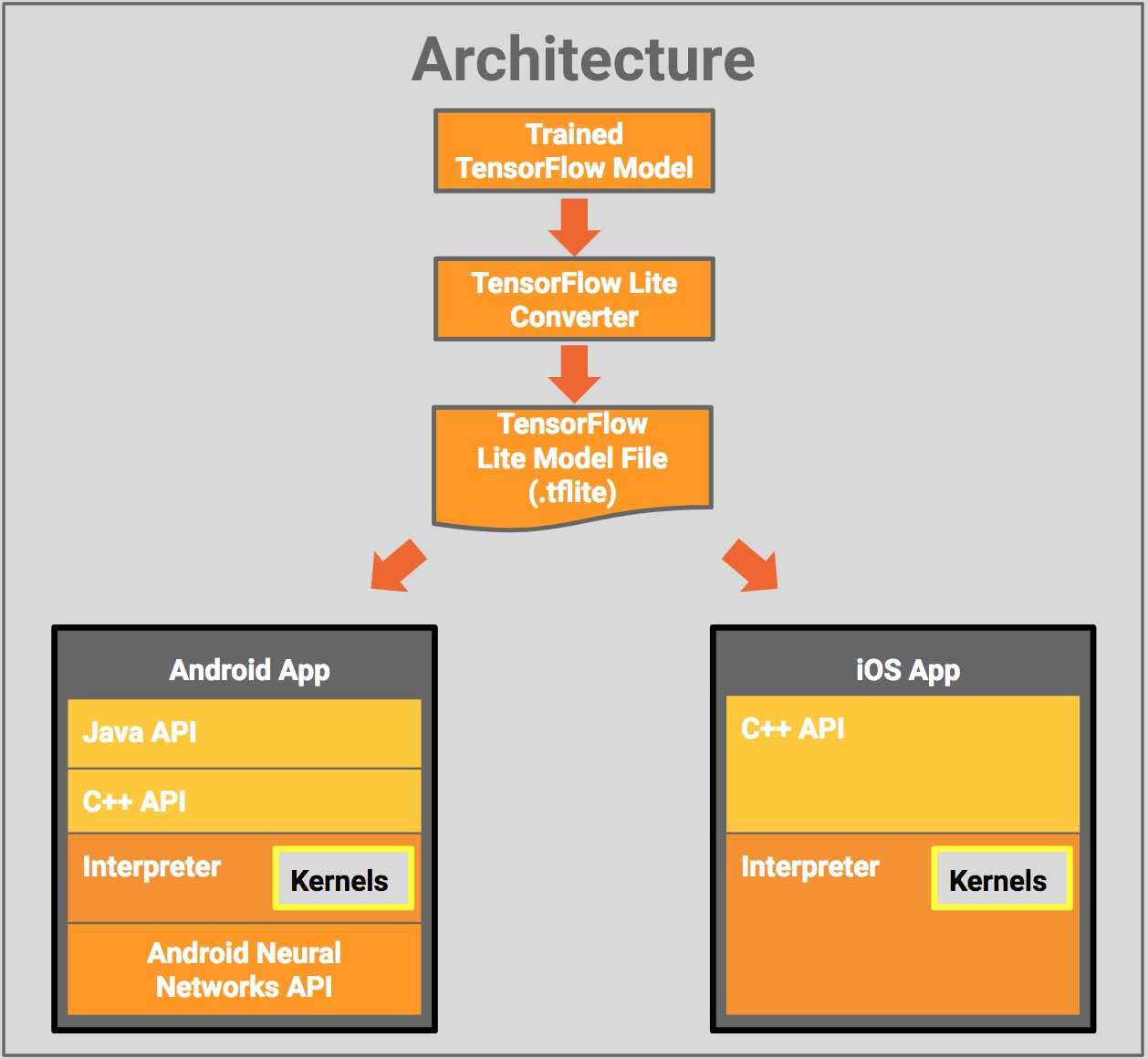


Fig: Tensorflow Lite Architecture

**CHAPTER 4**

**SYSTEM DESIGN**

**System design** is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

**4.1 Architecture**

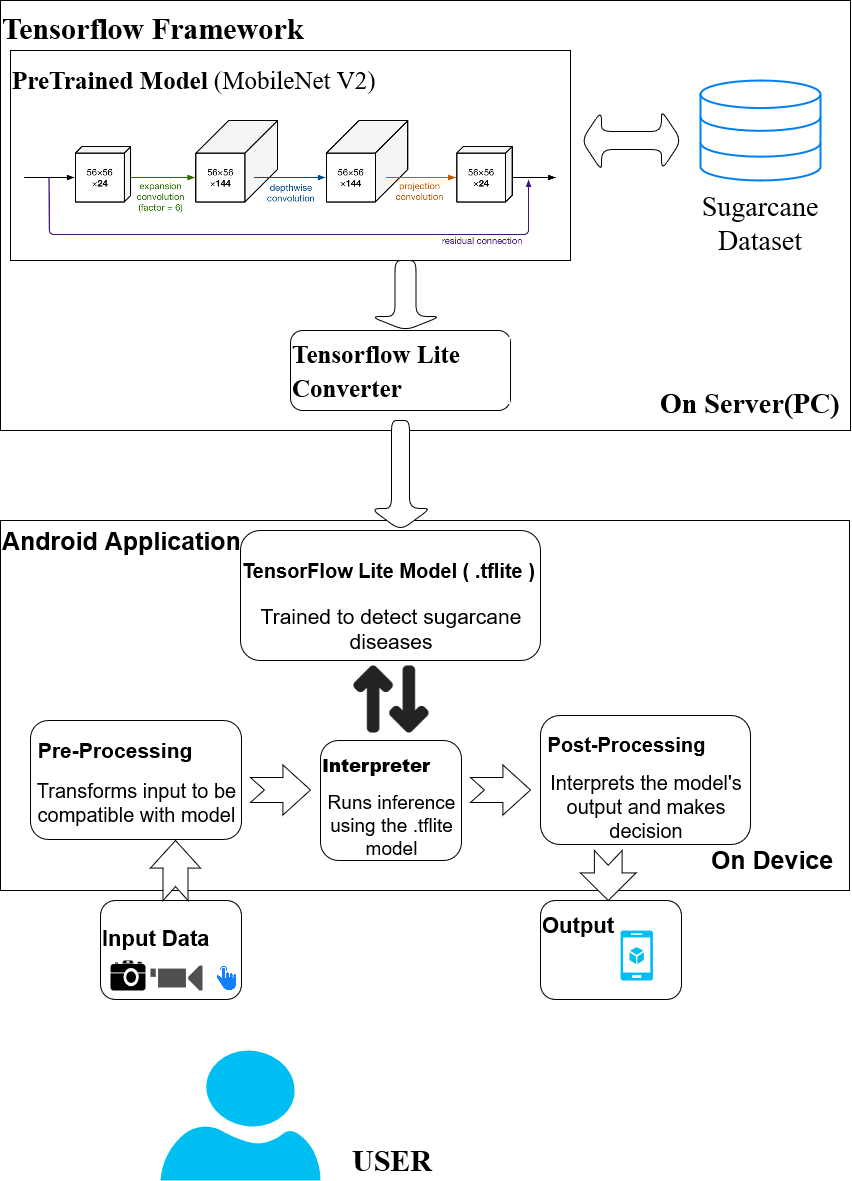
****

FIGURE 4.1: SYSTEM ARCHITECTURE OF PROPOSED SYSTEM.

