AROGYA - AN INTELLIGENT AYURVEDIC HERB MANAGEMENT PLATFORM

N.J. Pathiranage - (IT17129404)

Nilfa M.S.F – (IT17145930)

R.A. Mithula Nithmali – (IT17089500)

K.A.G.Y. Nadee Kumari – (IT17014250)

BSc (Hons) in Information Technology

Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

September 2020

AROGYA – AN INTELLIGENT AYURVEDIC HERB MANAGEMENT PLATFORM

N.J. Pathiranage - (IT17129404)

Nilfa M.S.F – (IT17145930)

R.A. Mithula Nithmali – (IT17089500)

K.A.G.Y. Nadee Kumari – (IT17014250)

(Final documentation in partial fulfilment of the requirement for the Degree of Bachelor of Science Special (honors) In Information Technology Specializing in Software Engineering)

BSc (Hons) in Information Technology

Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

September 2020

DECLARATION OF THE CANDIDATE AND SUPERVISOR

We declare that this is our own work and this proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
N.J. Pathiranage (Leader)	IT17129404	
Nilfa M.S. F	IT17145930	
R.A. Mithula Nithmali	IT17089500	
K.A.G.Y. Nadee Kumari	IT17014250	

The supervisor/s should certify the proposal report with the following declaration.

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

…….. ……

Signature of the supervisor:

(Mrs. Lokesha Weerasinghe) Date

…….. ……

Signature of the co-supervisor:

(Ms. Ishara Weerathunga) Date

ABSTRACT

Ayurvedic means a science of life and well-being with its unique approaches to social and spiritual life. There is a considerable species of plants in our environment which will serve us as home remedies for almost all diseases and keeps us healthy when taken for daily consumption. Especially in Sri Lanka we have our own set of rare ayurvedic herbs which have been utilized by generations as medicinal treatments for a variety of diseases. Absence of specialists in this area makes proper identification as well as classification of valuable herbal plants a tedious task, which is essential for better treatment. Hence, a fully automated system for medicinal plant detection and classification, information visualization regarding medicine, track geographical locations of newly identified plants, as well as display the most used herbal sample of each category of disease and visualizing some of herbal remedies related to them are highly desirable. There are existing applications which can identify plants with low prediction accuracies, as well as to give information regarding them. However, these applications are based on foreign plant data sets that do not include valuable herbs and shrubs with medicinal qualities. Hence this research proposes a centralized social media application unique to medicinal plants, which can perform all these functionalities in both online and offline approach. Here, a new ayurvedic plant dataset prepared from scratch, and preliminary results for classification of 5 types of herbs, compared with several CNN models based on transfer learning are presented. Experimental results indicate Marker-based Watershed algorithm as the best leaf detection algorithm in a complex background, VGG-16 as the best CNN classification model which reached a promising testing accuracy of 99.53%, and Seq2Seq LSTM model as the best deep learning model with optimum accuracy in abstractive information summarization. Conclusively, the outcome of this study will be used by locals to identify, track and use herbal plants, recipes and remedies precisely.

Keywords: abstractive information summarization, ayurvedic medication, classification, detection, geographical information system, transfer learning, object detection

ACKNOWLEDGEMENT

Several people played important roles in accomplishing this research. First, I would like to express my deepest appreciation and full-hearted gratitude to my supervisor Mrs. Lokesha Weerasinghe and co-supervisor Ms. Ishara Weerathunga who guided us to success in this research and gave entire support to complete this project. Also, I would like to thank Sri Lanka Institute of Information Technology for encouraging us to carry out this study and the academic staff for the continuous guidance, support, and insightful comments, which motivated us to move further. Furthermore, I am grateful to all the resource people of Ayurveda Research Institute at Navinna for giving an ultimate support for collecting relevant information as well as photographs of some selected valuable ayurvedic plants in Sri Lanka. Finally, I wish to extend my warm appreciation to my family members for their encouragement and support to complete this task.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN-Artificial Neural Network

CNN-Convolutional Neural Network

MSF-CNN-Multi-Scale fusion Convolutional Neural Network

DNN-Deep Neural Network

WHO-World Health Organization

HOG- Histograms of Oriented Gradients

LBP- Local Binary Patterns

SVM- Support Vector Machines

ELM- Extreme Learning Machines

PCA- Extreme Learning Machines

KNN- K-Nearest Neighbor

PNN- Probabilistic Neural Network

CBIR- Content-Based Image Retrieval

1. INTRODUCTION

Ayurveda is an ancient medicinal system evolved in India around thousands of years ago, still followed by many people as it is purely natural and has no side effects [2]. It is very relevant from ancient to this most modern time because of its power to cure chronic diseases [2]. The parts like leaf, flower, root, bark, and fruit are mainly used in the preparation of medicines in Ayurveda [2]. In 2009, World Health Organization (WHO) has stated that 80% of the people worldwide still count on ayurvedic drugs and medicine. They have remarkable contributions towards human lives and play a predominant role for the health of the global inhabitants.

According to Ayurveda every plant on earth has some medicinal value, so it is important to protect the plant and identify its medicinal values [30]. Studies have proved that consuming so much of allopathic medicines may lead to side effects as it carries out many chemical reactions within the body [30]. A general fact about Allopathy is that once it is taken, it requires taking another medicine to cure the side effects which has happened due to the previous medicine [30]. In general, process of consuming medicines will not end. Allopathic treatments are meant to treat the Symptoms of a disease whereas Ayurveda treats the root of the disease [30].

One of the major advantages of Ayurveda is that it does not have any side effects as it is purely natural, that is relevant in this most modern time as new diseases evolve due to changed lifestyle and changed diet [4]. Also, majority prefer Ayurveda medicine over Western medicine due to some versatile factors such as Completely natural, Massage treatment, Positive influence on mental health, Fights inflammation, adds to weight loss, Improves heart health, Healthy detox, Individually prescribed methods, etc. [4]. So, it is important for every human being to return to Ayurveda. Almost all general diseases can be cured through Ayurveda using the shrubs and herbs that are around us. Ayurveda also brings lots of foreign money to the country since many foreign countries are inclining towards it. These days a potential class of users exists who favor ayurvedic medicines than modern medicines. Therefore, the knowledge regarding medicinal plants carried by different age groups should be preserved and protected.

Computer vision, image processing technologies as well as pattern recognition give optimistic results for the identification and classification of herbs. Recognizing a herb with required biological and medicinal values is one of the crucial tasks [20] because it plays a prominent role in the preparation of Ayurveda related medicines and recipes. In addition, real classification of

herbs is significant for almost all the persons who are involved in the preparation of herbal medicines [20]. But absence of expert taxonomists is one of the main issues in this research area. Although ayurvedic medicine has less side effects when compared with synthetic drugs, treatment using an incorrectly recognized herbal plant may claim the life of a patient.

Computer vision, pattern recognition, and image processing technologies provide promising results for identification and classification of medicinal plants. Identifying a medicinal plant with required medicinal values is one of the major challenging tasks [20]. It plays a crucial role in the preparation of ayurvedic medicines. Additionally, proper classification of medicinal plants, important information sources regarding them, knowing about the locations of their growth in Sri Lanka as well as providing medical recipes for diseases of specialized categories are important for agronomist,
botanist, ayurvedic medicinal practitioners, forest department officials and those who are involved in the preparation of ayurvedic medicines [20]. But lack of expert taxonomists is a major issue in this area. Even though herbal medicine has no side effects, treatment using a wrongly identified medicinal plant may claim the life of a patient. Hence, a fully automated system to satisfy all the above-mentioned requirements regarding the local medicinal plants is inevitable at this point of time.

Hence, the proposed solution is to develop a centralized social media platform (android mobile application) unique to herbal plants, which allows users to detect and classify a group of selected valuable ayurvedic plant species accurately, based on the photograph of the plant part (leaf, root, fruit, etc) as the basic functionality. Additionally, an abstractive summary description of the identified herb’s medicinal properties, biological value, as well as reliable information sources about remedies and recipes for specific diseases associated with those plants are provided. The normal functionalities of a social media platform like creating user accounts, publishing posts only related to herbal plants, liking, commenting, and sharing them, and dynamic search with either words or web are also available with this application.

Additionally, the user is capable of adding the locations of herbal plants into the map and track locations (availability and growth) with the aim of preservation of valuable Ayurvedic plants worldwide. So, others can use the locations of previously tracked plants, in order to know about rare and unknown species which are used as ingredients in medical recipes. Therefore, anyone

without prior knowledge also will be able to identify ayurvedic plants hopefully and use them properly in their medications. On a mobile device, the Ayurveda plant detection has to be done with time, battery life critical manner, especially when it has to be done in a forest area. In the proposed system the whole identification process takes place on mobile devices and it does not require the internet. Therefore, this will be a great solution to identify Ayurveda plants in deep forest areas, where mobile network coverage is not available. The development strategy and methodology used in this approach will be able to be used and extended to identify any ayurvedic herb furthermore.

1.1 Background & Literature Survey

An overall idea about the importance of Ayurveda in Sri Lanka, current problems faced by people in this domain, apparent tasks that should be there in any automated identification and classification, information resources, tracking locations of existing plants, herbal remedies for categories of diseases with specific characteristics as well as the limitations of previous researches carried out regarding herbal plant management and preservation worldwide have been discussed in the introduction section. This section brings several works and important attention in the focus of this research.

Following are some literature reviews done with some existing mobile applications regarding medicinal herb management, which are somewhat similar to our platform to be implemented.

In 2013, the study of “Pl@ntNet Mobile App” [40] has been proposed as an image sharing and retrieval application for the identification of plants, available on iPhone and iPad devices. Contrary to previous content-based identification applications it can work with several parts of the plant including flowers, leaves, fruits, and bark. It also allows integrating user’s observations in the database thanks to a collaborative workflow involving the members of a social network specialized on plants. Data collected so far makes it one of the largest mobile plant identification tools.

The iPhone app itself basically contains four functionalities: an image feeds reader to explore the last contributions of the community, a taxonomic browser with full text search options, a user

profile and personal contents management screen and the image-based identification tool itself. This one first asks the user to take a picture and then let him choose among 4 icons representing a flower, a leaf, a fruit and a bark. Up to five pictures of the same plant can be acquired in this way and the complete set of pictures can finally be submitted as a query plant to the remote visual search engine. Retrieved species with confidence scores and matched images are finally returned to the device and displayed on the result screen by decreasing confidence. Selecting one of the retrieved species opens a detailed view screen with all matched pictures (classified by organ galleries) allowing a first stage of refinement in the determination process. A second stage of refinement can be achieved by accessing either e Flore fact sheets (the most complete db on France flora) or Wikipedia mobile pages. If the user believes having found the right species, he can finally contribute by sending his observation with pictures, date, gps and author's name (in Creative Commons). The observation will then be integrated in the collaborative web tool.

In 2013, the research work named "MedLeaf: Mobile Application for Medicinal Plant Identification Based on Leaf Image" [42] has proposed MedLeaf as a new mobile application for medicinal plants identification based on leaf image. The application runs on the Android operating system. MedLeaf has two main functionalities, i.e. medicinal plants identification and document searching of medicinal plant. They have used Local Binary Pattern to extract leaf texture and Probabilistic Neural Network to classify the image. In this research, it has used 30 species of Indonesian medicinal plants and each species consists of 48 digital leaf images. To evaluate user satisfaction of the application they have used questionnaire based on heuristic evaluation. The evaluation result shows that MedLeaf is promising for medicinal plants identification. MedLeaf will help botanical garden or natural reserve park management to identify medicinal plant, discover new plant species, plant taxonomy and so on. Also, it will help individual, groups and communities to find unused and undeveloped their skill to optimize the potential of medicinal plants. As the results, MedLeaf will increase of their resources, capitals, and economic wealth. Keywords—Heuristic evaluation, medicinal plant, identification, Local Binary Patterns, Probabilistic Neural Network.

In 2013, in the research study of "ApLeafis: An Android-Based Plant Leaf Identification System" [43], an Android-based mobile application has been designed to automatically identify plant species by the photographs of tree leaves is described. In this application, one leaf image can be either a digital image from the existing leaf image database or a picture collected by a camera. The picture should be a single leaf placed on a light and un-textured background without other clutter. The identification process consists of totally three steps: leaf image segmentation, feature extraction, and species identification. The demo system is evaluated on the ImageCLEF2012 Plant Identification database which contains 126 tree species from the French Mediterranean area. The output of the system to users is the top several species which match the query leaf image the best,

as well as the textual descriptions and additional images about leaves, flowers, etc., of theirs. The system works well with state-of-the-art identification performance.

In 2012, the study of “Leafsnap: A Computer Vision System for Automatic Plant Species Identification” [41] describes the first mobile app for identifying plant species using automatic visual recognition. The system called Leafsnap identifies tree species from photographs of their leaves. Key to this system are computer vision components for discarding non-leaf images, segmenting the leaf from an untextured background, extracting features representing the curvature of the leaf's contour over multiple scales, and identifying the species from a dataset of the 184 trees in the Northeastern United States. Their system obtains state-of-the-art performance on the real-world images from the new Leafsnap Dataset, which is the largest of its kind. Throughout the paper, it has been documented about many of the practical steps needed to produce a computer vision system such as this, which currently has nearly a million users.

1.1.1. Selecting the Most Accurate and the Highest Performance Segmentation Technique using Experimental Outcomes of Plants in Sri Lanka

Biodiversity is slowly decreasing across the globe. The current rate of extinction is largely the result of direct and indirect human activities [63]. It is necessary for the future survival of biodiversity to develop accurate knowledge of the identity and geographic distribution of plants [64]. Therefore, for efficient biodiversity research and management, rapid and precise identification of plants is important.

As in the introductory part, Researchers tried a number of methodologies to automatically extract the features and identify the plant leaf. According to previous work done, there are existing applications which can detect leaf with low detection accuracies.

However, existing applications are based on data from foreign plants, which do not include valuable herbs and shrubs of medicinal quality. Additionally, most of these methods make use of the combination of many parameters like color, shape, texture, features etc.

In a manually identification process, botanist use distinctive plant characteristics as identity keys, which might be used sequentially and adaptively to discover plant species. In essence, a person of an identification key's answering a chain of questions about one or more attributes of an unknown plant continuously focusing at the most discriminating traits and narrowing down the set of candidate species. This series of answered questions leads ultimately to the desired species.

However, the determination of plant species requires a tremendous botanical expertise. Sometimes botanists themselves species identification is often a tough task. The situation is further exacerbated by means of the increasing scarcity of professional taxonomists [65]. The declining and partially nonexistent taxonomic knowledge within the standard public has been termed "taxonomic crisis" [66].And other hand, it decreases user satisfaction about the application. Because these applications allow users to identify the plant using traits such as flower structure, leaf shape, flower color or fruit color. Sometimes users would not prefer to go several steps to identify the plant.

To overcome these issues, we are going to use object detections methods. So that user would not be unsatisfied about the product because user just has to take an image of a picture. After capturing the image, using image segmentation we can detect the object with its shape, color and with other features. Because our application gives deepest attention to identify the objects (leaf or digested root) accurately reason is why identifying objects correctly increases the application accuracy, performance and user satisfaction. Otherwise, it will give incorrect results at the end.

Shape is the most common feature, whether it be manual or automatic plant identification, used in plant identification [67]. So that we are going to use image segmentation techniques rather than using object detection technique. The reason is that Object detection [68] builds a bounding box corresponding to each class in the image. But for each object in the image, image segmentation creates a pixel wise mask. This technique provides us a much more granular idea of the object(s) in the image. So we can get clear idea of the shape of leaf, flower and etc.

Researchers have planned and suggested several segmentation techniques, but there is no general technique that can be used for all images. However, Keri Wood [69] suggested that good image segmentation need to meet the subsequent requirements:
Every pixel in the image must belong to a place and each place must be homogeneous with admire to a chosen feature, which might be syntactic e.g. color, depth or texture or the function primarily based on semantic interpretation.

1) Every region have to be linked and non- overlapping i.e. any pixels in a selected region need to be connected by a line that does depart the region.
2) It should not be viable to merge two adjoining regions to shape a single homogeneous region.
3) These characteristics also include in our segmentation process in order to get successful results.

Color images can increase the quality of segmentation but complexity of the problem is also increased. Segmentation of a colored image having different varieties of texture areas is a difficult problem, in particular if an exact texture subject is to be computed and a choice is to be made regarding optimum number of segments in the image. The problem will become further complicated if the image includes similar and/or "non-stationary texture fields. Each pixel of colored images is depicted using three component values that is red, green and blue and as such these are more complex as a long way as segmentation is concerned, than gray scale images which have a single intensity value for a pixel. Colored image segmentation can clear up many issues in medical imaging, bioinformatics, and material sciences [63]. It leads to lost lots of information of the leaf, flower or fruits. The lots of existing plant identifications applications regenerate the user entered pictures into greyscale images due to complexness of the color images. It results in lost voluminous info of the leaf, flower or fruits.

K.Deepak and A.N.Vinoth proposed to develop an application to identify plant species on android platform [70]. They used edge detection as there segmentation technique. This proposed method contain several major drawbacks. It converts the image into gray scale image. It ends with lost plenty of information approximately the detected object. It will be a major disadvantage when the detected object goes to classification model. And also in order to use this technique, there should be better contrast between objects. Otherwise, this technique isn't going to work and it doesn't detect the object accurately. It leads the application failed among the users. This indicates how the importance of detecting object accurately.

Lin-Hai Ma, Zhong-Qiu Zhao and Jing Wang designed plant identification system called APLeafis [71].They used threshold segmentation for the segmentation process. It is the simplest segmentation method. This application contains several drawbacks. In this application, one leaf image can be either a digital image from the existing leaf image database or a picture collected by a camera. The picture should be a single leaf placed on a light and untextured background without other clutter. Those facts reduce the user satisfaction because before taking a picture, user has to make the background that they mentioned about otherwise this application not going to detect the particular leaf from the image. If the application doesn't select it accurately other all processing things are not going to work. The application is failed. Because now a days, modern people always do and buy things which their lives make easier.

Desta Sandya Prasvita and Yeni Herdiyeni proposed MedLeaf as a new mobile application for medicinal plants identification based on a leaf image [72]. Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas and David W. Jacobs proposed a system called Leafsnap which is a mobile app that helps users identify trees from photographs of their leaves [73]. Above two authors proposed

systems they identify the plant only using the leaf. The major drawback of that kind of system is that sometimes some leaves have similar features but they are different plants. Because Ayurvedic plants are used by people as their medicine. Thus, it affects to human health also. We have to identify those plants separately. So that our proposed system detect not only leaf but also flower and fruit also. It leads to get best accurate result.

As mentioned above several problems occur when detecting the objects (leaf, flower or fruit). Therefore our proposed method is going to use the most suitable image segmentation techniques to detect the objects even if the image background is complex. In computer vision, we cannot directly apply an algorithm by believing that matches well with our application. We have to do several experiments and get outputs of those and then we should perform some evaluations on final outcomes to select best approach.

1.1.2. Classification of Ayurvedic Plants in Sri Lanka using transfer learning based on deep CNN Using A Mobile Application in An Offline Environment Approach

According to many known sources, researchers have tried many methodologies to extract the features and identify the plant species automatically. According to previous work done, there are existing applications which can identify plants with low prediction accuracies. However, these applications are based on foreign plant data sets that do not include valuable herbs and shrubs with medicinal qualities. Additionally, most of these methods make use of the combination of many parameters like color, shape, texture features etc.

In 2019, research paper “Oak Leaf Classification: An Analysis of Features and Classifiers” [14] has presented a new oak leaf dataset with preliminary results for classification of 8 types of oak trees. The new research findings include comparative analysis of a small set of hand-crafted geometric features as well as mostly used high-dimensional appearance features which includes Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG). It has been further compared with Support Vector Machines (SVM) classifier with Extreme Learning Machines (ELM). Results indicate an accuracy of 75% with a small set of geometric features, while an accuracy of 92% with high dimensional appearance features.

In 2019, in the study of “Combination of Deep Features and KNN Algorithm for Classification of Leaf-Based Plant Species” [15], an approach has been proposed which depicts a combination of deep architectures together. Deep features have been extracted from the leaves using the fc6 layer of AlexNet and VGG16 models. After that, dimension reduction of deep features using the Principal Component Analysis (PCA) method has been applied efficiently and the best

differentiated features were acquired. Finally, performances against classifications have been calculated using the K-Nearest Neighbor (KNN) algorithm. Flavia and Swedish, which are two popular plant leaf datasets, have been used for the testing of the system proposed. According to the experimental results, accuracy scores 99.42% and 99.64% were achieved for Flavia and Swedish leaf datasets, respectively.

In 2019, in the study of "AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants"[20], a deep learning based Convolutional Neural Network (CNN) model has been proposed to a system named AyurLeaf, in order to classify herbs using leaf features which included shape, size, color, texture etc. In addition, a standard dataset for medicinal plants has been proposed by this research work, which are habitually visible in various regions of Kerala. This dataset contains samples of leaves from 40 medicinal plants. A deep neural network inspired by Alexnet is utilized for the efficient feature extraction from the dataset. Finally, the classification is performed using Softmax and SVM classifiers. A classification accuracy of 96.76% has been achieved upon 5-cross validation, by their proposed model on the AyurLeaf dataset.

In 2019, in the study of "Fine-Grained Plant Identification using wide and deep learning model" [21], a model to address the fine-grained plant image classification task by using the wide and deep learning framework which combines a linear model and a deep learning model has been proposed. It sums the result of the wide and deep learning model using a logistic function so that discrete features can be considered simultaneously with continuous image content. Their works have used metadata such as the date of flowering and locational information for the wide model. This shows that the proposed method gives better performance than a baseline method.

In 2018, the study of "A Multiscale Fusion Convolutional Neural Network for Plant Leaf Recognition" [16], a multiscale fusion convolutional neural network (MSF-CNN) is proposed for plant leaf recognition at multiple scales. First, an input image is down-sampled into multiples low resolution im- ages with a list of bilinear interpolation operations. Then, these input images with different scales are step-by-step fed into the MSF-CNN architecture to learn discriminative features at different depths. At this stage, the feature fusion between two different scales is realized by a concatenation operation, which concatenates feature maps learned on different scale images from a channel view. Along with the depth of the MSF-CNN, multiscale images are progressively handled, and the corresponding features are fused. Third, the last layer of the MSF-CNN aggregates all discriminative information to obtain the final feature for predicting the plant species of the input image. Experiments show the proposed MSF-CNN method is superior to multiple state-of-the art plant leaf recognition methods on the MalayaKew Leaf dataset and the Leaf Snap Plant Leaf dataset.

In 2018, the study of "Improving Plant Recognition using Hybrid features from Connectionist and Knowledge-Based Approaches" [22] has proposed architecture that combined knowledge-based approach to improve the accuracy of plant recognition. Towards this, hybrid features are constructed by merging three types of knowledge-based features; morphological feature, texture feature and color feature with convolutional neural network extracted features. Their architecture consists of three main stages which are data pre-processing, feature extraction and classification. Before features are extracted, images will be resized and augmented in the pre-processing stage. To classify the species of leaf, they consider decision tree and artificial neural network as a classifier. They have experimented on two datasets: Flavia and Swedish dataset. The experimental result shows that the proposed architecture can predict unseen images correctly more than existing models.

In 2017, the study of "Plant Classification using Convolutional Neural Networks" [23] has proposed a Convolutional Neural Network (CNN) architecture to classify the type of plants from the image sequences collected from smart agro-stations. First challenges introduced by illumination changes and deblurring are eliminated with some preprocessing steps. Following the preprocessing step, Convolutional Neural Network architecture is employed to extract the features of images. The construction of the CNN architecture and the depth of CNN are crucial points that should be emphasized since they affect the recognition capability of the architecture of neural networks. In order to evaluate the performance of the approach proposed in this paper, the results obtained through CNN model are compared with those obtained by employing SVM classifier with different kernels, as well as feature descriptors such as LBP and GIST. The performance of the approach is tested on dataset collected through a government supported project, TARBIL, for which over 1200 agro-stations are placed throughout Turkey. The experimental results on TARBIL dataset confirm that the proposed method is quite effective.

In 2015, the study of "Recognition of Whole and Deformed Plant Leaves using Statistical Shape Features and Neuro-Fuzzy Classifier" [24] has proposed a methodology for recognition of plant species by using a set of statistical features obtained from digital leaf images. As the features are sensitive to geometric transformations of the leaf image, a preprocessing step is initially performed to make the features invariant to transformations like translation, rotation and scaling. Images are classified to 32 pre-defined classes using a Neuro fuzzy classifier. Comparisons are also done with Neural Network and k-Nearest Neighbor classifiers. Recognizing the fact that leaves are fragile and prone to deformations due to various environmental and biological factors, the basic technique is subsequently extended to address recognition of leaves with small deformations. Experimentations using 640 leaf images varying in shape, size, orientations and deformations demonstrate that the technique produces acceptable recognition rates.

In 2015, the study of "Deep-Plant: Plant Identification with Convolutional Neural Networks" [26] studies convolutional neural networks (CNN) to learn unsupervised feature representations for 44 different plant species, collected at the Royal Botanic Gardens, Kew, England. To gain intuition on the chosen features from the CNN model (opposed to a 'black box' solution), a visualization technique based on the deconvolutional networks (DN) is utilized. It is found that venations of different order have been chosen to uniquely represent each of the plant species. Experimental results using these CNN features with different classifiers show consistency and superiority compared to the state-of-the art solutions which rely on hand-crafted features.

In 2014, the study of "Ayurvedic leaf recognition for Plant Classification" [31], the performance of different features extraction methods is compared, different combinations of features and a number of classifiers applied for leaf identification process are also discussed.

In 2012, the study of "An Efficient Leaf Recognition Algorithm for Plant Classification Using Support Vector Machine" [27] uses an efficient machine learning approach for the classification purpose. This proposed approach consists of three phases such as preprocessing, feature extraction and classification. The preprocessing phase involves a typical image processing steps such as transforming to gray scale and boundary enhancement. The feature extraction phase derives the common DMF from five fundamental features. The main contribution of this approach is the Support Vector Machine (SVM) classification for efficient leaf recognition. 12 leaf features which are extracted and orthogonalized into 5 principal variables are given as input vector to the SVM. Classifier tested with flavia dataset and a real dataset and compared with k-NN approach, the proposed approach produces very high accuracy and takes very less execution time.

In 2012, the study of "Classification of Selected Medicinal Plants Leaf Using Image Processing" [32] has aimed at implementing a system using image processing with images of the plant leaves as a basis of classification. The software returns the closest match to the query. The proposed algorithm is implemented, and the efficiency of the system is found by testing it on 10 different plant species. The software is trained with 100 (10 number of each plant species) leaves and tested with different plant species) leaves. The efficiency of the implementation of the proposed algorithms is found to be 92%.

In 2010, the study of "Plant Leaf Classification Using Plant Leaves Based on Rough Set" [28] a method of plant leaf classification is proposed based on the neighborhood rough set. They mainly show that this is applicable to plant leaf classification. Experimental results on plant leaf image database demonstrate that the proposed method is effective and feasible for leaf classification.

In 2008, the study of "A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network" [29] they employ Probabilistic Neural Network (PNN) with image and data processing techniques to implement a general-purpose automated leaf recognition for plant classification. 12 leaf features are extracted and orthogonalize into 5 principal variables which consist the input vector of the PNN. The PNN is trained by 1800 leaves to classify 32 kinds of plants with an accuracy greater than 90%. Compared with other approaches, their algorithm is an accurate artificial intelligence approach which is fast in execution and easy in implementation.

1.1.3. Abstractive web page Information Summarization and Location Mapping with GIS technology on Ayurvedic Plants in Sri Lanka Using a Mobile Application in an Online Environmental Approach

According to many known sources, researchers have tried many methodologies to follow up dynamic information extraction and information summarization.

Most of the present Ayurvedic based applications follows only the medicinal plants identification process. The proposed system Arogya is capable of generating dynamic summaries regarding the identified medicinal plants. So, the system has to work with the online database, whenever the web pages get updated, the generated summary also should be adjusted in order to the current version of the changed website page. Hence it is kind of a real time information extraction, users also will get updated automatically. Currently the existing systems are just managing with the offline database of their own system, so the information will not be changed according to the timeline. So, the users will not get updated with the real time information of that particular medicinal plants.

In 2016, the research paper "An Automatic Multi document Text Summarization Approach Based on Naive Bayesian Classifier Using Timestamp Strategy" [3], discusses about an automatic text summarization approach is proposed which uses the Naive Bayesian Classification with the timestamp concept. This summarizer works on a wide variety of domains varying between international news, politics, sports, entertainment, and so on. Another useful feature is that the length of the summary can be adapted to the user's needs as can the number of articles to be summarized. The compression rate can be specified by the users so that they can choose the amount of information he wants to imbibe from the documents.

The research presented in 2012, "A Semantic-Based Framework for Summarization and Page Segmentation in Web Mining" [4], introduces a framework that can effectively support advanced Web mining tools. The proposed system addresses the analysis of the textual data provided by a

web page and exploits semantic networks to achieve multiple goals, the identification of the most relevant topics, the selection of the sentences that better correlates with a given topic, the automatic summarization of a textual resource. The eventual framework exploits those functionalities to tackle two tasks at the same time, text summarization and page segmentation.

In 2017, the research paper “Text Summarization Techniques: A Brief Survey” [5], emphasizes various extractive approaches for single and multi-document summarization. It described some of the most extensively used methods such as topic representation approaches, frequency-driven methods, graph-based and machine learning techniques. Although it is not feasible to explain all diverse algorithms and approaches comprehensively in this paper, it provides a good insight into recent trends and progresses in automatic summarization methods and describes the state-of-the-art in this research area.

In 2018, the paper “Multi-Document Text Summarization Using Deep Learning Algorithm with Fuzzy Logic” [6], presented a multi-document text summarization scheme using an unsupervised deep learning algorithm along with fuzzy logic. Feature matrix with seven features from the set of sample dataset from DUC2002(Document ID: AP880911- 0016). The feature matrix is applied through the various levels of the RBM and finally the efficient text summary is generated. The result indicates that this method generates efficient text summary when compared to previous methods based on evaluation metrics.

In 2019, the paper “Deep Learning Architecture for Multi-Document Summarization as a cascade of Abstractive and Extractive Summarization approaches” [7], extends the state-of-the-art abstractive summarization architecture for multi-document summarization. The proposed architecture produces comprehensive summary on a topic by combining the Abstractive and Extractive summarization approaches in a cascade. The state-of-the-art approach for abstractive summarization using pointer-generator model is limited to single-document summarization. Summarization of multiple news articles on a topic can be handled one by one independently which results in multiple summaries for the same topic with possible redundancy. In order to avoid redundancy, the authors propose Extractive summarization of the multiple summaries as the second phase in the proposed cascade framework. The effectiveness of the framework was established using ROUGE metric.

In 2011, the paper “Frequently Asked Questions Web Pages Automatic Text Summarization” [8], presented the preliminary results of our FAQ Web Pages automatic text summarization approach. Their approach is based on making use of the visual cues existing in the text of the questions and answers of the web page to detect Q/A segments. In addition, we devised a new combination of selection features to perform the actual summarization task. These features are namely, question-answer similarity, query overlap, sentence location in answer paragraphs and upper-case words frequency. The different document features were combined by a home-grown weighting score function. Pilot experimentations and analysis helped them in obtaining a suitable combination of

feature weights. In fact, the evaluation results showed that in general this approach seems to be promising where the overall average for all pages indicates statistically significant improvements for their approach in 62% of the cases when compared with another summarization tool.

In 2004, this paper "World Wide Web Site Summarization" [9], developed a new approach for generating summaries of Web sites. The approach applies machine learning and natural language processing techniques to extract and classify narrative paragraphs from the Web site, from which key-phrases are then extracted. Key-phrases are in turn used to extract key-sentences from the narrative paragraphs that form the summary, together with the top key-phrases. The summaries are demonstrated, although not in proper prose, are as informative as human-authored summaries, and significantly better than browsing the home page or the site for a limited time. The approach should be easy to transform into proper prose by human editors without having to browse the Web site. The performance of method depends on the availability of sufficient narrative content in the Web site, and the availability of explicit narrative statements describing the site.

In consequence of the above reviews, Arogya acts as a single platform of resource access of multiple functionalities to overcome all these difficulties in the current products in the market. Arogya's main goal is to be the feasible single unit of medicinal plants information hub which will satisfy all the user requirements regarding ayurvedic plants.

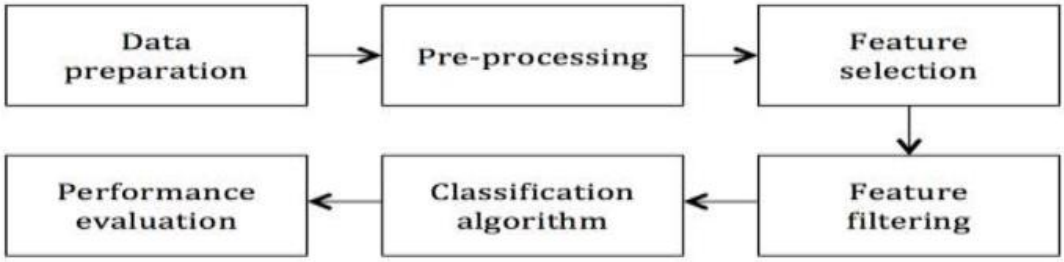
1.1.4. The Text classification of identifying a category type of diseases and visualizing ingredients in most probably used Ayurvedic prescription/recipes

Arogya is an android based mobile application which is going to overcome some of the problems mentioned in above. apart from there is a special feature of texting a description including characteristics of herbal remedies/ plants. Mainly, that option will be provided for identifying each type of herbal remedies / plants which represent in what category of diseases who may be seen or ever seen either knowing of their characteristics well. Proposed system will give some dropdown selections for selecting needed attributes/traits/properties/characteristics consistent in relevant text areas. If not there having a space to mention it in briefly. According to the application, there are some multiple categories to separate data which are entered by the user. Then analyzing what type of category after checking and rechecking using machine learning algorithms (Support vector, decision tree, etc.).

Analyzing the characteristics and predicting the category type of disease is the component for using machine learning algorithms to predict diseases. In this research component, I have to create a dataset manually for adding the specific characteristics of the relevant herbal plants.

There are some websites having different types of datasets to download for needed tasks. *Kaggle* is the famous web site which can be given the massive datasets with download freely. According to the many know sources; researchers have tried to classify many different types of datasets for predicting the specific things/outputs.

In this app going to implement an automation text classification on Machine Learning algorithms related to applications following the below steps.



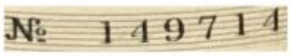
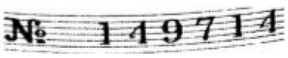
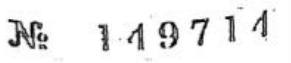
Analysis Overall process components

So that, this proposed system is more important to identify the herbal remedies when people failed in its identification and not understanding in better ayurvedic samples/prescriptions medicine even though in Offline status. In ancient Indian medical system, also known as Ayurveda, is based on ancient writings that rely on a natural and holistic approach to physical and mental health. This is also world's oldest medical systems and remains one of India's traditional health care system. Every person engaged in various type of activities/ treatments internet either by using social media or online identification activities. Many numbers of activities go through on internet as well as treatments in Ayurvedic and western approach medicine for health-related issues.[76].

Nowadays, many online communities are focused on health-related issues in western medicine approach and Ayurveda treatments. In order to that it clarify different types of diseases such as diabetes, Arthritis, Cancers etc. [82] However, they introduced dynamic exercise plans, nutritious to treatment and diagnosis for curving everything else related diseases. [77]. Therefore, today created many numbers of application for each of different tasks, but our proposed system will give special options as clarifying a disease using texts selections. According to this text classification and picture extraction function is somewhat different from the extract process of text classification apply in this application. There interface displays with dropdown icons for selecting needed plant details. Hence, there analyzing a categorized process is on feature extraction stage using Naive Bayes classifiers. [79]

There are some OCR techniques using various occasions for detecting texts clearly. In Health information systems are followed these OCR for clinical activities also. In 2019, research paper "A comparative study of optical character recognition in Health Information System" [86] presented the how use OCR techniques and the process of implementing the identification of images into texts using these techniques.

In 2018, the paper “The use of Tesseract OCR number recognition for food tracking and Tracing” [84], presented the comparison different OCR tools. The paper was presented after experiment with several OCR tools, and consider, taking an open source software called Tesseract OCR engine was selected for the pilot solution of this research food tracking and tracing.

Image type	*Image*	*Azure CV*	*Google CV*	*Tesseract*
Original		-	14971	749714
After thresholding and Otsu filtering		-	-	974
After removing horizontal lines		7	149714	149714

Comparison between different OCR tools

In 2018 paper, “Document segmentation and language translation using Tesseract-OCR” [85], presented the basic components and the process of Tesseract-OCR understandably. “python-tesseract” module in python implement tesseract-OCR to convert the image into text. The best OCR engine is considered as Tesseract OCR and ABBYY FineReader. Because of that reason and the above figure 1.1 2 comparison due to focus on Tesseract V4 for developing this research component in better way.

There are some software and tools for converting images into string type. This system was aimed to use a base64 for converting. Today many numbers of project are built with using OCR for recognition of the text inside images. Document segmentation and language translation, Health information systems, Food tracking and tracing [85,86,84] are some examples for the current projects that are used for OCR techniques. OCR technology is used for converting virtually any kind of images containing written text (typed, handwritten, or printed) into machine readable text data.

1.2. Research gap

In computer vision, despite many efforts [8–13], plant identification, classification, tracking locations of valuable plants, defining information sources with summary reports, as well as details of medical recipes for specific categories of diseases are still considered challenging and unsolved problems. It is very challenging since the unavailability of a single platform which can manage all of the above requirements regarding Ayurveda in Sri Lanka.

In Sri Lanka we have our own set of rare Ayurvedic herbs. But most of us are unable to identify these plants due to lack of knowledge. Dissemination of knowledge regarding herbal plants is restricted mainly to very limited group of people and is passed down from generation to generation who practice traditional medicine.

So, recognizing these endemic herbal plants is a challenging problem in the fields of Ayurvedic medicine, computer vision, and machine learning, because although leaf classification apps have been implemented for leaves in other countries, such an automated app with a reliable identification and classification mechanism has not still been implemented for the purpose of locals. There are existing applications which can identify plants with low prediction accuracies. However, these applications are based on foreign plant data sets that do not include valuable herbs and shrubs with medicinal qualities. Therefore, it is the main research gap which can be identified through the existing research and implemented automated applications regarding plant identification and classification.

If the main research gap is summarized, it is the “Lack of a single platform for the management and preservation of Ayurvedic plants in Sri Lanka using computer vision, and unavailability of a full-automated mobile application for the particular purpose.” This is mostly important in the system of medicine called Ayurveda in Sri Lanka because identification and classification of medicinal plants is considered an important activity in the preparation of herbal medicines. Because treatment using a wrongly identified medicinal plant may claim the life of a patient. Additionally, it is important for Ayurveda practitioners and also traditional botanists to know how to identify and classify the medicinal plants through computers [5]. They should apply their knowledge of Ayurveda with technical approach too.

Following table shows the comparison of several existing mobile applications with plant classification functionality with our system, and the gaps are well defined through this structure.

	Agrobase	**Plantex**	**PlantSnap**	**Pl@ntNet**	**MedLeaf**	**Arogya**
Search Plants	✓	✓	✓	✓	✓	✓
Identify and classify plants	✓	✓	✓	✓	✓	✓
Filters whether the searched plant is a medical plant or not	✓	✓	☒	☒	✓	✓
Identify and classify plants offline	☒	☒	☒	☒	☒	✓
Mapping the locations of newly identified plants	☒	☒	☒	☒	☒	✓
Show directions from users' current location	☒	☒	☒	☒	☒	✓
Ability to share plant location	☒	☒	☒	☒	☒	✓
Gives a Summarized information about the classified plant	✓	✓	✓	☒	✓	✓
Gives the user an AR experience	☒	☒	☒	☒	☒	✓
Gives remedies for specific categories of diseases and propose medical recipes for them	☒	☒	☒	☒	☒	✓
Gives user a novel social media experience limited to trees (upload/view/share posts only related to plants) and makes users aware of the importance of plants to our world	☒	☒	☒	☒	☒	✓

Provide a space for typing a characteristic of desired herbal plants.	☒	☒	☒	☒	☒	✓
Identifying a category type of diseases using a text classification	☒	☒	☒	☒	☒	✓
Displaying recipes related to the classified category	☒	☒	☒	☒	☒	✓
Picture Extraction of ingredients in recipes	☒	☒	☒	☒	☒	✓
Getting results in Offline stage	☒	☒	☒	☒	☒	✓
Image Identification (Handwriting recipes by users)	☒	☒	☒	☒	☒	✓
Text classify of the post regarding whether its Ayurvedic or not. If not, it will not share in the wall	☒	☒	☒	☒	☒	✓

Table 1.2.1: Comparison with existing systems and identifying gaps

1.3. Research problem

The problem regarding the identification and familiarization of the plants, getting information about the value of them precisely, tracking locations of their growth and availability, providing remedies and recipes for specific categories of diseases limited to the specified herbal medicine plant leaves, along with the seen necessity for the development of herbal medicine plant identification should be addressed by a study. Especially in Sri Lanka we have our own set of rare Ayurvedic herbs. But most of us are unable to identify these plants due to lack of knowledge. Though many are aware of the existence of different herbal medicine plants and their familiar applications, many are still unable to identify which are these herbal medicine plants from the vast diversity of plants in the environment, so as their noted and approved applications by health professionals.

There are several methods to recognize a plant and use medical recipes assigned with. At present, plants are identified, as well as recipes are predicted manually by taxonomist, which are prone to human errors. But manual process requires prior knowledge and also it is a lengthy process. Leaf Identification by mechanical means often leads to wrong identification. Identification and classification of medicinal plants are essential for better treatment. Lack of experts in this field makes proper identification and classification of medicinal plants a tedious task. This is one of the major problems in this domain.

Additionally, researchers in the field of botany, medicine, chemical structure analysis, agriculture, and ayurvedic medicinal practitioners, forest department officials, those who are involved in the preparation of ayurvedic medicines and others who are concerned with plant studies are faced with the application of considerable effort on identifying plants. It is stated that plant identification demands extensive knowledge and uses complex terminology, even professional botanists need to take much time in the field to master plant identification [7]. Identifying a medicinal plant with required medicinal values is one of the major challenging tasks faced by them. It plays a crucial role in the preparation of ayurvedic medicines. But lack of expert taxonomists is a 23 major issue in this area. Even though herbal medicine has no side effects, treatment using a wrongly identified medicinal plant may claim the life of a patient.

Below are the summarized research problems those are going to be addressed through our research studies.

1. Lack of technological research carried out on the Recognition and Classification of Ayurvedic plants in Sri Lanka using machine learning and computer vision.
2. Absence of a single platform to detect and classify Sri Lankan ayurvedic leaves, give important information about them and to show the locations from where those plants can be found in Sri Lanka.
3. Lack of a system which provides many features as services of giving Traditional treatments.
4. Lack of a system which identifies the category type of disease in an unknown herbal remedy by giving its special characteristics/properties and which displays the most used herbal sample of each category and visualizing some of herbal remedies that are included.
5. Current mapping systems used are inaccurate and lacks use of a common GIS platform.

6. Only manual location records are available.
7. Dissemination (Circulation) of knowledge regarding herbal plants is restricted mainly to a very limited group of people and is passed down from generation to generation who only practice traditional medicine.

Plants are an indispensable part of our ecosystem and the dwindling number of plant varieties is a serious concern. Another major problem is that there is a growing scientific consensus that plant habitats have been altered and species are disappearing at rates never witnessed before. The biodiversity crisis is not just about the perilous state of plant species but also of the specialists who know them This initially requires data about various plant varieties, so that they could be monitored, protected and can be used for future. Plants form the backbone of Ayurveda and today's modern-day medicine and are a great source of revenue. Due to Deforestation and Pollution, a lot of medicinal plant leaves have almost become extinct. So, there is an urgent need for us to identify them and regrow them for the use of future generations. Due to growing illegal trade and malpractices in the crude drug industry on one hand and lack of sufficient experts on the other hand, an automated and reliable identification and classification mechanism in order to handle the bulk of data and to curb the malpractices is needed.

1.4. Research Objectives

1.4.1. Main Objective

The primary objective of the proposed solution is to develop a Centralized Social Media platform (android mobile application) that gives user the ability to identify and classify a group of selected important ayurvedic plant species in Sri Lanka, based on photographs of the plant's leaf as well as other parts such as digested root and fruit, taken with mobile phone, to share information sources of them through information extraction, get information about remedies and recipes for specific disease categories as well as the tracked locations with the aim of preservation of valuable Ayurvedic plants in Sri Lanka.

Therefore, anyone without prior knowledge will be able to identify and classify the particular ayurvedic plant hopefully. The development strategy and methodology used in this approach will be able to be used and extended to identify any ayurvedic herb in Sri Lanka furthermore.

