ANALYZING THE CATEGORY OF HERB DISEASES AND IDENTIFIYING AYURVEDIC RELATED POST/RECIPES USING TEXT SIMILARITY AND OCR

Final Individual Thesis Report

K.A.G.Y. Nadee Kumari - IT17014250

B.Sc. (Hons) in Information Technology

Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2020

ANALYZING THE CATEGORY OF HERB DISEASES AND IDENTIFIYING AYURVEDIC RELATED POST/RECIPES USING TEXT SIMILARITY AND OCR

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

B.Sc. (Hons) in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2020

DECLARATION

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

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)."

Name	Student	Signature
K.A.G.Y. Nadee Kumari	IT17014250	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

…… ……

Signature of the supervisor: Date

(Mrs. Lokesha Weerasinghe)

ACKNOWLEDGEMENT

I would like to express our sincere gratitude to our project supervisor Ms. Lokesha Weerasinghe for providing her invaluable guidance, encouragement, and attention. her valuable feedback and time throughout the course of our research project for complete success.

Moreover, I would like to thank our lecture in-charge, Dr. Janaka weerakoon and CDAP team for providing the guidance to reach the successful ending of the project. Finally,

I would like to express our special thanks to friends who supported me throughout this project and especially grateful to our parents for their support and encouragement.

ABSTRACT

Ayurvedic means a science of Life well- being with its unique approaches of social and spiritual life is in practice science centuries in the Indian sub-content. Some treatments failed in the western medical approach, there are most of the people believe Ayurvedic approach is the best way of providing treatments because of having a less error prone of giving a feedback in any type of treatments and Although some of unknown diseases, bone fractured are curved in better condition way and it keeps in reducing much number of side effects. Usually, Ayurvedic medicine plants are used for treatments are undergone long period of time after taking the western medical approach. The proposed system will provide many features as the services of giving traditional treatments in Ayurvedic. As a special feature of this Arogya app is identifying the category type of disease in an unknown medicine plants giving its special characteristics/properties. After displaying the category, the system will display the most used medicine sample of each category and visualizing some of herbs that are included. Hence, there visualized images can be taken from the already used a database using in Image processing stage. In here, users can select dropdowns giving in the text area according to the herb's properties. If not mentioning relevant traits, typing field is given for including special herb characteristics within limited words. All the needed variables are stored in a relevant database. Although using some machine learning algorithm for analyzing the categories of each type of medicine plants. Feature extraction and filtering options should be done under the preprocessing stage around Natural Language processing (NLP). Analyzing the Ayurveda related post/recipes using text similarity and OCR techniques are other features included in this app Arogya. Nowadays, people mostly used day to day tasks depend on variety of technologies, therefore identifying the ingredients in the prescription medicine/ recipes and identify the different recipes which are uploaded by other users are very tedious task for people who are depend on technology to move with them familiarly. So that, this proposed system will support to predict most usable prescription for given category and visual appearance of its included ingredients and only uploaded Ayurveda related post/recipes on the wall.

Arogya App is the most important to find unknown medicine plants and identify the better solutions for health-issues neglected in western approaches and the long period of traditional ayurvedic treatments.

Keywords:
Machine Learning, Feature Extraction, Characteristics, Categories, Natural Language Processing, OCR, Text similarity, Ayurveda recipes

Table of Contents

List of figures

List of tables

List of abbreviations

OCR	Optical Character Recognition
IAMP	Intelligent Ayurvedic Management Platform
NLTK	Natural Language Tool Kit
NLP	Natural Language Processing

List of Appendixes

1. INTRODUCTION

Ayurvedic medicine is more affordable for a long time of health-issues in living beings rather than the approach of western medicine. Some countries like India, Sri Lanka believe those treatments can be reduced more side effects and avoiding less chance of having a threat to their lives. Although those are helping to provide stabilizing hormones and metabolism, natural healing, and strength in the immune system. Traditional ayurvedic treatments are mostly going on generation to generation by a specific person of a family. Some type of ayurvedic treatments can support avoiding either the dangerous effects of health-issues having in western approaches.