The components of the system architecture are explained below:

1. **USER:** The user here is the client who would be using the android application. The user needs to upload an image of the region to be inspected for diseases in a sugarcane crop. On which the inference will be returned to the user on device.
2. **Android Application:** It consists of the TensorFlow Lite model which will perform on device inference by invoking the TensorFlow Lite Interpreter. It also pre-processes the input data to be compatible with the .tflite model and post-processes the output result and displays the final Result to the user.
3. **TensorFlow Framework:** A pre-trained model( here, **MobileNet V2)** is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. This pre-trained model is the base model on which the Sugarcane dataset is trained to obtain a custom model. This is a resource intensive process that occurs in the Server machine and is known as Transfer Learning. The Custom CNN model is saved and converted to .tflite format by TensorFlow Lite Converter.

## 4.2 Major Algorithm

# MobileNetV2 Model

MobileNetV2 is a neural network model developed at Google, and pre-trained on the ImageNet dataset, a large dataset of 1.4M images and 1000 classes of web images. MobileNets are small, low-latency, low-power models parameterised to meet the resource constraints of a variety of use cases that is optimised for mobile devices.

MobileNetV2 improves the state-of-the-art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. It is a very effective feature extractor for object detection and segmentation. For instance, for detection, when paired with Single Shot Detector Lite, MobileNetV2 is about 35 percent faster with the same accuracy than MobileNetV1.

It builds upon the idea of using depth-wise separable convolutions as efficient building blocks. MobileNet has two new features:

* Linear bottlenecks between the layers: Experimental evidence suggests that using linear layers is crucial as it prevents nonlinearities from destroying too much information. Using non-linear layers in bottlenecks indeed hurts the performance by several percent, further validating our hypothesis
* Shortcut connections between the bottlenecks

### The Basic Structure of MobileNetV2

The bottlenecks of the MobileNetV2 encode the intermediate inputs and outputs while the inner layer encapsulates the model’s ability to transform from lower-level concepts such as pixels to higher level descriptors such as image categories. With traditional residual connections, shortcuts enable faster training and better accuracy.

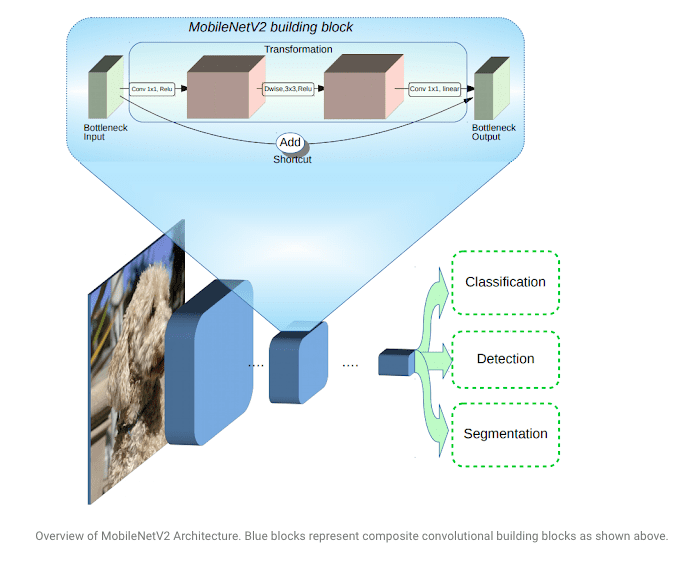


Fig: MobileNet V2 Architecture

The basic building block is a bottleneck depth-separable convolution with residuals. The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. The researchers have tailored the architecture to different performance points, by using the input image resolution and width multiplier as tunable hyperparameters, that can be adjusted depending on desired accuracy or performance trade-offs. The primary network  (width multiplier 1, 224 × 224), has a computational cost of 300 million multiply-adds and uses 3.4 million parameters. The network computational cost ranges from 7 multiply-adds to 585M MAdds, while the model size varies between 1.7M and 6.9M parameters.

MobileNet has several properties that make it suitable for mobile applications and allows very memory-efficient inference and utilises standard operations present in all neural frameworks. Thus MobileNetV2 provides a very efficient mobile-oriented model that can be used as a base for many visual recognition tasks like disease detection in plants.

**CHAPTER 5**

**IMPLEMENTATION**

**Module 1: Image Acquisition**

Appropriate datasets are required at all stages of image recognition research, starting from training phase to evaluating the performance of recognition algorithms. All the images collected for the dataset were manually clicked from the fields. Images in the dataset were grouped into four different classes addressing four major sugarcane diseases *Eye spot, Red rot,White rust* and *Yellow leaf* .

In order to distinguish healthy leaves from diseased ones, one more class was added in the dataset. It contains only images of healthy leaves. An extra class in the dataset with background images was beneficial to get more accurate classification. Thus, deep neural network could be trained to differentiate the leaves from the surrounding.

Next step was to enrich the dataset with augmented images. The main goal of the presented study is to train the network to learn the features that distinguish one class from the others. Therefore, when using more augmented images, the chance for the network to learn the appropriate features has been increased.

**Module 2: Image Augmentation and Pre-processing**

The main purpose of applying augmentation is to increase the dataset and introduce slight distortion to the images which helps in reducing overfitting during the training stage. In machine learning, as well as in statistics, overfitting appears when a statistical model describes random noise or error rather than underlying relationship . The image augmentation contained one of several transformation techniques including flipping, rotating etc. Functions of keras like ImageDataGenerator(),fit(),keras.preprocessing are used to generate and export augmented image files to a folder in order to build up a giant dataset of altered images.

Furthermore, procedure of image preprocessing involved cropping of all the images manually, making the square around the leaves, in order to highlight the region of interest. During the phase of collecting the images for the dataset, images with smaller resolution and dimension less than 500 px were not considered as valid images for the dataset. In addition, only the images where the region of interest was in higher resolution were marked as eligible candidates for the dataset. In that way, it was ensured that images contain all the needed information for feature learning. Images used for the dataset were image resized to 224 X 224 to standardize the dataset images.

**Module 3: Training the model for disease identification and classification**

Training the deep convolutional neural network for making an image classification model from the dataset described. There are several well-known state-of-the-art deep learning based pre-trained models ,of which MobilenetV2 is suitable for both research experiments and industry deployment**.** The pre-processed images were then fed into the MobileNetV2 which is pre-trained on the ImageNet dataset, a large dataset of 1.4M images and 1000 classes of web images.

MobileNetV2 architecture is considered a starting point, but modified and adjusted to support our 4 categories of diseases.