1.4.2. Specific Objectives

Following are some specific objectives those are going to be achieved through our new product

1. Recognizing and classifying the leaves of the herbs/plants used for Ayurvedic treatment once the user inputs an image of a leaf captured by mobile camera.
2. Providing a description of the identified herb's medicinal usage in reference to illnesses by going through feed data.
3. Dynamic search for the names of the plant leaves through a word search, or to do either a web search or generate a summary of the plant.
4. Display onsite information on medicinal plants and provide a summarized detail regarding the values and uses of them through abstractive information extraction and summarization.
5. Identify the category type of disease in an unknown herbal remedy by giving its special Characteristics/properties. After displaying the category, The System will display the most used herbal sample of each category of disease.
6. User is cable of adding the location of plants into the Map, as well as to get use of the locations of the previously tracked plants (availability and growth)
7. After the identification process users can choose to publish the post and other users can comment on them.
8. Can be seen as rare or unknown medicine plants which are included as ingredients of medicine recipes.

On a mobile device, the Ayurveda plant detection has to be done with time, battery life critical manner, especially when it has to be done in a forest area. In the proposed system the whole identification process will take place on mobile devices and it doesn't require the internet. Therefore, this will be a great solution to identify Ayurveda plants in deep forest areas, where mobile network coverage is not available.

2. METHODOLOGY

A methodology is a set of guidelines or principles that can be applied to a specific situation. In the project duration, members followed below guidelines and approaches in each of their individual components. A methodology could also be a specific template, approach, forms, and testing used over the project life cycle. A formal project methodology should lead the work of all team members throughout the life cycle of a project.

In methodology part, it shows the way we managed our research and how we developed our research. This section exaggerates the project scope, project features, and final outcome of project within time duration one year of period.

To fulfill the above outcomes, Agile methodology was used throughout the software development. Agile software development methodology refers to a group of software development methodologies It is based on iterative model.

2.1. Understanding the key pillars of the research domain

The system mainly relies on the key pillars of Deep Learning, Transfer Learning based on deep CNN and Data Augmentation.

2.1.1. Deep Learning

According to many previous research works done [10-30], recognition of herbs has been carried out for the last few decades using classical image processing techniques. They have used shape, texture, and color-based features for the performance purpose. Compactness, aspect ratio, skewness, energy, eccentricity, kurtosis, correlation, sum variance and entropy are some of these features [20]. Excess computation time acquired for handcrafted feature extraction is one of the major issues associated with these classical ways. At present, all the classical methods are replaced by machine learning techniques.

Deep learning can be defined as a class of machine learning in which a computational model learns to perform classification tasks directly from images. It is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example [37]. Ability to handle huge volumes of image data, higher accuracy, inbuilt ability to use GPUs for parallel computation as well as availability of inbuilt pre-trained convolutional neural networks contribute towards the majority use of deep learning. Since Arogya mobile app performs classification on huge volume of image data, this approach will be the best choice.

2.1.2. Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data [39]. A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing [39].

CNN has been recognized as a competent method for image recognition in past few years, on mobile devices in an offline environment. CNNs work somewhat similar to neurons in brain. It also performs image transformation with a certain degree of rotation and distortion [35]. Since this network is avoiding the complex preprocessing of the image, we will be able to input the original image of the suspected plant directly.

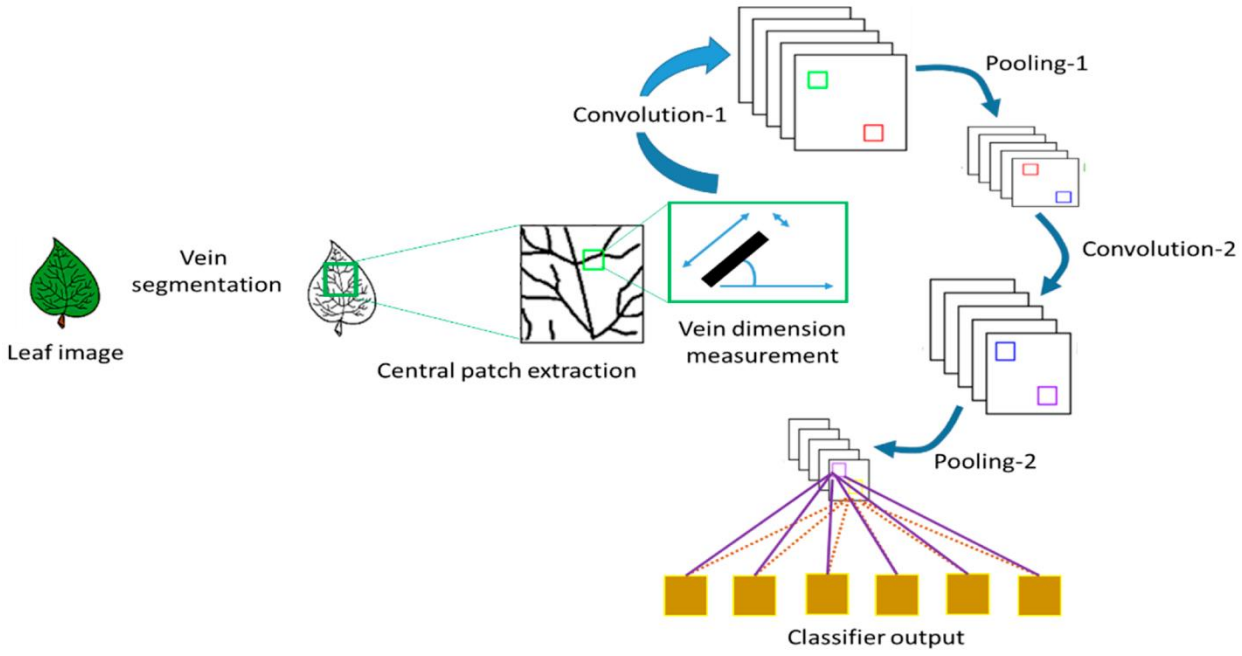


Figure 2.1.2.1: Review on techniques for plant leaves CNN:

Source: https://www.google.com/https://www.mdpi.com

2.1.3. Transfer Learning based on deep CNN

An interesting benefit of deep learning neural networks is that they can be reused on related problems. Transfer learning refers to a technique for predictive modeling on a different but somehow similar problem that can then be reused partly or wholly to accelerate the training and improve the performance of a model on the problem of interest [38]. In deep learning, this means

reusing the weights in one or more layers from a pre-trained network model in a new model and either keeping the weights fixed, fine tuning them, or adapting the weights entirely when training the model [38].

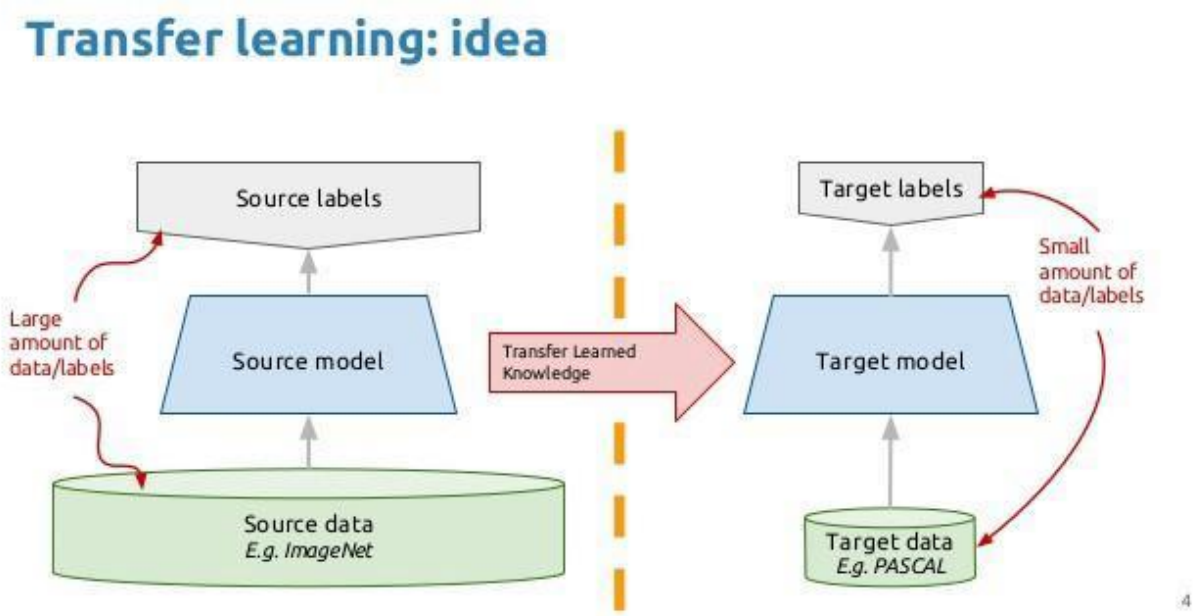


Figure 2.1.3.1: Transfer Learning with Convolutional Neural Networks in PyTorch: Source: https://www.google.com/towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch

Following is the general outline for transfer learning for object recognition [18]:

1. Load in a pre-trained CNN model trained on a large dataset
2. Freeze parameters (weights) in model's lower convolutional layers
3. Add custom classifier with several layers of trainable parameters to model
4. Train classifier layers on training data available for task
5. Fine-tune hyperparameters and unfreeze more layers as needed

2.1.4. Data Augmentation

As mentioned in many researches in this domain, among majority of DNN structures, CNN are used currently as the main tool for the image analysis and classification purposes [17]. Even though great achievements and perspectives have been populated, deep neural networks and accompanied learning algorithms have some relevant challenges to be tackled [17]. The most frequently mentioned issue in the area of machine learning and deep learning, is the lack of sufficient amount

of training data or uneven class balance within the datasets [17]. One of the ways of dealing with this problem is so called data augmentation [17]. For training the convolutional neural network that we use for extracting features, there is an idea for training images to be augmented. As a CNN requires a large number of images to train it, various data augmenting methods is planned to be applied to the collected dataset to increase the size of the dataset.

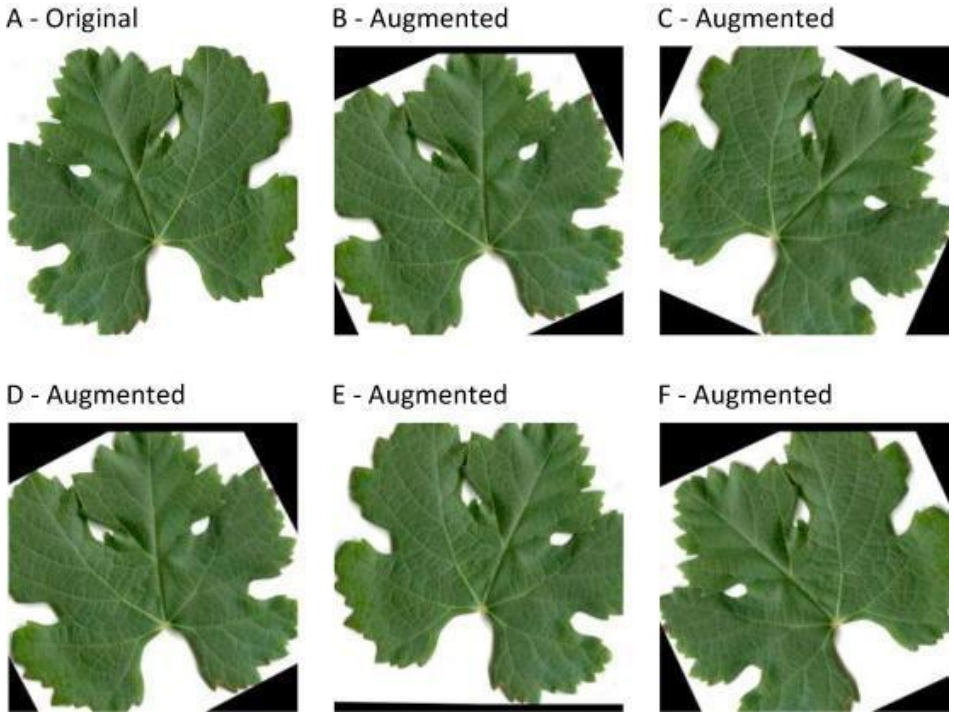


Figure 2.1.4.1: Data Augmentation with leaves:

Source: https://www.google.com/https/www.sciencedirect.com/science/article/pii

2.2. Methodologies of each approach

2.2.1. Selecting the Most Accurate and the Highest Performance Segmentation Technique using Experimental Outcomes of Plants in Sri Lanka

Object detection

Object detection is one of the classic computer vision problems where you function to distinguish what and where-exactly what objects are within a given image and also where they are within the image. The object detection problem is more complicated than classification, which can also identify objects but does not indicate where the object is located in the image. Classification also doesn't work on photographs that contain more than one thing.

Research Methodology

The system provides a way to upload an image of the plant to be identified. Detecting the objects accurately is more important because it helps the classification process to do a better job using a leaf or digested root. In addition, detecting plant leaf or digested root from complex and noisy backgrounds should also be done accurately, especially in this scenario. Therefore, much attention was given to detect the specific object accurately at the beginning. In order to achieve this objective, it was experimented with YOLO and marker-based watershed algorithms and the most accurate one was selected based on the results. At the begging backend application is developed using python and Front end application is developed using Android.

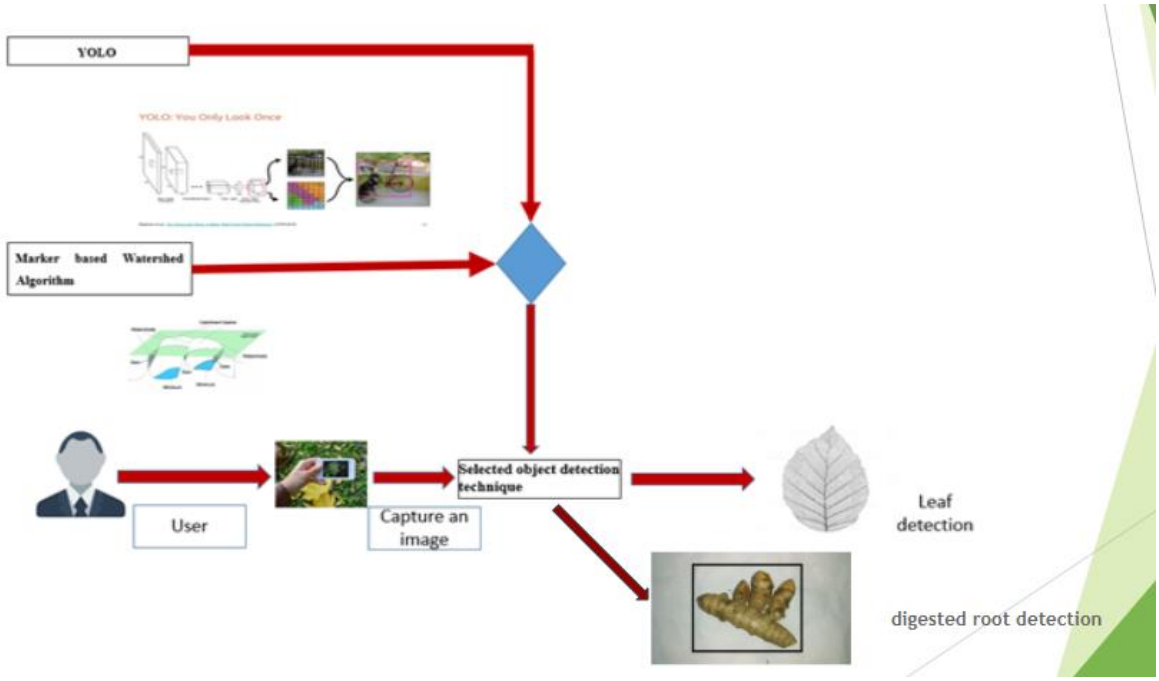


Figure 2.2.1.1-High level architecture diagram for leaf detection

1) YOLO (You only look once)

Those bounding boxes are weighted by the probabilities predicted. YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. The YOLO network consists of three main pieces.

1) **Backbone -** A convolutional neural network that aggregates and forms image features at different granularities.
2) **Neck -** A series of layers to mix and combine image features to pass them forward to prediction.
3) **Head -** Consume features from the neck and take box and class prediction steps.

Dataset

The first important step, as with any deep learning step, is to prepare the dataset. The dataset is the fuel that runs every model of deep learning**.** Our Ayurvedic dataset was made using captured photos of the tree leaves/digested roots, web images, etc.). After the process of data annotation, labelling into 2 classes (leaf or digested root) for the purpose of annotation and totally pre-processing; the prepared dataset will be divided into 3 parts as:

1. Training dataset -70% of data for training the model
    - Training set: Data sample used for model fit. The real data set which we use to train the model (for a Neural Network, weights and biases). From this knowledge, the model sees and learns.

2. 15% of data for validating the model – Validation dataset

    - Validation set: The sample of data used to provide an impartial model assessment fits into the training dataset while hyper parameters are adjusted to the model. As knowledge on the validation dataset is integrated into the model setup, the assessment becomes more biased. Often called the Dev set or the Creation set, the validation set is also known. This makes sense because during the model's "growth" period, this dataset assists.

3. 15% of data for testing the model – Test dataset.

    - Test set: The sample of data used to provide an impartial assessment of a final model fits into the dataset of the training. The Test dataset provides the gold standard used for the model evaluation. It is only used once a model (using the trains and validation sets) is fully educated. In general, the test set is what is used to compare competing models. The validation set is often used as the test set, but it is not good practice. In general, the test collection is well selected. It contains carefully sampled data which, when used in the real world, spans the various classes that the model will face.

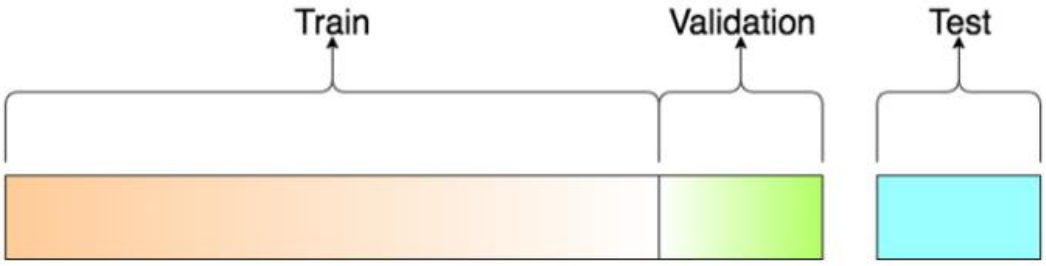


Figure 2.2.2.2-Data Visualization

Data Annotation

In here I train the **YOLO v3** version and **YOLO v5** version which are the latest versions of the YOLO algorithm using our annotated Ayurvedic dataset. To use the YOLO algorithm, first we annotate our dataset using LabelIng tool around 2000 images for 5 categories of plant species.

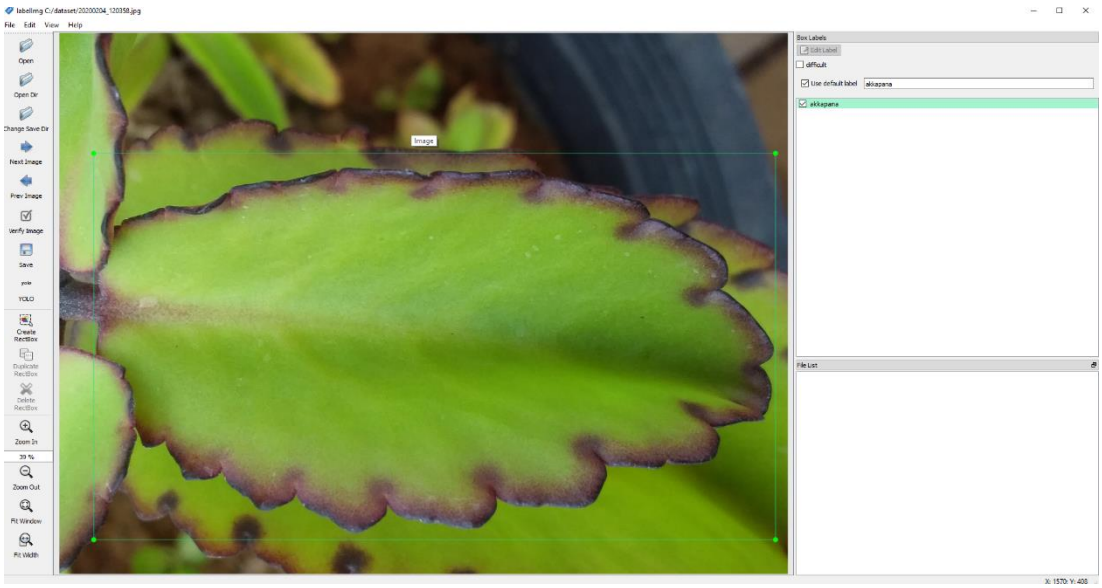


Figure 2.2.2.3 -Data annotation using LabelIng tool

YOLO v3

YOLO is a completely coevolutionary network, and by applying a 1 x 1 kernel on a feature map, its eventual output is generated.

The detection is achieved in YOLO v3 by applying 1 x 1 detection kernels to function maps of three different sizes at three different network locations. The most critical aspect of v3 is that it allows three distinct scales of detection [74].

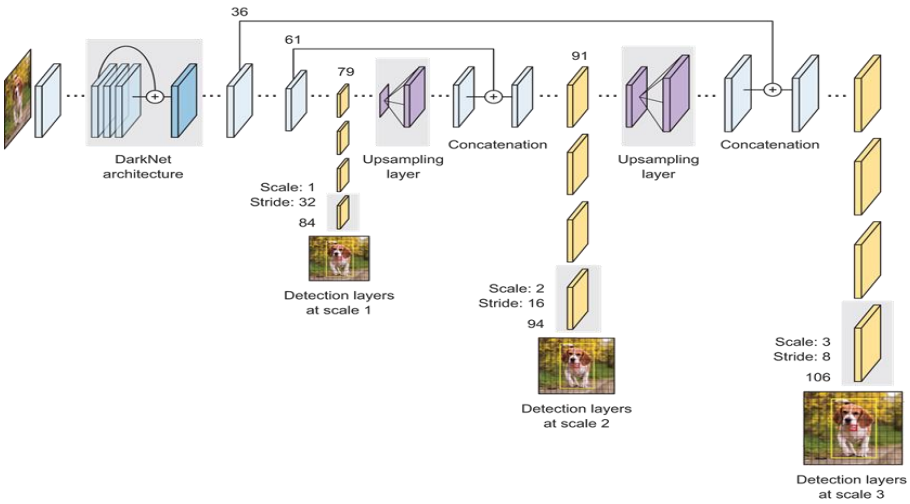


Figure 2.2.2.4-YOLO V3 Architecture

Source: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

To load the YOLO V3, we need to have YOLO V3 training file and the YOLO V3 Configuration file. We can get the configuration file from the Darknet (framework used to run and train YOLO) repository. To train the image dataset we use the free server offered by Google colab. Google Colab is a free service that Google provides, where you can run python scripts and use machine learning libraries to benefit from their powerful hardware. To obtain the weight file we should train our model using our Ayurvedic dataset. The only downside is that it can be used for 12 hours in a row, from which it will be disconnected and our files will be removed. Therefore, I connected it with my Google drive. Thus, all files are saved if it is disconnected. It takes more than five hours to train the model using our custom dataset.

YOLO V5

YOLOv5 is a recent release from the YOLO model family. Initially YOLO was implemented as the first model of object detection to merge bounding box prediction and object classification into a single end-to - end differentiable networkIt was written in a framework called Darknet and is preserved. The first of the YOLO models to be written in the PyTorch framework is YOLOv5 and it is much lighter and simpler to use. YOLOv5 is written in the Ultralytics PyTorch framework, which is very intuitive to use and inferences very fast. In fact, we and many others would often translate YOLOv3 and YOLOv4 Darknet weights to the Ultralytics PyTorch weights in order to inference faster with a lighter library. YOLOv5 is definitely easier to use and, based on our initial runs, it is very powerful on custom data.Therefore, YOLO V5 has more performance than YOLO V3.

2) Marker Based watershed algorithm

Watershed segmentation is region-based method that is regarded as a topological landscape with ridges and valleys. The elevation values of the landscape are typically defined by the gray values of the respective pixels or their gradient magnitude. Due to noise, direct application of the watershed segmentation caused over-segmentation. Markers were used as an approach to control over segmentation. One of the most complicated operations of image processing was to distinguish touching objects in image. The watershed segmentation is often applied to this issue. This image segmentation algorithm proposed here was capable of segmenting the images with minimum limitations to under and over segmentation. Segmentation of the marker-controlled watershed has been shown to be a robust and versatile method for segmenting artifacts with closed contours, where the boundaries are represented as ridges.

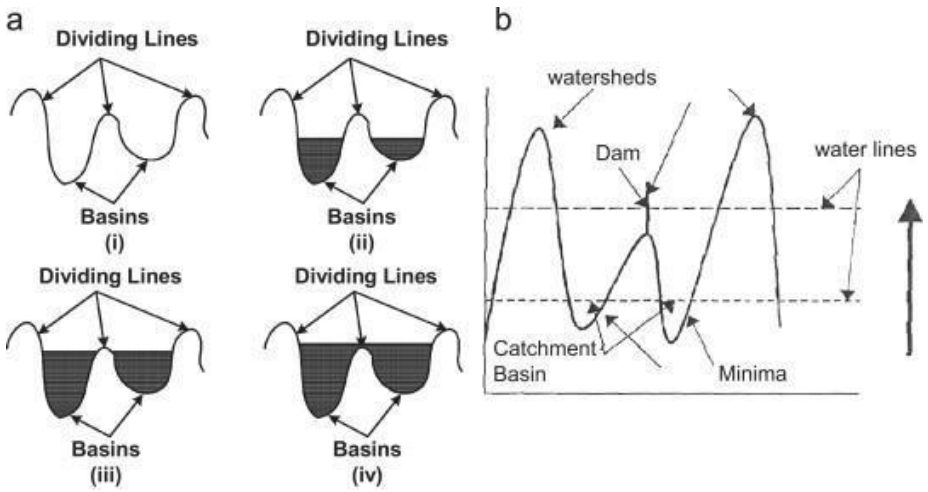


Figure 2.2.2.5-high level architecture for watershed segmentation

To identify a leaf, vein structure takes a prominent place. Vein is a vascular bundle within a leaf that frequently spreads from the middle of the leaf to the edge [75]. In this proposed method introduces a precise and fully automated image segmentation for complex context images of medicinal plant leaves. As the first step, green color range of the image is filtered. To get the wide green color range, at the begging of the process image is converted from RGB to HSV color format. By strictly applying the objective rule that the veins are linear and the accuracy of gradient angles of adjacent venous points is relatively high, the veins in the gradient magnitude images are further improved by the normal gradient angle deviation to obtain the vein enhancement image. Based on this image, a binary image taking the veins as the foreground is obtained using the OTSU process, and the main veins are identified from it. In the vein enhancement diagram, fine veins are found in other areas outside the main veins, and then attached to the main veins. Then a foreground marker image is obtained which targets the veins. In each component image of the color image, the background marker image is segmented using the OTSU process, and then

screened after a certain ratio is determined. The marker-controlled watershed approach is applied to obtain the outcome of binary segmentation, based on the foreground markers and the background markers.

To identify the digested root, Yellow and brown color range area is extracted from the image.

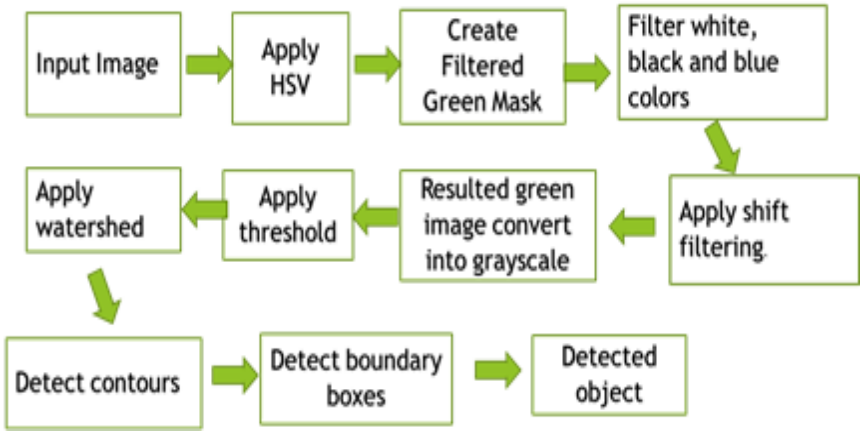


Figure 2.2.2.6 -Summary of the proposed marker based watershed segmentation technique

Research	**Research Work**	**Methods and Algorithms to be used**	**Expected Accuracy**	**Remarks**
Leaf detection from the complex background	Use different object detection techniques on collected dataset.	YOLO architectures (Such as YOLO v3, YOLO v4, VGG-16, marker based watershed algorithm.) will be used to choose the best technique	Higher Accuracy will be expected	Several existing algorithms will be analyzed, and accuracy will be tested in order to select the suitable algorithms to detect the leaf from the image accurately.

Table 2.2.2.1 – Summary of the methodology

2.2.2. Classification of Ayurvedic Plants in Sri Lanka using transfer learning based on deep CNN Using A Mobile Application in An Offline Environment Approach

The following diagram illustrates the methodology followed in order to implement the classification of the selected ayurvedic herbal plants in Sri Lanka using a mobile application in an offline environment.

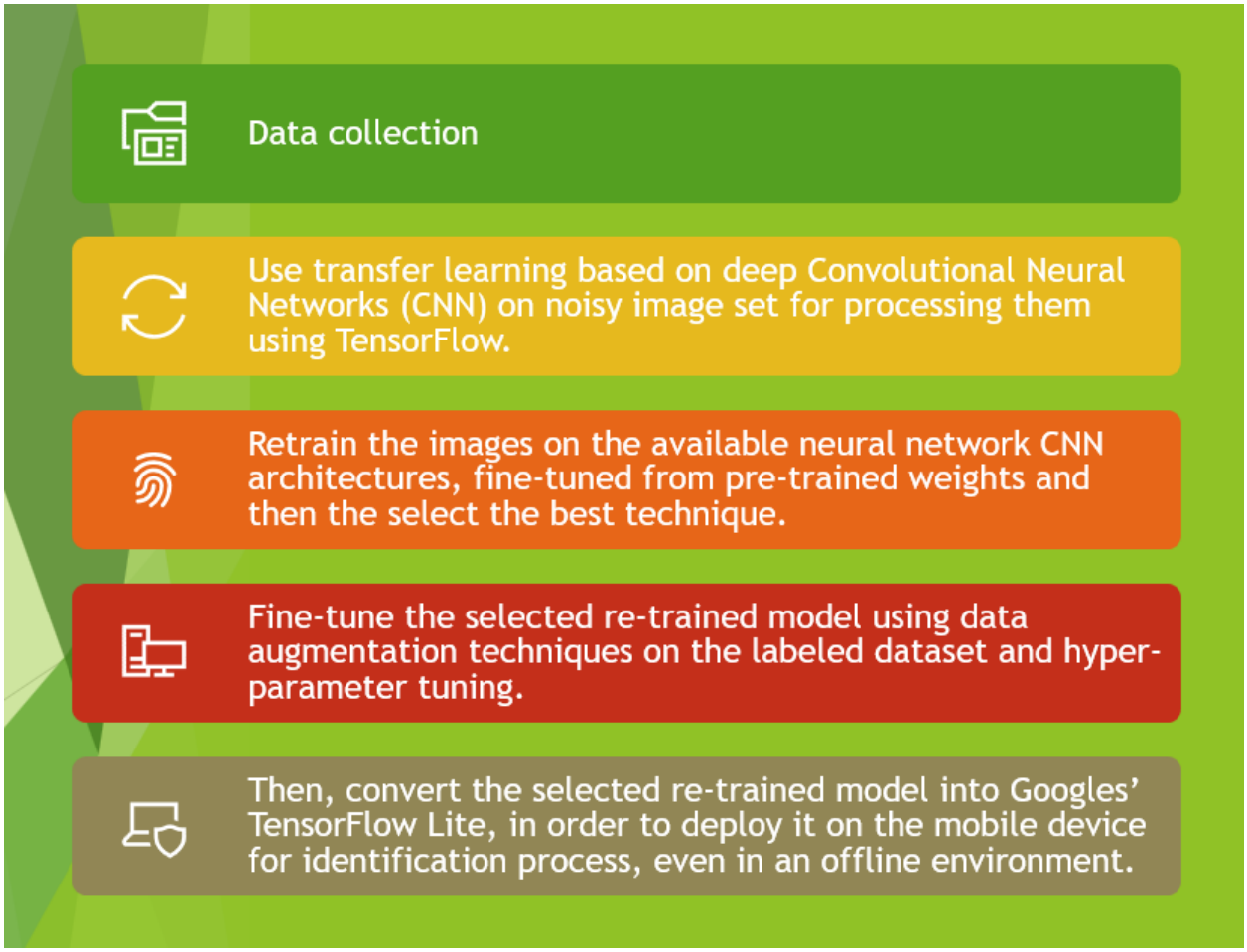


Figure 2.2.2.1: Methodology of the approach

2.2.2.1. Data Collection

In this research, the images of the leaf/digested root/fruit that was to be scanned using the application acted as the input for this phase. Among many herbal plants, 5 of the most important ayurvedic plants in Sri Lanka were chosen to analyze further in detail and the images of the plants were acquired from plant nursery of Navinna, Ayurveda Medical Hospital, social media, Institute of Ayurveda, alternative medicine website and blogs related to Sri Lankan herbal plants creating a noisy web data set.

The 5 types of plants chosen to be classified were:

1. Akkapana (leaf)
2. Cinnamon (leaf)
3. Katupila (leaf and fruit)
4. Kohomba (leaf)
5. Turmeric (leaf and digested root)

The following are some camera-captured images of the above mentioned ayurvedic herbs, acquired from the Research Institute at Navinna.

Figure 2.2.2.1.1: Akkapana leaf

Figure 2.2.2.1.2: Cinnamon leaf

Figure 2.2.2.1.3: Katupila leaf

Figure 2.2.2.1.4: Katupila leaf

Figure 2.2.2.1.5: Turmeric leaf

Figure 2.2.2.1.6: Turmeric root

Figure 2.2.2.1.7: Kohomba leaf

A set of 1348 camera captured as well as lab images were collected which included 214 Akkapana leaves, 230 Cinnamon leaves, 289 Katupila leaves including fruit, 249 Kohomba leaves, 158 Turmeric leaves and 208 turmeric digested roots. These were not the augmented images, but original images.

2.2.2.2. Methodology

The main target was to analyze several deep CNN models with the acquired training and testing data from scratch, get the final testing accuracy from each in order to compare and achieve the model with the highest accuracy, and then to use it as the finalized model in the herbal plant classification purpose in Arogya, based on images as the input from the mobile camera module.

The proposed identification method was based on running CNN, which has been recognized as a competent method for image recognition in the past few years, on mobile devices in an offline environment. CNNs work somewhat similar to neurons in the brain. It also performs image transformation with a certain degree of rotation and distortion [3]. Since this network is avoiding the complex preprocessing of the image, we were able to input the original image of the selected Ayurveda plant directly.

After acquiring the images of leaves/fruits/digested roots of the selected Ayurveda plants, they were labelled and annotated using “VGG Image Annotator” tool for multi-class classification, by forming 6 classes for the specific plant. After the dataset was prepared, transfer learning based on deep Convolutional Neural Networks (CNN) was used on the prepared image set for processing them using TensorFlow, in the local computer. When taken as overall, a sample of 1348 images were prepared, where 48%, 26% and 26% of that were used as training, testing, and validation splits, without data augmentation.

With the requirements of this functionality, we identified that it is easy to use transfer learning than building and training a model from scratch. However, training a model from scratch is too costly. To overcome this challenge transfer learning was applied in the Ayurvedic dataset. In transfer learning it is able to retrain an already implemented model with a custom dataset. A colab's [12] free GPU which is offered for 12 hours free, was used to re-train the deep learning CNN model.

2.2.2.2.1. Transfer learning vs. Non-transfer learning

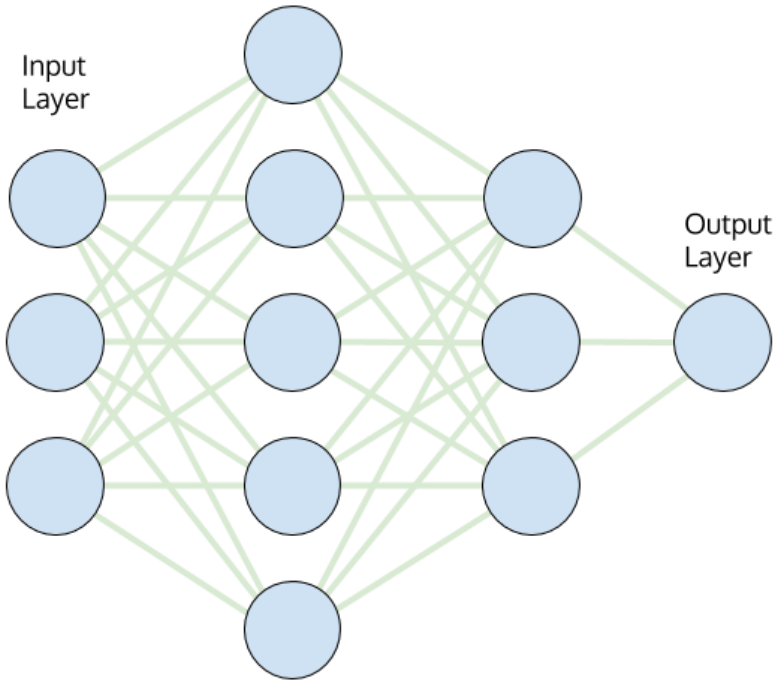


Figure 2.2.2.2.1.1: Transfer-learning neural network model, green color indicating the training of just the final layer's weights and biases and red color indicating fixed weights and biases
Source: https://towardsdatascience.com/how-to-train-your-model-dramatically-faster-9ad063f0f718

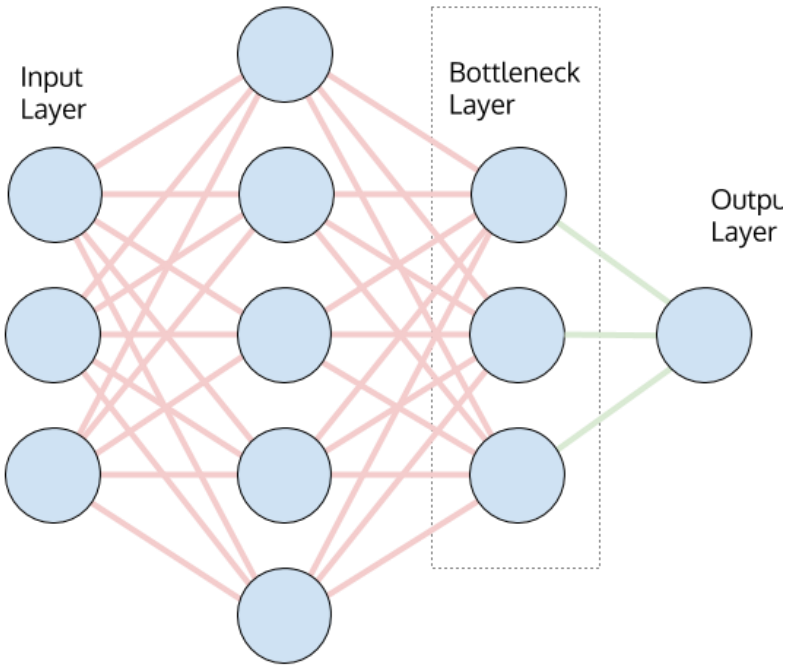


Figure 2.2.2.2.1.2: Transfer-learning neural network model, green color indicating the training of just the final layer's weights and biases and red color indicating fixed weights and biases
Source: https://towardsdatascience.com/how-to-train-your-model-dramatically-faster-9ad063f0f718

Following are the main benefits of using transfer learning in this problem,

1. Training on a new dataset is faster than implementing from scratch.
2. We can solve the problem with lesser amount of training data than the amount of data that we need if we start from scratch.

After that, the dataset was uploaded to google drive and were retrained with colab on the available neural network CNN architectures, then was fine-tuned from pre-trained weights and then the best technique with the highest accuracy was selected. While the training dataset was augmented, testing and validation datasets were not augmented for higher accuracy purposes.

The variants of deep Convolutional Neural Networks used to compare the initial training and testing accuracies included:

- InceptionV3
- MobileNetV2
- InceptionResNetV2
- Xception
- DenseNet121
- ResNet50
- VGG16

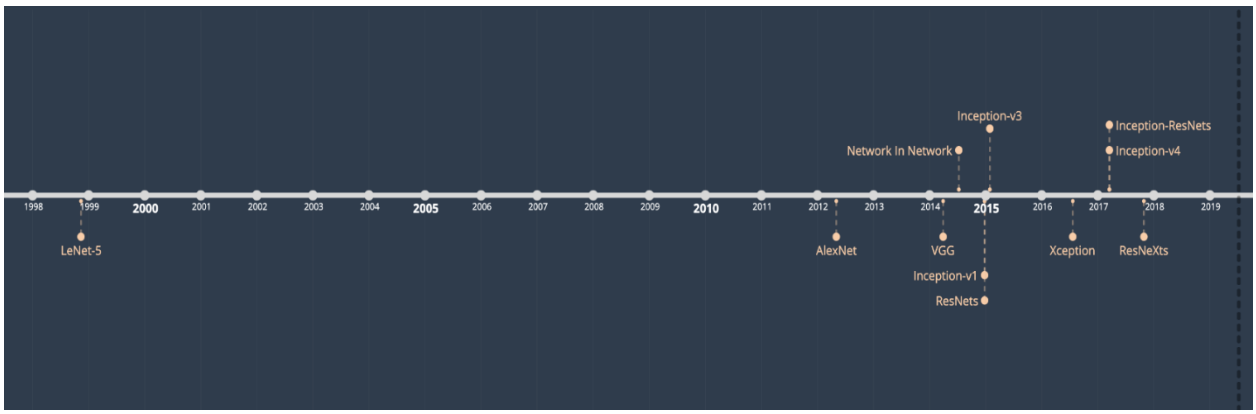


Figure 2.2.2.1: The 10 architectures and the year their papers were published.

Source: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614

Then, the selected re-trained model with the best accuracy was fine-tuned using data augmentation techniques on the labeled dataset and hyper-parameter tuning. While the training dataset was augmented, testing and validation datasets were not augmented for higher accuracy purposes. It was re-trained with the dataset using Keras which is an open source neural network library, written in python. It is user friendly, modular and extensible since it has been designed for fast experimentations with deep neural networks.

The next step was to deploy the re-trained model on mobile devices. Therefore, the re-trained model was converted for the deployment on mobile devices with Googles' TensorFlow Lite. TensorFlow Lite is a TensorFlow's lightweight solution for mobile devices which provides on-device machine learning inference with a small binary size and low latency. TensorFlow provides an interface for expressing machine learning algorithms and an application
for executing these algorithms [36].

Following diagram shows the architecture of TensorFlow lite.

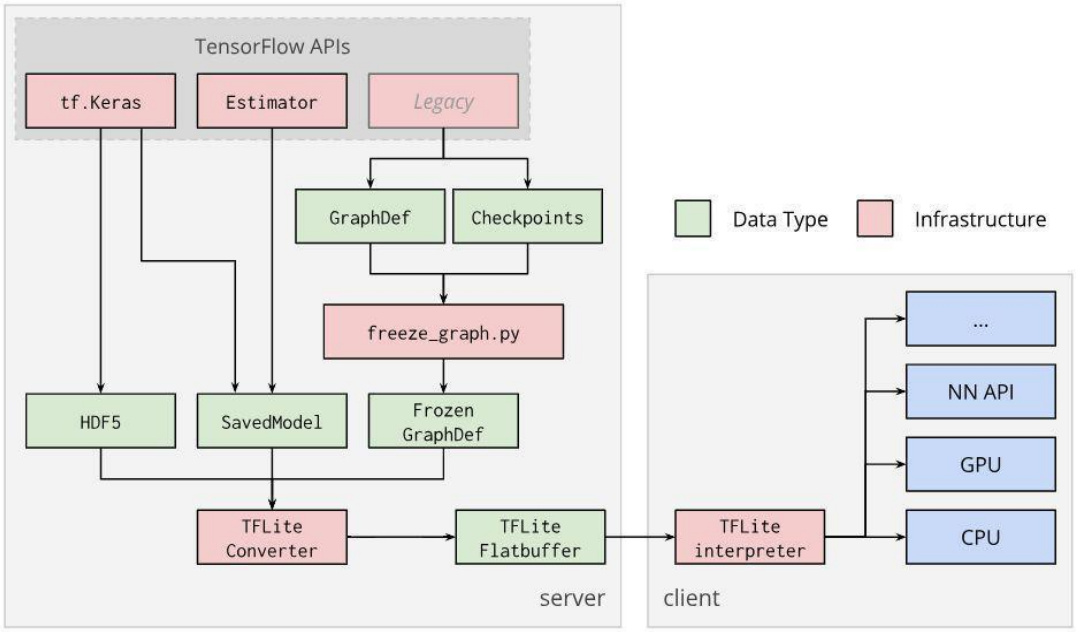


Figure 2.2.2.2.2: TensorFlow Lite Converter
Source: https://www.tensorflow.org/lite/convert

TensorFlow lite converter has been used to convert the re-trained model to TensorFlow Lite.