Arogya is a mobile application app for an Intelligent Ayurvedic Management Platform (IAMP). It provides many services in ayurvedic treatments for undergone alongside health-issues by neglecting in a western approach. Detecting the ayurvedic plants, the process of medical plants classification using detected ayurvedic plants, displaying the full-detailed information of prescription medicine samples of each category type of diseases such as Cancer, Diabetics, Arthritis etc. for provided herbal plants[2], displaying locations where the identified medical plants are spread out using a geographical map and analyzing the category of diseases and picture extraction of ingredients in prescription medicine. Not only that, develop another two functions as the new features of this app Arogya. Identifying the ayurvedic related post/recipes to upload to the wall of the app using text similarity technique and Ayurvedic related image identification using OCR following image processing. Such are the main components of focusing on this Arogya app.

Our proposed system concerns with these four main components for giving better solutions for the neglecting of health issues in western approach and traditional health issues in Ayurvedic treatments. There are most of the people who do not know which one used for what, the details of curving process and what are the best prescription

medicine (recipes, samples) etc. Although not able to spend time on finding different types of herbal remedies or medical plants within a short period of treatment, therefore this Arogya App will provide a good chance for these problems are going on busy daily routines.

There are some specific services like features provided by Arogya app as mentioned in earlier paragraph. When considering about the new features; If the user wants to upload a recipe or something related to the ayurvedic post on the social platform they given a chance to upload only related Ayurvedic posts and recipes. If they are not related to the Ayurveda, then it will block and not posted in the wall of the social media platform in Arogya. Today there are so many apps stored in play store for using ayurvedic treatments, but they are not consisting with new user-friendly features like posting recipes which are known and use like Facebook social media app.

Using OCR techniques to identify the uploaded image recipes, it will be displayed the included ingredients on it. Any user can upload recipes to the wall which appears to relevant types of diseases. Then it will recognize the text and display what things are included there clearly. The general working of OCR process is given below.

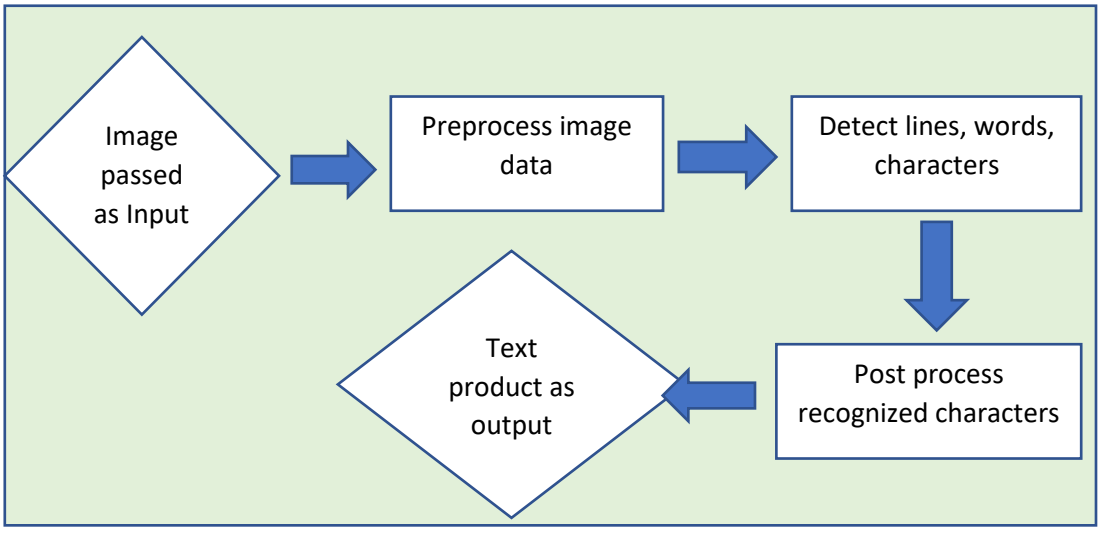


Figure 1: The General working of OCR Process

1.1 Background and Literature Survey

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 consisting of 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 known sources, researchers have tried to classify many different types of datasets for predicting the specific things/outputs.