Sophisticated deep learning models have millions of parameters (weights) and training them from scratch often requires large amounts of data of computing resources. Transfer learning is a technique that shortcuts much of this by taking a piece of a model that has already been trained on a related task and reusing it in a new model.

For the Sugarcane disease classification model, the intermediate layer of MobileNet V2 will be used for feature extraction. A common practice is to use the output of the very last layer before the flatten operation, the so-called "bottleneck layer". The reasoning here is that the following fully-connected layers will be too specialized to the task the network was trained on, and thus the features learned by these layers won't be very useful for a new task. The bottleneck features, however, retain much generality.

Thus an instantiated MobileNet V2 model pre-loaded with weights trained on ImageNet. By specifying the include\_top=False argument, we load a network that doesn't include the classification layers at the top, which is ideal for feature extraction. A classification head is added on top of this base model(MobileNet V2) with 4 layers for classification.

* **A convolutional layer(Conv2D)**  that extracts features from the image or parts of an image.
* A **subsampling or pooling layer** that reduces the dimensionality of each feature to focus on the most important elements (Here, GlobalAveragePooling2D Layers).
* **A Dropout layer is used to** remove a random number of neurons from the neural network. This works very well for two reasons: The first is that neighbouring neurons often end up with similar weights, which can lead to overfitting, so dropping some out at random can remove this. The second is that often a neuron can over-weigh the input from a neuron in the previous layer, and can over specialize as a result. Thus, dropping out can break the neural network out of this potential negative impact.
* And finally a **fully connected Dense layer** that takes a flattened form of the features identified in the previous layers, and uses them to make a prediction about the image.

Finally to increase the performance even further, train (or "fine-tune") the weights of the top layers of the pre-trained model alongside the training of the classifier layer added. The training process will force the weights to be tuned from generic features maps to features associated specifically to Sugarcane dataset. The trained model is then obtained as the saved model.

**Module 4: Deployment of trained model into Android application.**

The Trained model which is obtained in form of saved model is not compatible to integrate with Android Application. Hence, TensorflowLite framework is used to convert the saved model into .tflite format which can be deployed into the android Application.

TensorFlowLite has two main components. TensorflowLite convertor and TensorflowLite interpreter. The **TensorFlow Lite converter** , converts TensorFlow models(the saved model) into an efficient form(.tflite) for use by the interpreter, and can introduce optimizations to improve binary size and performance. The **TensorFlow Lite interpreter** then runs specially optimized models on mobile phones to identify and classify the images into appropriate category of disease based on the pre-trained model analysis.

**Module 5:** **Identification and Classification of diseased plants with suggested remedies.**

By this phase, the trained model is integrated with the android application. In this phase the images to be tested are clicked and uploaded to the application after granting the required permissions to it. The test images are then processed according to the trained model and are classified into a particular category of identified disease which is then returned to the end user through the Application UI. The application is also fed with the information regarding the diseases classified and their remedial measures. Based on the disease classified the application fetches the relevant information from the data fed and returns the suggested remedies to the end user via the application UI.

**CHAPTER 6**

**TESTING**

In this chapter, an overview of testing is provided to verify the correctness and the functionality of the system. Software testing is the process of analysing a software item to detect the differences between the existing and the required conditions and to evaluate the features of the software item. Software testing is an activity that should be done throughout the development process. Software testing is a task intended to detect defects in software by contrasting a computer program’s expected results with its actual results for a given set of inputs.

## Test Environment:

## The software was tested on the following Environments:

1. Python 3.7
2. Windows environment.

**Test Case:**

Set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

* Features to be tested
* Purpose of testing
* Pass/Fail Criteria

**6.1 Unit Testing**

Unit testing is the testing of individual hardware or software units or groups of related units. Using the unit test plans prepared in the design phase, important control paths are tested to uncover errors within the boundary of the modules. The interfaces of each of the modules are tested to ensure proper flow of the information into and out of the modules under consideration.

Each unit in this project was thoroughly tested to check if it might fail in any possible situation. This testing was carried out at the completion of each unit. At the end of the unit testing phase, each unit was found to be working satisfactorily in regard to the expected output from the module.

**Table** **6.1:** **Gives test cases for Unit testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE ID** | **STEP DESCRIPTION** | **EXPECTED RESULT** | **ACTUAL RESULT** | **PASS/FAIL** |
| 1 | User give the recorded audio files to pre-processing unit. | An audio files with removed noises and enhancement of speech signal. | An audio files without noise and enhancement in the speech signal | PASS |
| 2 | Pre-processed audio files are moved to feature extractor module. | The features that are to be extracted are the MFCCs, zero crossing rate, and the spectral centroid. | 13 MFCCs coefficients, zero crossing rate and spectral centroid are extracted. | PASS |
| 3 | Extracted features are passed on to the machine learning model for training. | The model is trained on the basis of the features extracted from previous module. | The model is trained. | PASS |
| 4 | Voice data will be passed to the classifier module. | The data is classified into one of the seven classes of emotions Anger, Happiness, Disgust, Sad, Fear, Boredom and Neutral. | The data is pre-processed and classified. | PASS |

**6.2 Integration Testing**

Integration testing is the testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them. The various modules are tested for their accuracy and compatibility. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together.