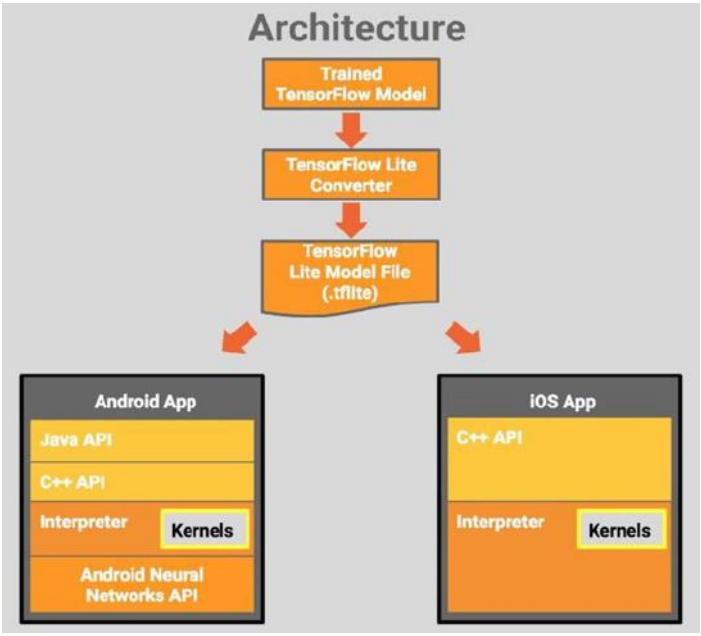


Figure 2.2.2.2.3: TensorFlow Lite Architecture
Source: https://www.tensorflow.org/lite/guide

One of the main challenges in detection of Ayurveda plants are similarity between different plant species in different phases of their life cycles and complexity of the background since the end user

will be using this application in deep forest environment. Therefore, we cannot just rely on a single feature, such as color, texture, or shape to distinguish them. The same species of leaves will be different because of the shades of colors, shape, scale, viewpoint etc. [35].

On a mobile device the Ayurveda plant detection has to be done with time, battery life critical manner, especially when it has to be done in a forest area .In the proposed system the whole identification process will take place on mobile devices and it doesn't require internet .Therefore this will be a great solution to identify invasive plants in deep forest areas, where mobile network coverage is not available .The mobile application will be built using android.

2.2.2.3. Summary of Methodology

The following table summarizes the methodology to be used in the functionality of Ayurvedic leaf classification precisely.

Research	**Research Work**	**Methods and Algorithms to be used**	**Expected Accuracy**	**Remarks**
Arogya – An Intelligent Ayurvedic Herb management Platform -> Classification of a group of rare ayurvedic leaves in Sri Lanka	Use transfer learning based on deep Convolutional Neural Networks (CNN) on collected image set for processing them using TensorFlow (The selected re-trained model will be finetuned using data augmentation techniques on the labeled dataset and hyper-parameter tuning.)	CNN Architectures (Such as Inception v3, ResNet50, VGG-16, MobileNet v2, etc.) will be used to choose the best technique	Higher Accuracy will be expected	Several existing algorithms will be analyzed, and accuracy will be tested in order to select the suitable algorithms to classify the selected 5 plants accurately.

Table 2.2.3.1 : Summary of methodology

2.2.2.4. High-Level Architecture Diagram

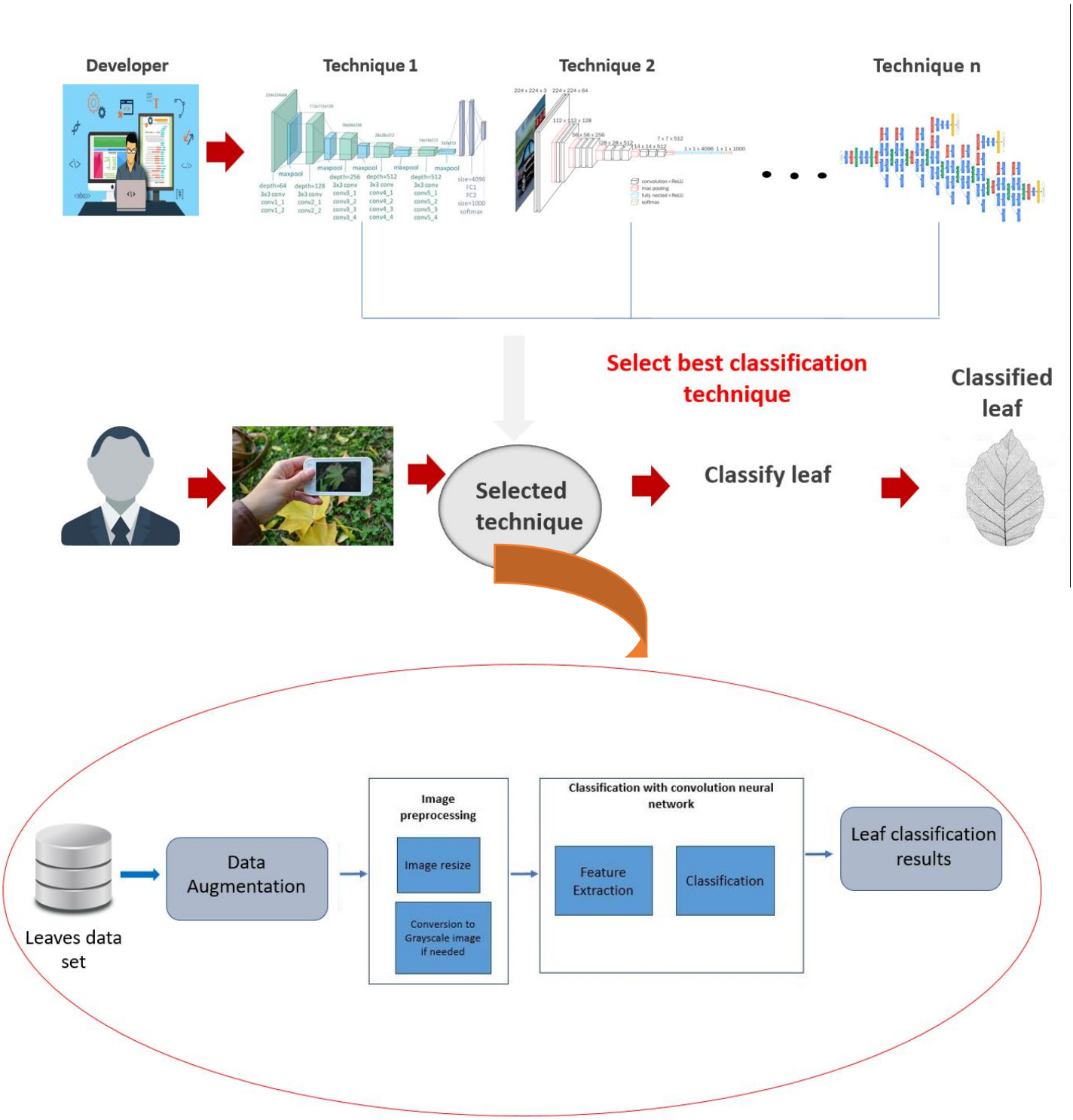


Figure 2.2.4.1: Software Architectural diagram

2.2.3. Abstractive web page Information Summarization and Location Mapping with GIS technology on Ayurvedic Plants in Sri Lanka Using a Mobile Application in an Online Environmental Approach

Main research area of this component was based on natural language processing techniques. Target was to extract dynamic information from multiple web pages and generate a summary on ayurvedic plants. So, in order to achieve this goal, many techniques were followed to select the best model out of them. Thus, several existing algorithms were analyzed, and accuracy was tested in order to select the suitable algorithms to extract the required information from dynamic web pages to generate a summary.

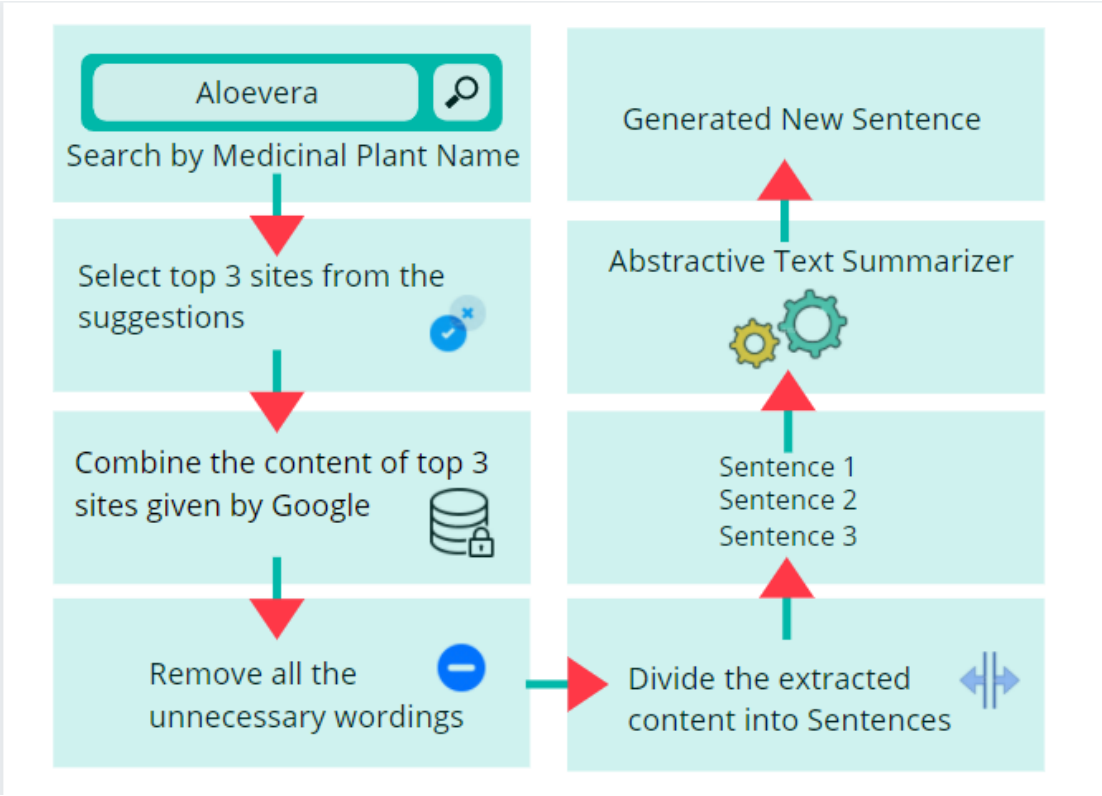


Figure 2.2.3. 1: Abstractive summarization process

The process of text summarization can be defined as generating an incisive and communicative synopsis while conserving the main content of the description and the overall outline. In general, for the process of text summarization, two main approaches are being executed called Extractive Summarization and Abstractive Summarization. As the methodology for this research, Abstractive Summarization was used. Instead of existing sentences, new sentences were generated which were totally differentiated from the original text. In contrast, what extractive summarization did was prepare the summary with the most important sentences which were extracted from the original text.

According to our working plan, the input was a set of descriptions of medicinal plants which were extracted from highly ranked multiple websites and the output was a short summary of sequence of words. Hence, this was modeled as a Many to Many Seq2Seq problem. Encoder and the Decoder are the two major components of a Seq2Seq model. In this research component Long Short-Term Memory was used as the encoder and the decoder. The Encoder-Decoder was set in two phases as training phase and Inference phase. As the first step Encoder and the Decoder were set in the training phase. The model was trained to predict the target sequence offset by one timestep. The way Encoder and the Decoder were set was as follows: the whole input sequence was read by the long short-term memory model at each timestep, one word was counted. Information at each timestep was processed and related information present in the input was captured. Decoder also worked as a Long Short-Term Memory network. Entire target sequence was read word by word to predict the same sequence offset by the decoder. Decoder was capable of predicting the next word in the sequence given the previous word. At the inference phase after doing training, the model was evaluated on a new source sequence. Target sequence was an unknown one. Inference Architecture was to be set up to decode a test sequence. At the final stage, the user retrieved a summarized paragraph on the specific Ayurvedic plant.

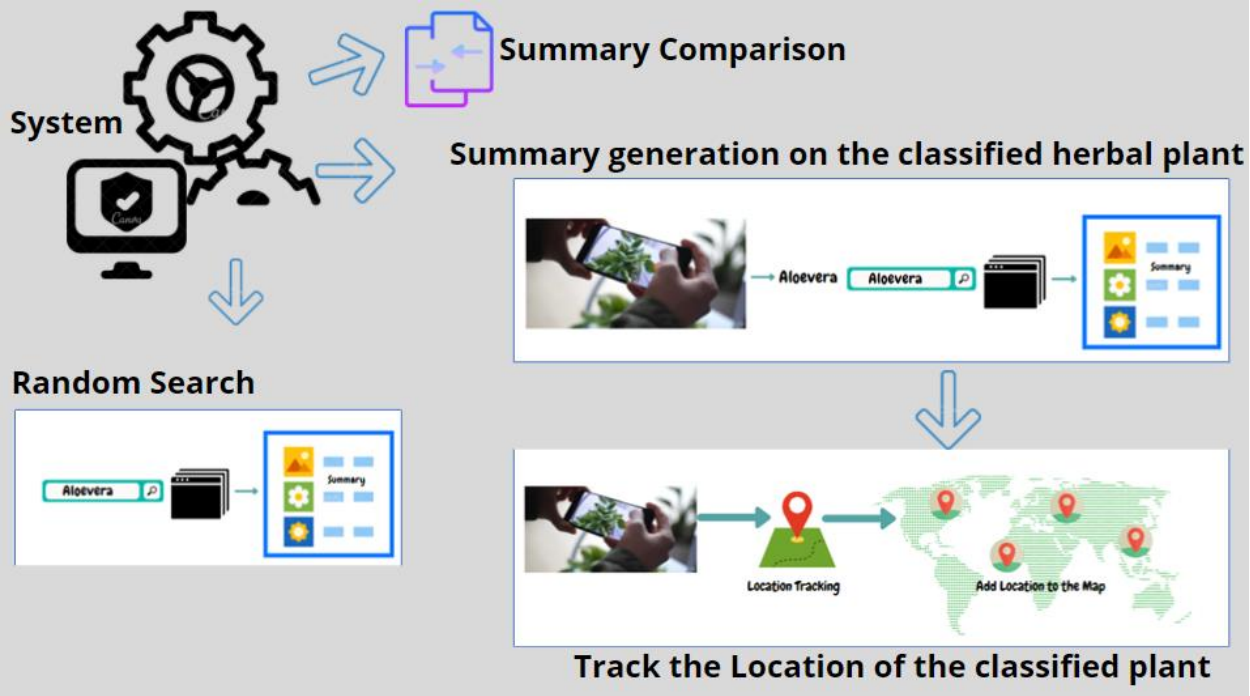


Figure 2.2.3. 2: System overview diagram

Expectation is to build a text summarizer where the input is a long sequence of words extracted from web pages and the output is a short summary of sequence of words. Thus, we can model this as a Many-to-Many Seq2Seq problem. Below is a typical Seq2Seq model architecture:

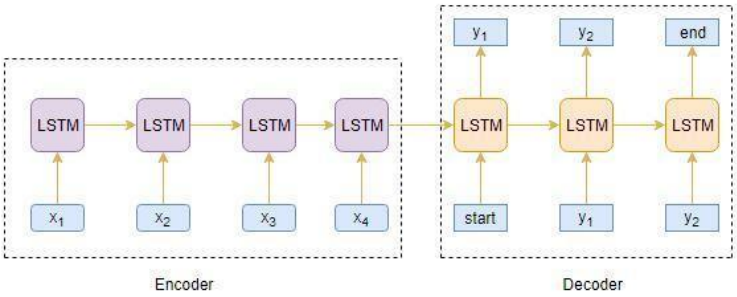


Figure 2.2.3. 3: Seq2Seq model

There are two major components of a Seq2Seq model as, Encoder and the Decoder.

Generally, variants of Recurrent Neural Networks (RNNs), i.e. Gated Recurrent Neural Network (GRU) or Long Short-Term Memory (LSTM), are preferred as the encoder and decoder components. This is because they are capable of capturing long term dependencies by overcoming the problem of vanishing gradient.

We can set up the Encoder-Decoder in 2 phases:

- Training phase
- Inference phase

2.2.3.1. Training phase

In the training phase, we will first set up the encoder and decoder. We will then train the model to predict the target sequence offset by one timestep. Let us see in detail on how to set up the encoder and decoder.

2.2.3.2. Encoder

An Encoder Long Short-Term Memory model (LSTM) reads the entire input sequence wherein, at each timestep, one word is fed into the encoder. It then processes the information at every timestep and captures the contextual information present in the input sequence.

Below diagram illustrates this process:

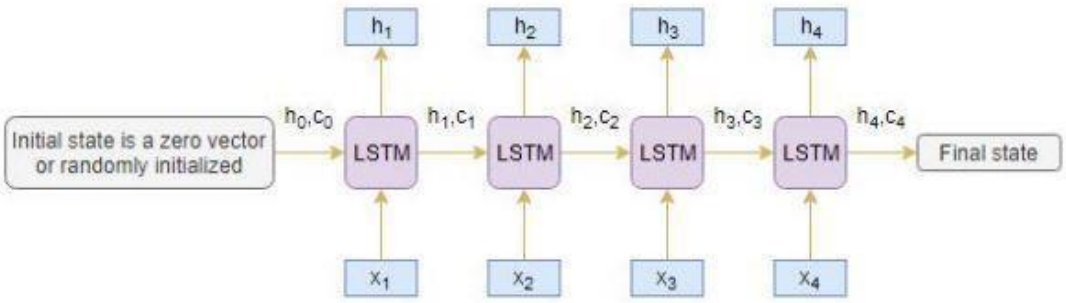


Figure 2.2.3.2. 1: Encoder Process

The hidden state (hi) and cell state (ci) of the last time step are used to initialize the decoder. Remember, this is because the encoder and decoder are two different sets of the LSTM architecture.

2.2.3.3. Decoder

The decoder is also an LSTM network which reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. The decoder is trained to predict the next word in the sequence given the previous word.

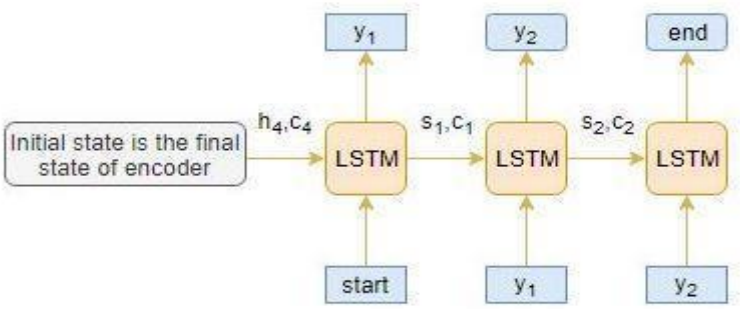


Figure 2.2.3.3. 1: Decoder Process

<start> and <end> are the special tokens which are added to the target sequence before feeding it into the decoder. The target sequence is unknown while decoding the test sequence. So, we start predicting the target sequence by passing the first word into the decoder which would be always the <start> token. And the <end> token signals the end of the sentence.

2.2.3.4. Inference Phase

After training, the model is tested on new source sequences for which the target sequence is unknown. So, we need to set up the inference architecture to decode a test sequence:

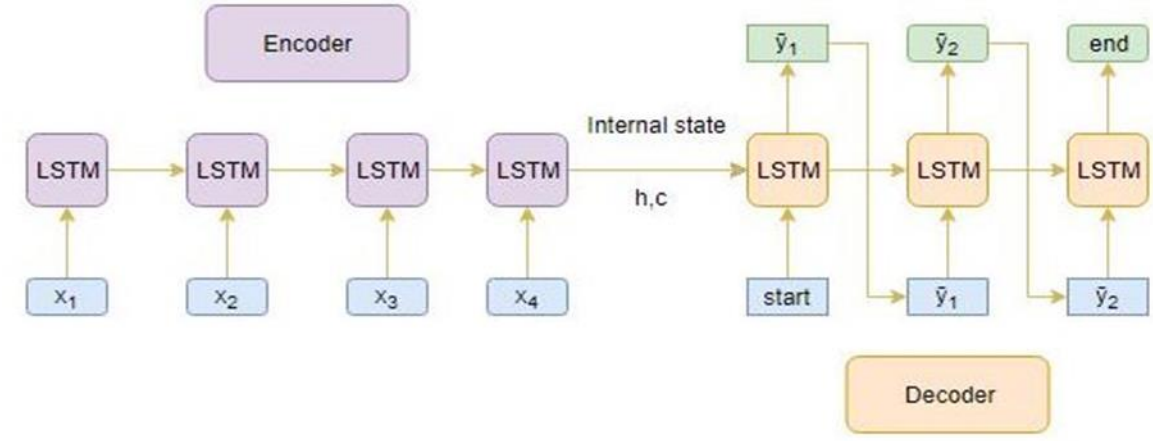


Figure 2.2.3.4 1: Inference Process

Here are the steps to decode the test sequence:

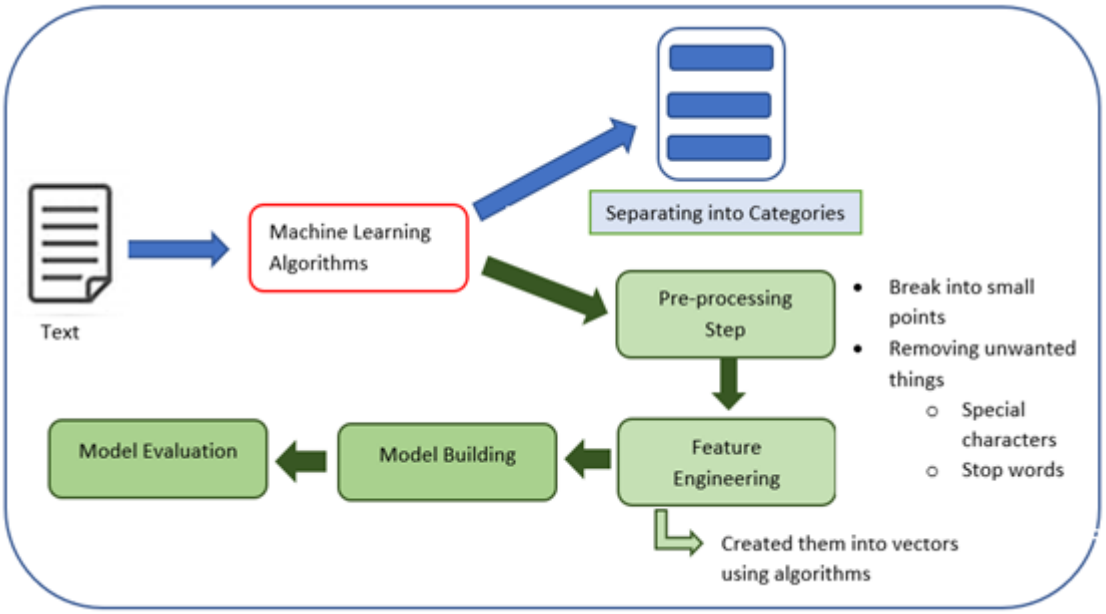
1.Encode the entire input sequence and initialize the decoder with internal states of the encoder
2.Pass <start> token as an input to the decoder
3.Run the decoder for one timestep with the internal states
4.The output will be the probability for the next word. The word with the maximum probability will be selected
5.Pass the sampled word as an input to the decoder in the next timestep and update the internal states with the current time step
6.Repeat steps 3 – 5 until we generate <end> token or hit the maximum length of the target sequence
So that, with the support of encoder-decoder architecture Arogya will be able to manage the summary generation functionality.

2.2.4. Text classification of Herbal plants and predict the category type of diseases and Identifying the Ayurveda image texts using OCR techniques and ayurveda related posts/recipes using cosine text similarity.

2.2.4.1 Text classification of Herbal plants and predict the category type of diseases

As considering the Background Process of Text Classification using NLP,

This is used for a Text classification process while typing its sentiments of needed herbs. Generated data has a variety of tabular data columns that have either numerical or categorical data. NLP (Natural Language Processing) is applicable in several problems from speech recognition, language translation. Classifying documents to information extraction. This helps identify the sentiment of herbs, finding entities in the sentence category of texts. NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it.



System Architecture

Text classification process regarding the flow is implemented under the NLP. In order to that, NLP supports here to identify the category of texts and Information extraction. Their end to end text classification which is a pipeline consists of three main components. According to the Figure . It represents Dataset Preparation (Preprocessing Text), Feature Engineering, and Model Training are the major components of this classification process. Therefore, consisting some points regarding the each of levels describing briefly as per in below.

2.2.4.1.1 Data collection

For this disease prediction component, this project needs different data that needs to be collected from external entities. Creating a dataset with 85 records for analyzing the relevant disease. Dataset consists of 33 detailed information for analyzing its parts of the entire herbal plants. There having only 87 records for predicting the relevant disease types.

Category, Propagation, Cultivation, seedsShape, seedsColor, rootSystem, rootBehaviur, rootSizeCm, rootColor, stem behaviour, stemColor, stemSizeCm, HavingFruits, FruitColor, FruitShape, FlowerBlooming, FlowerBehaviour, Flower color, Leaf_Look, Leaf Base, LeafLengthRangeCm, Leaf_color, petiole, Venation, Leaf_Shapes, Leaf Types, Phyllotaxy, rhizomeBehaviour, Types, Parts of used, Family name, Scientific name, Sankrit Name, English name, Local name, Target

These are the columns that were taken for analyzing the relevant disease type.

2.2.4.1.2 Data preprocessing

During the pre-processing stage, unwanted data columns and null rows were removed.

Loading a relevant dataset and performing basic pre-processing. If there are unwanted things which should be ejected on this stage. Hence, the dataset split into train and validation sets. The Traditional text classifiers usually break the documents into small word fragments(n-grams) and locate them as separate dimensions in the fragment hyperspace. These steps will be used for a typing field provide for special properties/features

According to the application, Giving an Interface includes a separated dropdown button for selecting most appropriate words related to the medicine plants. After selecting, all the dropdowns there will be displayed with a button for prediction of the disease type.

Raw dataset transforms into flat features using Machine Learning model and This process is going on creating new features from the existing data. Raw text data will be transformed into feature vectors and new features will be created using an existing text data. According to the new features from the dataset can be identified different ideas. Such as, Count vectors as features, TF-IDF Vectors as features, Word as features, Text NLP based features, Topic model as features. But this function not able to do such type of

bored things to classify. Hence, Naïve Bayes algorithm is used for all the NLP based TF-IDF.

According to the application, providing a same dropdown option to selecting the specific characteristics if the plants are consisting on. If this dropdown will not be included in needed characteristics, then giving a limited space for typing it with main points. So that, there only typing field must consider for the feature extraction part. There is a space for typing any special feature of the medicine plant in the provided area. There should be considered filtering options such as removing emoji, punctuation marks, spaces, special characteristics etc. Then it will be transformed into a feature vector. This field is not a required field and it’s not always because most of the special characteristics are included in the dropdown field.

2.2.4.1.3 Model Creation and Training

Machine learning models are trained on a labeled dataset. In there, data which are taken from dropdown fields should be moved to analyze options using Algorithms. Other data which are from typing areas should have to label and categorize according to the relevant types of category diseases. Before that these types of data came after doing previous steps. (Pre-processing and Feature Extraction). After going on both components, I should have to Improve the performances of text classifiers.

According to the application,

This is the Planned Mobile UI of this Analyzing the category of diseases part. All the Fields should be required excepting a provided typing area. Finally, all the files are filed then pressing a button “Analyzing” move to the next interface. Then displaying a Category after taking some time period of loading. Apart from that this interface will be displayed some recipes related to the displayed category. Then the user can select the most usable recipe and give a chance of visualizing some of the rare ingredients which are included.

Text Classification Using Tools

Both NLTK and TextBlob perform well in Text Classification processing. These are the features explained in the table below.

NLTK	**TextBlob**
NLTK is a very big library holding 1.5GB and has been trained on huge data with proving different dataset in multiple languages which can deploy according to the functionality it's required. NLTK is a powerful Python package that provides a set of diverse natural language algorithms.	TextBlob library which is a python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as noun phrase extraction, sentiments analysis, classification, translation etc.

this function will be applied for classification task. Under the Features there are some points to follow on.

1. Classification – Using Naïve Bayes and Decision Tree

2. Tokenization – Splitting text into words and sentences

3. Spelling correction

4. Emojis

Comparison of Text classification tools

There are lots of tools to work with NLP. NLTK, Spacy, Stanford Core NLP. These are the comparison of properties of tools.

	NLTK	Spacy	Stanford Core NLP	TensorFlow	Allen NLP
Build an end-to-end production application	✓	✓	✓	✓	✗
Efficiency on CPU	✓	✓	✗	✗	✗
Train models from own data	✓	✓	✓	✓	✓
Different neural network architectures for NLP	✗	✗	✗	✓	✓

Comparison of text classification tools

Text Classification Using Algorithms

Consisting of the most common algorithms such as *tokenizing*, part-of-speech tagging, topic segmentation, named entity recognition, etc. When considering this function, it helps the computer analyze, pre-process, and understand the written text. In This part, only use for the

typing field and this field consist of limited words as key words of herbal characteristics. Because if we give a large description there should be more stuff.

Text Classification Steps for typing field:

1. Loading data and Creating classifiers:

   First create custom classifiers using the TextBlob module. Before that creating some training and test data. Then creating a Naïve Bayes Classifier for passing the training data into Constructor.

2. Loading data from Files:

   Loading data from common file formats including CSV, JSON, and TSV

3. Classifying Text or Classifying TextBlobs

4. Evaluating Classifiers

   Compute the accuracy of the test data using a relevant method

5. Updating Classifiers with New Data

6. Feature Extractions.

   We can use our own extractors to identify each specialty.

Text Classification – Flow Chart

This application will classify only 3 or 4 category type of diseases such as Diabetics, Arthritis and Cancer or Sugar. According to the above categories giving by the users(dataset), has been labeled into some variables like color, length, shape etc.

Then using feature extraction functionalities like locate vectors etc. and for classifying them into model.

So that easily predicting a solution on a model evaluation step.

After that selecting a Most usable medicine recipe and visualizing ingredients as Feature extraction. Here take those images from the database which is used to store all the images in initial stage

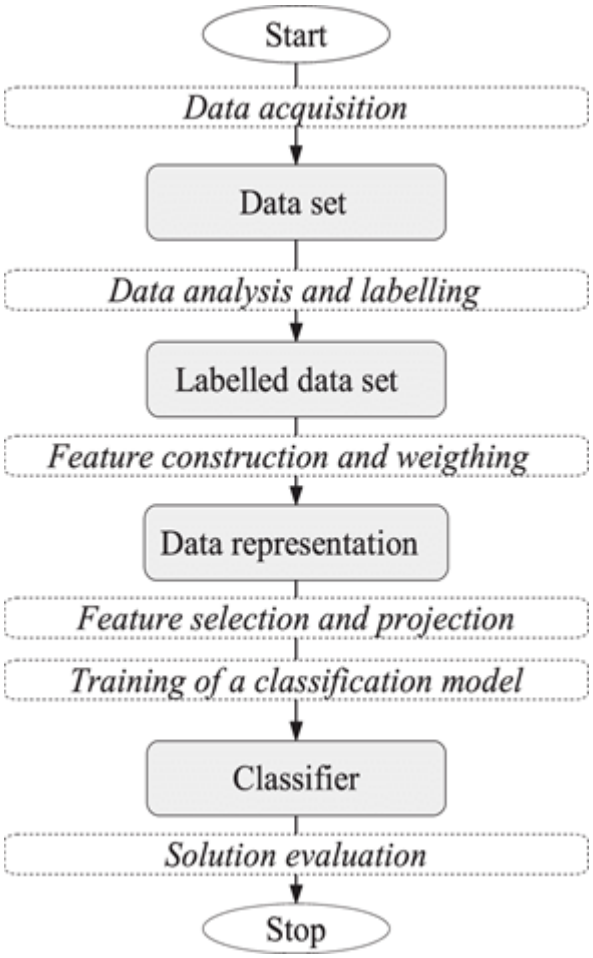
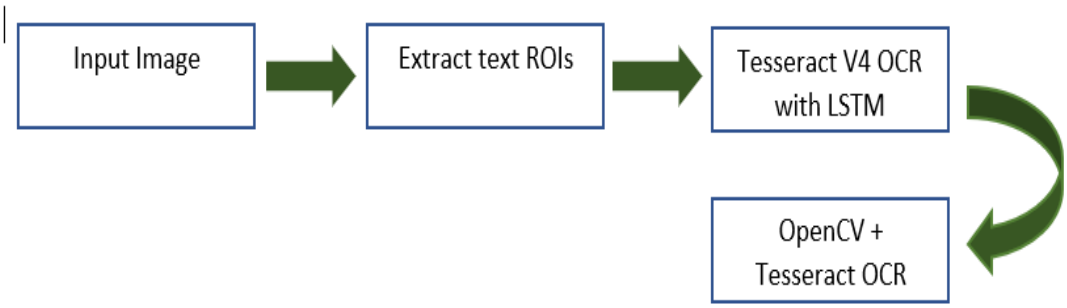


Figure 3.5 – Text classification flow chart

2.2.4.2 OpenCV OCR and text recognition with Tesseract



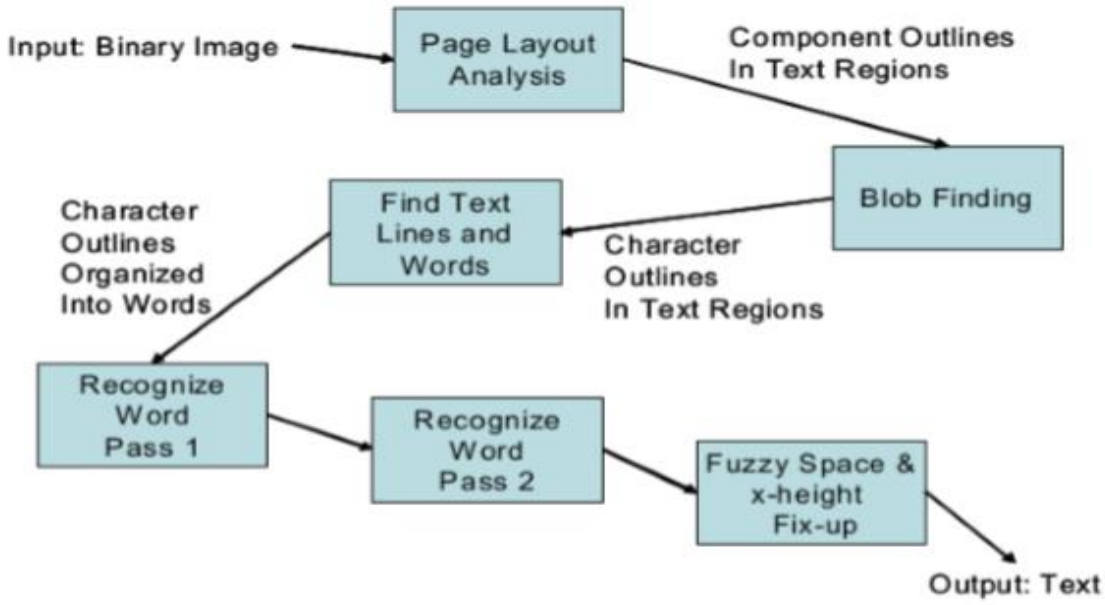
The OpenCV OCR pipeline

Identify the image wrapped with text for recognition using OpenCV, Python and Tesseract. Tesseract v4 which includes a highly accurate deep learning- based model for text recognition.

So that, firstly I need to install Tesseract V4 to the machine as the main thing to identify the text recognition of the included image. The latest release of Tesseract (v4) supports deep learning-based OCR that is significantly more accurate.

Firstly, install the pytesseract package to access Tesseract via the Python programming language. Then develop a simple python script to load an image. Then binarize it and pass it through the Tesseract OCR system. Finally, test OCR pipeline on some images and review the results. Tesseract is one of the most accurate open-source OCR engines in which development is sponsored by Google.

Tesseract is designed to be language independent. At the beginning the aim of Tesseract was to recognize white on black. Which led to the design in a way of connected parts analysis and operating on outlines of parts.



Block diagram of basic components of Tesseract

Extracting text from the image,

Here used the Tesseract-OCR for an image for a text to be extracted. This Tesseract has a very highly optimized group of algorithms itself. "python-tesseract" module in python implements tesseract-OCR to convert the image into text. Their implementation is simple to carry out because the module and call the defined method 'image_to_string' converts the given image to string. Tesseract converts and returns the text available in an image.

2.2.4.2.1 Installing packages

- Install Tesseract + Python bindings

Need to install the Tesseract + Python bindings to create python scripts. Then it could communicate with Tesseract and perform OCR on images processed by OpenCV.

- Install PIL (Python Imaging Library)

Used to pip install pillow, a more Python-friendly version of PIL (a dependency) which is followed by pytesseract and imutils.

- pip install pillow
- pip install pytesseract

- Install *argparse* Package

This is included with Python and handles command line arguments

Finally, below packages are installed to identify the OpenCV OCR image recognition. Handle below imports and the class Image is required to load the input image from disk in PIL format using pytesseract.

There are two types of line arguments to be considered. Image and the preprocess.

- Image: The path to the image which is sending through the OCR system
- Preprocess: This is the preprocessing method. There are two values thresh (threshold) or blur

Then load the image and use the preprocessing method specified by the command line either threshold or blur.

Load the Image from disk into memory and convert it to grayscale.

Image = CV2.imdecode(np_data, cv2.IMREAD_UNCHANGED) And the

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) are used for this.

This is performed at a threshold to segment the foreground from the background using cv2.THRESH_BINARY and cv2.THRESH_OTSU flags. This thresholding method can be very useful to read the dark text included in the image that is overlaid upon gray scale.
After that, applying a median blurring to remove the noise of the image. This median blurring could be helped to reduce the salt and paper noise. Although it makes it easier for Tesseract to correctly OCR the image.
After preprocessing the image, using the command "*os.getpid*" to derive a temporary image filename based on process ID of python script. Finally, apply OCR to the image using Tesseract python "bindings". Applied commands are displayed under the Implementation section.

Converts the contents of the image into desired string text using "*pytesseract.image_to_string*". This will be passing a reference to the temporary image file residing on disk.
Using "*os.remove(filename)*" to do some cleanup process where the file was deleted in temporary. Then it will be printed text to the terminal.

2.2.4.3 Identifying the Ayurveda related posts using cosine text similarity.

When considering the two similarity words or sentences using the techniques to identify is popular in Jaccard similarity and Cosine similarity. In here using Cosine similarity for identify the text similarity for the recipes which are Ayurvedic related or not.

2.2.4.3.1 Cosine similarity Algorithm

Cosine similarity is used for measuring the similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. This similarity has a score ranges from 0 to 1, with 0 being the lowest (at least similar) and 1 being the highest (the most similar)
Example of the Cosine similarity calculation for two vectors A and B in below.

$$\text{Similarity} = \cos(\text{Ø}) = \frac{A.B}{} = \frac{\sum AB}{}.$$

$$|A||B| \qquad \sqrt{\sum A_i^2}.\ \sqrt{\sum B_i^2}$$

Applying the cosine formula for this research component as per n below figure.

```
# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
results.append(cosine)
```

Apply the Cosine formula

2.2.4.3.2 Installing Packages

Install the NLTK package which is the massive toolkit related to the natural language processing (NLP). Below packages should be installed for identify the text similarity.

- Pip install nltk (Then entire the python shell in displayed terminal by simply typing **python**)
- Type import nltk
- nltk.download('all')
- pip install pickle

Pickle is basically used for the serialization and deserialization a python object structure. As further describing, it is the process of converting a python object into a byte stream to store in it a file or database, thus maintain program state across sessions or transport data over the network. *Importing json* is the meaning of importing the json it is the string version that can be read or written to a file. Python has a built-in package called json which can be used to work with json data also.

2.2.4.3.3 Process of Text similarity

When considering the demonstration of this process, the important one is to consider the angle between the two vectors, not the magnitude of the vectors.
If the angle between two vectors is 0, then the similarity would be 1. Conversely
If the angle between two vectors is 90, then the similarity would be 0.
If the two vectors with an angle greater than 90, then the similarity would be 0 also.

Mainly consider the two functions as nltk.tokenization and nltk.corpus.

nltk.tokenization is the process by which a big quantity of text is divided into smaller parts called as tokens.

The meaning of **nltk.corpus** is used to get a list of stop words. “the”, “a”, “an”, “in” are the stop words we are commonly used in.

Each entity that is the part of whatever was spilt up. Each word is a token when a sentence is tokenized into words. Basically, tokenizing involves splitting sentences and words from the body of the text.

2.3. High Level System Architecture Diagram of the whole system

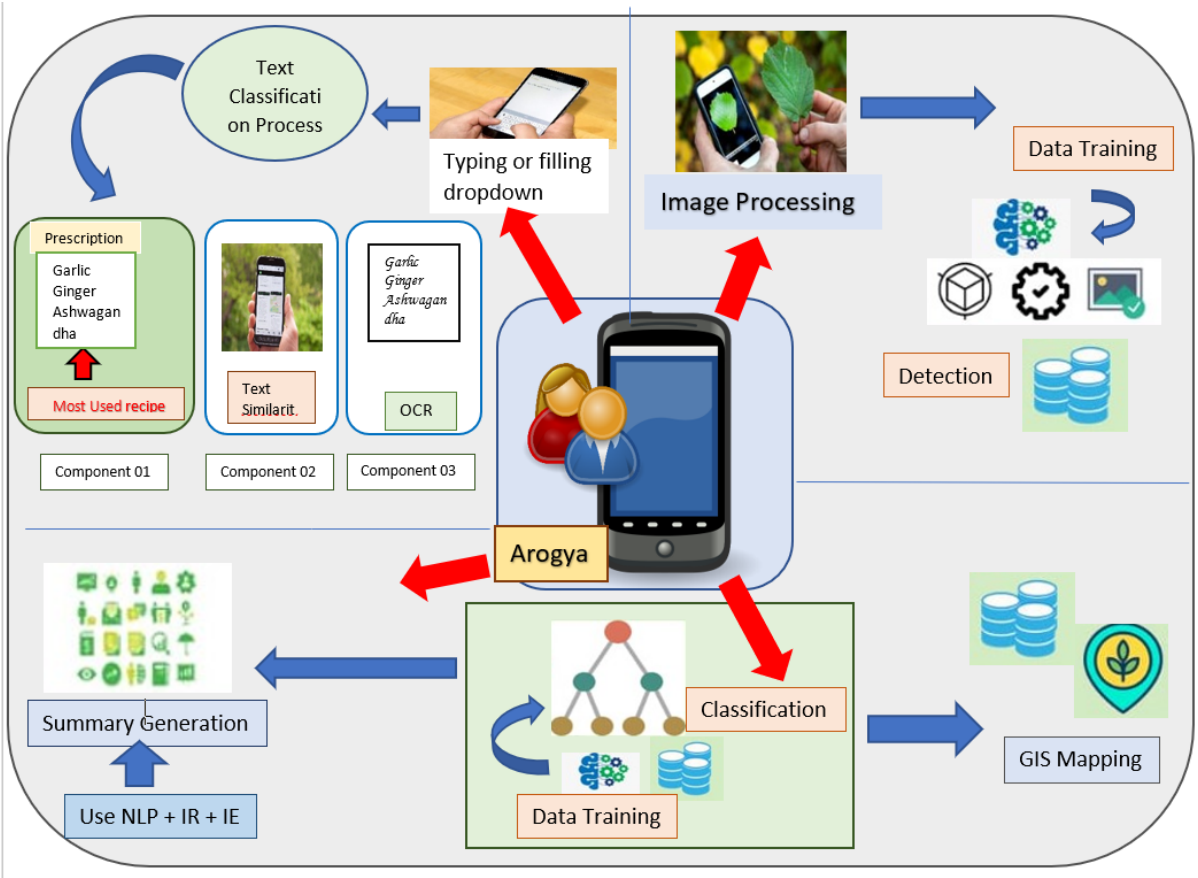


Figure 2.3.1: System high level architecture diagram

2.4. Project Requirements that have been achieved

2.4.1. Functional Requirements

1. Classification of a set of selected Ayurveda herbs in an offline environment has been achieved.
2. Selecting the best classification technique with comparing several transfer learning models based on deep CNN has been achieved.
3. Comparison of the accuracies of these models with justification has been achieved.
4. Request to re-train the existing model with new datasets by premium uses can be done.
5. Showing the classification history on the mobile device has been implemented.
6. Management of Ayurveda herb details in mobile device has been implemented.
7. Display information on top of the plant species
8. Summary should be generated regarding the medicinal plant
9. Application should be able to track the location of users
10. Medicinal plant’s description should be available for offline view
11. Locations of plants should be able to view as a Map view

2.4.2. Non-Functional Requirements

1. The application is able to give the results **as fast as possible**.
2. Accuracy is high, and results are efficient. The **reliability** of this research function can guarantee 95% in average. (for Akkapana – 100%, Cinnamon – 90%, Katupila – 90%, Kohomba – 100%, Turmeric root – 100%, Turmeric leaf – 95%)
3. **Feasible** in order to understand the functionalities of the app.
4. **Usability**- Usability means how easy to use this mobile application, also it includes how easy it is to learn and how easily user can use this application. This mobile application has user guidelines which provide all the information about how to use this mobile application properly.
5. **User friendly** interfaces have been provided. User interfaces of mobile app are looking pleasant and even attractive, often promoting confidence in its use. The way to use the mobile application is understandable by the image icons in the interfaces.
6. **User Experiences** are properly managed in order to achieve the specific functionality.
7. **Safety**- This application is not causing harm, injury, or damage to the user or his/her mobile.
8. **Security**- Security of this app is high because only a registered user can use this app and they have to log in before they use this application. Also, user has to provide suitable username and password to access this mobile app which was given at the registration.
9. **Modifiability**- The system has the ability to add new features and new data. Therefore, whole application is built with object oriented and module concepts.