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

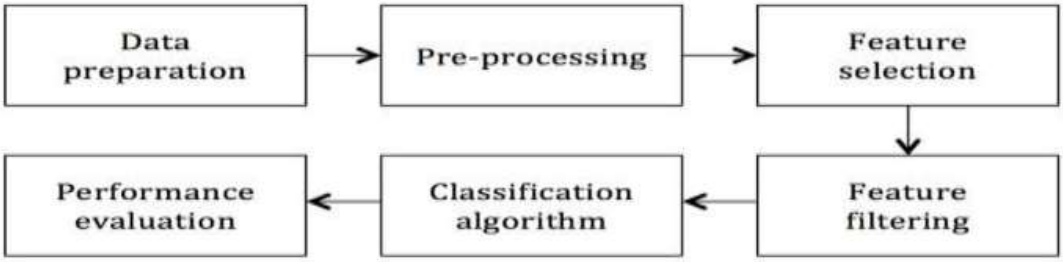


Figure 1.1 1: Analysis Overall process components

So, this proposed system is more important to identify the herbal remedies when people fail in its identification and not understanding in better ayurvedic samples/prescriptions medicine even though in Offline status. The 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 the world's oldest medical system and remains one of India's traditional health care systems. Every person engaged in various type of activities/ treatments on the internet either by using social media or online identification activities. Many activities go through on the internet as well as treatments in Ayurvedic and western approach medicine for health-related issues.[1].

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. [7] However, they introduced dynamic exercise plans, nutritious treatment and diagnosis for curving everything else related to diseases. [2]. Therefore, today created many applications for each of different tasks, but our proposed system will give special options as clarifying a disease using text selections. According to this text classification and picture extraction function is somewhat different from the exact process of text classification applied 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. [4]

2019, research paper “A comparative study of optical character recognition in Health Information System” [11] presented the use of 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” [9], presented the comparison of different OCR tools. The paper was presented after experiment with several OCR tools, and considering, taking an open source software called Tesseract OCR engine was selected for the pilot solution of this research food tracking and tracing.

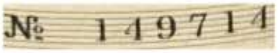
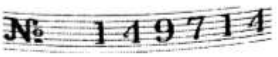
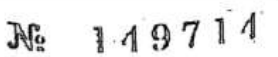
Image type	*Image*	*Azure CV*	*Google CV*	*Tesseract*
Original		-	14971	749714
After thresholding and Otsu filtering		-	-	974
After removing horizontal lines		7	149714	149714

Figure 1.1 2 : Comparison between different OCR tools

In the 2018 paper, “Document segmentation and language translation using Tesseract-OCR” [10], presented the basic components and the process of Tesseract-OCR understandably. “python-tesseract” module in python implements 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 a 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 projects are built using OCR for recognition of the text inside images. Document segmentation and language translation, Health information systems, Food tracking and tracing [10,11,9] 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. Text similarity is implemented by various type of algorithms such as Cosine. Jaccas etc. Cosine similarity is the best algorithm for identifying the text similarities in most of the applications.[12]. Considering the Cosine similarity used applications predicted better accuracy ranges.

Comparison of the point of text classification and picture extraction in recipes between Arogya App and existing Mobile apps

	Agrobase	Plantex	PlantSnap	MedLeaf	Arogya
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 1.1 1 :Comparison between existing apps with Arogya app

1.2 Research Gap

Ayurvedic medicine systems are mostly produced to introduce some herbs details and prescriptions for either daily diseases or traditional diseases. The existing apps were introduced only for detailed information regarding methods of ayurvedic treatments. In here developed the text identification through the dropdown selections and identify its related to Ayurvedic type or not. If it is a Ayurvedic related herbal plant, then display the it used for what type of disease.

Even thought it was identified handwriting recognition and Text identification of Ayurvedic related recipes or posts which were uploaded by the users. Until The existing apps were not developed for all the features which were mentioned in Arogya.

1.3 Research Problem

But Our proposed system will provide many services for busy daily schedule of lives. According to this function “Analyzing the category of diseases and picture extraction of prescription medicine” is proceed on offline status. So that people can take this service whatever place they used. apart from that identify the ingredients in the prescription/ recipes is a tedious task for people who are depend on technologies. If we consider the medical/herbal leaves, plants, roots etc. most of the people are not familiar with their visual appearance. So that, this system will support to display most appropriate recipes in predicting categories and visual images of included ingredients in prescriptions/recipes.