**6.3 System Testing**

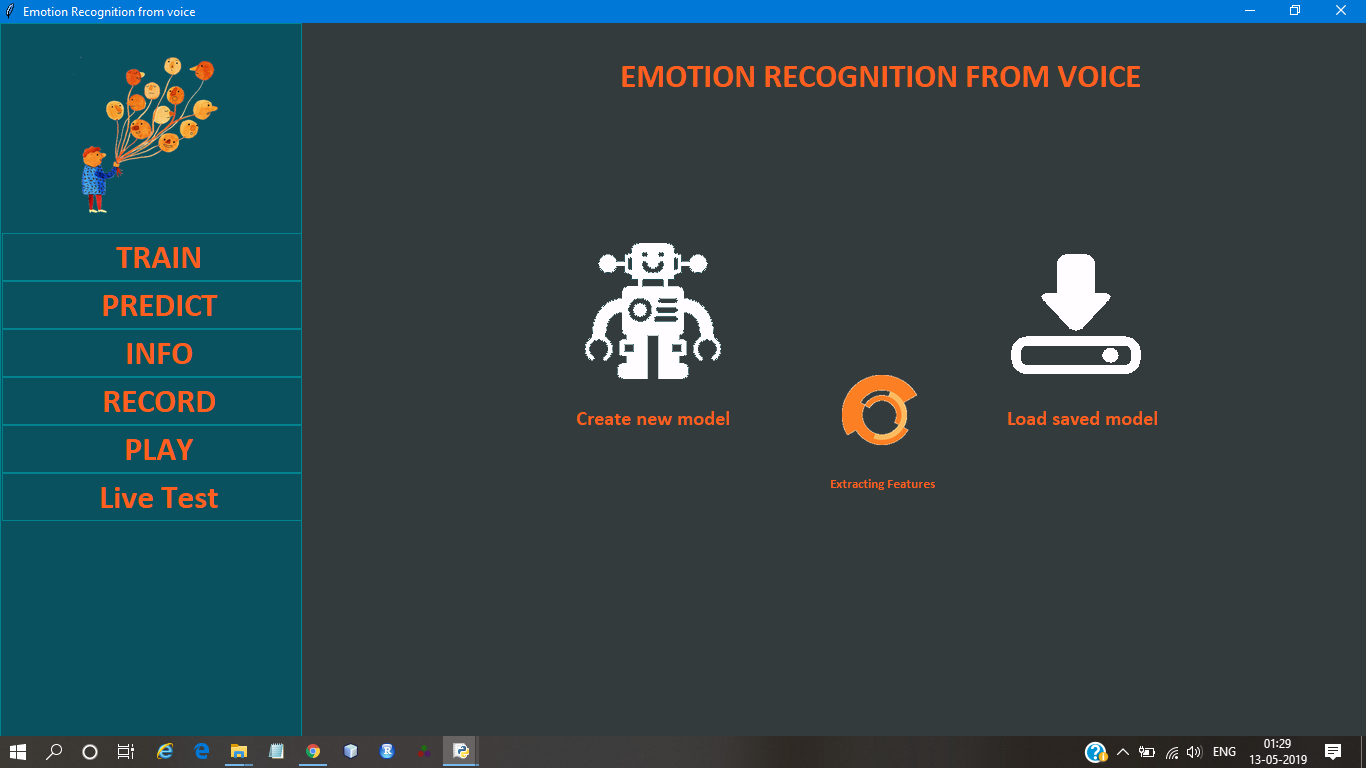
System testing is the testing conducted on a complete, integrated system to evaluate the system compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passes integration testing.

**Table 6.3: System test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST CASE ID** | **STEP DESCRIPTION** | **INPUT** | **EXPECTED RESULT** | **ACTUAL RESULT** | **PASS/FAIL** |
| 1 | Test case for recording, pre-processing, features extracting and classification. | Audio files. | The test data is pre-processed and classified. | The test data is classified accurately. | PASS. |

**SNAPSHOTS**

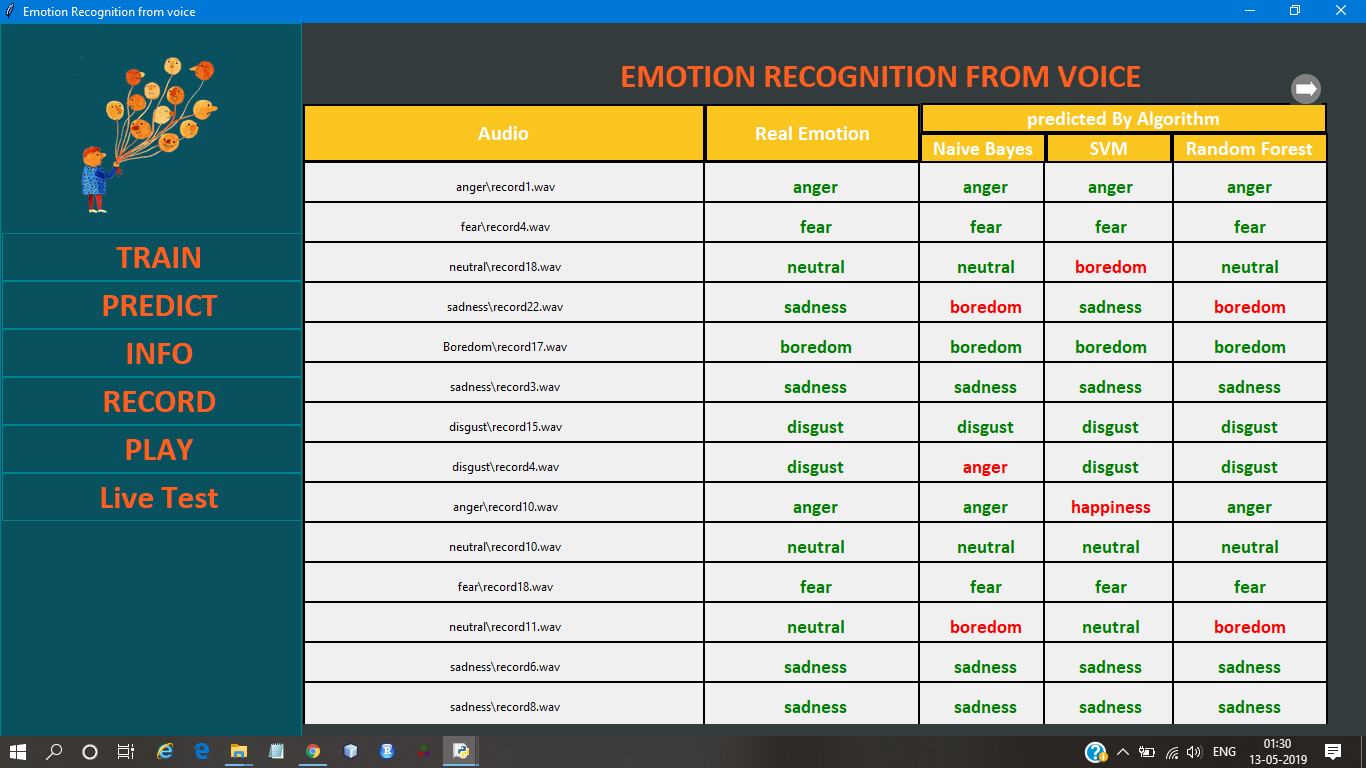
**6.1 Welcome Page**



**Figure: 6.1 welcome page**

The welcome page presents the user with the navigation bar from where the user has access points to various functionalities of the project – training the model, predictions associated with test cases, visual information associated with each test audio, recording live voice for predicting the underlying emotion. The user is required to train the machine learning model before the actual classification can be performed.