2.4.3. Other Requirements

To run this mobile app, user needs an android mobile which has the capability to store 30MB storage space. No internet connection is needed to use this mobile application, can be used in an offline approach.

2.5. Consideration of the aspects of the system

Standards: We followed coding standards when doing our individual coding parts. Coding standards tell developers how they should write their codes. All the group members used object-oriented concepts to maintain coding standards. Inside the code, we commented important things. When writing reports and referencing, we followed IEEE format.

2.5.1. Social aspects

This mobile application can be used by any person who is not aware about but keen to experience Ayurveda medication worldwide. For example, people who use and wish to use ayurvedic medicines and treatment, researchers in the field of botany, medicine, chemical structure analysis, agriculture, ayurvedic medicinal practitioners, taxonomists, forest department officials, those who are involved in the preparation of ayurvedic medicines and others who are concerned with plant studies, as well as doctors, students, locals, foreigners, Ayurvedic plant sellers and many more can use this application wisely. Therefore, a great appreciation as well as a demand can be expected from the society for this mobile application. Especially, there is no age limitations for the users, no need of advanced computer literacy, as well as no need of advanced knowledge in Ayurveda field to use this app. Only a simple knowledge of how to use a smart mobile phone will be sufficient for this. In addition, the name of the identified plant is given in native as well as in the scientific name of the plant, which would support any person including foreigners. So, this would be very beneficial for the whole society which would do a great service in uplifting our ancient Ayurveda, which is anyway better than modern medicine due to its naturality.

2.5.2. Security aspects

Security of this mobile application is high, because only a registered user can use this app and they have to log in before they use this application. Also, user has to provide suitable username and password (correct credentials) to access this mobile app which were given at the registration process. Credentials of all the registered users are stored in the firebase database with high security rules, so there will not be any unauthorized access for them because it is pretty similar to the security rules followed when creating the Facebook application. (this mobile application is created as a centralized social media platform) Server-side security is also very high because all the

services are hosted in AWS (Amazon Web Services) with IAM (Identity Access Management) user credentials.

2.5.3. Ethical aspects

This application is not causing any harm, injury, or damage to the user or his/her mobile. No unethical behaviors, rules or principles have been followed in the implementation of this mobile application. This will be capable of identifying majority of ayurvedic plants through any part of it like leaves, root, fruit, flower, etc. if retrained with more datasets by professionals. However, this does not replace the role of a plant taxonomist or an Ayurveda doctor, but it supports any person without any background knowledge about Ayurveda to classify a particular Ayurveda plant and to know about the medicinal value of it hopefully. This would more be a learning resource for almost all persons who need to experience Ayurveda.

2.5.4. Limitations

- This application is currently built only in English. Further, this can be applied in native languages such as Sinhala, Tamil as well as in any of the other languages preferred.
- Since the system is designed only as a mobile application, later can be improvised to a web application with the same functionality and content.
- Currently this mobile application is developed only for Android users, so have to consider about other mobile platforms and operating systems as well.
- The development strategy in this approach is only to identify 5 categories of plants, but the same strategy and methodology can be used and extended to identify any ayurvedic herb worldwide.
- If the user ends up with doubts and clarifications regarding this procedure, this application can be facilitated with consultation help from Ayurveda doctors.

2.6. Commercialization aspects of the product

2.6.1. Target Audience

- People who use ayurvedic treatment
- Researchers in the field of botany, medicine, chemical structure analysis, agriculture, ayurvedic medicinal practitioners, forest department officials, those who are involved in the preparation of ayurvedic medicines and others who are concerned with plant studies
- Doctors, Students, locals, and foreigners
- Ayurvedic plant sellers

2.6.2. Market Space

- No age limitations for the users
- No need of advance computer literacy
- No need of advance knowledge in Ayurveda field

2.6.3 Revenue Earning

- Through subscription fee
- Revenue via additional services

2.7. Testing and Implementation

Testing plays a major role in this research. The research is related to the health sector, due to the interconnectivity with Ayurveda. So, the implemented functions must be accurate since the software deals with the lives of people and affects the ingredients of medical recipes.

The initial process of the mobile application is detecting the constitution based on a photo that incorporates image processing. The image is uploaded for distinguishing the plant part (leaf/fruit/root) needed to be classified. There is a possibility where analysis can lead to inaccurate information if the conditions that affect a photo change. Therefore, the testing of image processing part should be tested under different conditions of lighting, the angle of capture, brightness, zoom, rotation and the distance between the object and capture device. All these different conditions which varies from each other were obtained through data augmentation technique.

2.7.1. Selecting the Most Accurate and the Highest Performance Segmentation Technique using Experimental Outcomes of Plants in Sri Lanka

Arogya is a mobile application which is divided into sub sections and developed individual component wise by the Arogya research team members. Developing individual components includes designing, analyzing, coding and unit testing etc. Server-side implementation and all back-end related services implemented with the use of python and front-end is an android application.

Software interfaces

Since developing a mobile application, below indicated software to use for development purposes.

- Android studio

  This IDE is used for front-end development of the application.

Figure 2.5.1-Android Studio IDE

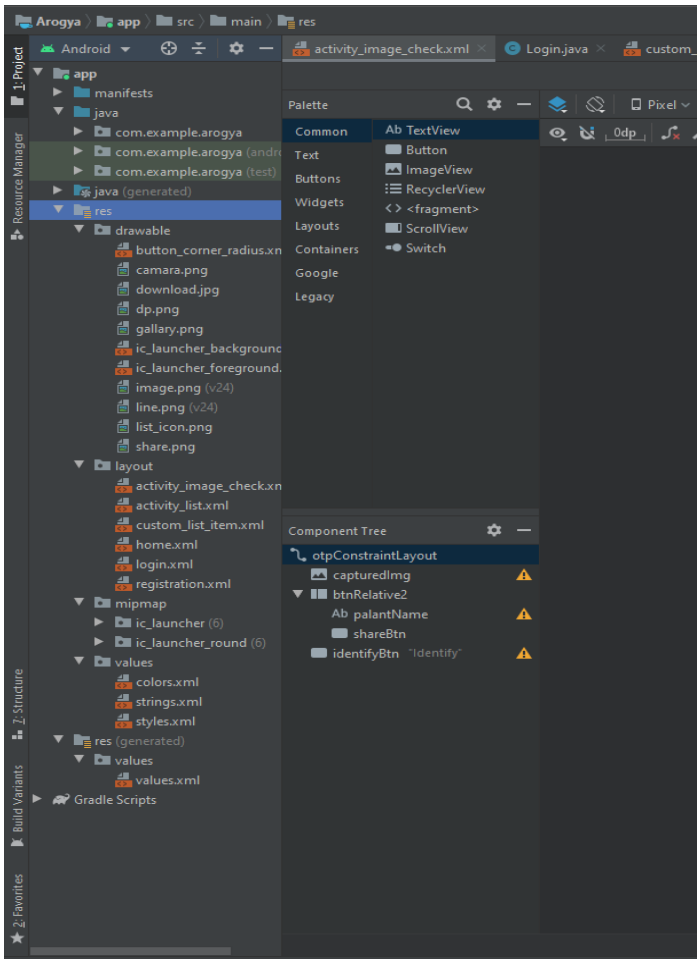
- Spyder

  This IDE is used for front-end development of the application.

Figure 2.5.2-Spyder IDE

Frontend Structure

The below figure shows the directory structure of the Android project.



Backend Structure

Used Technologies

- **Python 3.6**
- **OpenCV**
- **Keras**
- **Google Colab**
- **Roboflow**

From the starting point of the project, we followed an agile model of development until the end of the project. During the implementation period, we used Github for the version controlling of our application. Clearly identified features and independent development of them in different branches allowed the features to be integrated to the main system successfully without major merge conflicts.

Following code snippet indicates the implementations of the research component.

2.7.2. Classification of Ayurvedic Plants in Sri Lanka Using transfer learning based on deep CNN in a Mobile Application in An Offline Environment Approach

2.7.2.1 Implementation

The client side runs as a mobile application which was built on the top of Android. The user login and registration along with profile management, herb classification and providing biological details about the classified herb run on the mobile client itself.

The finalized deep CNN model achieved with the highest accuracy of 99.53% related to herb classification (VGG-16) was trained on Jupiter notebook, and retrained on Google Colab itself with data augmentation techniques. The data augmentation techniques included position (crop, scale, zoom, rotate, padding) and color (brightness, contrast, hue) techniques. The implementation of the model was based on Python environment, Keras, Tensorflow and OpenCV. After the best model was obtained, it was converted to Tensorflow Lite in order to be deployed in the Android Operating System.

Server-side API s related to herb classification, giving the result related to the correctly classified herb and giving the biological details of the specific herb were hosted and deployed on Azure as cloud services. The activation of those deployed services depends on the starting and stopping of particular services deployed in Azure. The best model achieved was deployed as a service on a virtual machine and was integrated with plant detection mechanism.

2.7.2.1.1 Frontend Implementation

This component comes with the frontend of mobile application.

Environment: Android IDE with Java.

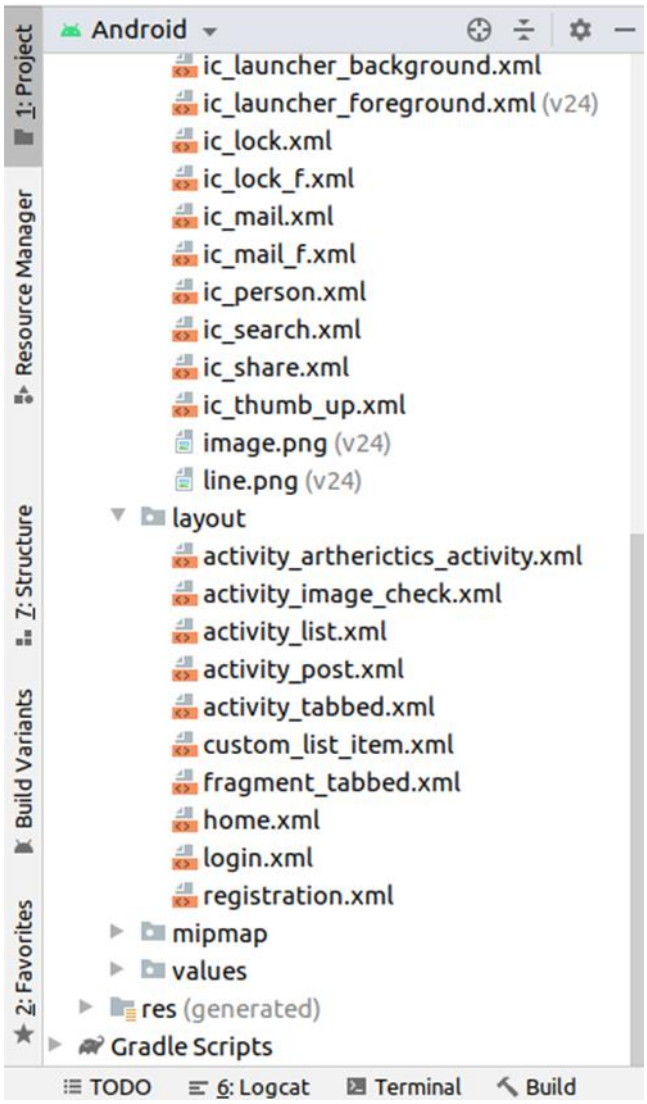


Figure 2.7.2.1.1.1: Frontend Mobile Folder Structure with layout files

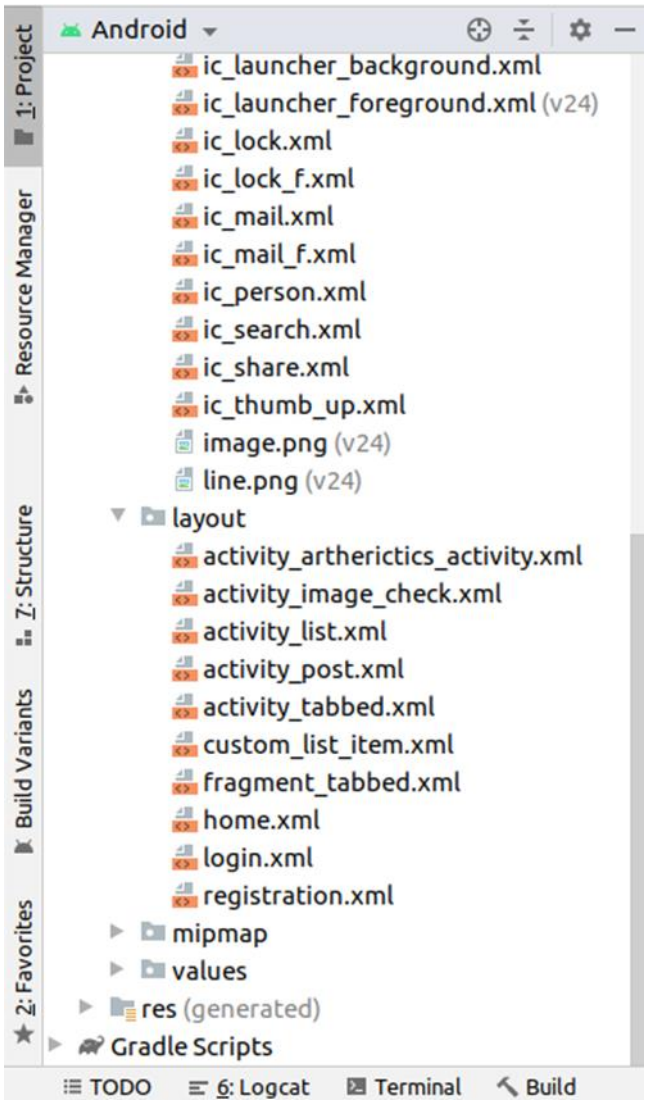


Figure 2.7.2.1.1.2: Frontend Mobile Folder Structure with java files

2.7.2.1.2Backend Implementation

Database Structure

Firebase has been used as the database to store user authentication details (login and registration with account creation) and classified herbal plants details.

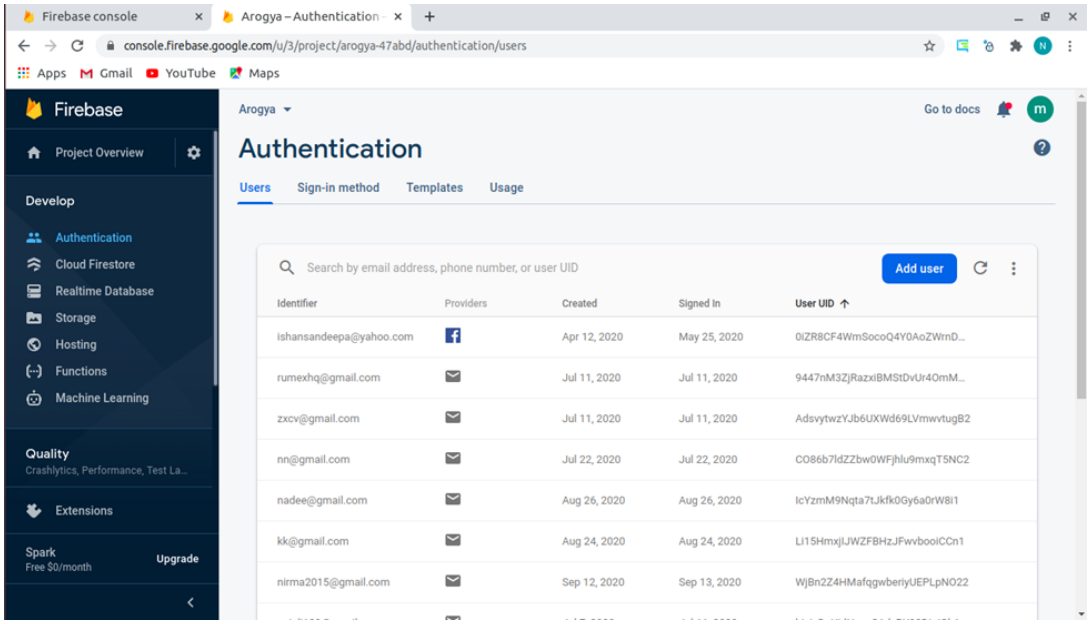


Figure 2.7.2.1.2.1: Firebase database for user authentication and account creation

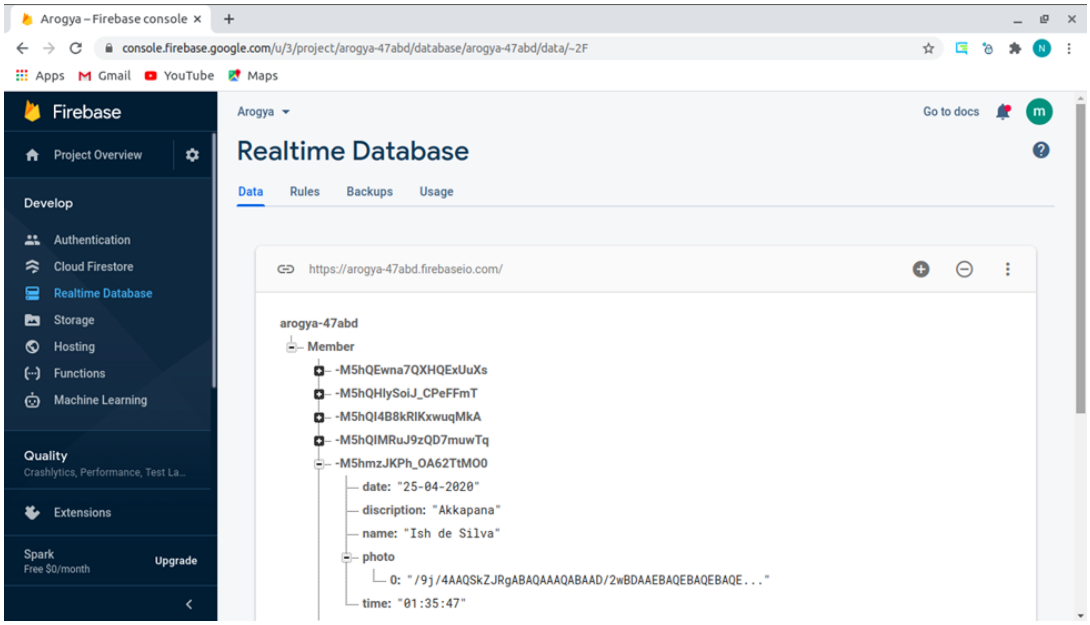


Figure 2.7.2.1.2.2: Firebase database for uploading posts related to ayurvedic leaves

Cloud Services Structure

Azure has been used as the cloud service for the hosting and deployment of services related with herbal plant classification.

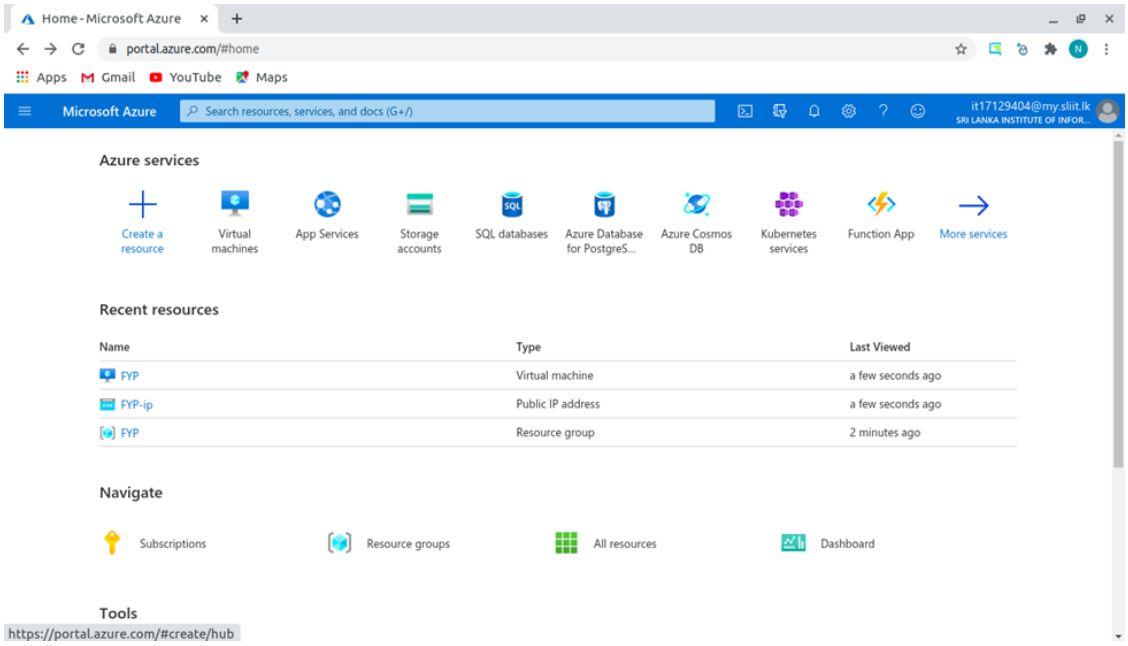


Figure 2.7.2.1.2.3: Azure cloud resources allocated for services

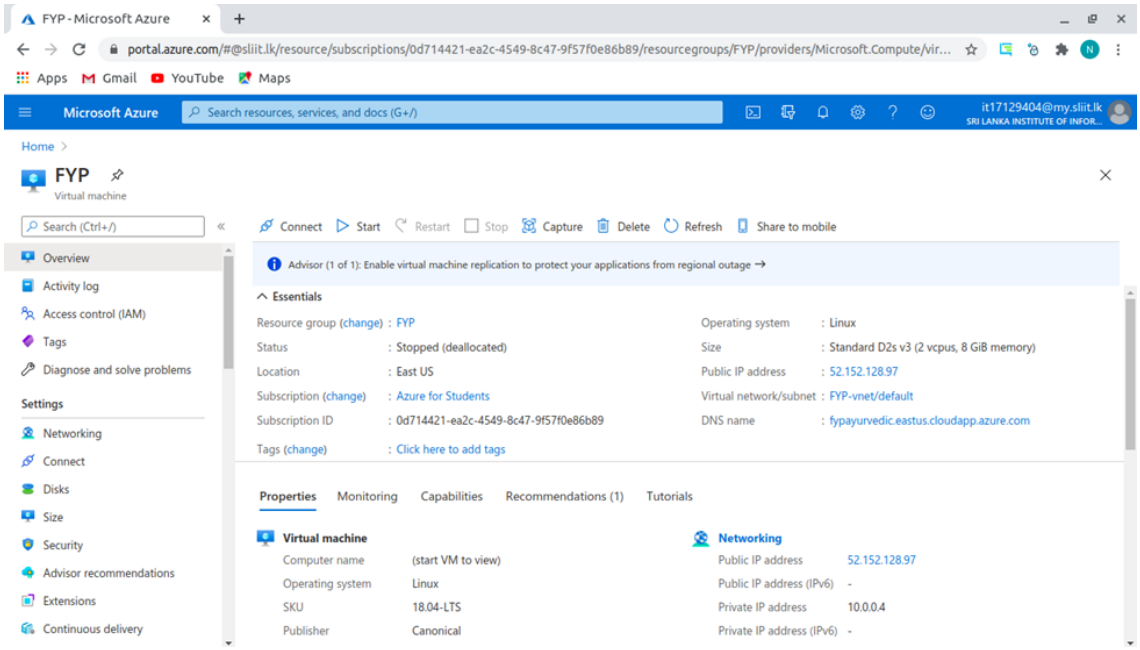


Figure 2.7.2.1.2.4: Azure virtual machine created for deployment of plant detection and classification

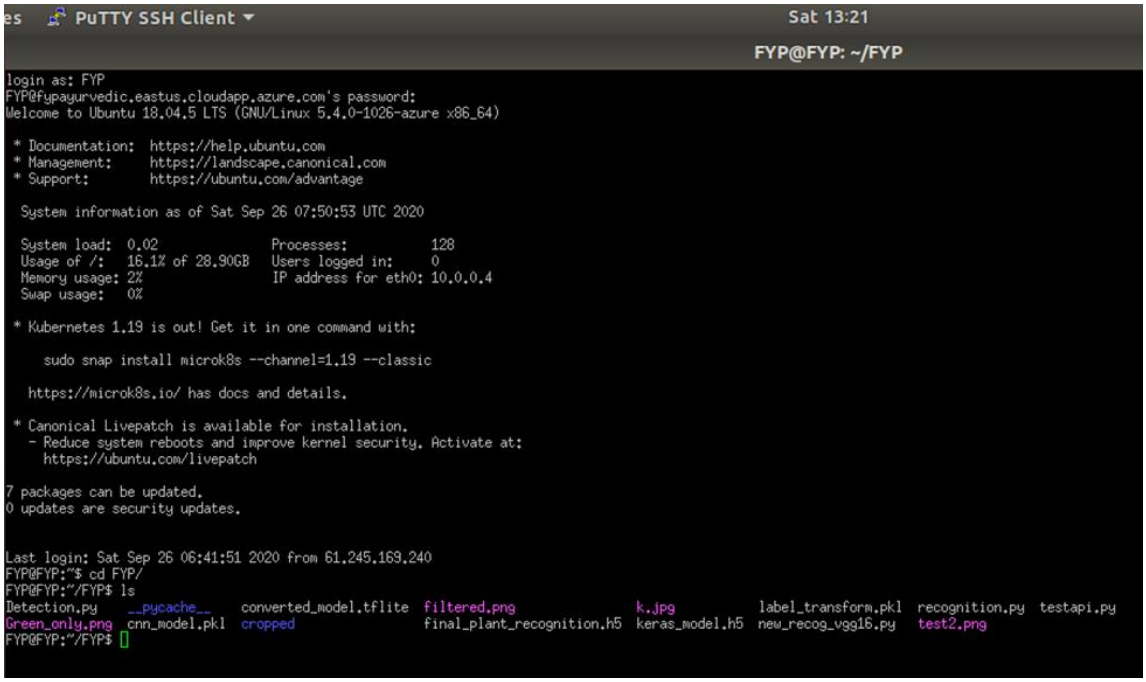


Figure 2.7.2.1.2.5: Model with highest accuracy hosted and deployed in the virtual machine

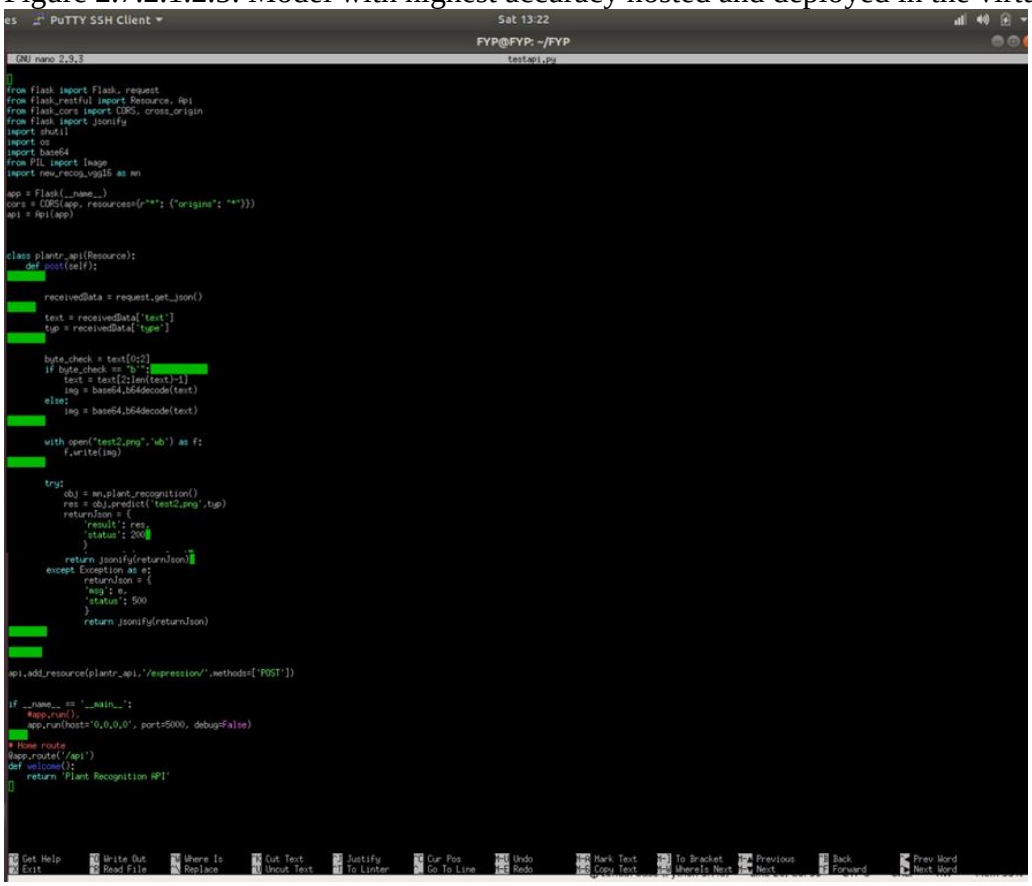


Figure 2.7.2.1.2.6: Code snippet with integrated models in the virtual machine

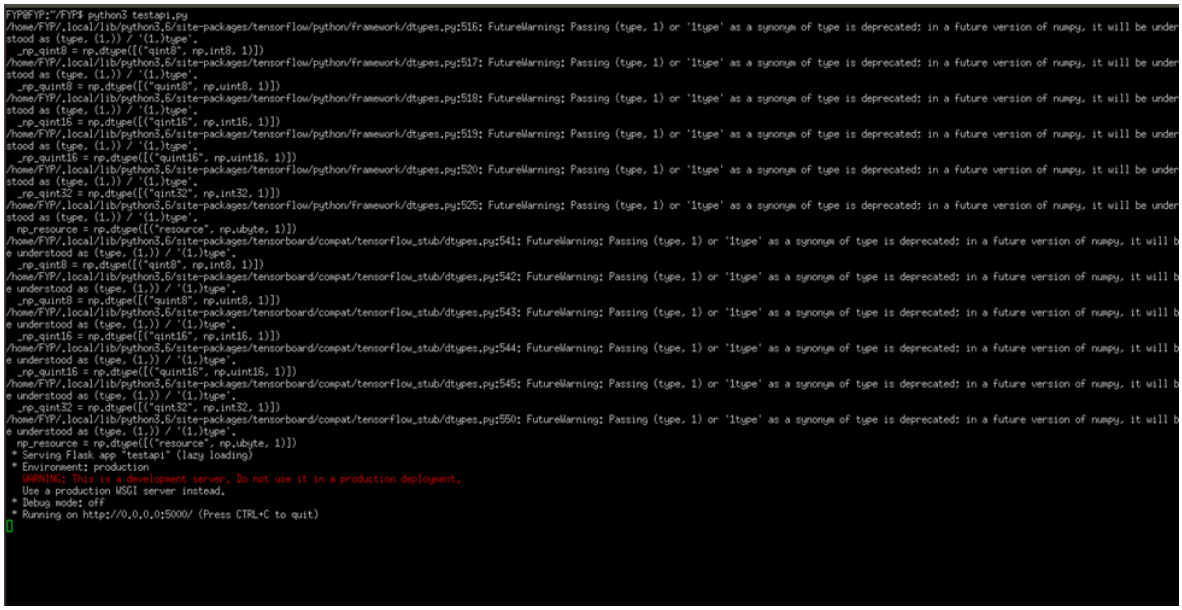


Figure 2.7.2.1.2.7: Backend running as a service in the virtual machine

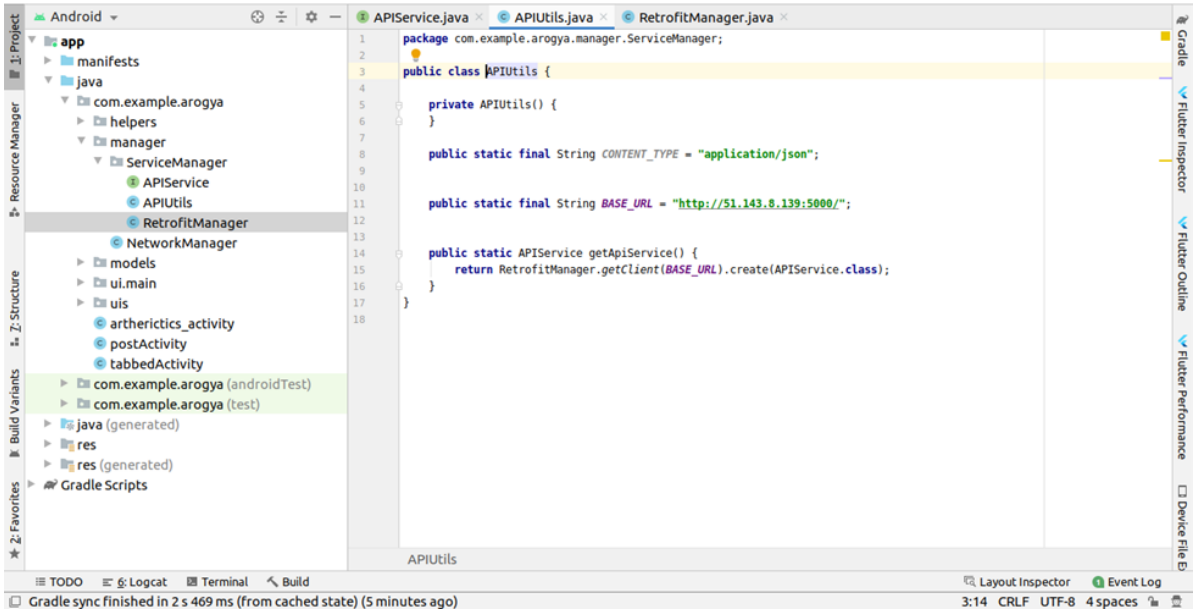


Figure 2.7.2.1.2.8: Code snippet to connect frontend mobile client with backend API in cloud

2.7.2.2 Code Implementations for the research part

Used Technologies

- Python, Keras, Tensorflow, TensorFlow-Backend installed Anaconda Navigator
- **IDE:** Jupiter Notebook and Google Colab

The code implementation part of the specific research component starts with the application of transfer learning based on deep Convolutional Neural Networks (CNN), to the collected and annotated dataset from scratch, for processing them using TensorFlow.

Throughout this research, the variants of deep Convolutional Neural Networks used to compare the initial training and testing accuracies included:

- InceptionV3
- MobileNetV2
- InceptionResNetV2
- Xception
- DenseNet121
- ResNet50
- VGG16

I. InceptionV3

1. Created the base model from the InceptionV3 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.keras.applications.inception_v3 import InceptionV3

conv_base=InceptionV3(weights='imagenet',include_top=False,input_shape=(100,100,3))
```

Figure 2.8.1.3.1: Code snippet to get InceptionV3 base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.2: Code snippet to freeze InceptionV3 base model

3. Added a global_average_pooling2d layer, which reduced the tendency of overfitting.
4. Added a dropout layer, with a rate of 0.5 for regularization purpose.
5. Added a dense layer for 6-class classification, with the activation function as ‘softmax’.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, MaxPooling2D, BatchNormalization, Glob
alAveragePooling2D, Concatenate

from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.3: Code snippet to add extra layers in InceptionV3 base

6. Compiled the model before training it - here, ‘categorical_crossentropy’ was used as the loss function, ‘SGD’ as the optimizer, (1e-4) as the learning rate, 0.9 as the momentum and ‘categorical_accuracy’ as metrics.

```
from keras import optimizers, losses, activations, models

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
              metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.4: Code snippet to compile InceptionV3 newly created model

7. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.5: Code snippet to load and generate train and test datasets in InceptionV3

8. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 50, and the number of steps per epoch testing as 25, including a callback.

```
from tensorflow.python.keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBo
ard

checkpoint = ModelCheckpoint(filepath = 'best_InceptionV3.hdf5', monitor='val_loss', save_best_only=True, mode='auto')

early = EarlyStopping(monitor="val_loss", mode="auto", patience=3)

callbacks_list = [checkpoint, early] #early

fit_history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25,
      callbacks=callbacks_list)
```

Figure 2.8.1.3.6: Code snippet to train InceptionV3

II. MobileNetV2

1. Created the base model from the MobileNetV2 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2

conv_base=MobileNetV2(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

Figure 2.8.1.3.7: Code snippet to get MobileNetV2 base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.8: Code snippet to freeze MobileNetV2 base model

3. Added a conv2d layer, with filters=32, kernel_size= (3,3) and activation function as ‘relu’.
4. Added a dropout layer, with a rate of 0.2 for regularization purpose.
5. Added a global_average_pooling2d layer, which reduced the tendency of overfitting.
6. Added a dense layer for 6-class classification, with the activation function as ‘softmax’.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, MaxPooling2D, BatchNormalization, Glob
alAveragePooling2D, Concatenate

from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)

model.add(layers.Conv2D(32, 3, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.GlobalAveragePooling2D())

model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.9: Code snippet to add extra layers in MobileNetV2 base

7. Compiled the model before training it - here, ‘categorical_crossentropy’ was used as the loss function, (1e-5) as the learning rate, ‘Adam’ as the optimizer and ‘categorical_accuracy’ as metrics.

```
from tensorflow.keras import optimizers

Adam=optimizers.Adam(1e-5)

model.compile(loss='categorical_crossentropy',
              optimizer=Adam,
              metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.10: Code snippet to compile MobileNetV2 newly created model

8. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
·····
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.11: Code snippet to load and generate train and test datasets in MobileNetV2

9. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 50, and the number of steps per epoch testing as 25.

```
fit_history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25)
```

Figure 2.8.1.3.12: Code snippet to train MobileNetV2

III. InceptionResNetV2

1. Created the base model from the InceptionResNetV2 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2

conv_base=InceptionResNetV2(weights='imagenet',include_top=False,input_shape=(100,100,3))
```

Figure 2.8.1.3.13: Code snippet to get InceptionResNetV2 base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.14: Code snippet to freeze InceptionResNetV2 base model

3. Added a flatten layer, which did not affect the batch size.
4. Added a dropout layer, with a rate of 0.5 for regularization purpose.
5. Added a dense layer, with 1x512, with the activation function as ‘relu’.
6. Added a dropout layer, with a rate of 0.5 for regularization purpose.
7. Added a dense layer, with 1x256, with the activation function as ‘relu’ (adding of dense layers was decided by considering the number of classes)
8. Added a dense layer for 6-class classification, with the activation function as ‘softmax’.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, MaxPooling2D, BatchNormalization, Glob
alAveragePooling2D, Concatenate

from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.15: Code snippet to add extra layers in InceptionResNetV2

9. Compiled the model before training it - here, ‘categorical_crossentropy’ was used as the loss function, ‘Adam’ as the optimizer, (1e-3) as the learning rate and ‘categorical_accuracy’ as metrics.

```
from keras import optimizers, losses, activations, models

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(1e-3),
              metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.16: Code snippet to compile InceptionResNetV2 newly created model

10. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.17: Code snippet to load and generate train and test datasets in InceptionResNetV2

11. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 50, and the number of steps per epoch testing as 25, including a callback.

```
from tensorflow.python.keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBo
ard

checkpoint = ModelCheckpoint(filepath = 'best_InceptionResNetV2.hdf5', monitor='val_loss', save_best_only=True, mode='auto')

early = EarlyStopping(monitor="val_loss", mode="auto", patience=3)

callbacks_list = [checkpoint, early] #early

fit_history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25,
      callbacks=callbacks_list)
```

Figure 2.8.1.3.18: Code snippet to train InceptionResNetV2

IV. Xception

1. Created the base model from the Xception model, which is pretrained on the ImageNet dataset.

```
from tensorflow.keras.applications.xception import Xception

conv_base=Xception(weights='imagenet',include_top=False,input_shape=(100,100,3))
```

Figure 2.8.1.3.19: Code snippet to get Xception base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.20: Code snippet to freeze Xception base model

3. Added a global_average_pooling2d layer, which reduced the tendency of overfitting.
4. Added a dense layer, with 1x512, with the activation function as ‘relu’ (adding of dense layers was decided by considering the number of classes)
5. Added a dense layer for 6-class classification, with the activation function as ‘softmax’.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, MaxPooling2D, BatchNormalization, Glob
alAveragePooling2D, Concatenate

from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.21: Code snippet to add extra layers in Xception base model

6. Compiled the model before training it - here, ‘categorical_crossentropy’ was used as the loss function, ‘Nadam’ as the optimizer and ‘categorical_accuracy’ as metrics.

```
from keras import optimizers, losses, activations, models

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Nadam(),
              metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.22: Code snippet to compile Xception newly created model

7. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.23: Code snippet to load and generate train and test datasets in Xception

8. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 10, and the number of steps per epoch testing as 25, including a callback.

```
from tensorflow.python.keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBo
ard

checkpoint = ModelCheckpoint(filepath = 'best_Xception.hdf5', monitor='val_loss', save_best_only=True, mode='auto')

early = EarlyStopping(monitor="val_loss", mode="auto", patience=5)

callbacks_list = [checkpoint, early] #early

fit_history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25,
      callbacks=callbacks_list)
```

Figure 2.8.1.3.24: Code snippet to train Xception

V. DenseNet121

1. Created the base model from the DenseNet121 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.python.keras.applications.densenet import DenseNet121

conv_base=DenseNet121(weights='imagenet',include_top=False,input_shape=(100,100,3))
```

Figure 2.8.1.3.25: Code snippet to get DenseNet121 base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.26: Code snippet to freeze DenseNet121 base model

3. Added a global_average_pooling2d layer, which reduced the tendency of overfitting.
4. Added a dense layer for 6-class classification, with the activation function as ‘softmax’.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, MaxPooling2D, BatchNormalization, Glob
alAveragePooling2D, Concatenate

from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.27: Code snippet to add extra layers in DenseNet121 base model

5. Compiled the model before training it - here, ‘categorical_crossentropy’ was used as the loss function, ‘Adam’ as the optimizer and ‘categorical_accuracy’ as metrics.

```
from tensorflow.keras import optimizers

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.28: Code snippet to compile DenseNet121 newly created model

6. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.29: Code snippet to load and generate train and test datasets in DenseNet121

7. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 50, and the number of steps per epoch testing as 25.

```
from tensorflow.python.keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBo
ard

fit_history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=10,
      validation_data=validation_generator,
      validation_steps=25)
```

Figure 2.8.1.3.30: Code snippet to train DenseNet121

VI. ResNet50

1. Created the base model from the ResNet50 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.keras.applications import ResNet50

# Define transfer learning network model

conv_base = ResNet50(include_top = False, pooling = RESNET50_POOLING_AVERAGE, weights='imagenet', input_shape=(100,100,3))
```

Figure 2.8.1.3.31: Code snippet to get ResNet50 base model

2. Extracted the features through freezing the convolutional base, while making conv5_block1_1_conv layer unfreezed.

```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'conv5_block1_1_conv':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

Figure 2.8.1.3.32: Code snippet to freeze and unfreeze ResNet50 base model

3. Added a dense layer for 6-class classification, with the activation function as 'softmax'.

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(NUM_CLASSES, activation = DENSE_LAYER_ACTIVATION))
```

Figure 2.8.1.3.33: Code snippet to add extra layers in ResNet50 base model

4. Compiled the model before training it - here, 'accuracy' was used as the loss function, 0.01 as the learning rate, (1e-6) as the decay, 0.9 as the momentum, 'SGD' as the optimizer and 'categorical_crossentropy' as metrics.

```
# Compile transfer learning model

from tensorflow.keras import optimizers

sgd = optimizers.SGD(lr = 0.01, decay = 1e-6, momentum = 0.9, nesterov = True)
model.compile(optimizer = sgd, loss = OBJECTIVE_FUNCTION, metrics = LOSS_METRICS)
```

Figure 2.8.1.3.34: Code snippet to compile ResNet50 newly created model

5. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.35: Code snippet to load and generate train and test datasets in ResNet50

6. Finally, the dataset was trained for 10 epochs, with using the number of steps per epoch training as 50, and the number of steps per epoch testing as 25.

```
from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = EARLY_STOP_PATIENCE)
cb_checkpointer = ModelCheckpoint(filepath = 'best_Res50.hdf5', monitor = 'val_loss', save_best_only = True, mode = 'auto')
```

```
fit_history = model.fit_generator(
        train_generator, steps_per_epoch=STEPS_PER_EPOCH_TRAINING, epochs = NUM_EPOCHS,
        validation_data=validation_generator, validation_steps=STEPS_PER_EPOCH_VALIDATION,
        callbacks=[cb_checkpointer, cb_early_stopper])
```

Figure 2.8.1.3.36: Code snippet to train ResNet50

VII. VGG16

1. Created the base model from the VGG16 model, which is pretrained on the ImageNet dataset.

```
from tensorflow.python.keras.applications import VGG16

conv_base=VGG16(weights='imagenet',include_top=False,input_shape=(100,100,3))
```

Figure 2.8.1.3.37: Code snippet to get VGG16 base model

2. Extracted the features through freezing the convolutional base.

```
conv_base.trainable = False
```

Figure 2.8.1.3.38: Code snippet to freeze VGG16 base model

3. Added a flatten layer, which did not affect the batch size.
4. Added a dense layer, with 1x256, with the activation function as 'relu' (adding of dense layers was decided by considering the number of classes)
5. Added a dense layer for 6-class classification, with the activation function as 'softmax'.