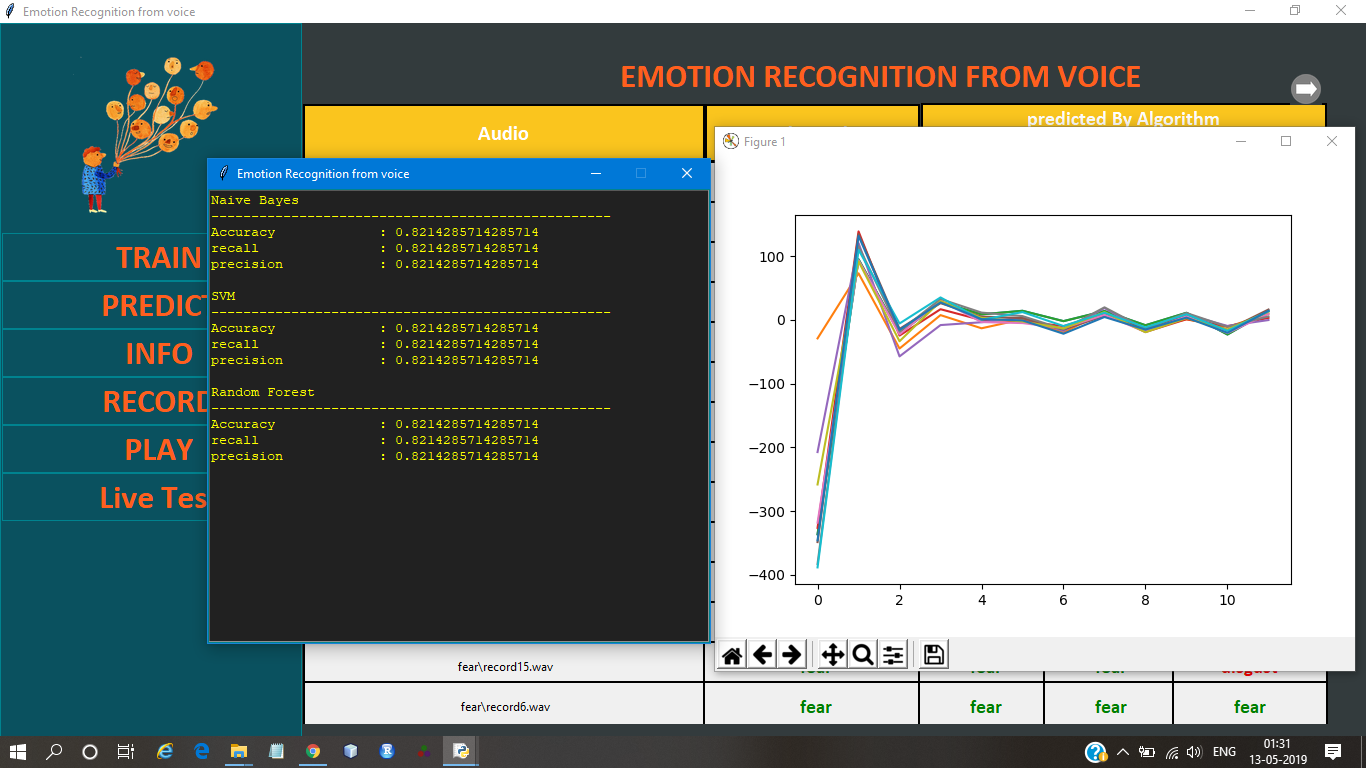
**6.2 Predictions Associated With the Test Set**



**Figure: 6.2. Predictions associated with the test set**

The predictions associated with each audio file of the test set are shown. The actual emotion, and the emotions to which all the three classifiers- Naïve Bayes, Support vector machines, and random forest classifier, are shown. The correctly classified instances are marked green. Those incorrectly classified are marked red. On clicking a particular test instance, the visual information associated with the features extracted with respect to the emotion into which it is classified, are displayed.

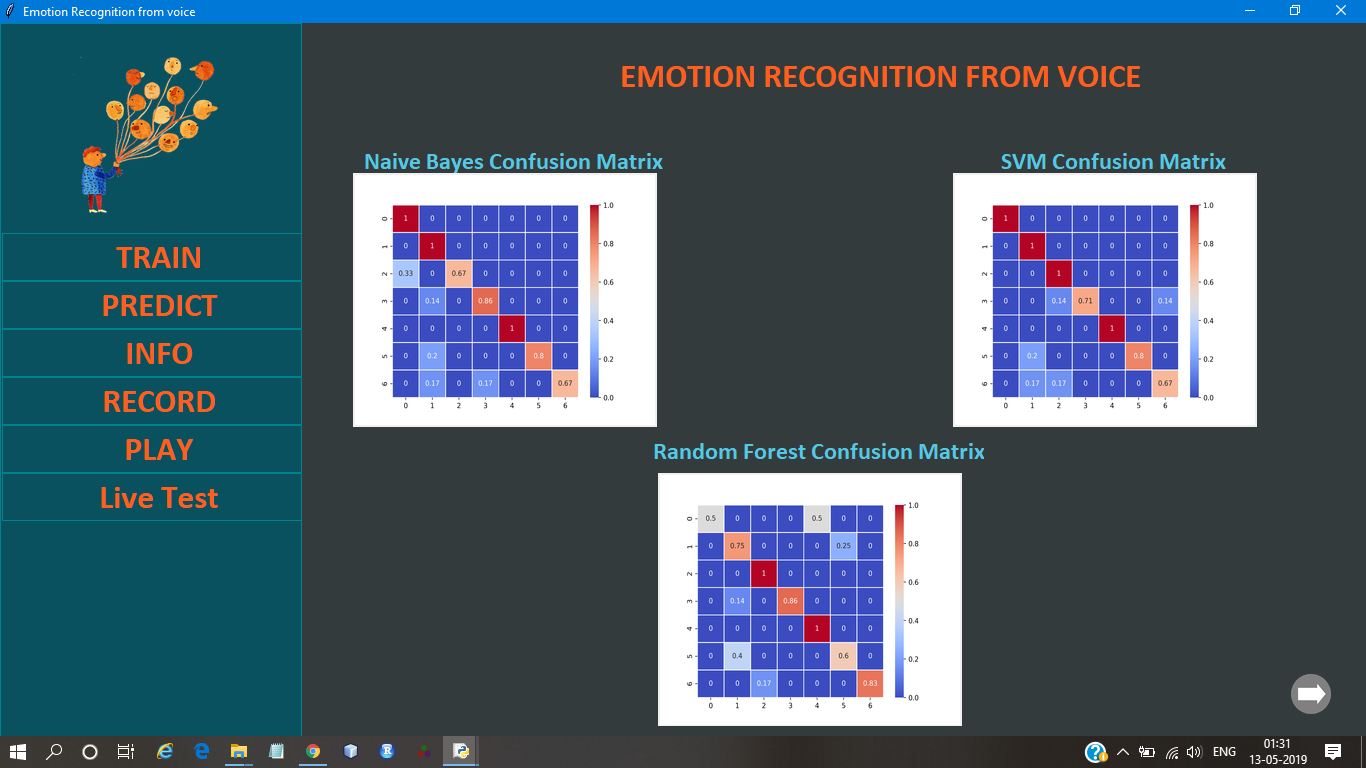
**6.3 Analysis of the Test Instances**



**Figure: 6.3. Analysis of the test instances**

The analytical information associated with each test instance can be viewed by clicking on that particular instance. A window is displayed that represents the values of the performance parameters obtained over the test set, for each of the three classifiers. These include Accuracy, precision and recall. The value of each coefficient of the MFCCs associated with a particular instance, is shown upon mouse click.