```
from tensorflow.python.keras import models
from tensorflow.python.keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(6, activation='softmax'))
```

Figure 2.8.1.3.39: Code snippet to add extra layers in VGG16 base model

6. Compiled the model before training it - here, 'categorical_crossentropy' was used as the loss function, 'RMSprop' as the optimizer, (1e-4) as the learning rate and 'categorical_accuracy' as metrics.

```
from tensorflow.python.keras import optimizers

model.compile(loss='categorical_crossentropy',optimizer=optimizers.RMSpropfruit_image_classification_using vgg16_transfe(lr=1e-4),metrics=['categorical_accuracy'])
```

Figure 2.8.1.3.40: Code snippet to compile VGG16 newly created model

7. Training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.
apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.
googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

```
train_dir='/content/drive/My Drive/plant/train'
```

```
validation_dir='/content/drive/My Drive/plant/val'
```

```
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import preprocess_input

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32)

validation_generator = test_datagen.flow_from_directory(
        # This is the target directory
        validation_dir,
        # All images will be resized to 100x100
        target_size=(100, 100),
        batch_size=32,
        shuffle=False)
```

Figure 2.8.1.3.41: Code snippet to load and generate train and test datasets in VGG16

8. Finally, the dataset was trained for 100 epochs, with using the number of steps per epoch training as 100, and the number of steps per epoch testing as 50.

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=100,
      validation_data=validation_generator,
      validation_steps=50)
```

Figure 2.8.1.3.42: Code snippet to train VGG16

After training for different number of epochs, number of steps per epoch training and number of steps per epoch testing; the testing accuracy was calculated for each of the above-mentioned CNN variant using the below equation:

```
steps_test=212/32
```

```
print(steps_test)
```

```
6.625
```

```
result = model.evaluate_generator(validation_generator, steps=steps_test)
print("Test-set classification accuracy: {0:.2%}".format(result[1]))
```

Figure 2.8.1.3.43: Code snippet to calculate testing accuracy

In addition, for collecting the history returned from training each CNN model and for easy reference of the performance from each, a plot of accuracy on the training and validation datasets over training epochs, as well as a plot of loss on the training and validation datasets over training epochs have been visualized and plotted with the following code.

```
import matplotlib.pyplot as plt
```

```
acc=history.history['categorical_accuracy']
val_acc=history.history['val_categorical_accuracy']
loss=history.history['loss']
val_loss=history.history['val_loss']
epochs=range(1,len(acc)+1)
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.plot(epochs, acc,  label='Training acc')
plt.plot(epochs, val_acc, label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, label='Training loss')
plt.plot(epochs, val_loss, label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Figure 2.8.1.3.44: Code snippet to generate accuracy and loss graphs against epochs trained

2.7.2.2 Testing

Testing plays a major role in this research. The research is related to the health sector, due to the interconnectivity with Ayurveda. So, the implemented functions must be accurate since the software deals with the lives of people and affects the ingredients of medical recipes.

The initial process of the mobile application was detecting the constitution based on a photo that incorporates image processing. The image was uploaded for distinguishing the plant part (leaf/fruit/root) needed to be classified. There is a possibility where analysis can lead to inaccurate information if the conditions that affect a photo changes. Therefore, the testing of image processing part should be tested under different conditions of lighting, the angle of capture, brightness, zoom, rotation and the distance between the object and capture device (mobile camera). All these different conditions which varies from each other were obtained through data augmentation techniques. In summary, testing phase is set to check the efficiency, effectiveness and reliability of the system and the final outputs, whether it satisfies the project requirements.

Unit Testing

The purpose of executing unit tests is to make sure, all the independent components work as expected according to the working plan. The system was divided into small components such as sign up (account creation), sign in (log in) and plant classification, and tested in order to get the results. They were compared to see whether the actual results were as similar as the expected results. The system was divided into modules accordingly. Finally, a verification process was run to check whether the system met its specifications as discussed by the team at the beginning.

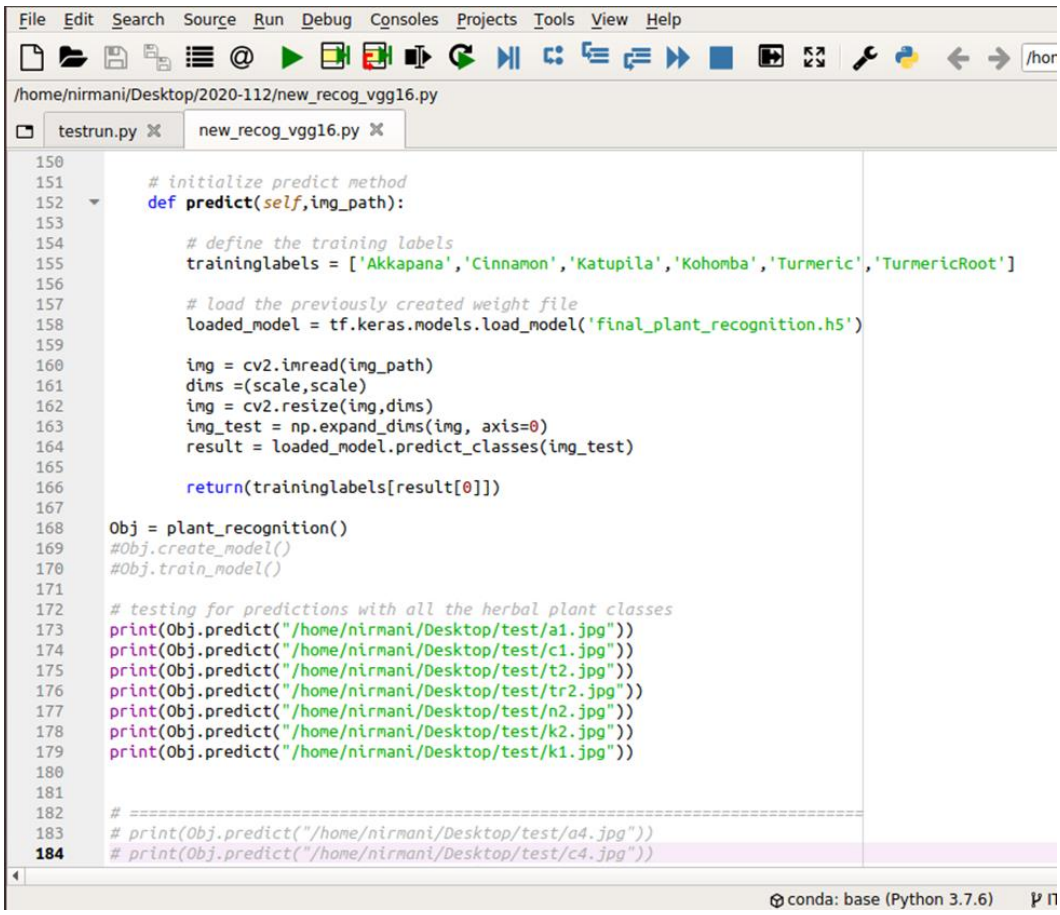


Figure 2.8.2.1: Code snippet of the method for unit testing of plant classification functionality

Integration Testing

Individual components such as system authentication, object detection and plant classification functionalities were combined and tested into a single module to check the quality of the overall product and make sure, there was no other internal conflicts on the combination of whole product. Object detection component was followed by the plant classification component. They were integrated as:

- ✓ First either a plant image captured through the mobile camera or uploaded from gallery was detected successfully whether it was a leaf/root/fruit from the object detection functionality.
- ✓ Then those detected objects were saved to a separate folder as images.
- ✓ After that, the particular folder path was given for the plant classification functionality, so that the input for that functionality was the image of detected object, which was the output from the object detection functionality.
- ✓ Likewise, those 2 main components were integrated and tested successfully.

Finally, the information summarizer came into picture as it produced the detailed summary report on the classified herbal plant. Integration testing was done individually by the group members according to their component scope.

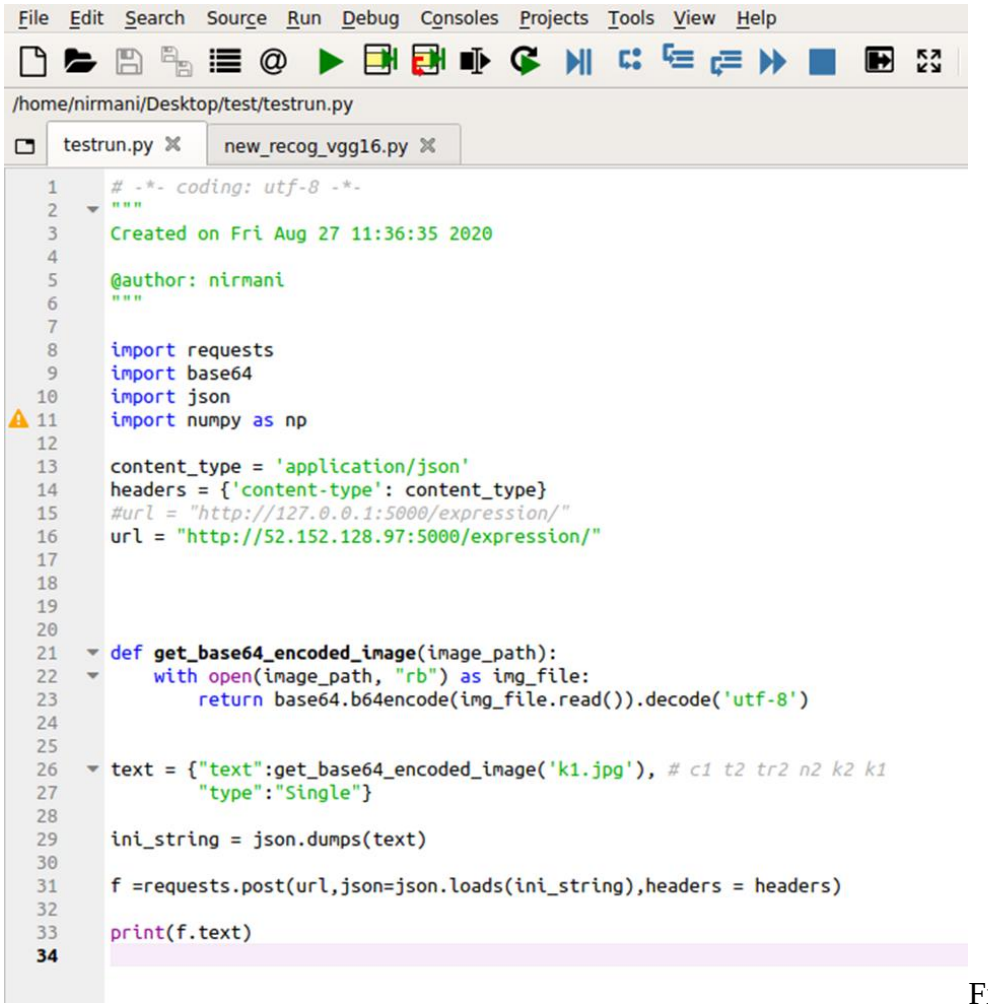


Figure 2.8.2.2: Code snippet of the class for integration testing of plant detection and classification functionalities with the backend API

System Testing

Whole system was tested according to the specification, is said to be a System Integration. This makes sure, the system to be compatible with all the other modules and work as the execution plan without arising any issues. This is more like, black box testing type of testing. If there is any bugs or issues, the overall team has to take the responsibility and relaunch the application with the needed modifications. System testing was done successfully for the whole product by integrating all the API s and bugs were resolved accordingly.

User Acceptance Testing

This is the phase, where the customer interacts with the developed trial product. The product will be tested by the customer to make sure whether it reaches the customer functional requirements. They will be given a demo version of the product, so that, they can test it. If the user accepts the products without making any issues and satisfied, that is the end of the testing life cycle. Otherwise, the team has to do some other modifications accordingly, and the cycle goes on until the user gets satisfied.

Test cases for Arogya mobile application

a. Account creation

Table 2.8.2.1: Test case for account creation with valid data

Test Case ID	TU001
Test Scenario	Check user account creation with valid data
Precondition	User should not have a previously created user account with the same email address.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Tap on "Register Here" 3. View the user registration page. 4. Enter a valid email address. 5. Enter a valid password. 6. Tap of the button "Register".
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$
Expected Results	User should be registered to the system by creating a new account with the given credentials.
Actual Results	User was successfully registered to the system and given a successful message as "User Created Successfully"
Pass/Fail	**Pass**

Table 2.8.2.2: Test case for account creation with invalid data

Test Case ID	TU002
Test Scenario	Check user account creation with invalid data
Precondition	User should not have a previously created user account with the same email address.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Tap on "Register Here" 3. View the user registration page. 4. Enter an invalid email address. 5. Enter a valid password. 6. Tap of the button "Register".
Test Data	Email Address: nirma2015123gmail.com Password: KFD4ert1@#$
Expected Results	User should be given an error message with real time validation as "enter a valid email address", and registration should be unsuccessful.
Actual Results	User was given an error message with real time validation as "enter a valid email address", and registration was unsuccessful.
Pass/Fail	**Pass**

b. Login to the application

Table 2.8.2.3: Test case for user login with valid data

Test Case ID	TU003
Test Scenario	Check user login with valid data
Precondition	User should have an account.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter the correct password. 4. Tap of the button “Login”.
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$
Expected Results	User should be able to login to the system successfully and redirected to the home page.
Actual Results	User was given a successful message as “Login Successful” and was redirected to the home page
Pass/Fail	**Pass**

Table 2.8.2.4: Test case for user login with invalid data

Test Case ID	TU004
Test Scenario	Check user login with invalid data
Precondition	User should have an account.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter an incorrect password. 4. Tap of the button “Login”.
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1$
Expected Results	User should be given an error message as “incorrect email address or password”, and login should be unsuccessful.
Actual Results	User was given an error message as “incorrect email address or password”, and login was unsuccessful.
Pass/Fail	**Pass**

c. Plant classification

Table 2.8.2.5: Test case for accurate plant classification with a camera-captured image

Test Case ID	TU005
Test Scenario	Check the plant classification function by a camera captured image
Precondition	User should have an account.

Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter the correct password. 4. Tap of the button “Login”. 5. View the home page 6. Tap on the “camera” icon in the toolbar 7. Capture an image of an Akkapana leaf 8. Tap on the button “Classify the herb”.
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$ Input image: a camera captured Akkapana leaf
Expected Results	The name of the classified plant should be displayed as “Akkapana”, and a detailed description of the particular species should be visualized to the user.
Actual Results	The name of the classified plant was displayed as “Akkapana”, and a detailed description of the particular species was visualized to the user.
Pass/Fail	**Pass**

Table 2.8.2.6: Test case for inaccurate plant classification with a camera-captured image

Test Case ID	TU006
Test Scenario	Check the plant classification function by a camera captured image
Precondition	User should have an account.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter the correct password. 4. Tap of the button “Login”. 5. View the home page 6. Tap on the “camera” icon in the toolbar 7. Capture an image of a Katupila leaf 8. Tap on the button “Classify the herb”.

Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$ Input image: a camera captured Katupila leaf
Expected Results	The name of the classified plant should be displayed as “Katupila”, and a detailed description of the particular species should be visualized to the user.
Actual Results	The name of the classified plant was displayed as “Cinnamon”, and a detailed description of the particular species was visualized to the user.
Pass/Fail	**Fail**

Table 2.8.2.7: Test case for accurate plant classification with an image uploaded from gallery

Test Case ID	TU007
Test Scenario	Check the plant classification function by an uploaded image from phone gallery.
Precondition	User should have an account.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter the correct password. 4. Tap of the button “Login”. 5. View the home page 6. Tap on the “from gallery” icon in the toolbar 7. Upload an image of a turmeric leaf 8. Tap on the button “Classify the herb”.
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$ Input image: a camera captured turmeric digested root
Expected Results	The name of the classified plant should be displayed as “Turmeric Root”, and a detailed description of the particular species should be visualized to the user.
Actual Results	The name of the classified plant was displayed as “Turmeric Root”, and a detailed description of the particular species was visualized to the user.
Pass/Fail	**Pass**

Table 2.8.2.8: Test case for inaccurate plant classification with an image uploaded from gallery

Test Case ID	TU008

Test Scenario	Check the plant classification function by an uploaded image from phone gallery.
Precondition	User should have an account.
Assumption	User has installed the app in his android mobile device and has internet connectivity
Test Steps	1. Tap on the Arogya icon and open the mobile app to view the login page. 2. Enter the correct email address. 3. Enter the correct password. 4. Tap of the button “Login”. 5. View the home page 6. Tap on the “from gallery” icon in the toolbar 7. Upload an image of a turmeric leaf 8. Tap on the button “Classify the herb”.
Test Data	Email Address: nirma2015@gmail.com Password: KFD4ert1@#$ Input image: a camera captured turmeric leaf
Expected Results	The name of the classified plant should be displayed as “Turmeric”, and a detailed description of the particular species should be visualized to the user.
Actual Results	The name of the classified plant was displayed as “Cinnamon”, and a detailed description of the particular species was visualized to the user.
Pass/Fail	**Fail**

Tools and Technologies

1. Python
2. Keras
3. TensorFlow
4. TensorFlow Lite
5. Jupiter Notebook
6. Google Colab
7. Android
8. Android Studio
9. Firebase – for the database
10. Azure – for backend API deployment

Research Area – Deep Learning Algorithms (deep neural network architectures based on transfer learning) and Image Processing Techniques

2.7.3. Abstractive web page Information Summarization and Location Mapping with GIS technology on Ayurvedic Plants in Sri Lanka Using a Mobile Application in an Online Environmental Approach

2.7.3.1. Implementation

The client side runs as a mobile application which was built on top of Android and Flutter (Dart). The User Login and Registration along with Profile Management and Geographical Location Mapping run on Mobile client itself while the Summary Generation component runs on the web server. Front-end technologies are based on HTML, CSS. Apart from this Bootstrap has been used in order to create responsive webpages. Server side runs on Flask server, which is a python-based framework and the implementation continues with Python, TensorFlow environment and using Anaconda navigator. Front-end Flutter web view call the web API to run the Summary generator process, which run on flask server. SpaCy, NLTK, Sumy, and Gensim are natural language processing python libraries which are related to summary generation process.

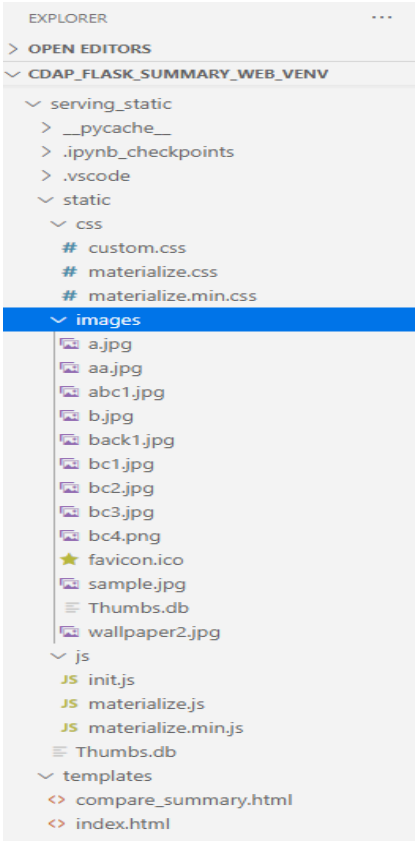


Figure 2.7.3 1: Frontend Website Folder Structure

2.7.3.1.1. Frontend Implementation

This component comes with two versions of Frontends

1.0. Web Application

2.0. Mobile Application

Web Application

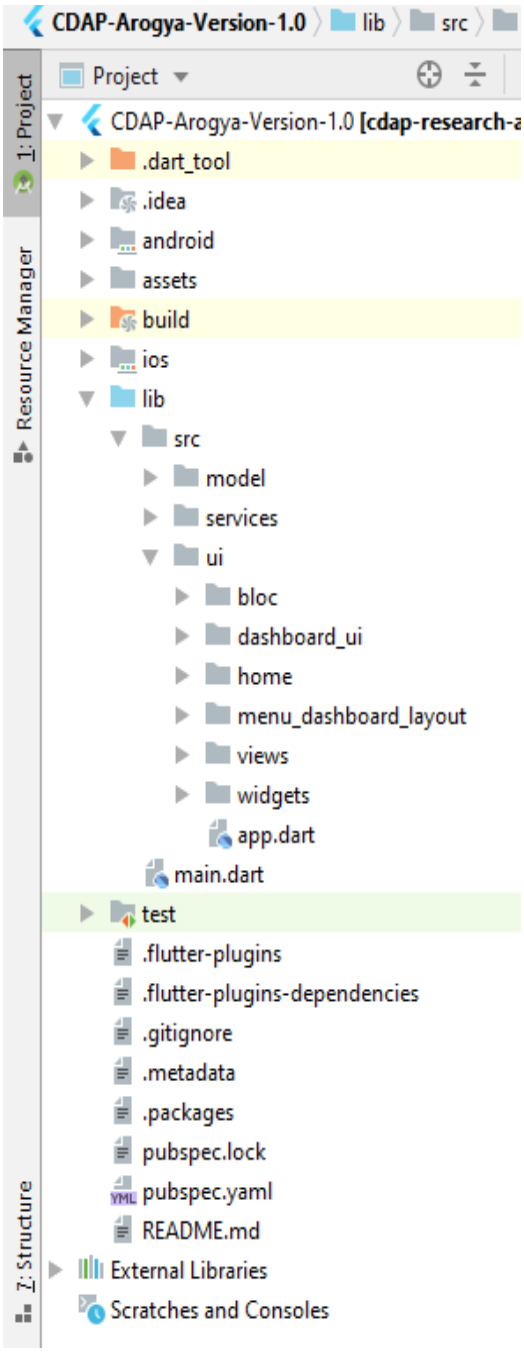
Environment: VS Code IDE

API end point of the Web Application is called by the Mobile Web View component.

Figure in the left side displays the folder structure of the web application.

compare_summary.html and **index.html** are the main web application templates.

Mobile Application



Environment: Android IDE with Flutter (Dart Language)

Left side figure shows the Directory structure of the Android project

Figure 2.7.3 2: Frontend Mobile Folder Structure

pubspec.yaml

Flutter commands

```yaml
name: flutter_arogya_plant_app
description: Flutter Herbal Plant App
version: 1.0.0+1

environment:
  sdk: ">=2.1.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  splashscreen: ^1.2.0
  google_fonts: ^1.1.0
  cupertino_icons: ^0.1.2
  dio: ^2.1.13
  json_serializable: ^3.2.3
  intl: ^0.16.0
  url_launcher: ^5.2.0
  flutter_webview_plugin: 0.3.0+2
  flutter_bloc: ^2.1.1
  cached_network_image: ^2.0.0-rc
  auto_size_text: ^2.0.2
  flutter_spinkit: ^3.1.0
  flutter_auth_buttons: ^0.5.0
  flutter_stetho: ^0.5.2
  cloud_firestore: ^0.12.6
  firebase_auth: ^0.11.1+10
  google_sign_in: ^4.0.4
  google_maps_flutter: 0.0.3+3
  geolocator: ^5.0.0
  flutter_map:
  geocoder:
  location: ^2.3.5
  fluttertoast: ^7.0.1

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_launcher_icons: ^0.7.4

flutter_icons:
  image_path: 'assets/images/pl.png'
  android: true
  ios: true
  build_runner: ^1.7.1

flutter:
  uses-material-design: true

  assets:
    - assets/images/
```

Figure 2.7.3 3: Packages in Pubspec.yaml

Many types of libraries were used in order to build up the application

Pubspec.yaml file indicates the all libraries which were used

2.7.3.1.2. Backend Implementation

Database Structure

Firebase has been used as the Database to store User Profile Information, Herbal Plants and Geographical Locations details.

As shown in the following figure, **userData** is the collection which stores all the records related to a particular user on a time session.

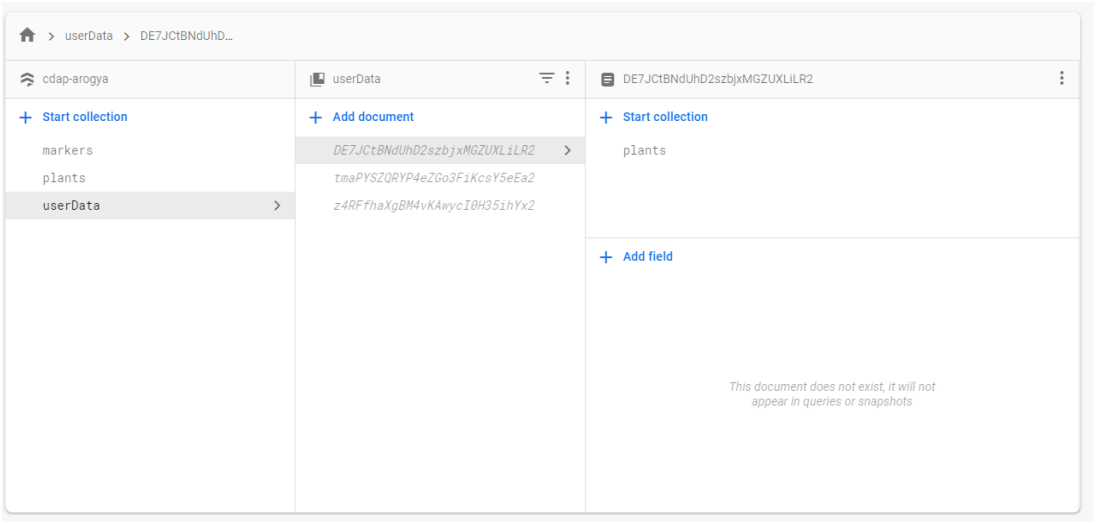


Figure 2.7.3 4: User Data

As shown in the following figure, **markers** is the collection which stores all the records related to a particular Geographical Location.

Each document consists with three main fields:

- clientName
- location
- plantName

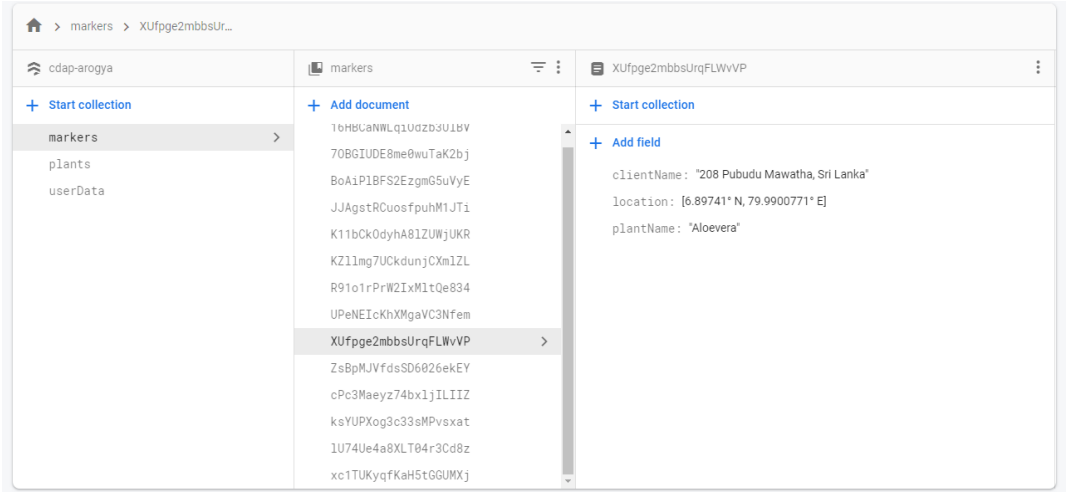


Figure 2.7.3 5: Geographical Location Details

Following figure indicates the plants collection which is related to the information of the herbal plants.

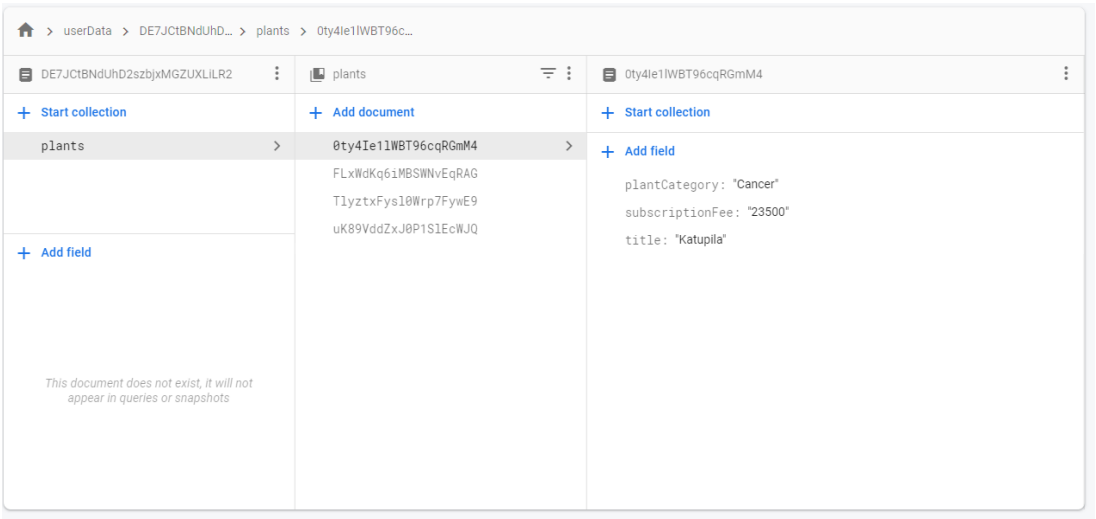


Figure 2.7.3 6: Herbal Plant Details

Multi-Level user authentication is also implemented in this system. Such as, Google Sign In, Email User Sign In, Anonymous Sign In.

Server-Side Implementation

Used Technologies:

- Python, TensorFlow-Backend installed Anaconda Navigator
- Flask web application Framework
- IDE: Visual Studio Code

Following code snippet indicates the imported libraries to the project, you can check the comments to get an idea about its contribution.

app.ipynb

```
[-]
#When using this, all string literals become unicode literals
from __future__ import unicode_literals

#Flask is a light weight WSGI WEB APPLICATION FRAMEWORK
from flask import Flask,render_template,url_for,request

#My Created files for summarization:start
from spacy_summarization import text_summarizer
from nltk_summarization import nltk_summarizer
from sumy_summarization import sumy_summary
#My Created files for summarization:end

#This module summarizes the given text by extracting the important
#points
from gensim.summarization import summarize

#This is stateful, keeps track of the current figure and plotting area
#And the plotting functions are directed to the current axes
import matplotlib.pyplot as plt

#Python based data analysis toolkit
import pandas as pd
#This is using for opening the URL's
import urllib
#This is for Regular Expression syntax
import re
#Allow tosend HTTP/1.1 requests
import requests

#Using Fake user-agent to send a string to a website on each visit by the
#browser or the application
from fake_useragent import UserAgent
#Returns the number of seconds passed
import time

#A platform for managing human language data built on Python language
import nltk

#This tokenizer can divide the text into list of sentences
#Unsupervised Algorithm is used to build the model for abbreviations
#need to download at the first time execution only
#nltk.download('punkt')

#Open source software library for advanced natural
#language processing
import spacy

nlp = spacy.load('en_core_web_sm')
app = Flask(__name__)

# Web Scraping Pkg
from bs4 import BeautifulSoup

#Old way of importing
#from urllib import urlopen

#Updated way of importing
from urllib.request import urlopen
```

nltk_summarization.py

serving_static > nltk_summarization.py

```python
import nltk

#Corpus is a large structured set of texts
#can be used to access the corpora in the NLTK data pakage
from nltk.corpus import stopwords

#Tokenization is a way to split text into tokens
#as paragraphs,sentences, words
from nltk.tokenize import word_tokenize, sent_tokenize

#Heap queue algorithm
#It's a special tree structure, each parent node is less than or equal to its child node
import heapq

```

spacy_summarization.py

serving_static > spacy_summarization.py > text_summarizer

```python
# NLP Packages
import spacy
nlp = spacy.load('en_core_web_sm')
# Packages for Normalizing Text
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
# Import Heapq for Finding the Top N Sentences
from heapq import nlargest

```

sumy_summarization.py

serving_static > sumy_summarization.py > ...

```python
#Module for automatic summarization of text documents and HTML pages
from sumy.parsers.plaintext import PlaintextParser
#Tokenizer
from sumy.nlp.tokenizers import Tokenizer
#LexRank is an unsupervised approach to text summarization based
#on graph-based centrality scoring of sentences. The main idea is
#that sentences “recommend” other similar sentences to the reader
from sumy.summarizers.lex_rank import LexRankSummarizer

```

Figure 2.7.3 7: Imported packages

The following code snippet indicates the Reading Time method and how to fetch the text from URL with the help of Beautiful Soup library

```
# Reading Time
def readingTime(mytext):
    total_words = len([ token.text for token in nlp(mytext)])
    estimatedTime = total_words/200.0
    return estimatedTime

# Fetch Text From Url
def get_text(url):
    page = urlopen(url)
    soup = BeautifulSoup(page,'lxml')
    fetched_text = ' '.join(map(lambda p:p.text,soup.find_all('p')))
    return fetched_text
```

Figure 2.7.3 8: Reading Time & Fetch Text

The following code snippet indicates the main Summary generate function for copy & pasted text

```
@app.route('/analyze',methods=['GET','POST'])
def analyze():
    start = time.time()
    if request.method == 'POST':
        rawtext = request.form['rawtext']
        final_reading_time = readingTime(rawtext)
        final_summary = text_summarizer(rawtext)
        summary_reading_time = readingTime(final_summary)
        end = time.time()
        final_time = end-start
        text_word_count = []
        summary_word_count = []

        for i in rawtext:
            text_word_count.append(len(i.split()))

        for i in final_summary:
            summary_word_count.append(len(i.split()))

        b=len(text_word_count)
        print(b)

        a=len(summary_word_count)
        print(a)

        df = pd.DataFrame({
            "Charactor count in Text and Summary": ["Text", "Summary"],
            "Count":[b,a]})

        df.set_index("Charactor count in Text and Summary",drop=True,inplace=True)
        df.plot.bar()
        plt.show()

    return render_template('index.html',ctext=rawtext,final_summary=final_summary,final_time=final_time,
final_reading_time=final_reading_time,summary_reading_time=summary_reading_time)
```

Figure 2.7.3 9: Summary Generation Function for Custom Text

The following code snippet indicates the main Summary generate function for the extracted text from a single URL

```
@app.route('/analyze_url',methods=['GET','POST'])
def analyze_url():
    start = time.time()
    if request.method == 'POST':
        raw_url = request.form['raw_url']
        rawtext = get_text(raw_url)
        final_reading_time = readingTime(rawtext)
        final_summary = text_summarizer(rawtext)
        summary_reading_time = readingTime(final_summary)
        end = time.time()
        final_time = end-start
        text_word_count = []
        summary_word_count = []

        for i in rawtext:
            text_word_count.append(len(i.split()))

        for i in final_summary:
            summary_word_count.append(len(i.split()))

        b=len(text_word_count)
        print(b)

        a=len(summary_word_count)
        print(a)

        df = pd.DataFrame({
            "Charactor count in Text and Summary": ["Text", "Summary"],
            "Count":[b,a]})

        df.set_index("Charactor count in Text and Summary",drop=True,inplace=True)
        df.plot.bar()
        plt.show()
    return render_template('index.html',ctext=rawtext,final_summary=final_summary,final_time=final_time,
final_reading_time=final_reading_time,summary_reading_time=summary_reading_time)
```

Figure 2.7.3 10: Extract Text from Single URL

The following code snippet indicates the main Summary generate function for the extracted text from a multiple URL

```
@app.route('/analyze_multiple_url',methods=['GET','POST'])
def analyze_multiple_url():
    start = time.time()
    if request.method == 'POST':
        query = request.form['query']
        # print(query)

    # query = "'aloe vera'"
    query = urllib.parse.quote_plus(query) # Format into URL encoding
    number_result = 20

    ua = UserAgent()

    #google_url = "https://www.google.com/search?q=%27trade+war%27&num=20"
    google_url = "https://www.google.com/search?q=" + query + "&num=" + str(number_result)
    #print(google_url)
    response = requests.get(google_url, {"User-Agent": ua.random})
    soup = BeautifulSoup(response.text, "html.parser")
    export = str(soup)
```

Figure 2.7.3 11: Extract Text from Multiple Sites

2.7.3.2. Testing

Testing phase is set to check the efficacy of the system and the final outputs, whether it satisfies the project requirements.

S = Set of words extracted by analyzing the sentences present in each document

There are many parameters against which you can evaluate your summarization system. like,

Precision = Number of important sentences / Total number of sentences summarized

Recall =Total number of important sentences Retrieved /Total number of important sentences present

Unit Testing

The purpose of executing unit testing is to make sure, all the independent components work as expected according to the work plan. The system should be divided into small components and tested in order to get a good quality report. The system can be divided into modules according to the design document. Finally, a verification process should be run to check whether the system meets its specifications as discussed by the team at the beginning.

Integration Testing

Individual components are combined and tested into a single module to check the quality of the overall product and make sure, there's no other internal conflicts on the combination of whole product.

Object Detection component is followed by the Image classification component. Finally, the Information Summarizer comes into picture as it produces the detailed summary report on the classified herbal plant. Integration testing were done individually by the group members according to their component scope.

System Testing

Whole system is tested according to the specification, is said to be a System Integration. This makes sure, the system to be compatible with all the other modules and work as the execution plan

without arising any issues. This is more like, black box testing type of testing. If there's any bugs or issues, the overall team has to take the responsibility and relaunch the application with the needed modifications

User Acceptance Testing

This is the phase, where the customer interacts with the developed trial product. The product will be tested by the customer to make sure whether it reaches the customer functional requirements. They will be given a demo version of the product, so that, they can test it. If the user accepts the products without making any issues and satisfied, that is the end of the testing life cycle. Otherwise, the team has to do some other modifications accordingly, and the cycle goes on until the user gets satisfied.

Test cases for the Arogya Application: Summary generation and Geographical Location Mapping

Table 2.7.3 1: Test Case 1

Test case ID	1.1.	1.2.
Description	Login to the Application (If you have an account already)	
Input Data	● Username/E-mail ● Password	
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the Sign In button 4. Type your E-mail and Password 5. Click on the **Sign In** button	1. Keep Password and E-mail empty 2. Click on the Sign In button
Expected Output	Successfully Redirect to the Dashboard	Validation Error: Required fields are empty
Actual Output	Successfully Redirect to the Dashboard	Validation Error: Required fields are empty
Status		

Pass -  Fail - 	Pass - 	Pass - 

Table 2.7.3 2: Test Case 2

Test case ID	2.1.	2.2.
Description	Sign up to the Application (If you do not have an account already)	
Input Data	• Username/E-mail • Password	
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Get Started** button 4. Type your Name, E-mail and Password 5. Click on the Sign-Up button	1. Keep Name, Password and E-mail empty 2. Click on the Sign-Up button
Expected Output	Successfully Redirect to the Dashboard	Validation Error: Required fields are empty
Actual Output	Successfully Redirect to the Dashboard	Validation Error: Required fields are empty
Status Pass -  Fail - 	Pass - 	Pass - 

Table 2.7.3 3: Test Case 3

Test case ID	3.
Description	Reset Password (If you forgot your password)

Input Data	• Username/E-mail
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Click the **Forget Password** link 5. Click on the **Submit** button
Expected Output	Receive an email with the password reset link
Actual Output	Receive an email with the password reset link
Status Pass -  Fail - 	Pass - 

Table 2.7.3 4: Test Case 4

Test case ID	4.
Description	Add a New Location to Map (If the plant is classified as a Herbal One)
Input Data	• Geographical Latitude and Longitude coordinates • Plant Name
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Location Tracker** button 6. Click the **Plus Mark** in the header 7. Enter the Plant Name 8. Click the add plant name into the Map
Expected Output	Location will be added to the Map view
Actual Output	Location is added to the Map view
Status Pass -  Fail - 	Pass - 

Table 2.7.3 5: Test Case 5

Test case ID	5.
Description	View the Herbal Plant list (If the Login is successful)
Input Data	● No Input
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Top Right Corner Square** button 6. You can view the List of Herbal plants
Expected Output	Display the Herbal Plants List view
Actual Output	Display the Herbal Plants List view
Status Pass -  Fail - 	Pass - 

Table 2.7.3 6: Test Case 6

Test case ID	6.
Description	Add a New Herbal Plant as a post (If the Login is successful)
Input Data	● Plant Name ● Plant Specialization
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Top Right Corner Square** button 6. You can view the List of Herbal plants 7. Click the **Plus Mark** in the header 8. Enter the Plant Name 9. Click the **Continue** Button 10. Enter the Disease name it is specialized for 11. Click the **Submit** button
Expected Output	Display the newly added plat in the Herbal Plants List view

Actual Output	Display the newly added plat in the Herbal Plants List view
Status Pass -  Fail - 	Pass - 

Table 2.7.3 7: Test Case 7

Test case ID	7.
Description	Edit a Plant from the List (If the Login is successful)
Input Data	• No Input
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Top Right Corner Square** button 6. You can view the List of Herbal Plants 7. Click on a record from the list 8. Change the values accordingly and click **Update**
Expected Output	Plant List will be updated with the new value
Actual Output	Plant List will be updated with the new value
Status Pass -  Fail - 	Pass - 

Table 2.7.3 8: Test Case 8

Test case ID	8.
Description	Delete a Plant from the List (If the Login is successful)
Input Data	• Select a plant

Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Top Right Corner Square** button 6. You can view the List of Herbal Plants 7. Click on a record from the list 8. Click **Delete**
Expected Output	Plant will be deleted from the Plant List
Actual Output	Plant is deleted from the Plant List
Status Pass -  Fail - 	Pass - 

Table 2.7.3 9: Test Case 9

Test case ID	9.
Description	Compare the summary report on an herbal plant (If the Login is successful)
Input Data	• Input the Extracted text from the Internet
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Summarizer** button 6. Click the left most **Menu** button 7. Copy and paste the text extracted from the **Cinnamon Summarization** 8. Click **Summarize** button 9. Compare the Custom text area and Summary text area generated in each column
Expected Output	Summary will be generated
Actual Output	Summary is generated
Status Pass -  	Pass - 

Fail -	

Table 2.7.3 10 Test Case 10

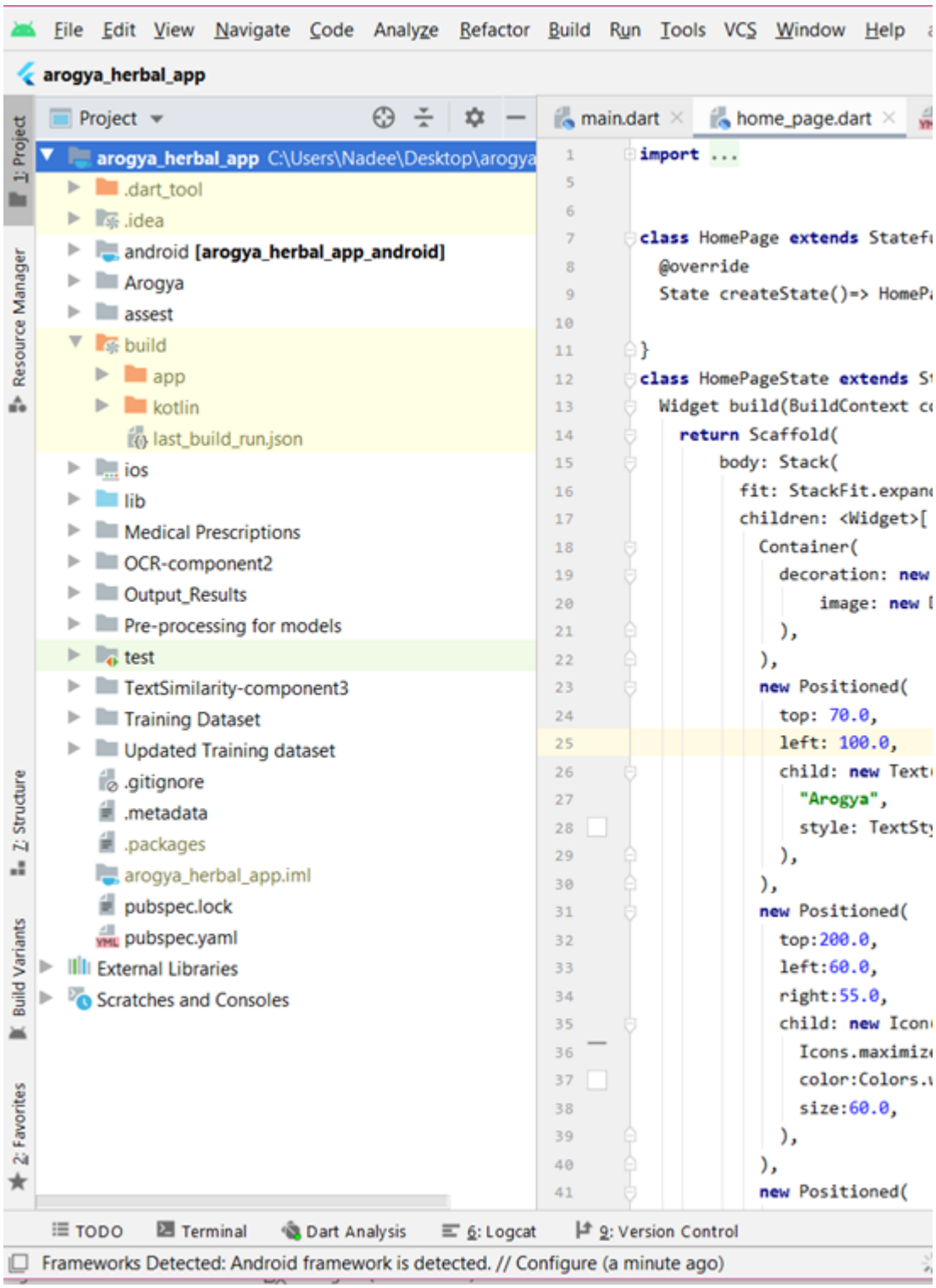
Test case ID	10.
Description	Get a summary report on an herbal plant (If the Login is successful)
Input Data	• Input the Extracted text from the Internet
Steps	1. Select the **Arogya icon** from the app list 2. Wait for the Loading 3. Click the **Sign In** button 4. Redirect to the Dashboard 5. Click on the **Summarizer** button 6. Scroll down and click on the button **Cinnamon Summarization** 7. Check the Custom text area and Summary text area
Expected Output	Summary will be generated
Actual Output	Summary is generated
Status Pass -  Fail - 	Pass - 

2.7.4. Text classification of Herbal plants and predict the category type of diseases and Identifying the Ayurveda image texts using OCR techniques and ayurveda related posts/recipes using cosine text similarity.

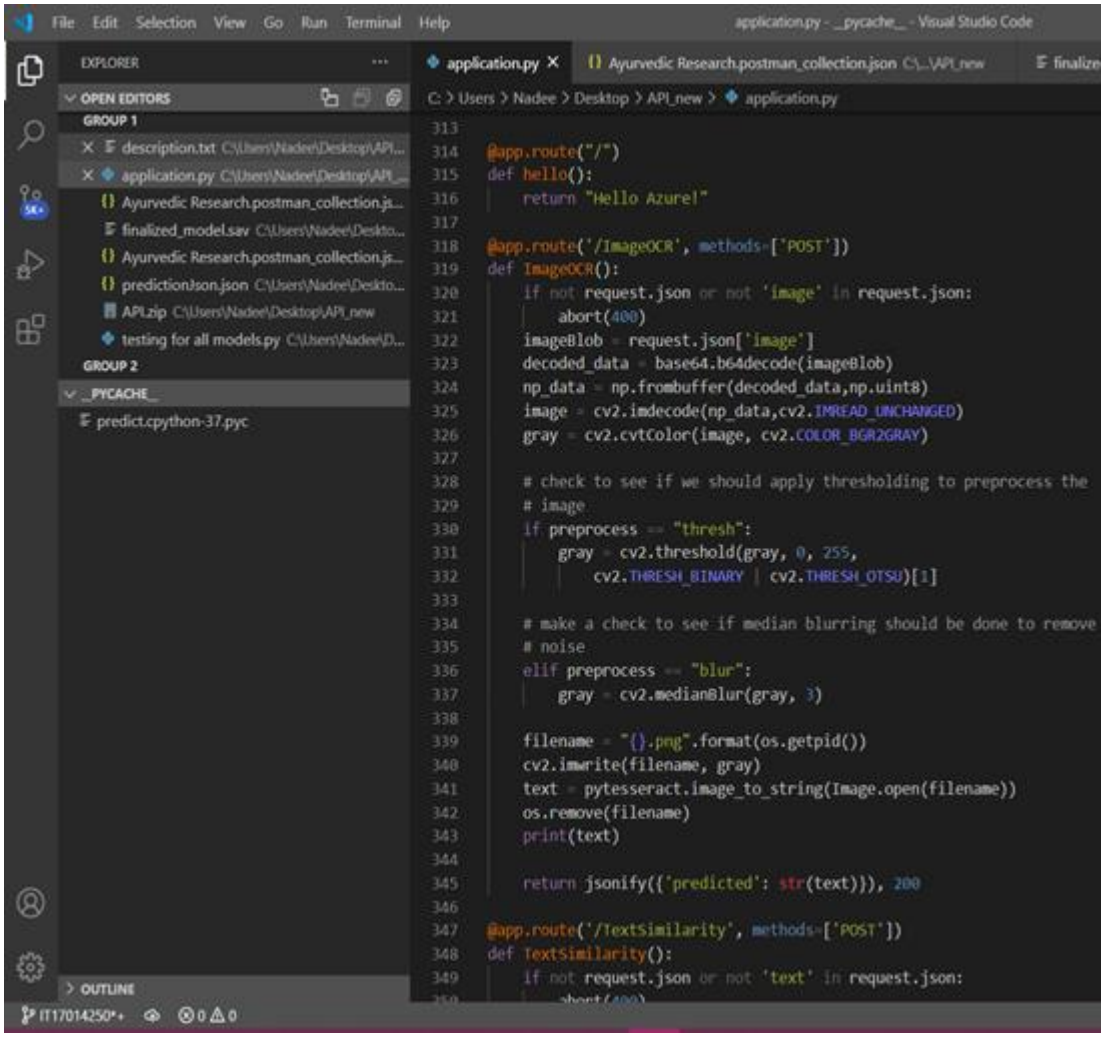
From the starting point of the project we followed an Agile based management model of until the end of the project. Unit testing was done for each individual component and its sub-components. During the implementation period, we used Gitlab as the version control system for version controlling our application.

Software testing is required each and every phase that we are following in software development life cycle like unit testing, integration testing, system testing and user acceptance testing. When consider this individual component testing is done for two systems.

Ø Frontend – Android application



Ø Backend – Machine learning model



Frontend Testing – Make sure the android app will be performing well without any failures.

2.7.4.1 Identifying the category type of Disease

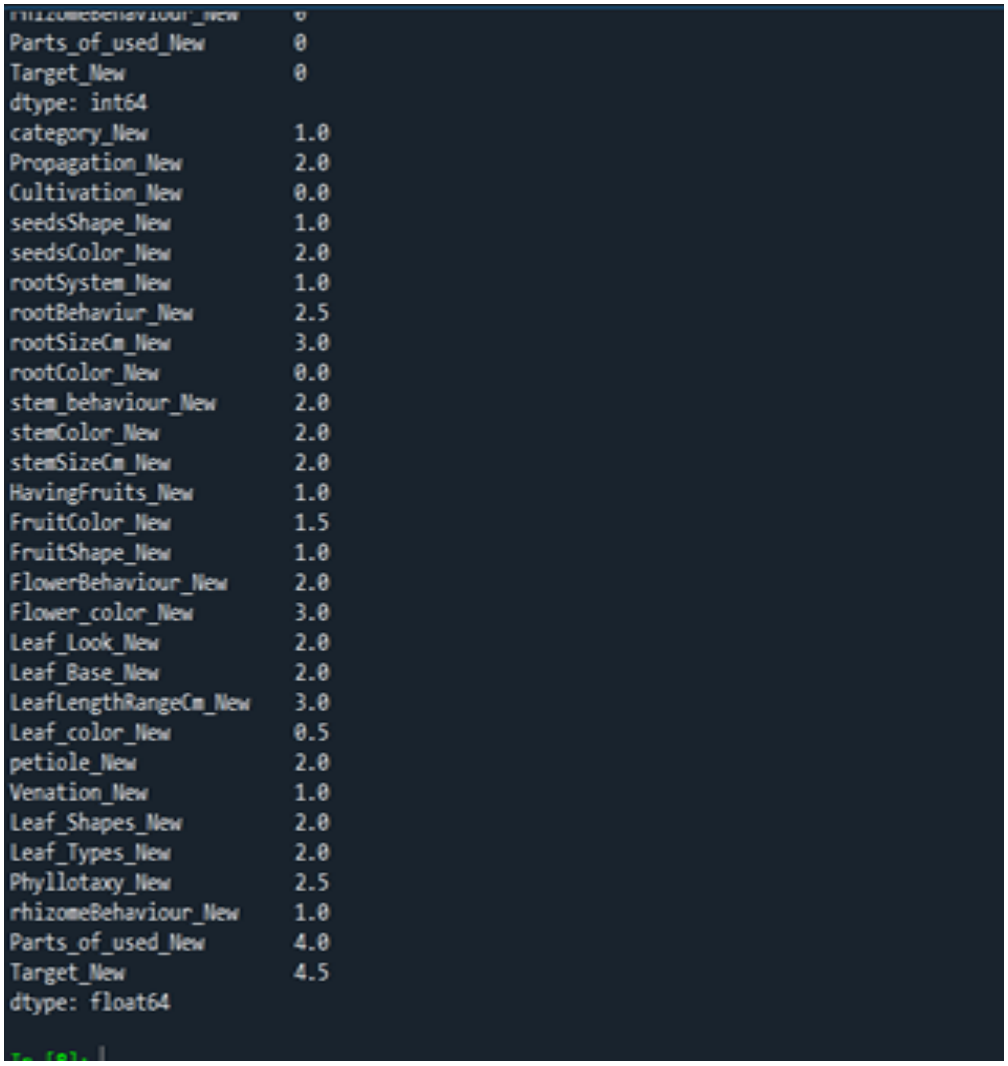
Implementation:

codes are listed in below.

Calculate the duplication rows and assign them into new rows as per in below.

Assign the new columns to numeric values into string type. After assigning them as per below results