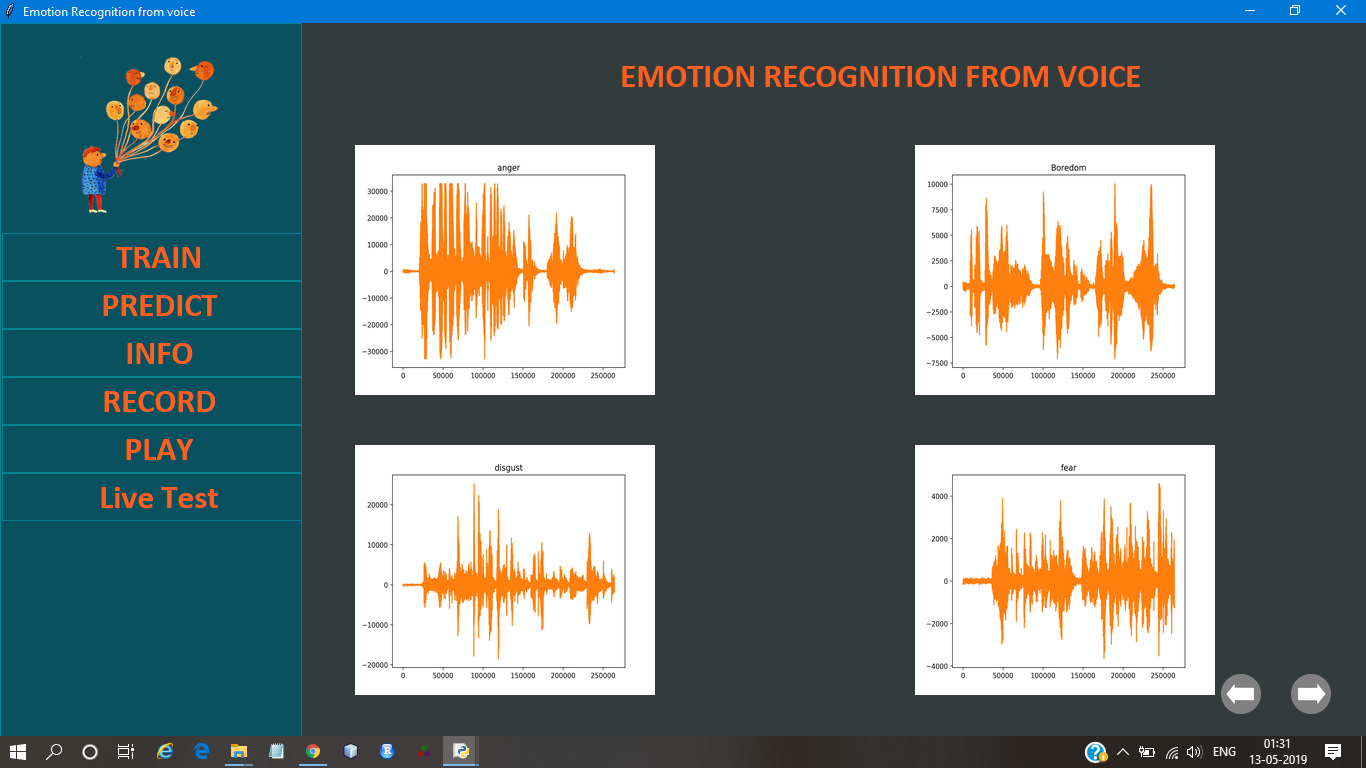
**6.4. Confusion Matrices for Each Classifier**



**Figure: 6.4. Confusion Matrices for each classifier**

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes. The confusion matrices are shown for each of the three classifiers - naïve Bayes, support vector machines and random forest classifier.

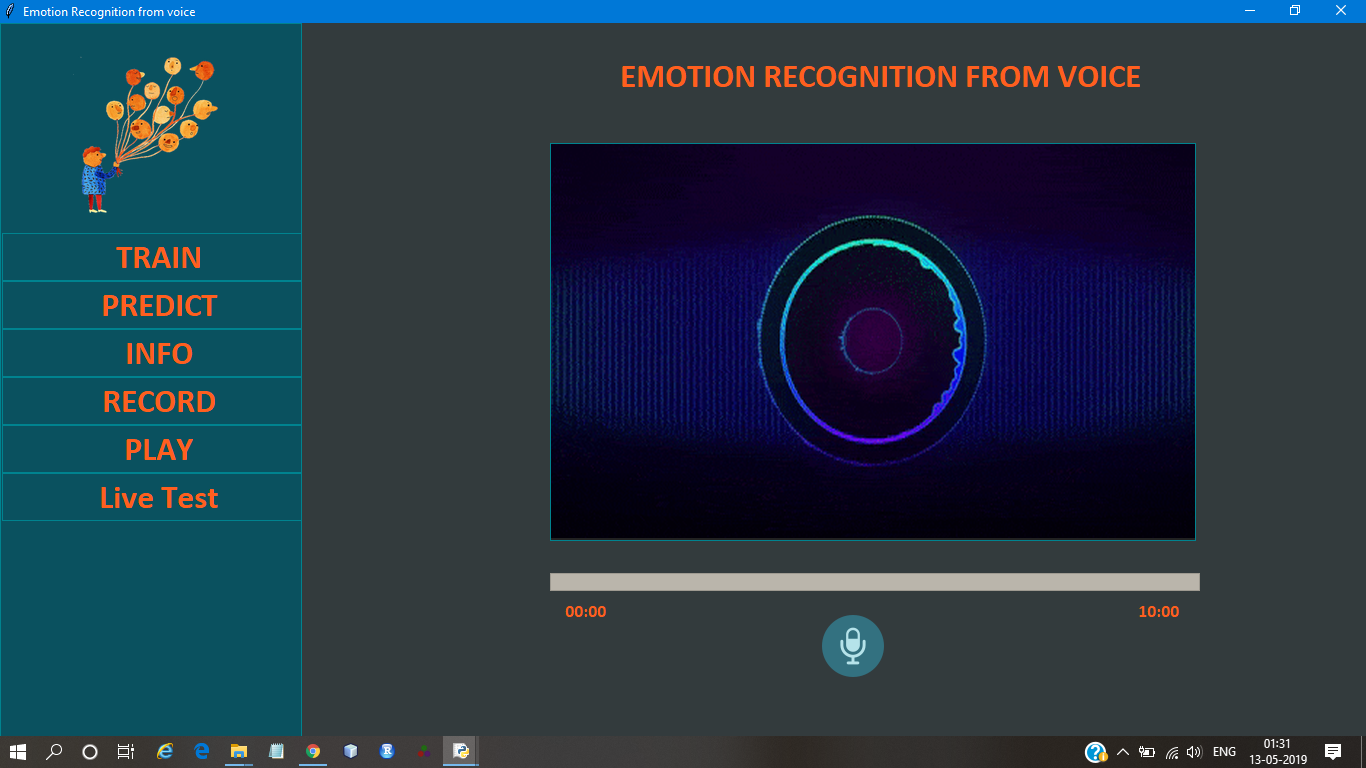
**6.5 Analytical Information of Each Emotion**



**Figure: 6.5 Analytical information of each emotion**

The auditory information associated with each emotion, gathered from the testing data, visualized as a plot between time and frequency is shown. The difference is clearly visible, and forms the basis of differentiability between two or more emotions.