```
In [7]: runfile('C:/Users/Nadee/Desktop/New folder/4 Plant Prediction/visualization.py', wdir='C:/Users/Nadee/
Desktop/New folder/4 Plant Prediction')
number of duplicate rows:  (66, 29)
category_New           0
Propagation_New        0
Cultivation_New        0
seedsShape_New         0
seedsColor_New         0
rootSystem_New         0
rootBehaviur_New       0
rootSizeCm_New         0
rootColor_New          0
stem_behaviour_New     0
stemColor_New          0
stemSizeCm_New         0
HavingFruits_New       0
FruitColor_New         0
FruitShape_New         0
FlowerBehaviour_New    0
Flower_color_New       0
Leaf_Look_New          0
Leaf_Base_New          0
LeafLengthRangeCm_New  1
Leaf_color_New         0
petiole_New            0
Venation_New           0
Leaf_Shapes_New        0
Leaf_Types_New         0
Phyllotaxy_New         0
rhizomeBehaviour_New   0
Parts_of_used_New      0
Target_New             0
dtype: int64
category_New           0
```



Test case ID	001
Test case scenario	Predicting the disease category
Test steps	1. Login to the app 2. Select the details using drop-down options display in the interface 2.1 select plant category 2.2 select propagation 2.3 select Leaf base 2.4 select venation 2.5 select Leaf shape 2.6 select Leaf type 2.7 select phylloataxy 2.8 select Flower bloomed 2.9 select Flower color 2.10 select flower behaviour 2.11 select having fruits

	2.12 select fruit shape 2.13 select parts of used 2.14 select cultivation 3. Click “Analyzing” button
Test Data	1. Username = nadee@gmail.com Password= nadee@321 2. 2.1 vine 2.2 seeds and tubers 2.3 Rounded 2.4 parallel-veined 2.5 Acicular 2.6 Compound 2.7 whorled 2.8 yes 2.9 Red berrish/white/ pinkis white 2.10 small and spikes 2.11 yes 2.12 oval 2.13 whole parts 2.14 wet zone
Expected Results	Diabetics
Actual Results	Diabetics
Pass/fail	Pass

Test for the predicting disease type

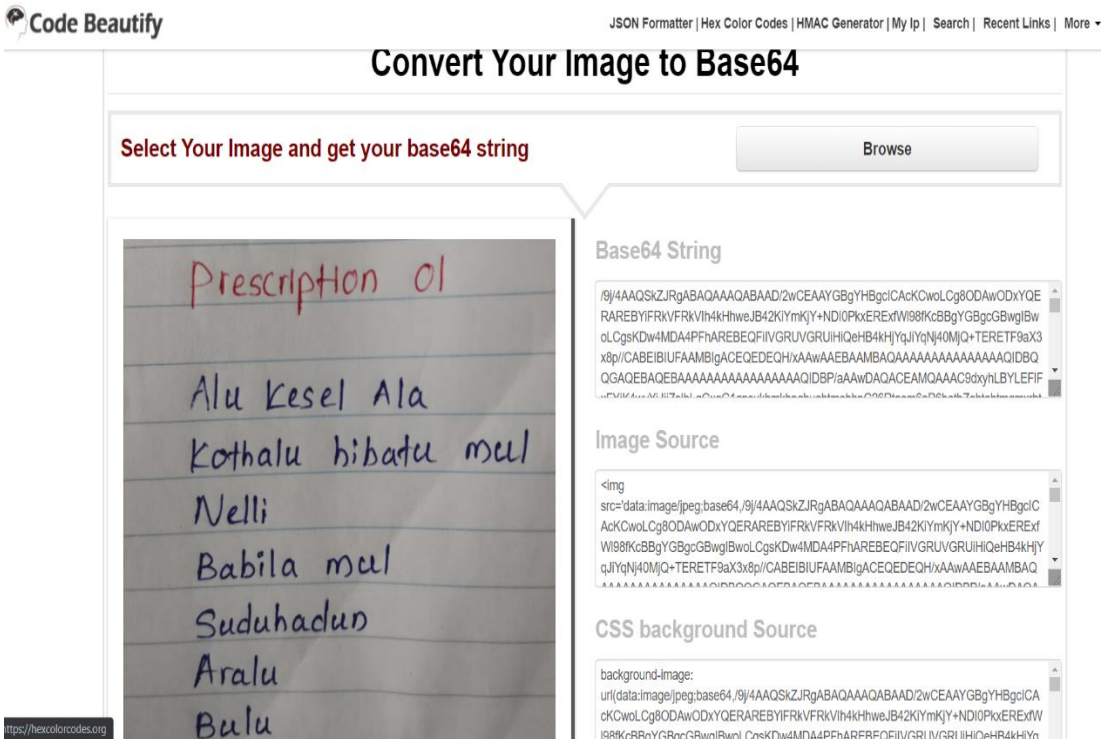
2.7.4.2 Analyzing the Image using OCR

This is the implementation of the component as per in below.
Load the Image from disk into memory and convert it to grayscale.

```
image = cv2.imdecode(np_data,cv2.IMREAD_UNCHANGED)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```python
@app.route('/ImageOCR', methods=['POST'])
def ImageOCR():
    if not request.json or not 'image' in request.json:
        abort(400)
    imageBlob = request.json['image']
    decoded_data = base64.b64decode(imageBlob)
    np_data = np.frombuffer(decoded_data,np.uint8)
    image = cv2.imdecode(np_data,cv2.IMREAD_UNCHANGED)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # check to see if we should apply thresholding to preprocess the
    # image
    if preprocess == "thresh":
        gray = cv2.threshold(gray, 0, 255,
            cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

    # make a check to see if median blurring should be done to remove
    # noise
    elif preprocess == "blur":
        gray = cv2.medianBlur(gray, 3)

    filename = "{}.png".format(os.getpid())
    cv2.imwrite(filename, gray)
    text = pytesseract.image_to_string(Image.open(filename))
    os.remove(filename)
    print(text)

#  # show the output images
# cv2.imshow("Image", image)
# cv2.imshow("Output", gray)
# cv2.waitKey(0)

    return jsonify({'predicted': str(text)}), 200
```

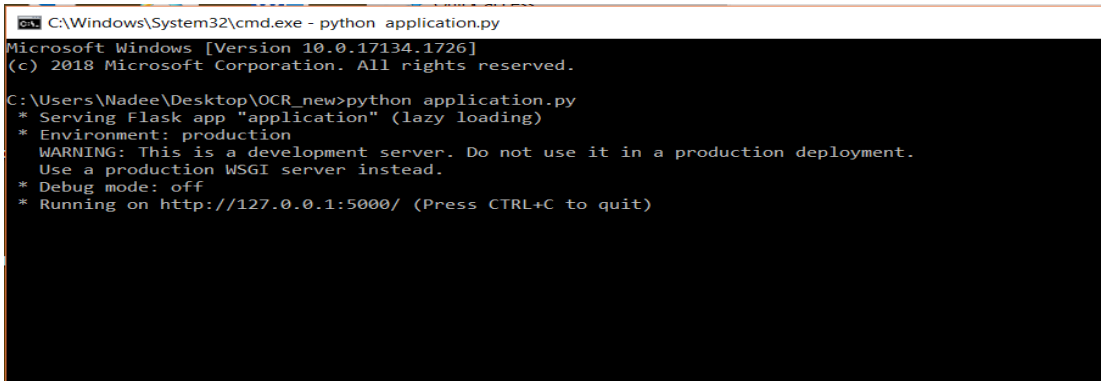
When upload an image which should have to convert image to string type using base64. This is the link used for converting as per in below.
https://codebeautify.org/image-to-base64-converter



Checking the results through the Postman - OCR

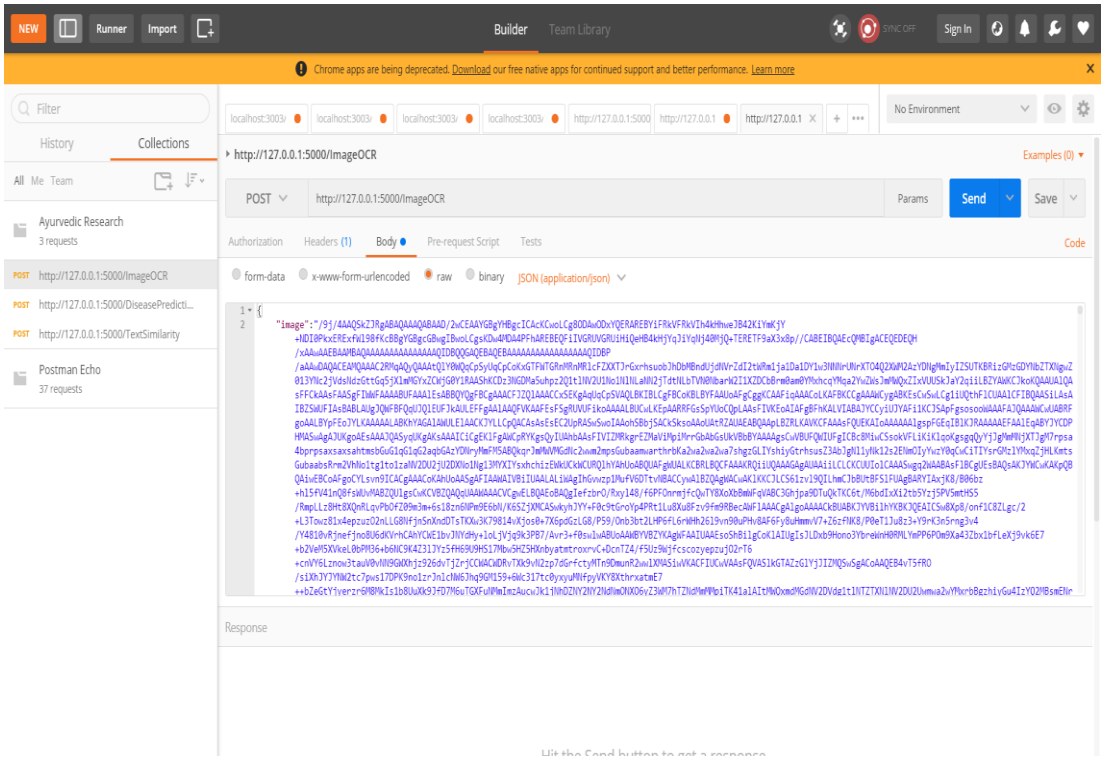
Firstly, open the needed folder which included all the python scripts, texts, images and the Json object which are used to pass through the URL. Then open the command prompt from this related folder path and run the application.py script. Type like this,

- Python application.py



Run the image sending through the URL to the object. Load the image and see the output results of Image OCR.

Apply the String type of image to the postman



Test case ID	002
Test case scenario	Upload the image and identify its text using camara or Gallery option
Test steps	1. Login to the Arogya app with correct credentials. 2. Navigate to the image/recipe/post upload interface 3. Select an image using Gallery or take an image using camara option 4. Click upload button
Test Data	1. Username = nadee@gmail.com Password= nadee@321 2. 
Expected Results	Prescription 01 Alu kesel Ala Kothalu hibatu mul Nelli Babila mul Suduhadun Aralu Bulu
Actual Results	As Expected,

Pass/fail	Pass

2.7.4.3 Identifying Ayurvedic related posts using Text similarity

```
@app.route('/TextSimilarity', methods=['POST'])
def TextSimilarity():
    if not request.json or not 'text' in request.json:
        abort(400)
    Y = request.json['text']
    Y_list = word_tokenize(Y)
    Y_set = {w for w in Y_list if not w in sw}
    results = []

    for X in sentences:
        if X:
            if X != '':
                # tokenization
                X_list = word_tokenize(X)
                l1 =[];l2 =[]
                # remove stop words from the string
                X_set = {w for w in X_list if not w in sw}
                # form a set containing keywords of both strings
                rvector = X_set.union(Y_set)
                for w in rvector:
                    if w in X_set: l1.append(1) # create a vector
                    else: l1.append(0)
                    if w in Y_set: l2.append(1)
                    else: l2.append(0)
                c = 0
                # cosine formula
                for i in range(len(rvector)):
                        c+= l1[i]*l2[i]
                cosine = c / float((sum(l1)*sum(l2))**0.5)
                results.append(cosine)

    return jsonify({'results': json.dumps(results)}), 200
```

Applying the text to check though postman

Test case ID	001
Test case scenario	Upload Ayurveda related post/recipe
Test steps	1. Login to the Arogya app with correct credentials. 2. Navigate to the post/recipe upload interface. 3. Select a post from the gallery 4. Upload the post/recipe
Test Data	1. Username = nadee@gmail.com Password= nadee@321 4. This recipie is for orthopedic in Ayurveda treatment. Treatments For wrist fracture. Needed Ayurveda remedies are nelli Arallu Bullu Kohobha pothu Sawandara mul. Ayurveda is considered by many scholars to be the oldest healing science. In Sanskrit Ayurveda means the science of life Ayurvedic knowledge. originated in India more than 5000 years ago and is often called the Mother of Healing It stems from the ancient vedic culture and was taught for many thousands of years in an oral tradition from accomplished masters to their disciples.
Expected Results	Identify the ayurvedic related or not. If it ayurvedic related, then upload to the wall. If not post will be blocked.
Actual Results	Upload to the wall
Pass/fail	Pass

In this research project we have implemented our mobile application by using the following tools and technologies.

1. Python
2. Keras
3. TensorFlow

4. TensorFlow Lite
5. Jupiter Notebook
6. Google Colab
7. Android
8. Android Studio
9. Firebase – for the database
10. OpenCv
11. Pyterrest
12. Base64

3. RESULTS AND DISCUSSION

3.1.Selecting the Most Accurate and the Highest Performance Segmentation Technique using Experimental Outcomes of Plants in Sri Lanka

Actual User Interfaces

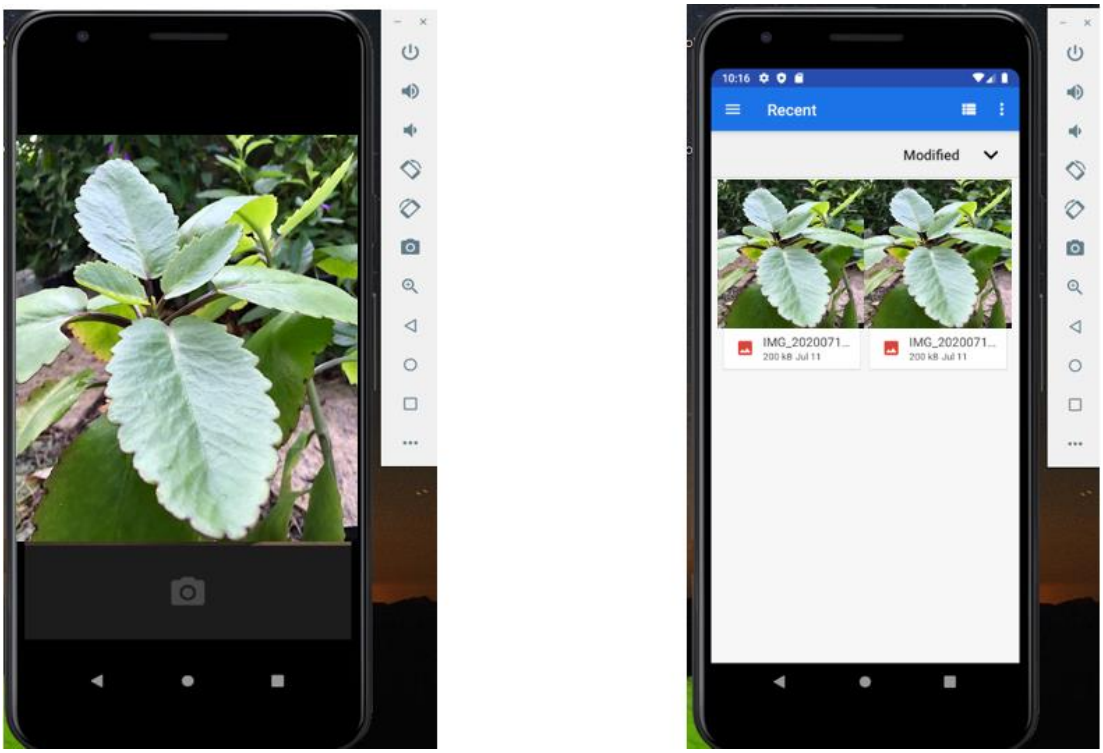


Figure 3.1.1.1 -Mobile UIs

3.1.2 Summary of the result

The following results depict clear evidence to prove the experiment results that the proposed watershed marker based algorithm obtained the highest performance and accuracy.

1) YOLO

Same Annotated dataset is used in YOLO v3 and YOLO v5. To compare the result of these two versions of YOLO detection algorithm, same test data is used. Results of the test data for each version are given below.

Display inference on test images.

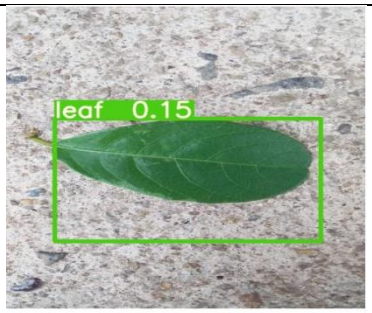
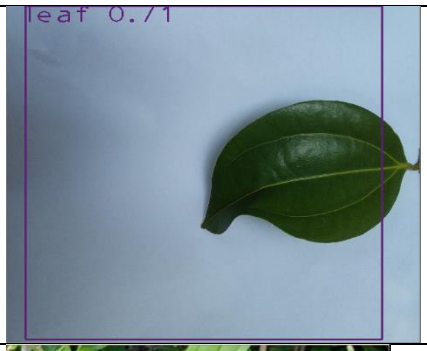
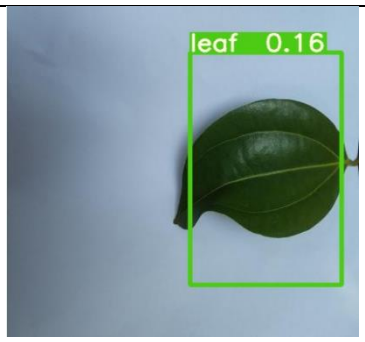
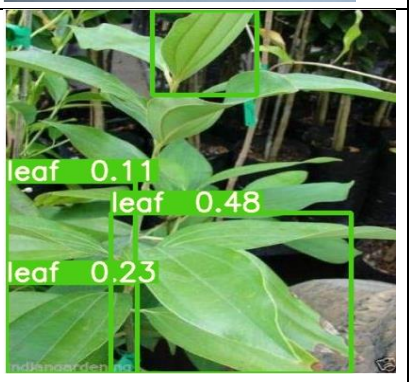
Input image	YOLO v3	YOLO v4
01 [Leaf in a colored background] 		
02 [Leaf in a white background] 		
03 [leaves in a complex background] 		
04)[leaves in a dark background] 		

Table 3.1.1.1-Compare results of YOLO V3 and YOLO V5

Evaluate Custom YOLOv5 Detector Performance

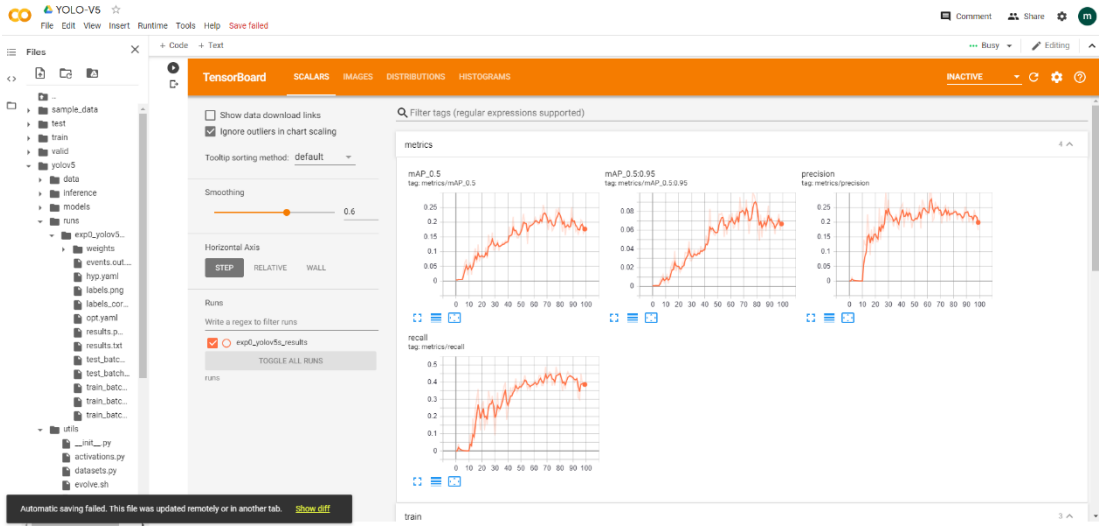


Figure 3.1.1.2-Performance Metrics of YOLO V5

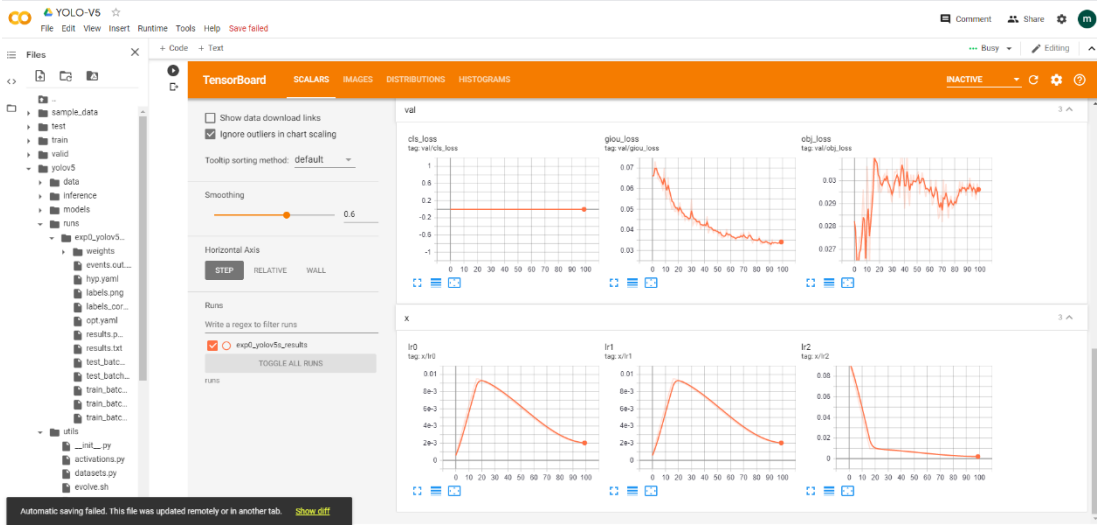


Figure 3.1.1.3-Validation performance of YOLO V5

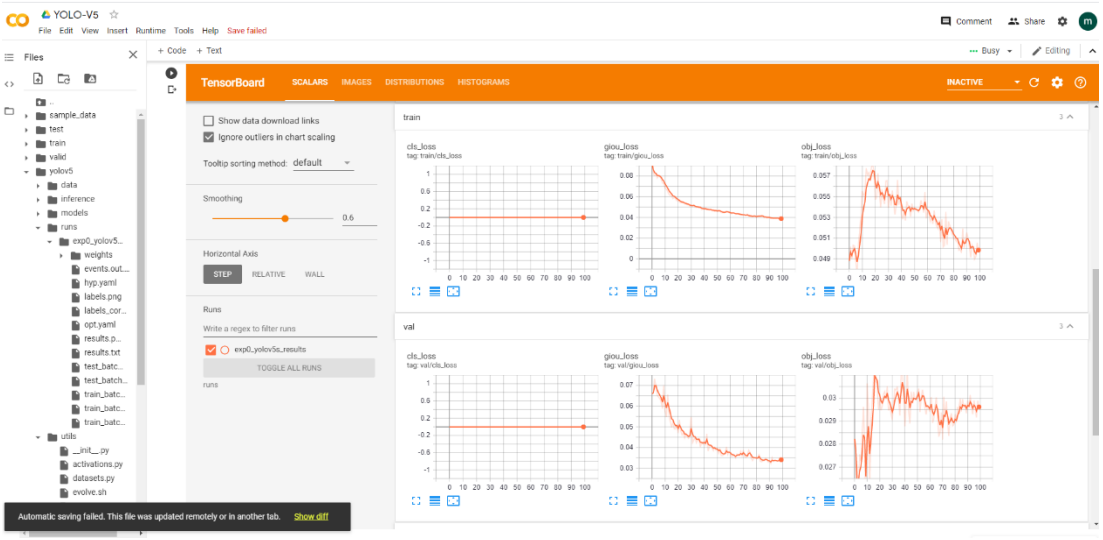
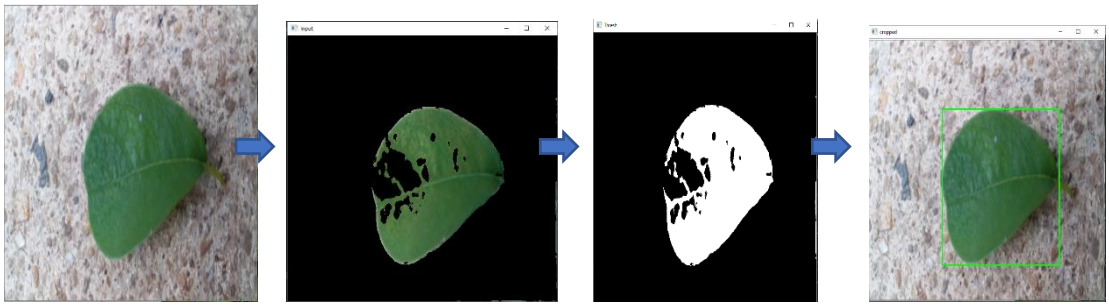


Figure 3.1.1.4-Training performance of YOLO V5

2) Proposed marker based watershed segmentation

Display inference on test images.

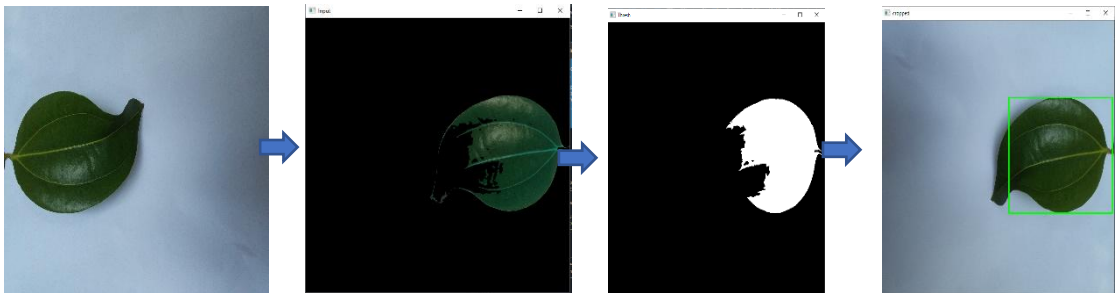
Above test data is used here also to compare the detection result.
Input 1: [leaf in a colored background]



input image green mask threshold detected object

Figure 3.1.1.5-Leaf in a colored background

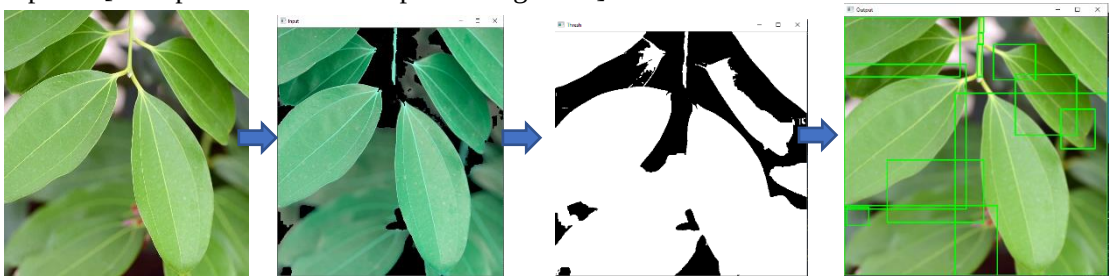
Input 2: [leaf in a white background]



input image green mask threshold detected object

Figure 3.1.1.6-Leaf in a white background

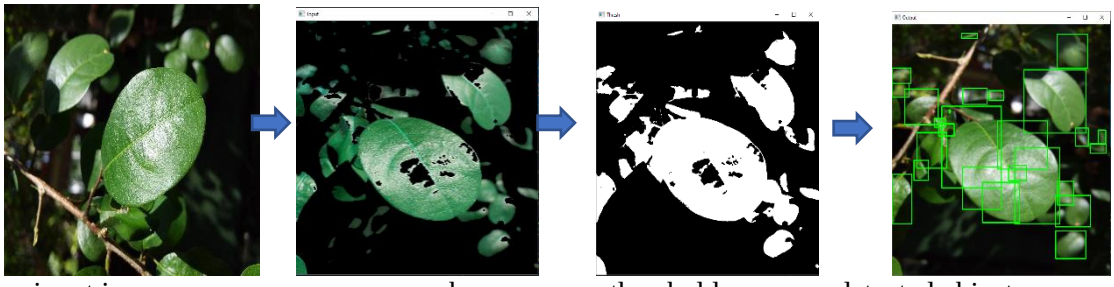
Input 3: [multiple leaves in a complex background]



input image green mask threshold detected object

Figure 3.1.1.7-Multiple leaves in a complex background

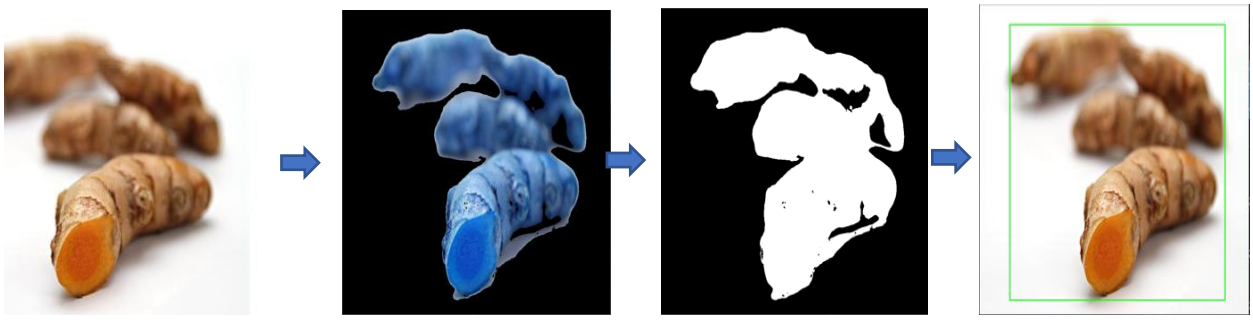
Input 4: [multiples leaves in a dark background]



input image green mask threshold detected object

Figure 3.1.1.8-Multiple leaves in a complex background

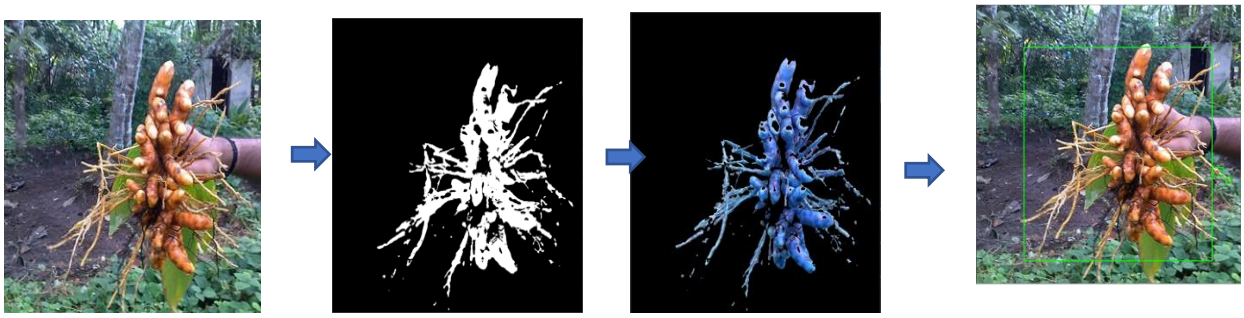
Input 5: [turmeric root in a simple background]



input image green mask threshold detected object

Figure 3.1.1.9- turmeric root in a simple background

Input 6: [turmeric root in a complex background]



input image green mask threshold detected object

Figure 3.1.1.10- turmeric root in a complex background

3.2. Classification of Ayurvedic Plants in Sri Lanka using transfer learning based on deep CNN Using A Mobile Application in An online Environment Approach

Summary of Results

Dataset

When taken as overall, a set of 1348 leaf images were retained, where 883, 235 and 230 images were used for training, validation, and testing purposes, respectively. These were the original images of the dataset those were not augmented. The class distribution of the samples over training, validation and testing folds are shown in the following table.

Table 3.2.1: Class distribution of samples over training, testing and validation folds

Herbal plant class	Train	Validation	Test	Total
Akkapana leaf	133	41	40	**214**
Cinnamon leaf	148	42	40	**230**
Katupila leaf with fruit	167	62	60	**289**
Kohomba	169	40	40	**249**
Turmeric leaf	78	40	40	**158**
Turmeric digested root	188	10	10	**208**
Total	**883**	**235**	**230**	**1348**

While retraining the dataset in order to obtain a higher accuracy from the selected best accurate CNN model, only the training dataset was augmented, because augmenting validation and testing datasets is not valid in order to obtain the most reliable value of accuracy.

Experimental Outcomes

The primary results obtained from transfer learning based on the trained deep CNN architectures can be illustrated as follows. Those results indicate the model summary, final testing accuracy, model performance accuracy against training epochs and model performance loss against training epochs, for each model which have been trained and tested throughout this research.

1. InceptionV3

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 1, 1, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 12294
=================================================================
Total params: 21,815,078
Trainable params: 12,294
Non-trainable params: 21,802,784
```

Figure 3.1.1: InceptionV3 model summary

- Finalized testing accuracy: **56.76%**

- Model performance accuracy graph

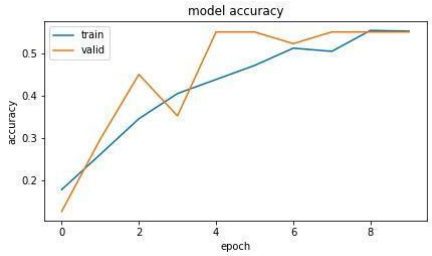


Figure 3.1.2: InceptionV3 accuracy performance

- Model performance loss graph

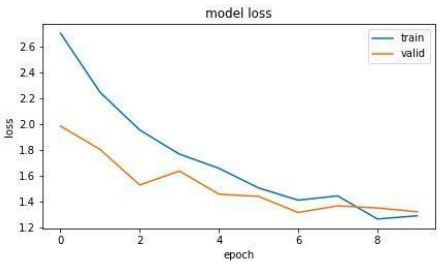


Figure 3.1.3: InceptionV3 loss performance

2. MobileNetV2

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
conv2d (Conv2D)              (None, 5, 5, 32)          368672
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 32)          0
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 6)                 198
=================================================================
Total params: 2,626,854
Trainable params: 368,870
Non-trainable params: 2,257,984
```

Figure 3.1.4: MobileNetV2 model summary

- Finalized testing accuracy: **63.58%**

- Model performance accuracy graph

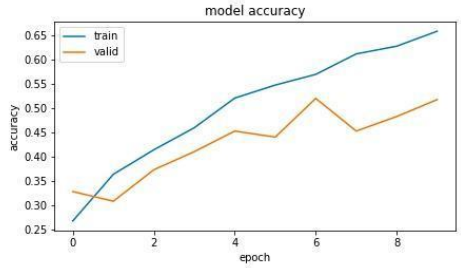


Figure 3.1.5: MobileNetV2 accuracy performance

- Model performance loss graph

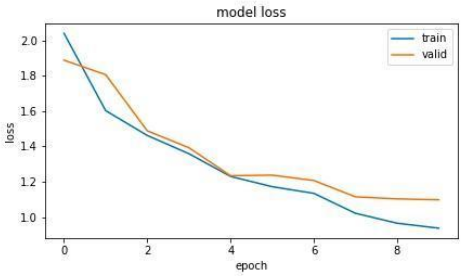


Figure 3.1.6: MobileNetV2 loss performance

3. InceptionResNetV2

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Model)  (None, 1, 1, 1536)        54336736
_________________________________________________________________
flatten (Flatten)            (None, 1536)              0
_________________________________________________________________
dropout (Dropout)            (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 512)               786944
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 1542
=================================================================
Total params: 55,256,550
Trainable params: 919,814
Non-trainable params: 54,336,736
```