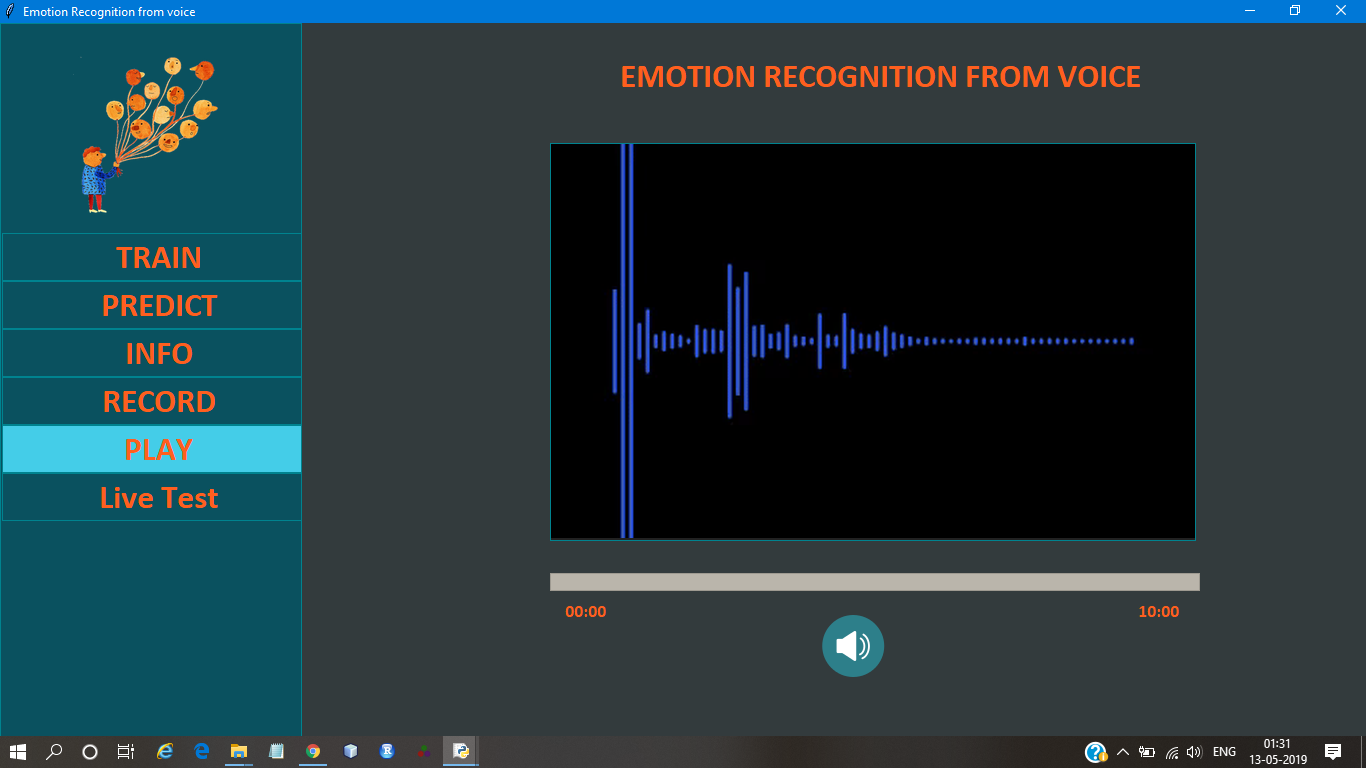
**6.6 Recording Phase**



**Figure: 6.6. Recording phase**

The recorder module enables the user to record his voice through the microphone. The audio recording takes place for a period of ten seconds. The file stored as wav is retrieved for classification by the classification module- the live test phase. The user may however, also input the audio from the file system.

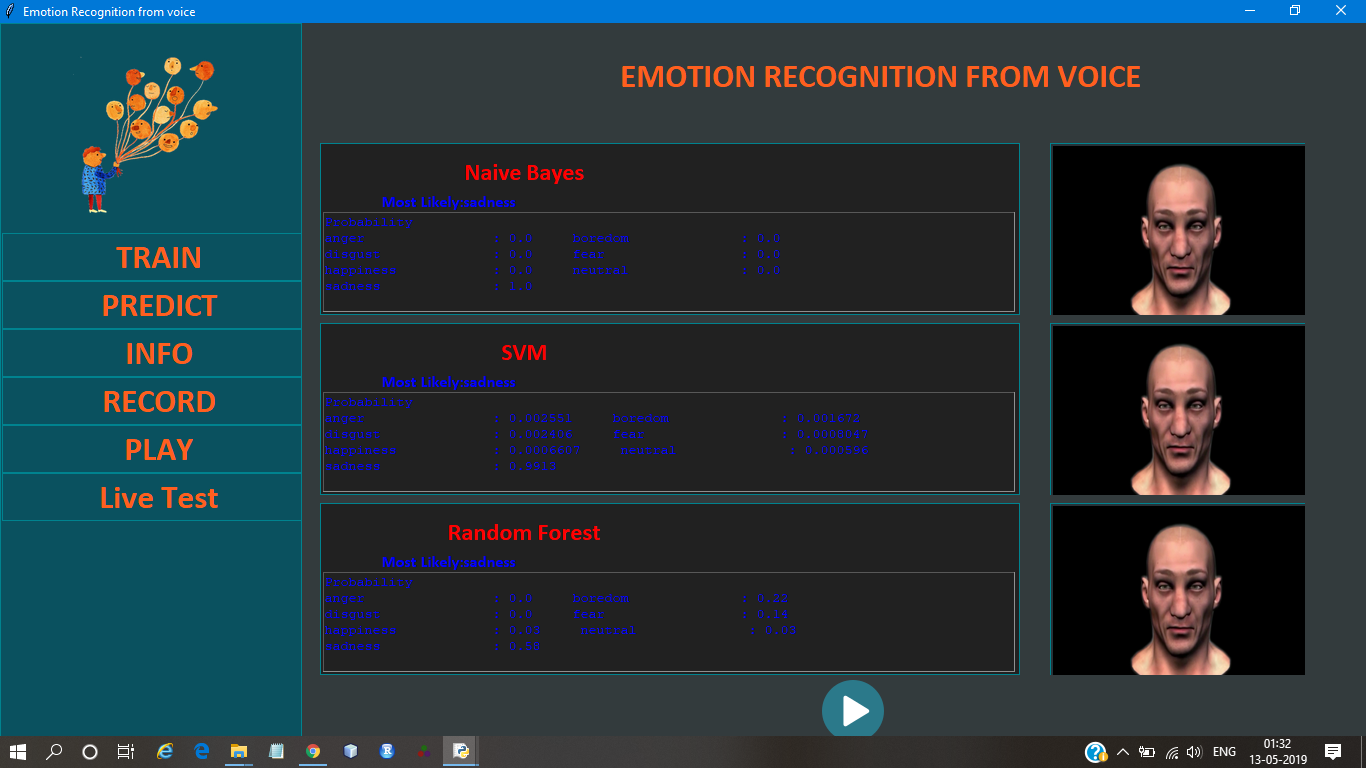
**6.7 Playback of the Recording**



**Figure: 6.7 Playback of the recording**

The recorded voice can be played back. Any speaker, either in built or an externally connected speaker is necessary to produce the sound. The variation of the amplitude with time is shown as a dynamically changing graph.

**6.8 Results of the live test**



**Figure: 6.8. Results of the live test**

The results obtained by classifying the live test instances, for each classifiers, is shown. The normalized probabilities with which the given live test instance lies into each of the six classes of emotion, are shown for each classifier. A 3d animated face model that expresses the most probable emotion into which the live test audio falls into, is shown for each classifier.

**CHAPTER 7**

**CONCLUSION AND FUTURE WORKS**

**7.1 Conclusion**

The problem of early stage disease identification in crops has always been a challenging task to the farmers. The proposed system is an efficient solution to address this issue. It automates the disease detection process by comparing the suspected plant features with the features exhibited by thousands of diseased plants and provides the results to farmers with a high accuracy ,thus helping the Farmers to take the preventing measures at the early stage to avoid crop loss. The System is designed to be a user friendly application with an interactive and easy to use UI. The system provides easily accessible solutions as it is a self inference system and does not require any internet accessibility to get the solutions .Thus, it can be concluded that the system is an efficient aid to farmers (regardless of the level of experience), enabling fast and efficient recognition of plant diseases and facilitating the decision-making process

**7.2 Future Enhancement**

An extension of this study will be on gathering images for enriching the database .The main goal for the future work is to enhance the system to address other diseases.

**APPENDICES**

**ABBREVIATIONS**

|  |  |
| --- | --- |
| **IDE** | Integrated Development Environment |
| **MFCCs** | Mel frequency cepstral coefficients |
| **SVM** | Support Vector Machine |
| **OS** | Operating System |
| **logMMSE** | Log minimum mean square error. |
| **URI** | Uniform Resource Identifier |
| **URL**  **OOB** | Uniform Resource Language  Out of Bag |
| **RBF** | Gaussian radial basis function |

**LANGUAGE DESCRIPTION**

**Core Tools and Technologies**

This section covers the complete development matrix. It identifies the complete technologies elements with guidelines and specifications for specific implementation.

Python includes development tools that help to implement the algorithm efficiently. These include the following:

**Code Analyser**

Checks the code for problems and recommends modifications to maximize performance and maintainability.

**Language Specification**

The Python language supports the vector and matrix operation functional to engineering and scientific problems. It enables fast development and execution.

**The Python Programming Language**

The Python programming language is a general purpose and high-level language that can be characterized by all of the following buzzwords.

* Simple
* Object oriented
* Free and open source
* Extensible
* Expressive language
* Interpreted language
* Cross platform language
* Large Standard Library
* Integrated
* GUI Programming support

In the python programming language, all source code is first written in plain text files ending with the .py extension.

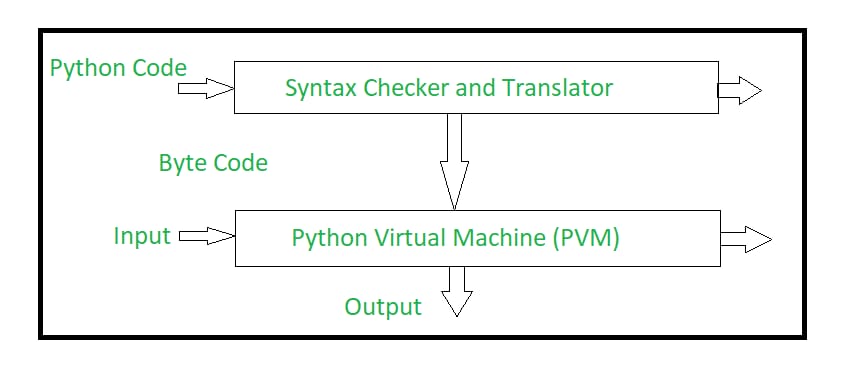
Python does not convert its code into machine code, something that hardware can understand. It actually converts it into something called byte code.

The Python interpreter performs following tasks to execute a Python program:

**Step 1:** The interpreter reads a python code or instruction. Then it verifies that the instruction is well formatted, i.e. it checks the syntax of each line. If it encounters any error, it immediately halts the translation and shows an error message.

**Step 2:** If there is no error, i.e. if the python instruction or code is well formatted then the interpreter translates it into its equivalent form in intermediate language called “Byte code”. Thus, after successful execution of Python script or code, it is completely translated into Byte code.

**Step 3:** Byte code is sent to the Python Virtual Machine (PVM).Here again the byte code is executed on PVM. If an error occurs during this execution then the execution is halted with an error message.



**An overview of the python program conversion**

**BIBLIOGRAPHY**

**Review through Books**

1. Makoto SHIMURA, Fumiaki MONMA, Shunji MITSUYOSHI, Masaki SHUZO, Taishi YAMAMOTO, Ichiro YAMADA, “Descriptive Analysis of Emotion and Feeling in Voice” The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan, AGI Inc., 6-3-13-4F Akasaka, Minato-ku, Tokyo 107-0052, Japan.
2. Subhasmita Sahoo,Aurobinda Routray “MFCC Feature With Optimized Frequency Range: An Essential Step For Emotion Recognition”.
3. M.S. Likitha, Sri Raksha R. Gupta, K. Hasitha and A. Upendra Raju “Speech Based Human Emotion Recognition Using MFCC”.
4. Eun Ho Kim, Kyung Hak Hyun and Yoon Keun Kwak “Improvement ofEmotion Recognition from Voice by Separating of Obstruents”.
5. Aishah AM. Razak, Mohd Hafizuddin Mohd Yusof, Prof. Ryoichi Komiya, “Towards Automatic Recognition of Emotion in Speech”.
6. Aishah Abdul Razak,Ryoichi komiya,Mohamad Izani zainal abidin “Comparison Between Fuzzy and NN Method for Speech Emotion Recognition”.
7. Zygmunt Ciota “Emotion Recognition on the Basis of Human Speech”.
8. Rupayan Chakraborty, Meghna Pandharipande, Sunil Kopparapu “EVENT BASED EMOTION RECOGNITION FOR REALISTIC NON·ACTED SPEECH”.
9. Pavol Harár, Radim Burget and Malay Kishore Dutta “Speech Emotion Recognition with Deep Learning”.
10. J.C. V´asquez-Correa1, N. Garc´ıa, J.R. Orozco-Arroyave, J.D. Arias-Londo˜no, J.F. Vargas-Bonilla, Elmar N¨oth “Emotion Recognition from Speech under Environmental Noise Conditions using Wavelet Decomposition”.

**Review through Web Reference:**

* msdn.microsoft.com
* https://www.datacamp.com/community/tutorials/rand om-forests-classifier-python
* https://www.stat.berkeley.edu/~breiman/RandomFore sts/cc\_home.htm