Figure 3.1.7: InceptionResNetV2 model summary

- Finalized testing accuracy: **82.77%**

- Model performance accuracy graph

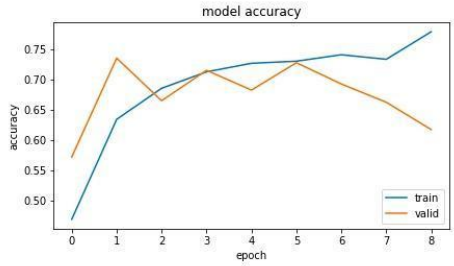


Figure 3.1.8: InceptionResNetV2 accuracy performance

- Model performance loss graph

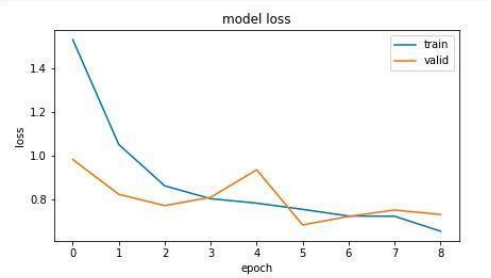


Figure 3.1.9: InceptionResNetV2 loss performance

4. Xception

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 3, 3, 2048)        20861480
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1049088
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 3078
=================================================================
Total params: 21,913,646
Trainable params: 1,052,166
Non-trainable params: 20,861,480
```

Figure 3.1.10: Xception model summary

- Finalized testing accuracy: **86.01%**

- Model performance accuracy graph

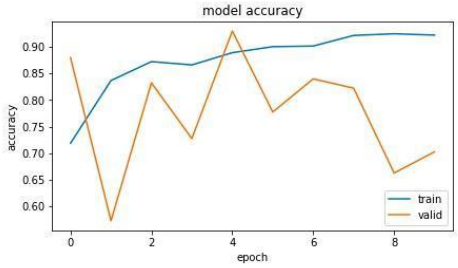


Figure 3.1.11: Xception accuracy performance

- Model performance loss graph

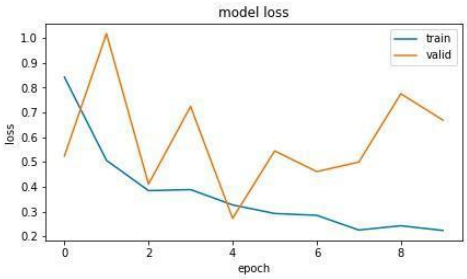


Figure 3.1.12: Xception loss performance

5. DenseNet121

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 3, 3, 1024)        7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 6150
=================================================================
Total params: 7,043,654
Trainable params: 6,150
Non-trainable params: 7,037,504
_________________________________________________________________
```

Figure 3.1.13: DenseNet121 model summary

- Finalized testing accuracy: **89.12%**

- Model performance accuracy graph

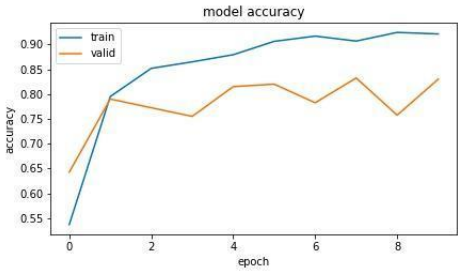


Figure 3.1.14: DenseNet121 accuracy performance

- Model performance loss graph

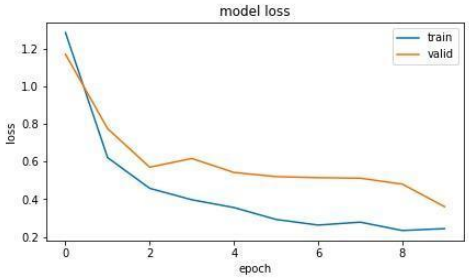


Figure 3.1.15: DenseNet121 loss performance

6. ResNet50

- Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 2048)              23587712
_________________________________________________________________
dense (Dense)                (None, 6)                 12294
=================================================================
Total params: 23,600,006
Trainable params: 14,988,294
Non-trainable params: 8,611,712
```

Figure 3.1.16: ResNet50 model summary

- Finalized testing accuracy: **98.92%**

- Model performance accuracy graph

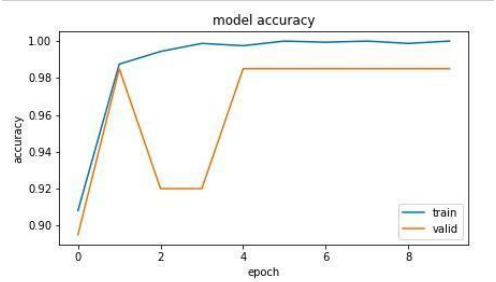


Figure 3.1.17: ResNet50 accuracy performance

- Model performance loss graph

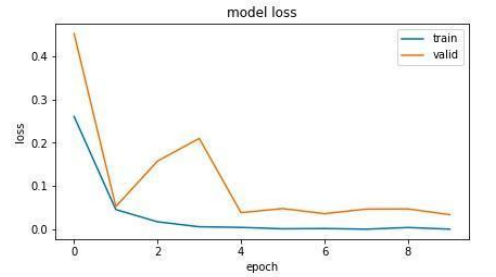


Figure 3.1.18: ResNet50 loss performance

7. VGG16

- Model Summary

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 3, 3, 512)	14714688
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
dense_1 (Dense)	(None, 6)	1542

Total params: 15,896,134
Trainable params: 15,896,134
Non-trainable params: 0

Figure 3.1.19: VGG16 model summary

- Finalized testing accuracy: **99.53%**

- Model performance accuracy graph

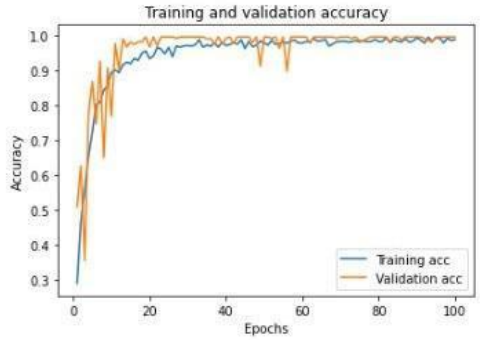


Figure 3.1.20: VGG16 accuracy performance

- Model performance loss graph

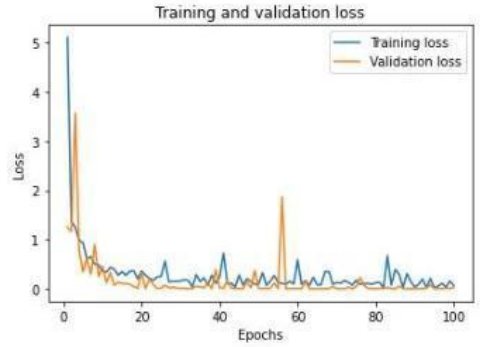


Figure 3.1.21: VGG16 loss performance

Classification results obtained from the finalized model

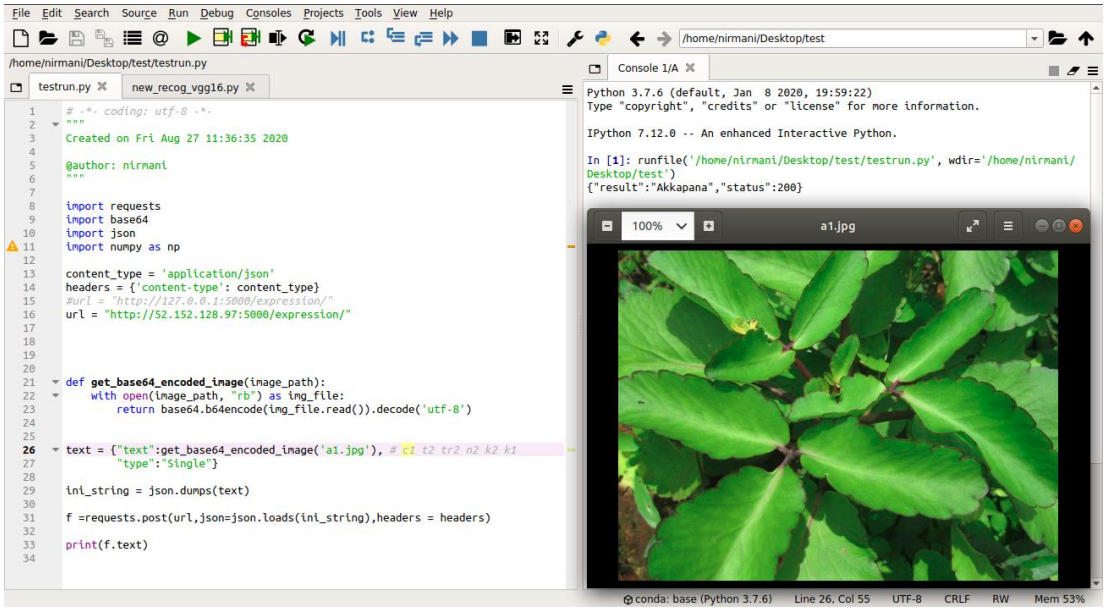


Figure 3.1.22: Classification of Akkapana

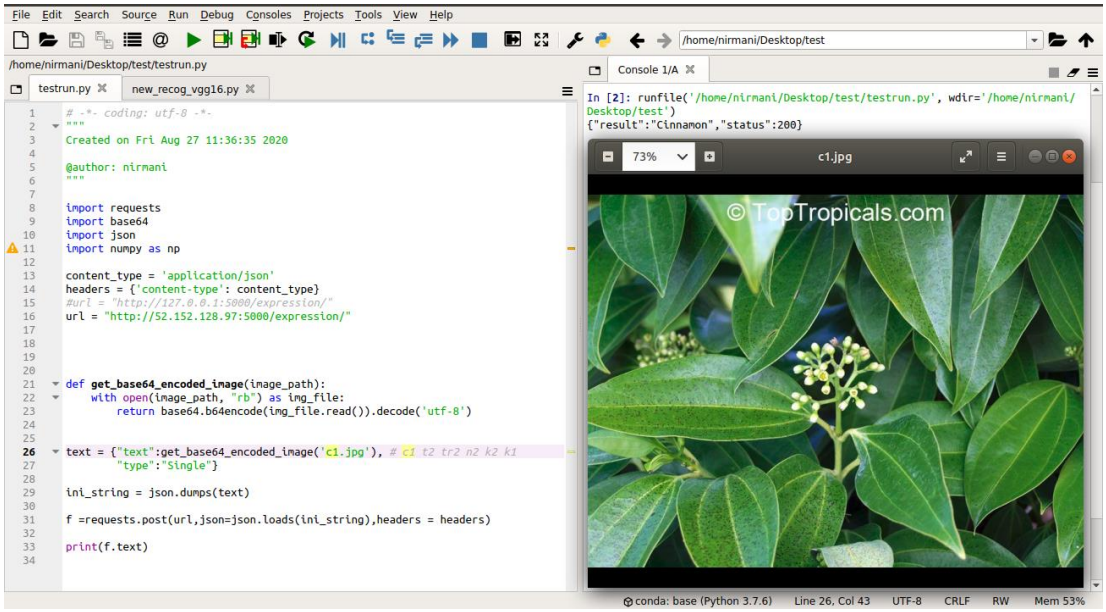


Figure 3.1.23: Classification of Cinnamon

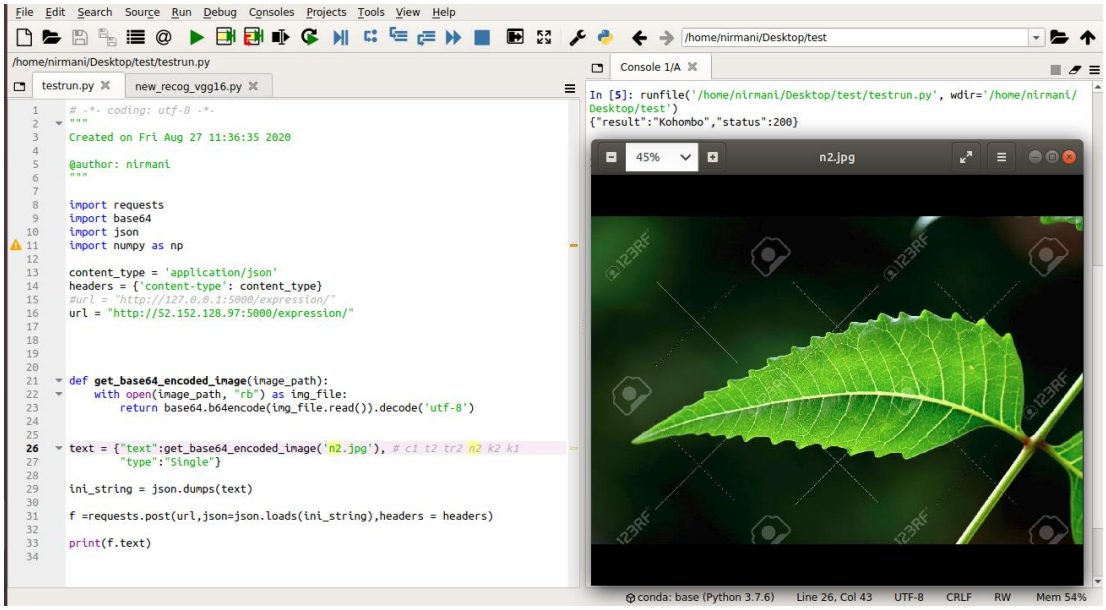


Figure 3.1.24: Classification of Kohomba

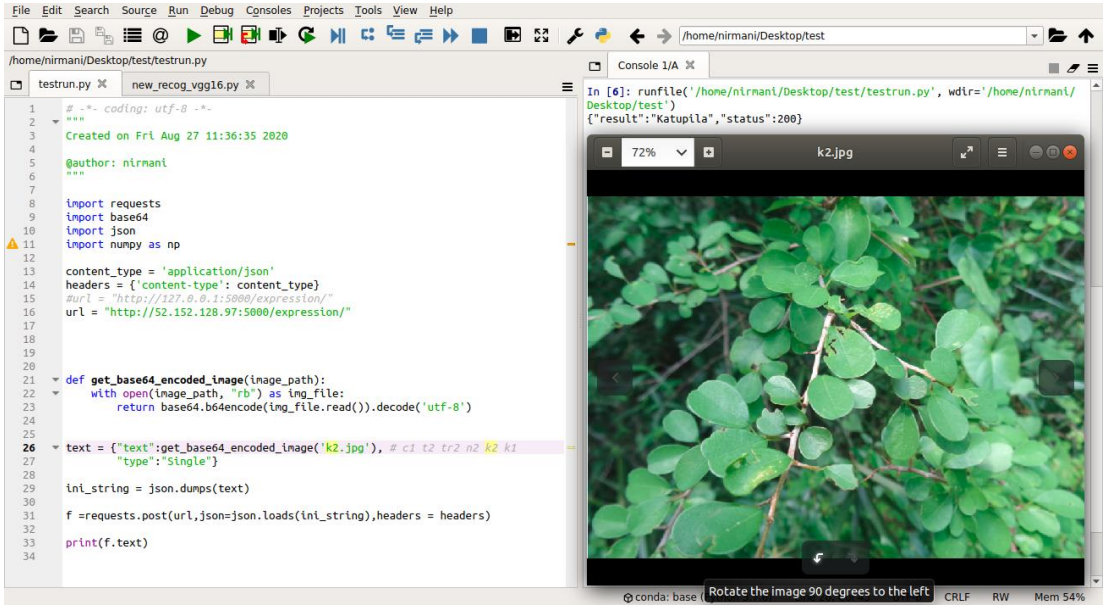


Figure 3.1.25: Classification of Katupila leaves

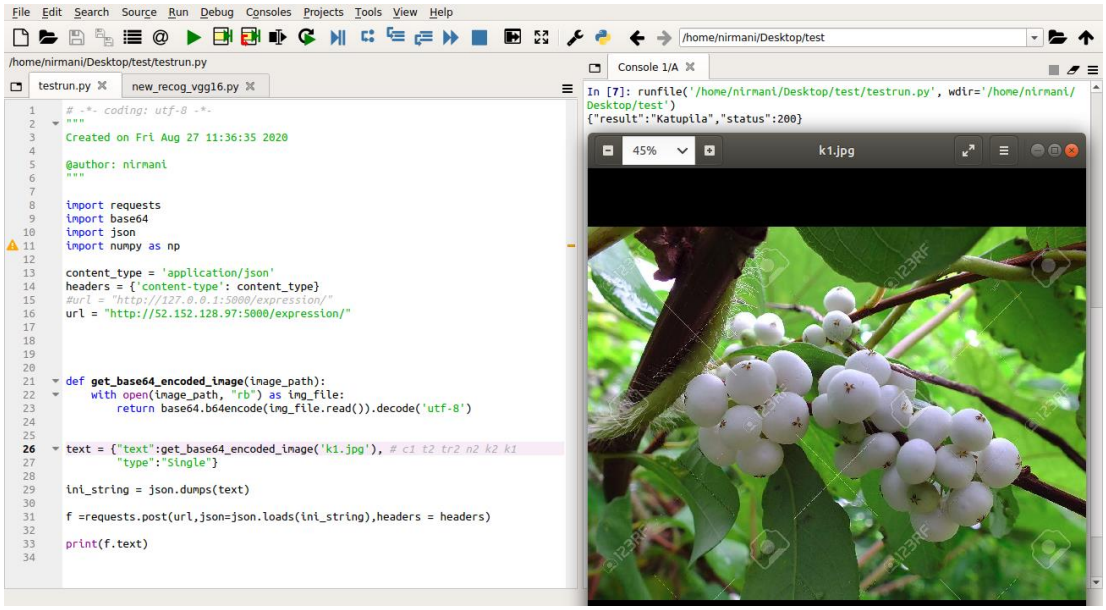


Figure 3.1.26: Classification of Katupila fruit

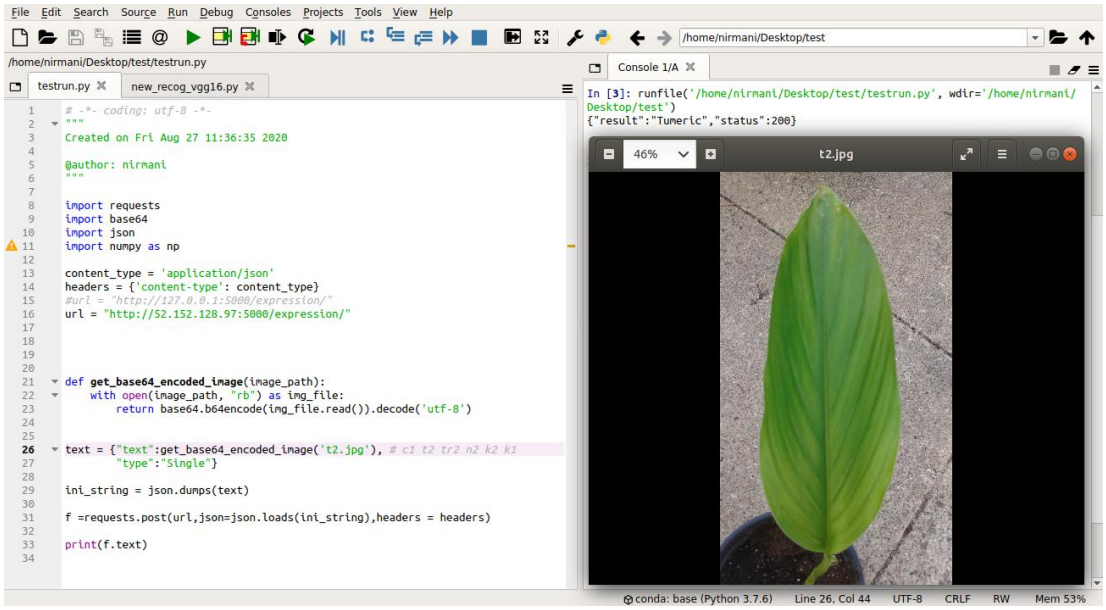


Figure 3.1.22: Classification of Turmeric leaf

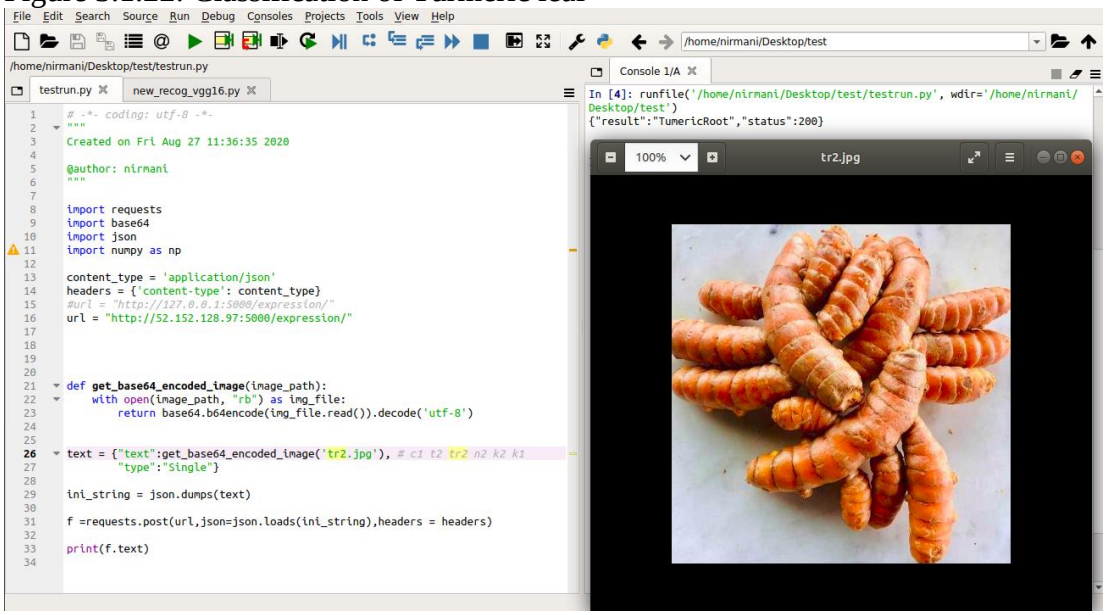


Figure 3.1.22: Classification of Turmeric root

3.3. Abstractive web page Information Summarization and Location Mapping with GIS technology on Ayurvedic Plants in Sri Lanka Using a Mobile Application in an Online Environmental Approach

Actual User Interfaces

3.3.1.1.Main Dashboard View

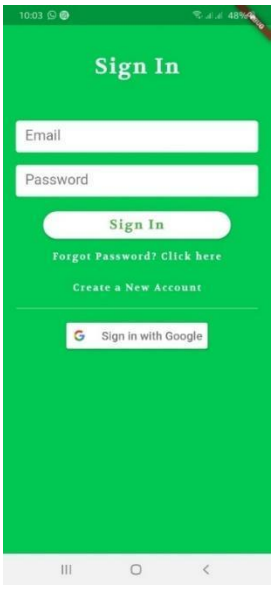
Table 3.3 1: Interface 1

Interface	Functionality
	• Main entry point to the Application-Main UI • Get Started button direct to the Sign-Up page • You get a chance to register into the system over there • Sign In button direct to the Sign In page • By providing the credentials, can login to the account

3.3.1.2.User Login

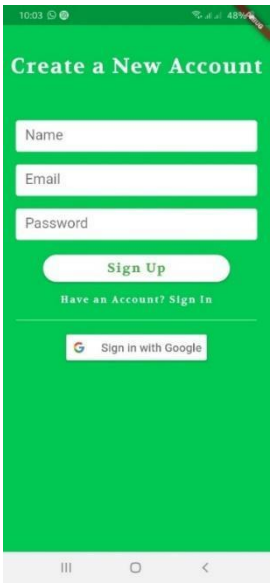
Table 3.3 2: Interface 2

Interface	Functionality

	• By providing the credentials, can login to the account • If you click **Sign in with Google**, you will have a chance of login with the google account credentials

3.3.1.3.User Registration

Table 3.3 3: Interface 3

Interface	Functionality
	• Sign Up to the system by providing Name, E-mail and Password • And also, you can use Google Sign Up method too

3.1.3.4. User Password Reset

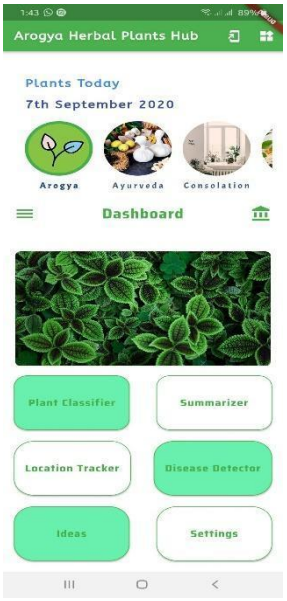
Table 3.3 4: Interface 4

Interface	Functionality

	• If you forgot your Password, provide your e-mail address, so that you can receive a reset link

3.1.3.5. Dashboard

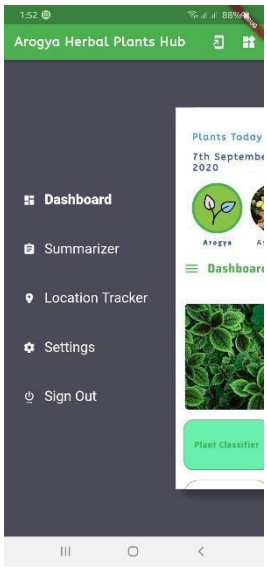
Table 3.3 5: Interface 5

Interface	Functionality
	• This is the Dashboard • You can access every main menu from this board

3.1.3.6. Side Bar Menu

Table 3.3 6: Interface 6

Interface	Functionality

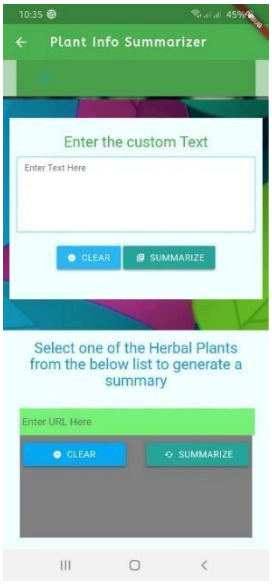
	• This is the side bar menu • Main menus are located in this view too

3.1.3.7. Herbal Plants Info Summarizer

Table 3.3 7: Interface 7

Interface	Functionality
	• This is the Summarizer view • User has to enter a customer text to generate a summary

Interface	Functionality

	• User has to paste the required websites' URL to be summarized • Then it will extract the content from that website and generate a summary

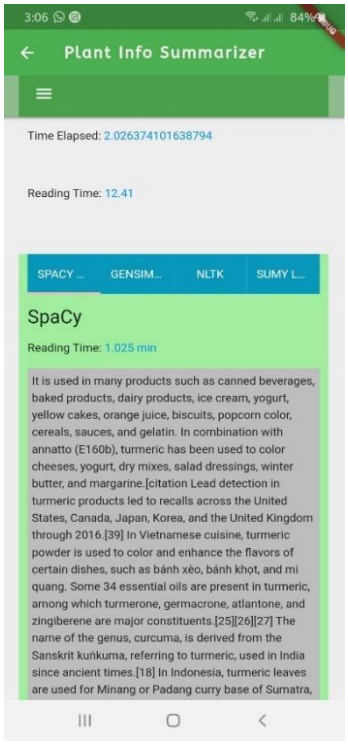
Interface	Functionality
	• 6 types of herbal plants are already listed in the view • If you want, you can select one of the plants from the list menu • Then the system will generate a summary on it

Interface	Functionality

	• Search for the plant by its name • Then it will extract the information from multiple websites • Finally, the system will generate a summary on it

3.1.3.8. Summary Comparison

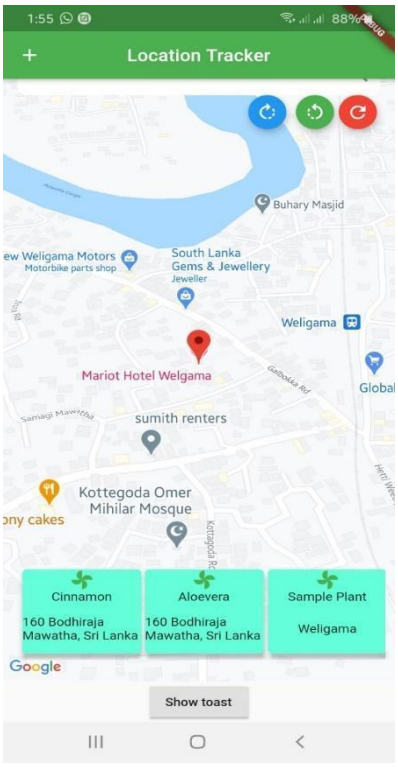
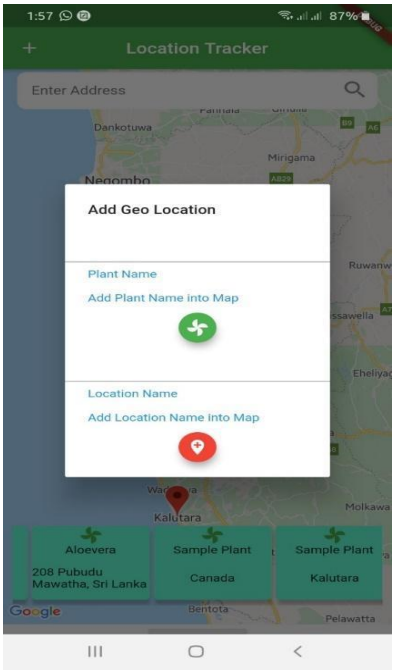
Table 3.3 8: Interface 8

Interface	Functionality
	• Generated summary is compared with four other models • So that, the user can obtain the most accurate results comparatively

3.1.3.9. Location Tracker

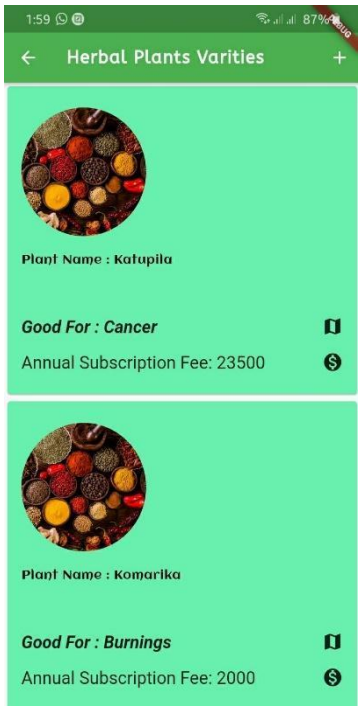
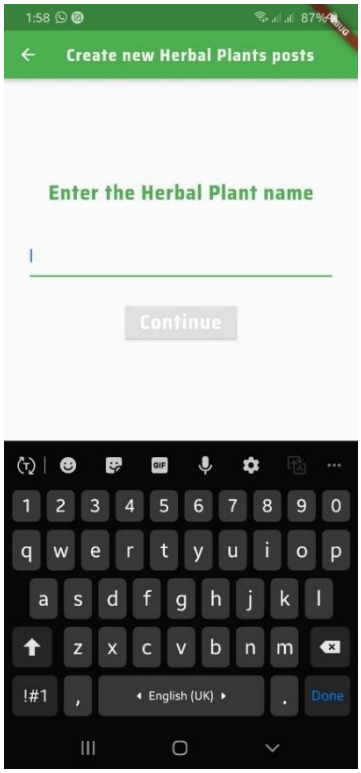
Table 3.3 9: Interface 9

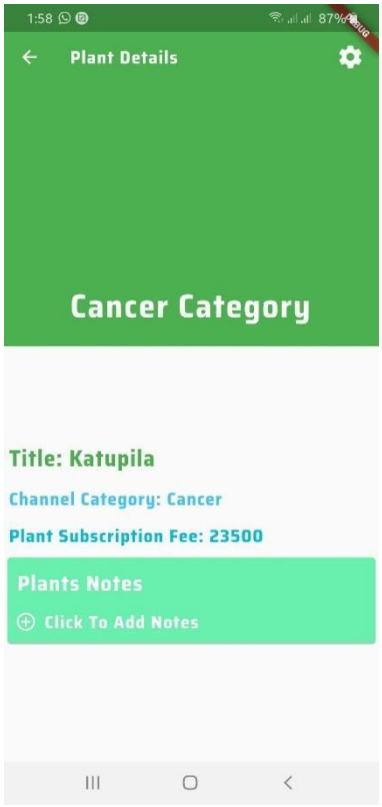
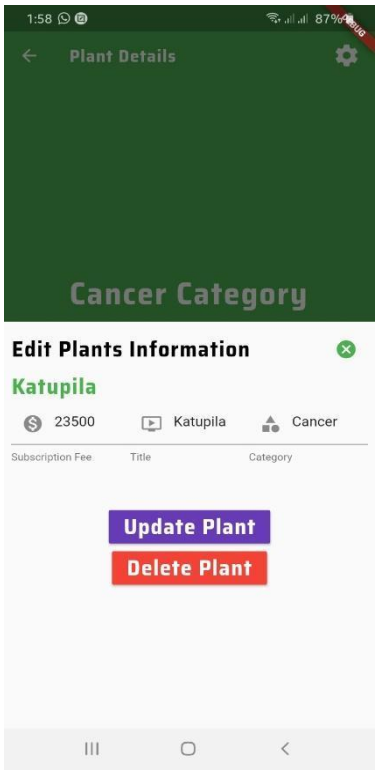
Interface	Functionality

	• Location View as a map • Same herbal plant can be grown in different locations • Many herbal plants can be grown in a one location
	• Track the current location • Add Plants into the Map view

3.1.3.10. Insert Herbal Plant details

Table 3.3 10: Interface 10

	• Insert required herbal plants in to the system
	• Enter the plant details to add into the system

	• More Information on the specific plant
	• Update herbal plant and Delete herbal plant

Summary of Results

The following graphs depict clear evidence to prove the experiment results that the Seq2Seq LSTM model obtained the highest accuracy. The graphs indicate the word count comparison between, the generated summary and the customized text which was extracted from the multiple web pages dynamically.

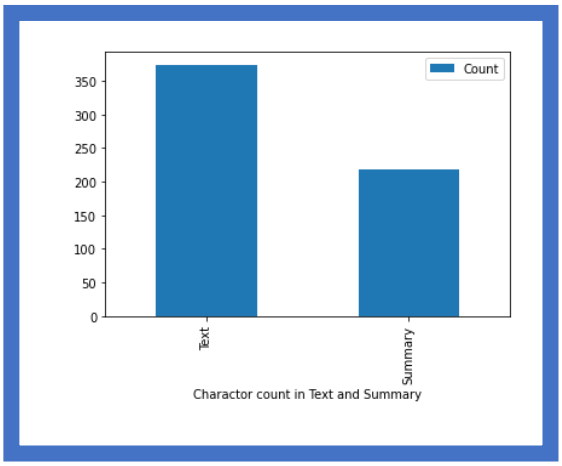


Figure 3.3 1: Instance 1 - Word count of Generated Summary in comparison to Customized text

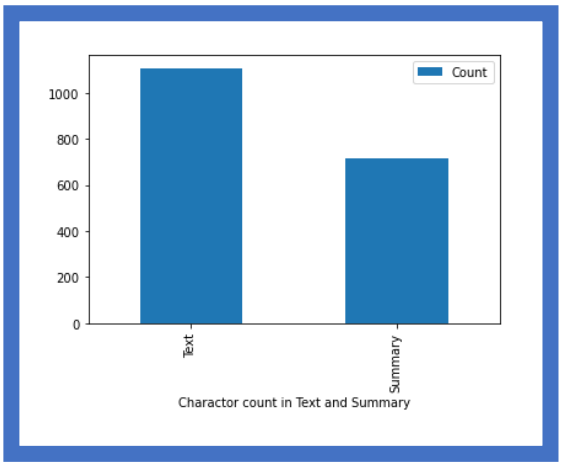
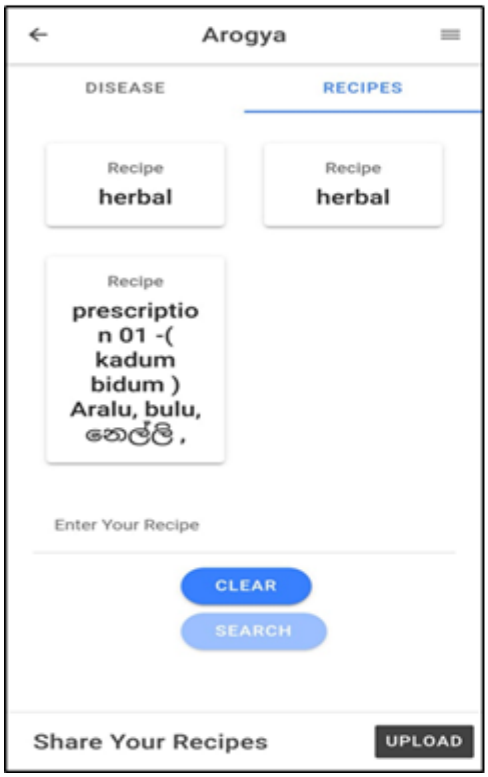
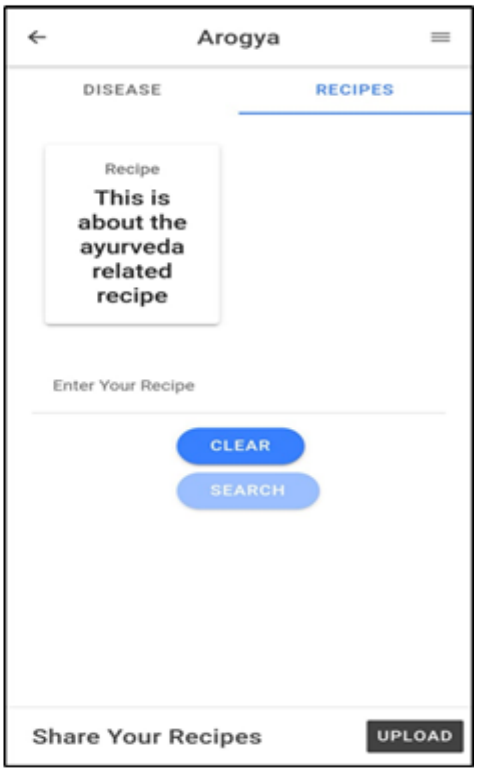


Figure 3.3 2: Instance 2 - Word count of Generated Summary in comparison to Customized text

3.4. Text classification of Herbal plants and predict the category type of diseases and Identifying the Ayurveda image texts using OCR techniques and ayurveda related posts/recipes using cosine text similarity.

Actual User Interfaces





Summary of Results

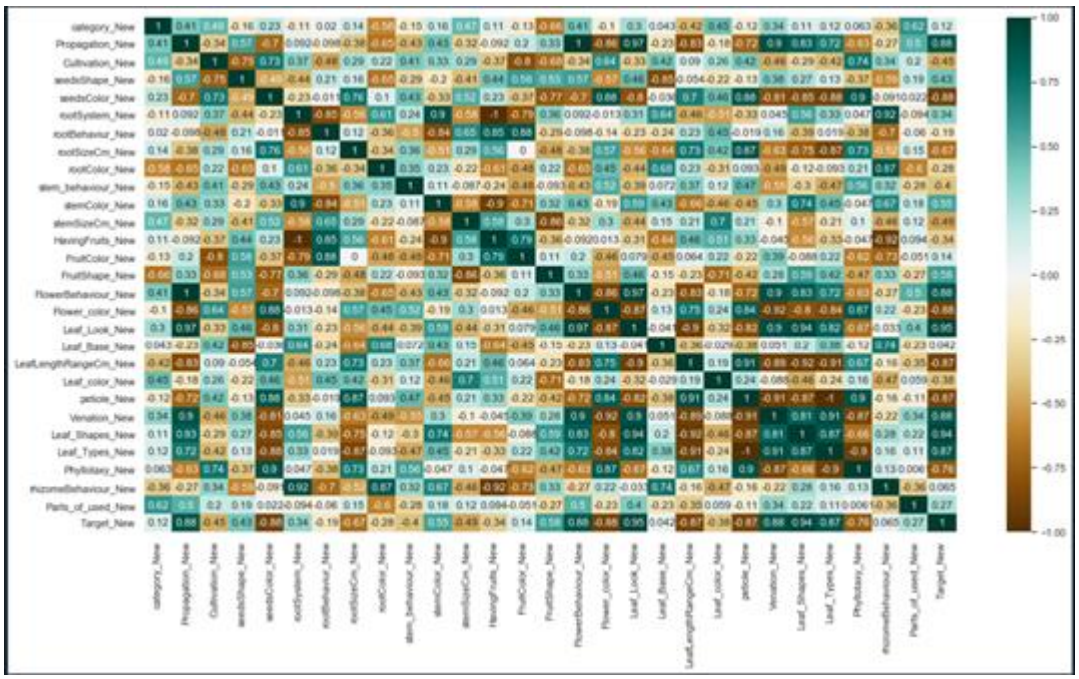
Arogya is an ayurvedic app which is consists of the different features according to the new technology. So that, in here developed various types of services to the users as they wish. Primarily there are four main components were consisted with this app. *Analyzing the category type of disease and Identifying the Ayurveda related recipes or posts using OCR techniques and text similarity* is the component, which is described here as among the four main components/functions. In this component/function consists of three main sub-components. Analyzing the category type of disease, Identifying the images of Ayurveda recipes/posts using OCR techniques, Analyzing Ayurveda recipes/posts using text similarity are the subcomponents of this component. There are some drop down selections that are listed there to choose the relevant characteristics of the herbal remedies or plants. Then it will predict the relevant disease which is used for curving. If it irrelevant then it will predict it will not be sufficient to curve the disease with an appropriate message. This component was developed using machine learning concepts. It will be tested by different models for getting better accuracy. Support vector machine, KNN, Bagging, RF and ET are the models that are used for the getting testing accuracy. The sub-components of Identifying the Image texts using OCR techniques and checking the Ayurveda related post using text similarity are other features adding to this App **Arogya.** Image processing and cosine similarity were used for implementing these functions in better way.

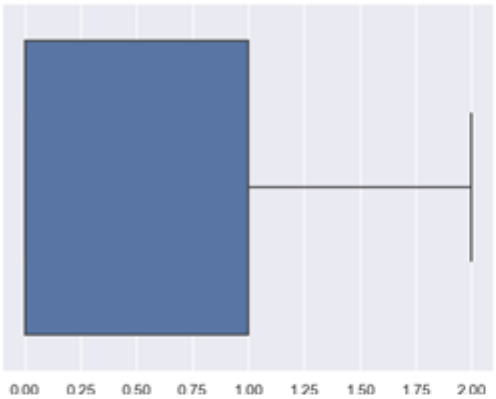
3.4.2.1 Predicting the results using dropdown selection – Identify the Disease

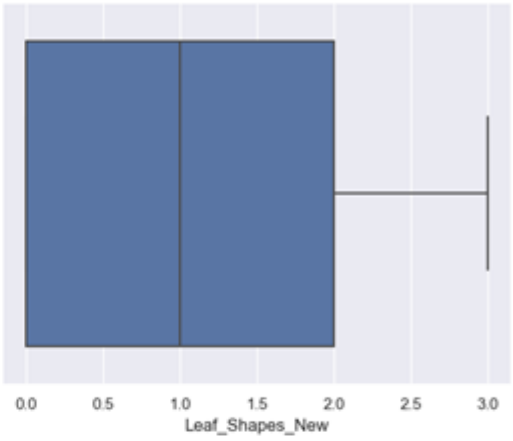
```
In [8]: runfile('C:/Users/Nadee/Desktop/New folder/4 Plant Prediction/train.py', wdir='C:/Users/Nadee/Desktop/New
folder/4 Plant Prediction')
>Predicted=diabetics

In [9]:
```

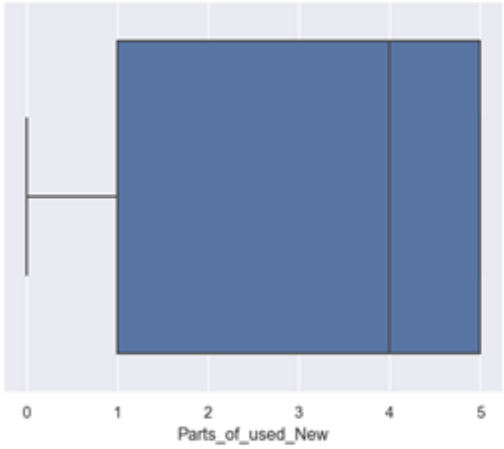
Predicting result of the component of the identify the disease type

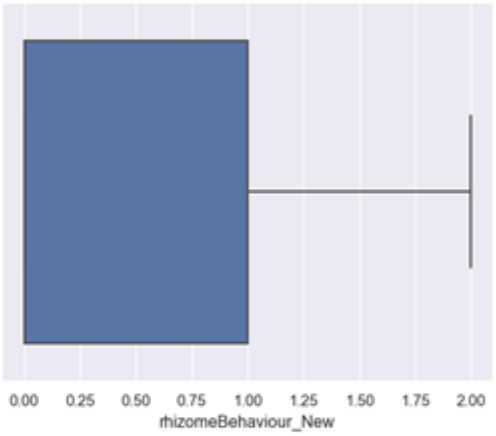
3.4.2.1.2 Visualization Graphs

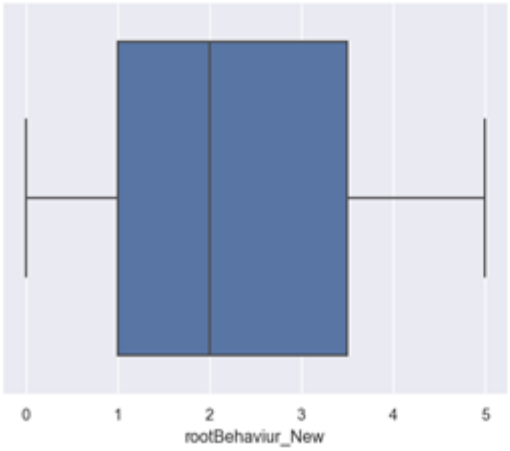


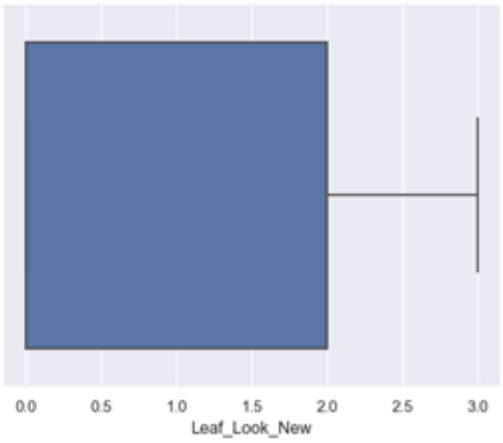

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


0.0
0.5
1.0
1.5
2.0
2.5
3.0
Leaf_Shapes_New


0
1
2
3
4
5
Parts_of_used_New


0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
rhizomeBehaviour_New


0
1
2
3
4
5
rootBehaviur_New


0.0
0.5
1.0
1.5
2.0
2.5
3.0
Leaf_Look_New

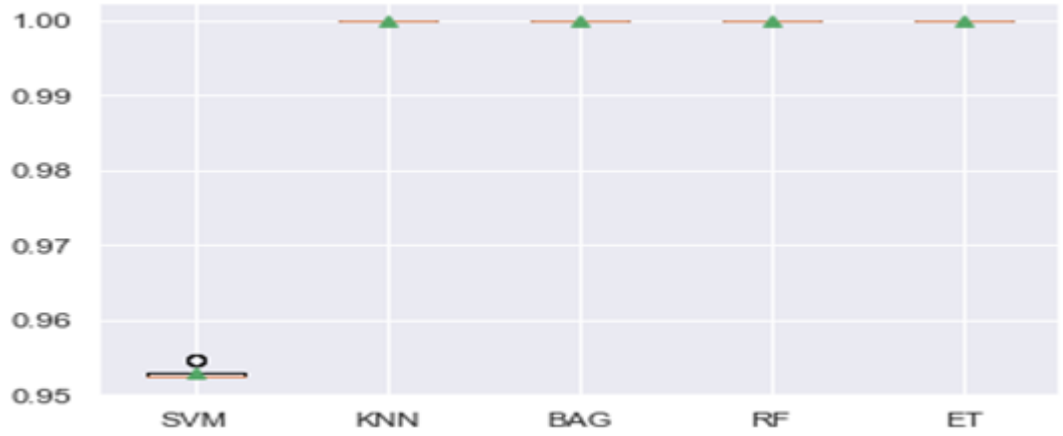
3.4.2.1.3 Testing for the all the models

Using Support vector machine, KNN, BAG, RF and ET are the models which are used for the testing.

```
In [9]: runfile('C:/Users/Nadee/Desktop/New folder/4 Plant Prediction/testing for all models.py', wdir='C:/Users/Nadee/
Desktop/New folder/4 Plant Prediction')
>SVM 0.953 (0.001)
>KNN 1.000 (0.000)
>BAG 1.000 (0.000)
>RF 1.000 (0.000)
>ET 1.000 (0.000)
```

Testing results of the all the models

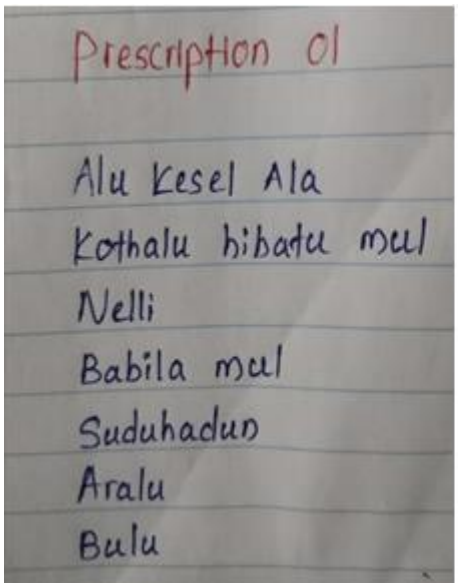
3.4.2.1.4 Accuracy Graph for the comparison between testing models



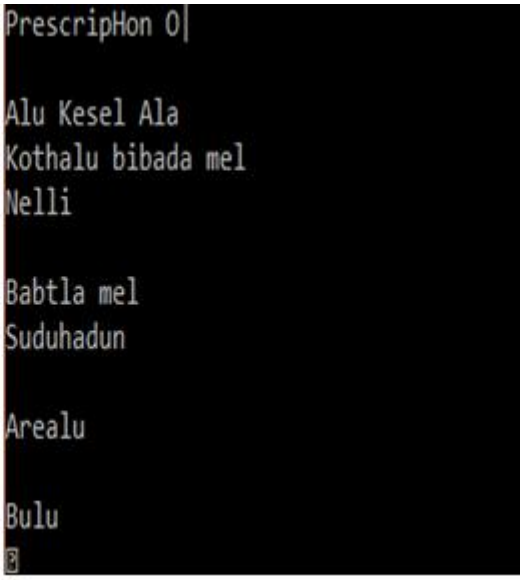
Accuracy graphs for the comparison between testing models

3.4.2.2 Identifying the results of Image OCR techniques

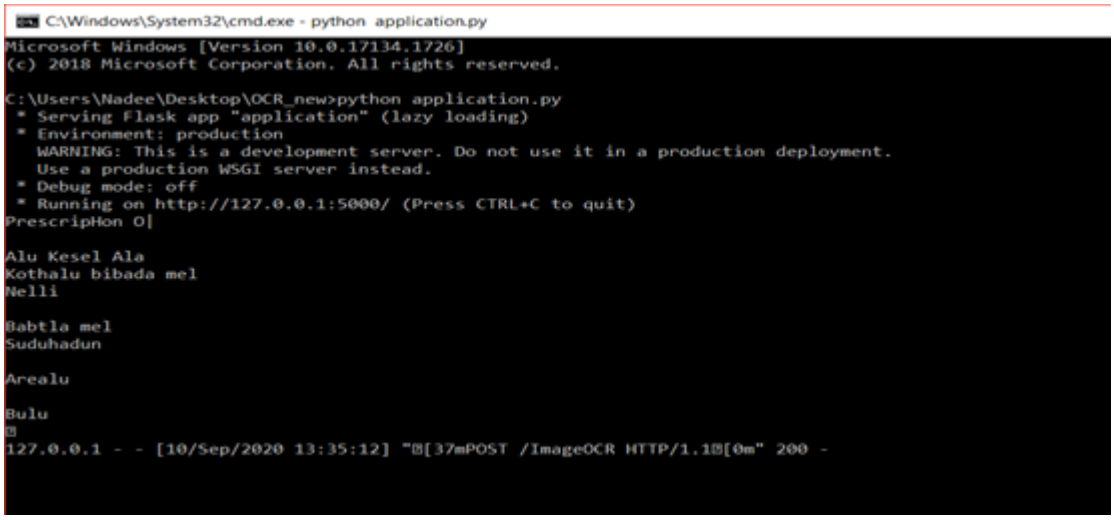
This is the Image of the prescription/recipe of the Ayurvedic medicine in figure and the results of the output in figure



Handwriting image

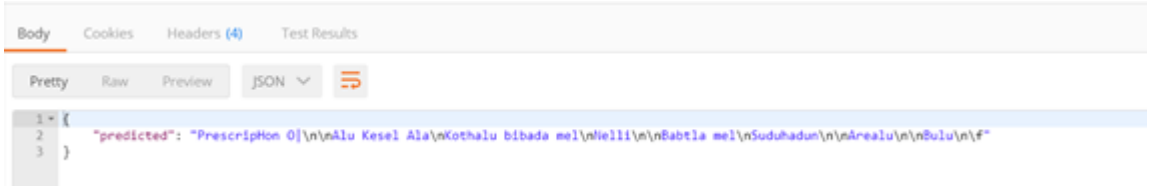


OCR output results



Results display in the Command prompt

This is the output result of the Image OCR using through the postman checking.



Postman checking the results of OCR

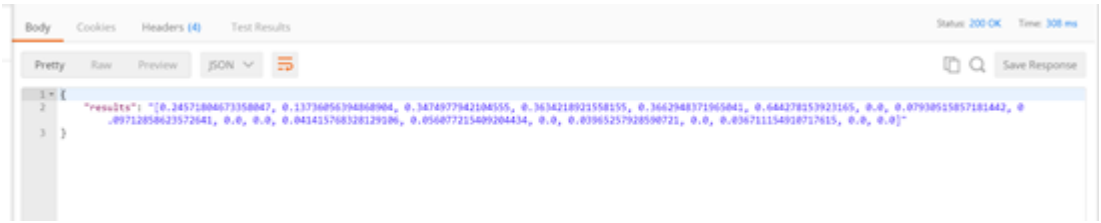
3.5.4.3 Identifying the results of text similarity

Text Description 01:

{

"text": "This recipe is for orthopedic in Ayurveda treatment. Treatments For wrist fracture. Needed Ayurveda remedies are nelli Arallu Bullu Kohobha pothu Sawandara mul.Ayurveda is considered by many scholars to be the oldest healing science. In Sanskrit Ayurveda means The science of life Ayurvedic knowledge. originated in India more than 5000 years ago and is often called the Mother of Healing It stems from the ancient Vedic culture and was taught for many thousands of years in an oral tradition from accomplished masters to their disciples."

}



The output results of the text including Ayurveda related text

3.5.Research Findings and Discussions

Selecting the Most Accurate and the Highest Performance Segmentation Technique using Experimental Outcomes of Plants in Sri Lanka

This document proposes an innovative solution to help the Arogya users to find out the plant most accurately by simply uploading the picture of the plant. When doing the research many irrelevant topics have to be examined and filtered before the real research could be done. But all this relevant content had to be read over and over again for it to be made clear and to be used in the design stage of the development.

When developing object detection component, it was very difficult for us to find a proper objects detection technique at a glance for our proposed system because there are number of detection techniques are available. Therefore, to get a better outcome which means to apply a better detection model to our proposed system, we did some experiment with selected models. Comparison result of them is given above section. And finally select the most suitable model for our proposed mobile application.

Identification and Classification of Ayurvedic Plants in Sri Lanka Using A Mobile Application in An Offline Environment Approach

The main objective of this individual research component was to analyze several deep CNN models with the acquired training and testing data from scratch, get the final testing accuracy from each in order to compare and achieve the model with the highest accuracy, and then to use it as the finalized model in the herbal plant classification purpose in Arogya, based on images as the input from the mobile camera module.

CNN Architecture	Finalized testing accuracy
InceptionV3	56.76%
MobileNetV2	63.58%
InceptionResNetV2	82..77%
Xception	86.01%
DenseNet121	89.12%
ResNet50	98.92%
VGG16	**99.53%**

Table 3.2.1: Comparison of accuracies for the selected CNN models in herbal plant classification

Hence, according to the comparison, the pre-trained model **VGG16** with a testing accuracy of **99.53%** was chosen as the best model with the highest accuracy. The results were highly promising, reaching over 99% accuracy using the VGG16 model. So, this was the finalized model to be deployed in Arogya mobile application in order to classify the selected Ayurveda leaves/roots/fruits by the user.

The following table illustrates the experimental classification accuracies obtained for each plant class wise when the selected VGG16 model was applied for classification.

Table 3.2.2: Experimental classification accuracy class wise

Plant Class	Classified amount	Misclassified amount	Accuracy
Akkapana leaf	40	0	100%
Cinnamon leaf	36	4	90%
Katupila leaf with fruit	36	4	90%
Turmeric leaf	38	2	95%
Turmeric digested root	40	0	100%
Kohomba	40	0	100%
Average			**95.83%**

So, it is obvious with the experimental results that, the reliability of this research function can guarantee **95% in average**.

Abstractive web page Information Summarization and Location Mapping with GIS technology on Ayurvedic Plants in Sri Lanka Using a Mobile Application in an Online Environmental Approach

Our final solution **Arogya** is a mobile application, which includes all four components in a single point. It handles Object detection, Object Classification, Information Summarization and Geographical Location Mapping.

Abstractive Text Summarization

The main research finding is based on Generating Summaries of the Herbal plants, which have been classified in the early stage by the object classifier. Information has been extracted from the internet and processed in order to get a productive output. Arogya is capable of preparing dynamic summaries on multiple web pages. So, to do that, information has been extracted from the first three top ranked web pages and combined all of them together by the system. Then starts the internal process of cleaning the content by removing all the unnecessary duplicates and passing back to the summarizer component to handle the rest of the process.

The task which was assigned was not easy. It was somewhat difficult to enter into the main flow of the function.

Figure 3.2. 1:Abstractive Text Summarization

Can search for a random herbal plant

Apart from generating summaries on classified herbal plants, in Arogya there is an option for searching random herbal plants according to the user's preference.

User has to type the plant name and search for it. Then the system will be generated a summary report on it as well

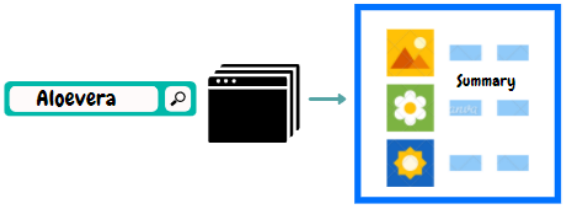


Figure 3.2. 2: Search for Random Plants

Summary Comparison with Multiple models

And also, there is a function implemented to compare the generated summaries with four main models SpaCy, NLTK, Sumy, and Gensim. So that user would gat a chance to choose the best summary document which they would prefer to go with as a solution for their prevailing requirement, based on Herbal plants information.

Ayurvedic Plants' Location tracking and mapping with GIS technology

The Geographical Information System, which we have implemented in the proposed system Arogya, mainly focuses on the locations of the Ayurvedic plants all over Sri Lanka. Whenever a user identifies an ayurvedic plant, the system is capable of tracking the current location and mapping it to the Map view. And also, they can search for a particular ayurvedic plant and find the areas which grow an abundance.

Figure 3.2. 3: Location Tracking and GIS Technology

User Account Login and Registration

The Mobile Application is well organized with a highly secured authentication system. So, the customers' data will be secured. The system is occupied with the Firebase Backend. Since the system is implemented with multiple user-level authentications, every user of the system can maintain their information separately.

There are three main **Login** Options available in the mobile application.

1. Regular user Login with Email and Passwords

   It is just the normal procedure user has to follow in order to login to the system. First of all, user has to Sign up in to the system by providing necessary information, email, and password as the credentials, then will be prompted to Login Interface.

2. Using Google - Gmail Login credentials

   Here, user will be able to create a new account by using their existing google Gmail account credentials.

3. Using Anonymous Login option

   Anonymous user login is especially available for random users, who do not want to maintain separate accounts for themselves. They just login to the system as outsiders and get the provided service by the system.

   As an example,

   Herbal Plants Classification is the main service provided by Arogya

   So, the anonymous user login to the system and just get classified some random plants whether it is an herbal plant or not.

   The user doesn't want to keep records after adding them to the database.

   They need only to identify the random plant, that's it.

Maintaining the Herbal Plants records in the system

If the classified plant is an Ayurvedic plant, and if it is very helpful for the user, they can add it to their account and keep a record of it. Later they can update the record or delete if it is not needed further.

1. Insert Herbal Plants
2. Update Plants
3. Delete Plants
4. Display Plants as a list view

3.5.4. Text classification of Herbal plants and predict the category type of diseases and Identifying the Ayurveda image texts using OCR techniques and ayurveda related posts/recipes using cosine text similarity.

As considering the component of analyzing the disease type of herbal remedies function was implemented using Machine learning algorithms. Using support vector machines, BF, KNN and are the models that are used for getting test accuracy. Some models were not giving better accuracy. Some models were giving overfitting issues because of the smaller number of records. We decided to focus on five types of herbal plants such as Iguru, Hathawariya, Ardathoda, Polpala and Gotukola. Due to that reason, gathering information of its characteristics its being limited with a smaller number of records. So, overfitting issues happened there. Removing duplicates records and null values records cause to reduce the number of records in the dataset preliminary. Hence considering the Support vector machine (SVM) model that gave the best accuracy percentage as 95%. Thus, used this model as the best accuracy model for training well. Analyzing the image text using OCR techniques, here identified the texts which were included in the image. Converting to the pure black and white and applying thresholding the image is the preprocessing stage of OCR identification. Median blurring should be done to remove noise and Identify the image text. Some handwriting words are not clearly identified as separated as characters. Due to that reason, it will not be predicted with higher accuracy also. Thus, currently having a medium accuracy level of this function. Identifying the text similarity is based to implement on Cosine text similarity. Firstly, tokenization and corpus functions are for preprocessing to separate into segments as tokens and keep separate into common stop words. After that, remove the stop words and create a vector. Then apply the cosine algorithm and calculate the similarity. Here 0.4 and above 0.4 is related to Ayurveda related posts/recipes and other similarity values are not related to Ayurveda. There is a problem regarding the accuracy level of this function also. Currently it has medium accuracy level. Taking more words and getting some higher accuracy is the main point

to research in the next step also. Finally, Arogya will give special options as the social media app to outside.

3.6. DISCUSSION

There is some research as well as mobile applications already available right now which classify plant leaves and give low prediction accuracies. However, these applications are based on foreign leaves, but not valuable medicinal plants or shrubs. So, Arogya mobile application would be a huge resource for almost everybody who keen to experience Ayurveda in their lives. Not only leaves, but also fruits, flowers and roots associated with Ayurveda will also be able to be classified with this application with a promising accuracy. This would be a great benefit for all the users since in Ayurveda not only leaves are used for preparation of medicine, but also the other parts of the plant are also used often for that purpose. In addition, the development strategy and methodology used in this approach will be able to be used and extended to identify any ayurvedic herb worldwide furthermore.

From the generated results of the object detection techniques, proposed marker based watershed algorithm perform well for our proposed system. When compare YOLO v3 and YOLO v5, YOLO v5 performs well. According to the result, YOLO v5 works well even if the image has complex background rather than YOLO v3. And also to train the dataset, YOLO V5 takes less time than YOLO v3.Thus, YOLO v5 is superfast than YOLO v3.However, YOLO has some drawbacks. Sometimes, it struggles with small objects that appear in groups, it struggles to generalize to objects in new or unusual aspect ratios or configurations, to identify the object accurately, and at least it needs 1000 images per category. Thus, we need to annotate large datasets, and need to train the dataset and it takes time. Sometimes, it doesn't work on unseen data.

Main purpose of the component is to detect the object (leaf) in real time even if it is in a complex background. According to the results, proposed marker-based watershed segmentation provided several advantages such as it required low computation time, and provided closed contours, the boundaries of the resulting regions always correspond to contours which appear in the image as obvious contours of objects, fast simple, no need to train the dataset .Thus no need to annotate data and saves time as well. This proposed algorithm works well in our application as described and this is used as the finalized model.

The component, Summary Generation on Herbal Plants and Geographical Location Mapping was started by analyzing the existing systems which were related to this and it was not an easy path to get here from the beginning. Several algorithms were studied and evaluated in order to get the best

results. Sometimes, had to spend more time on testing the product quality to get the best results rather than concentrating on further implementation. As the first step, an effective dataset should be prepared to train the model. To do that, website information was scrapped. At the beginning, even though the scraping information from a single webpage was manageable, extracting information from multiple websites with the help of python libraries was not possible. Even, there were many ups and downs throughout the project life span, finally we were able to cover the project scope somehow. Sometimes, to provide an ideal unique solution for its consumers, we had to do many great commitments. The application has been successfully completed along with its main functionalities. Finally, it was able to fulfil the consumers' requirements on Ayurvedic Plants.

As considering the component of analyzing the disease type of herbal remedies function was implemented using Machine learning algorithms. Using support vector machine, BF, KNN and are the models that are used for the getting test accuracy. Some models were not giving better accuracy. Some models were giving overfitting issues because of the smaller number of records. We decided to focus on five type of herbal plants as Iguru, Hathawariya, Ardathoda, Polpala and Gotukola. Due to that reason, gathering information of its characteristics its being limited with a smaller number of records. So that, overfitting issues were happened there. Removing duplicates records and null values records cause to reduce the number of records in dataset preliminary. Hence considering the Support vector machine (SVM) model that gave the best accuracy percentage as 95%. Thus, used this model as the best accuracy model for trained well. Analyzing the image text using OCR techniques, here identified the texts which included in the image. Converting to the pure black and white and apply thresholding the image is the preprocessing stage of OCR identification. Median blurring should be done to remove noise and Identify the image text. Some handwriting words are not clearly identified as separated as characters. Due to that reason, it will not be predicted the higher accuracy also. Thus, currently having a medium accuracy level of this function. Identifying the text similarity is based to implement on Cosine text similarity. Firstly, tokenization and corpus functions are having to do for preprocessing to separating into segments as tokens and keep separate into common stop words. After that, remove the stop words and create a vector. Then apply the cosine algorithm and calculate the similarity. In here 0.4 and above 0.4 is related to Ayurveda related posts/recipes and other similarity values are not related to Ayurveda. There is a problem regarding the accuracy level of this function also. Currently it has medium accuracy level. Taken more words and getting some higher accuracy is the main point to research in next step also. Finally, Arogya will give special options as the social media app to outside.

4. CONTRIBUTION

Table 4 1: Summary of Group Members' Contribution

IT17129404	N.J. Pathiranage
Functionality	Description
Classification of Ayurvedic plants using transfer learning based on deep CNN in an offline approach	● Data collection for 5 different types of herbal plants in Sri Lanka ● Annotate and label them to form 6 classes for classification ● Apply transfer learning based on deep CNN on the prepared dataset ● Do comparison with 7 variants of CNN in order to obtain the CNN with highest accuracy ● Do model comparison with all of them separately and obtain accuracies ● Retrain the selected best model with more data using data augmentation and hyper parameter tuning ● Optimizing the best model and getting the promising accuracy ● Testing using the best model (unit and integration testing) ● Deployment of the finalized model in Azure for connecting with the front end
User Login & Registration	● Implementation of User Levels Authentication with Firebase ● Multiple ways of Login methods ● Implementation of mobile client and connecting with the backend API deployed in cloud

Table 4 2: Summary of Group Members' Contribution

IT17145930	M.S.F. Nilfa
Functionality	Description
Summary Generation on Ayurvedic Herbal Plants	● Collect and prepare the dataset for training the model ● Single webpage extraction and multiple webpage extraction ● Cleaning the dataset by removing the redundant values ● Summary comparison with three other models
Geographical Location Mapping	● Using google Map API to Live Location tracking ● If the found plant is classified as an herbal plant, Add the current Location address for the Map as it is the plant found location.
User Login & Registration	● Implementation of User Levels Authentication with Firebase ● Multiple ways of Login methods
User Profile Maintenance	● Create, Edit, Delete, and View functions implementation on ayurvedic plants

IT17089500	R.A.M.Nithmali
Functionality	Description
Plant part detection in a complex background	● Collect and prepare the dataset for training the model ● Select the best accurate and the performance object detection technique by comparing the several object detection techniques.

CONCLUSION

The purpose of this research is to develop a centralized social media application which is mobile-based and unique to herbal plants. This solution would be a great chance for those who are keen to learn and use Ayurveda medicine and plants, but who do not have prior knowledge about the specific domain. The application mainly creates awareness among common people about Ayurveda plants, their medicinal usage and value, and about their growth and availability throughout the country. This proposed system has been tested in various situations and it is capable of providing the most reliable and accurate output to the user. According to the main research components focused, it has been experimentally proved in this research that the most suitable algorithm for plant detection based on our dataset prepared from scratch is Marker-based Watershed algorithm, and the most accurate CNN pre-trained model used for classification purpose is VGG-16, which achieved a testing accuracy of 99.53%. The results are highly promising, reaching over 99% accuracy using the VGG-16 model. Additionally, the Seq2Seq LSTM model which is a deep-learning model has been proved as the best model with optimum accuracy in the purpose of abstractive information summarization. This application is currently built only in English. Further, this can be applied in native languages such as Sinhala and Tamil. Since the system is designed only as a mobile application, later can be improvised to a web application with the same functionality and content. Additionally, the development strategy and methodology used in this approach will be able to be used and extended to identify any ayurvedic herb worldwide furthermore, and if the user ends up with doubts and clarifications regarding this procedure, this application can be facilitated with consultation help from Ayurveda doctors.

Future Work

We expect to expand this application according to some other consumer’s needs too. So that, will be able to increase the usability and productivity of this Application for further extent. And also, it is required to increase the accuracy of the summaries and to keep the content more concise to the point. Current model Seq2Seq is capable of predicting the next word, but sometimes the accuracy is less, because of the size of the dataset. Hence, if it is possible to collect more data as the training dataset on this project, it will direct this project into a successful direction with more accurate results.

REFERENCES

[1] Pathirage Kamal Perera, "Current scenario of herbal medicine in Sri Lanka", ASSOCHAM, 4th annual Herbal International Summit cum Exhibition on Medicinal & Aromatic Products, Spices and finished products(hi-MAPS), pp. 1-3, April 2012.

[2] Lakpura LLC, Ayurveda. Accessed on: February 15,2020. [Online] Available: https://lanka.com/about/interests/ayurveda/

[3] Y. R. Azeez, R. A. C. P. Rajapakse, "An application of image processing techniques in identifying herbal plants in Sri Lanka", Proceedings of the 3rd. International Research Symposium on Pure and Applied Sciences, pp.1, 26th October 2018.

[4] Ayurveda Bansko, Blog, Benefits of Ayurveda and the difference to the traditional western medicine, 2015. Accessed on: February 15,2020. [Online] Available: https://www.ayurvedabansko.com/benefits-ayurveda-and-difference-to-western-traditional-medicine/.

[5] Basavaraj S. Anami, Suvarna S. Nandyal, A. Govardhan, "A Combined Color, Texture and Edge Features Based Approach for Identification and Classification of Indian Medicinal Plants", International Journal of Computer Applications (0975 – 8887), Volume 6– No.12, pp.1-3, September 2010.

[6] Robert G. de Luna, Renann G. Baldovino, Ezekiel A. Cotoco, Anton Louise P. de Ocampo, Ira C. Valenzuela, Alvin B. Culaba, Elmer P. Dadios, "Identification of Philippine Herbal Medicine Plant Leaf Using Artificial Neural Network", IEEE, 978-1-5386-0912-5/17/$31.00,PP.1-2, 2017.

[7] Rademaker C. A. 2000, "The classification of plants in the United States Patent Classification system", World Patent Information 22: 301–307.

[8] Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John Kress, Ida C Lopez, João VB Soares, "Leafsnap: A computer vision system for automatic plant species identification," ECCV, pp. 502–516. Springer, 2012.

[9] Cem Kalyoncu, Önsen Toygar, "Geometric leaf classification," Computer Vision and Image Understanding", vol. 133, pp. 102–109, 2014.

[10] Abdul Kadir, Lukito Edi Nugroho, Adhi Susanto, Paulus Insap Santosa, "Leaf classification using shape, color, and texture features," arXiv preprint arXiv:1401.4447, 2013.

[11] Thibaut Beghin, James S Cope, Paolo Remagnino, SarahBarman, "Shape and texture based plant leaf classification," Advanced Concepts for Intelligent Vision Systems, 2010, pp. 345–353.

[12] James Charters, Zhiyong Wang, Zheru Chi, Ah Chung Tsoi, David Dagan Feng, "Eagle: A novel descriptor for identifying plant species using leaf lamina vascular features," ICME-Workshop, 2014, pp. 1–6.

[13] Heysem KAYA, Ilhan KEKLIK, Tolga ENSARI, Fatih ALKAN, Yagmur BIRICIK, "Oak Leaf Classification: An Analysis of Features and Classifiers", 978-1-7281-1013-4/19/$31.00, pp.1-4, 2019.

[14] James S Cope, Paolo Remagnino, Sarah Barman, Paul Wilkin, “The extraction of venation from leaf images by evolved vein classifiers and ant colony algorithms,” Advanced Concepts for Intelligent Vision Systems, pp. 135–144, 2010.

[15] Muammer TORKOGLU, Davut HANBAY, “Combination of Deep Features and KNN Algorithm for Classification of Leaf-Based Plant Species”.

[16] Jing Hu, Zhibo Chen, Meng Yang, Rongguo Zhang, Yaji Cui, “A Multiscale Fusion Convolutional Neural Network for Plant Leaf Recognition”, 1070-9908, IEEE SIGNAL PROCESSING LETTERS, VOL. 25, NO. 6, JUNE 2018.

[17] Agnieszka Mikołajczyk, Michał Grochowski, “Data augmentation for improving deep learning in image classification problem”, 17859931, ISBN: 978-1-5386-6143-7, May 2018.

[18] Will Koehrsen, Medium, towards data science, Transfer Learning with Convolutional Neural Networks in PyTorch, November 27, 2018. Accessed on: February 15,2020. [Online]
Available: https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce.

[19] Inside.Techlabs, Leaf Classification Project, October 23, 2018, Accessed on: February 15,2020. [Online]
Available:https://medium.com/@inside.techlabs/leaf-classification-project-89bb5c7577f3.

[20] Dileep M.R, Pournami P.N, “AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants”, IEEE, 978-1-7281-1895-6/19/$31.00, pp.1-4, 2019.

[21] Jun Woo Lee, Yeo Chan Yoon, "Fine-Grained Plant Identification using wide and deep learning model", Institute for Information & communications Technology Promotion (IITP), No. 2016-0-00010-003, Digital Content In-House R&D.

[22] Benjaphan Sommana, Thanaruk Theeramunkong, “Improving Plant Recognition using Hybrid features from Connectionist and Knowledge-Based Approaches”, IEEE, 978-1-7281-0161-3/18/$31.00, 2018.

[23] Hulya Yalcin, Salar Razavi, “Plant Classification using Convolutional Neural Networks”, Visual Intelligence Laboratory Electronics & Telecommunication Engineering Department Istanbul Technical University, 2017.

[24] Jyotismita Chaki, Ranjan Parekh, Samar Bhattacharya, “Recognition of Whole and Deformed Plant Leaves using Statistical Shape Features and Neuro-Fuzzy Classifier”, IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS), 978-1-4799-8349-0/15/$31.00, 2015.

[25] Chaoyun Zhang, Pan Zhou, Chenghua Li, Lijun Liu, “A Convolutional Neural Network for Leaves Recognition Using Data Augmentation”, IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.318, 978-1-5090-0154-5/15 $31.00, 2015.

[26] Sue Han Lee, Paul Wilkin, Chee Seng Chan, Paolo Remagnino, “DEEP-PLANT: PLANT IDENTIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS”, IEEE, 978-1-4799-8339-1/15/$31.00, 2015.

[27] ArunPriya C, Balasaravanan T, Antony Selvadoss Thanamani, “An Efficient Leaf Recognition Algorithm for Plant Classification Using Support Vector Machine”, Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering, 978-1-4673-1039-0/12/$31.00, 2012, March 21-23, 2012.

[28] Shanwen Zhang, YouQian Feng, “Plant Leaf Classification Using Plant Leaves Based on Rough Set”, 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 978-1-4244-7237-6/10/$26.00, 2010.

[29] Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang, Qiao-Liang Xiang, “A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network”, IEEE International Symposium on Signal Processing and Information Technology, 978-1 -4244-1 835-0/07/$25.00 , 2007.

[30] Pushpa BR, Anand C, Mithun Nambiar P, “Ayurvedic Plant Species Recognition using Statistical Parameters on Leaf Images”, International Journal of Applied Engineering Research, ISSN 0973-4562, Volume 11, Number 7 (2016) pp 5142-5147.

[31] Sethulekshmi A V, Sreekumar K, “Ayurvedic leaf recognition for Plant Classification”, Sethulekshmi A V et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6), ISSN:0975-9646, 2014, 8061-8066.

[32] A Gopal, S. Prudhveeswar Reddy, V. Gayatri, “Classification of Selected Medicinal Plants Leaf Using Image Processing”, IEEE, 978-1-4673-2322-2112/$31.00, 2012.

[33] J. W. Tan, S. Chang, S. Binti Abdul Kareem, H. J. Yap, K. Yong, “Deep learning for plant species classification using leaf vein morphometric”, pages 1–1, 2018.

[34] R. Janani, A. Gopal, “Identification of selected medicinal plant leaves using image features and ann”, 2013 International Conference on Advanced Electronic Systems (ICAES), pages 238–242, Sept 2013.

[35] Gavai, N. R., Jakhade, Y. A., Tribhuvan, S. A., & Bhattad, R. (2017, December). MobileNets for flower classification using TensorFlow. In 2017 International Conference on Big Data, IoT and Data Science (BID) (pp. 154-158). IEEE.

[36] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).

[37] MathWorks, Deep Learning. Accessed on: February 19,2020. [Online] Available: https://www.mathworks.com/discovery/deep-learning.html.

[38] Machine Learning Mastery, How to Improve Performance With Transfer Learning for Deep Learning Neural Networks. Accessed on: February 19,2020. [Online] Available:https://machinelearningmastery.com/how-to-improve-performance-with-transfer-learning-for-deep-learning-neural-networks/.

[39] TechTarget, Convolutional Neural Network. Accessed on: February 19, 2020. [Online]

Available:https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network

[40] Hervé Goëau, Pierre Bonnet, Alexis Joly, Julien Barbe, Souheil Selmi, Vera Baki´c, Jennifer Carré, Daniel Barthelemy, Nozha Boujemaa, "Pl@ntNet Mobile App", Proceedings of the 21st ACM international conference on Multimedia, DOI: 10.1145/2502081.2502251, ppt. 1-2, October 2013.

[41] Neeraj kumar Kumar, Peter N. Belhumeur, Arijit Biswas, João V. B. Soares, "Leafsnap: A Computer Vision System for Automatic Plant Species Identification", European Conference on Computer Vision, DOI:10.1007/978-3-642-33709-3_36, ppt. 1, October 2012.

[42] Desta Sandya, Yeni Herdiyeni, "MedLeaf: Mobile Application For Medicinal Plant Identification Based on Leaf Image", International Journal on Advanced Science, Engineering and Information Technology, Vol. 3 (2013) No. 2 ISSN: 2088-5334, ppt 1-3, May 2013.

[43] Lin-Hai Ma, Zhong-Qiu Zhao, Jing Wang, "ApLeafis: An Android-Based Plant Leaf Identification System", International Conference on Intelligent Computing, LNCS 7995, pp. 106–111, 2013.

[44] PredictiveAnalyticsWorld.com, 3 Ways to Test the Accuracy of Your Predictive Models. Accessed on: February 22, 2020. [Online] Available:http://https//www.predictiveanalyticsworld.com/machinelearningtimes/3-ways-test-accuracy-%20predictive-models/3295/

[45] Machine Learning Mastery, What is the Difference between Test and Validation Datasets? Accessed on: February 22, 2020. [Online] Available: https://machinelearningmastery.com/difference-test-validation-datasets/

[46] ResearchGate, What is training and testing data in machine learning? Accessed on: February 22, 2020. [Online] Available:https://www.researchgate.net/post/What_is_training_and_testing_data_in_machine_learning

[47] H. P. Edmundson, "New Methods in Automatic Extracting," *J. ACM*, 1969, doi: 10.1145/321510.321519.

[48] J. Kupiec, J. Pedersen, and F. Chen, "Trainable document summarizer," 1995, doi: 10.1145/215206.215333.

[49] N. Ramanujam and M. Kaliappan, "An automatic multidocument text summarization approach based on naïve Bayesian classifier using timestamp strategy," *Sci. World J.*, 2016, doi: 10.1155/2016/1784827.

[50] A. Leoncini, F. Sangiacomo, P. Gastaldo, and R. Zunino, "A Semantic-Based Framework for Summarization and Page Segmentation in Web Mining," in *Theory and Applications*

for Advanced Text Mining, 2012.

[51] M. Allahyari *et al.*, "Text Summarization Techniques: A Brief Survey," *Int. J. Adv. Comput. Sci. Appl.*, 2017, doi: 10.14569/ijacsa.2017.081052.

[52] S. S. Lakshmi and M. U. Rani, "Multi-Document Text Summarization Using Deep Learning Algorithm with Fuzzy Logic," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.3165331.

[53] A. K. Singh and M. Shashi, "Deep Learning Architecture for Multi-Document Summarization as a cascade of Abstractive and Extractive Summarization approaches," *Int. J. Comput. Sci. Eng.*, 2019, doi: 10.26438/ijcse/v7i3.950954.

[54] Y. M. Shaalan and A. Rafea, "Frequently asked questions web pages automatic text summarization," 2011, doi: 10.2316/P.2011.717-095.

[55] Y. Zhang, N. Zincir-Heywood, and E. Milios, "World Wide Web site summarization," *Web Intelligence and Agent Systems*. 2004.

[56] N. Kumar *et al.*, "Leafsnap: A computer vision system for automatic plant species identification," 2012, doi: 10.1007/978-3-642-33709-3_36.

[57] S. A. Babar and P. D. Patil, "Improving performance of text summarization," 2015, doi: 10.1016/j.procs.2015.02.031.

[58] S. P. Singh, A. Kumar, A. Mangal, and S. Singhal, "Bilingual automatic text summarization using unsupervised deep learning," 2016, doi: 10.1109/ICEEOT.2016.7754874.

[59] H. A. Chopade and M. Narvekar, "Hybrid auto text summarization using deep neural network and fuzzy logic system," 2018, doi: 10.1109/ICICI.2017.8365192.

[60] Y. R. Azeez and C. Rajapakse, "An Application of Transfer Learning Techniques in Identifying Herbal Plants in Sri Lanka," 2019, doi: 10.23919/SCSE.2019.8842681.

[61] R. G. De Luna *et al.*, "Identification of philippine herbal medicine plant leaf using artificial neural network," 2017, doi: 10.1109/HNICEM.2017.8269470.

[62] H. Kaya, I. Keklik, T. Ensari, F. Alkan, and Y. Biricik, "Oak leaf classification: An analysis of features and classifiers," 2019, doi: 10.1109/EBBT.2019.8742053.

[63] Pimm, S. L., Jenkins, C. N., Abell, R., Brooks, T. M., Gittleman, J. L., Joppa, L. N., Raven, P. H., Roberts, C. M., & Sexton, J. O. (2014). The biodiversity of species and their rates of extinction, distribution, and protection. *Science*, *344*(6187). https://doi.org/10.1126/science.1246752

[64] Joly, A., Müller, H., Goëau, H., Glotin, H., Spampinato, C., Rauber, A., Bonnet, P., Vellinga, W. P., & Fisher, B. (2014). LifeCLEF: Multimedia life species identification. *CEUR Workshop Proceedings*, *1222*(March), 7–13.

[65] De Luna, R. G., Baldovino, R. G., Cotoco, E. A., De Ocampo, A. L. P., Valenzuela, I. C., Culaba, A. B., & Gokongwei, E. P. D. (2017). Identification of philippine herbal medicine plant leaf using artificial neural network. *HNICEM 2017 - 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*, *2018-January*, 1–8. https://doi.org/10.1109/HNICEM.2017.8269470

[66] Anami, B. S., Nandyal, S. S., & Govardhan, A. (2010). A Combined Color, Texture and Edge Features Based Approach for Identification and Classification of Indian Medicinal Plants. *International Journal of Computer Applications*, *6*(12), 45–51. https://doi.org/10.5120/1122-1471

[67] Jamil, N., Hussin, N. A. C., Nordin, S., & Awang, K. (2015). Automatic Plant Identification: Is Shape the Key Feature? *Procedia Computer Science*, *76*(Iris), 436–442. https://doi.org/10.1016/j.procs.2015.12.287

[68]https://www.analyticsvidhya.com/blog/2019/04/introductionimagesegmentationtechniques-python/

[69] Singh, V. (2019). Sunflower leaf diseases detection using image segmentation based on particle swarm optimization. *Artificial Intelligence in Agriculture*, *3*, 62–68. https://doi.org/10.1016/j.aiia.2019.09.002

[70] De Luna, R. G., Baldovino, R. G., Cotoco, E. A., De Ocampo, A. L. P., Valenzuela, I. C., Culaba, A. B., & Gokongwei, E. P. D. (2017). Identification of philippine herbal medicine plant leaf using artificial neural network. *HNICEM 2017 - 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*, *2018-Janua*, 1–8. https://doi.org/10.1109/HNICEM.2017.8269470

[71] Ma, L. H., Zhao, Z. Q., & Wang, J. (2013). ApLeafis: An Android-based plant leaf identification system. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7995 LNCS*(61005007), 106–111. https://doi.org/10.1007/978-3-642-39479-9_13

[72] Prasvita, D. S., & Herdiyeni, Y. (2013). MedLeaf: Mobile Application for Medicinal Plant Identification Based on Leaf Image. *International Journal on Advanced Science, Engineering and Information Technology*, *3*(2), 103. https://doi.org/10.18517/ijaseit.3.2.287

[73] Barré, P., Stöver, B. C., Müller, K. F., & Steinhage, V. (2017). LeafNet: A computer vision system for automatic plant species identification. *Ecological Informatics*, *40*, 50–56. https://doi.org/10.1016/j.ecoinf.2017.05.005

[74] Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. http://arxiv.org/abs/1804.02767

[75] Pushpa, B. R., Anand, C., & Mithun Nambiar, P. (2016). 169Ayurvedic plant species recognition using statistical parameters on leaf images. *International Journal of Applied Engineering Research*, *11*(7), 5142–5147.

==========

[76]H. J. Gunathilaka, P. Vitharana, L. Udayanga, and N. Gunathilaka, “Assessment of Anxiety, Depression, Stress, and Associated Psychological Morbidities among Patients Receiving Ayurvedic Treatment for Different Health Issues: First Study from Sri Lanka,” BioMed Research International, vol. 2019, pp. 1–10, Nov. 2019, doi: 10.1155/2019/2940836.

[77]J.-R. Reichert, K. L. Kristensen, R. R. Mukkamala, and R. Vatrapu, “A supervised machine learning study of online discussion forums about type-2 diabetes,” in 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), 2017, doi: 10.1109/healthcom.2017.8210815.

[78] B. Ravishankar, “Journal of Ayurveda Medical Sciences – Peer Reviewed Journal for Rapid Publication of Ayurveda and Other Traditional Medicine Research,” Journal of Ayurveda Medical Sciences, vol. 1, no. 1, pp. 01–02, Oct. 2016.

[3] “Facts and Figures” [Online] Available: https://spacy.io/usage/facts-figures

[79] “Natural Language Processing (NLP) Techniques for Extracting Information” [Online] Available: https://www.searchtechnologies.com/blog/natural-language-processing-techniques

[80] “Text Analytics for beginners Using NLTK” [Online] Available: https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk

[81] “Machine Learning, NLP: Text Classification Using scikit-learn, python and NLTK” [Online] Available: https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikitlearn-python-and-nltk-c52b92a7c73a

[82] “Ayurveda modern medicine interface: A critical appraisal of studies of Ayurveda medicines to treat osteoarthritis and rheumatoid arthritis” [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3087360/

[83] Efficacy and Safety evaluation of Ayurvedic treatments (Ashwagandha power & sidh Makardhwaj) in rheumatoid arthritics patients: apilot prospective study [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4405924/

[84] S. Cakic, T. Popovic, S. Sandi, S. Krco, and A. Gazivoda, “The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing,” presented at the 2020 24th International Conference on Information Technology (IT), Feb. 2020, doi: 10.1109/it48810.2020.9070558.

[85] S. Thakare, A. Kamble, V. Thengne, and U. R. Kamble, “Document Segmentation and Language Translation Using Tesseract-OCR,” presented at the 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), Dec. 2018, doi: 10.1109/iciinfs.2018.8721372.

[86] M. R. M. Ribeiro, D. Julio, V. Abelha, A. Abelha, and J. Machado, “A Comparative Study of Optical Character Recognition in Health Information System,” presented at the 2019 International Conference in Engineering Applications (ICEA), Jul. 2019, doi: 10.1109/ceap.2019.8883448.

[87] S. Pattnaik and A. K. Nayak, “Summarization of Odia Text Document Using Cosine Similarity and Clustering,” presented at the 2019 International Conference on Applied Machine Learning (ICAML), May 2019, doi: 10.1109/icaml48257.2019.00035.

[88] Y. Liu, Q. Xu, and Z. Tang, “Research on Text Classification Method Based on PTF-IDF and Cosine Similarity,” presented at the 2019 4th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Nov. 2019, doi: 10.1109/iciibms46890.2019.8991542.
[14] S. Sohangir and D. Wang, “Document Understanding Using Improved Sqrt-Cosine Similarity,” presented at the 2017 IEEE 11th International Conference on Semantic Computing (ICSC), 2017, doi: 10.1109/icsc.2017.12.