This proposed system is a social media app based on IAMP. So that, we could upload recipes and posts to the wall in the app for given the additional knowledge to the others who are already used in Arogya app. Some recipes will be handwriting images. Then it

will read the details included in there successfully. When users upload images, which are not clearly mentioning the handwritings then it is very difficult to read and understand. Because of that reason, OCR will be given the better identification of the included details. Most of the existing ayurvedic related apps were consist in irrelevant posts and details. Then app will be unwanted app.

1.4 Research Objectives

The purpose of the Document /Text classification is to separate the contents of texts or documents for a one or multiple categories. It is mainly supported in providing information retrieval, document association and management. According to this function, "Text classification of herbal diseases and picture extraction of Ayurvedic recipes" have much of goals including this proposed system. Although developed as another two functions as "Identifying the ayurvedic related posts/recipes using OCR and Text similarity." Hence, users can upload recipes and posts as they wish but it will be uploaded whether its only related to the Ayurveda.

1.4.1 Main Objectives

There are three components included in this research function "Analyzing the category of herb disease and Identify ayurvedic related post/recipes using text similarity and OCR." Each of the components have main objectives which are mentioned in below.

As considering the Component of *Identifying the disease type,*

- Users can identify the category type of diseases, just including its specific properties of needed medicine plants and Ability to observe the visual appearance of ingredients in prescription/recipes which are most probably used and other recipes.

As considering the *Image Identification using OCR*,

- Identify the Handwriting images very clearly and understandably.
- Lack of systems were consisting only hard things, but here can identify the upload images very clearly.

As considering the *Text similarity*,

- Lack of systems were included irrelevant things but here, Only uploaded ayurvedic related posts and recipes.

1.4.2 Specific Objectives

There are some sub objectives related to this function as per in below.

1. Users can identify better prescription medicine/recipes which are already not used in before
2. In any place can search because it can be activated either in off status.
3. Identify more prescription samples according to the relevant category types.
4. Can be seen rare or unknown medicine plants which are included in as ingredients of medicine recipes.

1.5 Requirements

1.5.1 Social aspects

This Arogya mobile application is a friendly social media Ayurveda app. People who busy with their tedious tasks and not attending to find Ayurveda treatments without depend on technology. 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, it is possible to expect a great appreciation as well as a demand from society for this mobile application. In particular, there are no age limits for users, no need for advanced computer literacy, and no need for advanced expertise to use this software in the field of Ayurveda.

For this, only a basic knowledge of how to use a smart mobile phone would be adequate. In addition, both the native and the scientific name of the plant that would benefit any human, including foreigners are given the name of the plant described.

This would also be very good for the whole community and would do a great service in uplifting our ancient Ayurveda, and because of its naturalness, is even better than modern medicine.

1.5.2 Security aspects

Security of this mobile application is high; user should enter his/her own credentials (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, Instagram applications. (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 Microsoft Azure with IAM (Identity Access Management) user credentials.

1.5.3 Ethical aspects

This application is not causing any harm, injury, or damage to the user 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.

Hence, users who want to identify the various type of recipes relevant to the disease type, is easy to be known here easily.

1.5.4 Functional and Non-Functional Requirements

Functional Requirements	Non-Functional Requirements
Progress should be Efficient	Results should be efficient
Algorithms give prompt responses	Accuracy should be required
	Interfaces should be User-friendly
	Less Time-consuming for searching details of medicine plants

Table 1.5.4 1: Functional and non-functional requirements

1.5.5 User Requirements

- Have a medicine plant which was known one or unknown one but familiar with the characteristics/properties

1.5.6 System Requirements

- If want to updated DB data, then use strong strength of Internet connection.

2. METHODOLOGY

2.1 System overview

The outcome of this research project will produce a mobile application that including special features for Ayurvedic medicine. According to the "Analyzing the category of herb diseases with picture extraction of prescription medicine and identification of Ayurvedic related post/recipes with text similarity and using OCR" function is the describing here in detailed. There is a feature selecting texts using dropdown options for text analyzing is a separate process going on to identify unknown characteristics of herbal remedies via using machine learning algorithms. Then it will be predicting its own category type according to the giving details.

As considering the system overview of this function, it provides drop down sections for selecting characteristics/properties of the herbal plants. Hence, people can select whatever things they know about the plant in the selection list. All the fields are not required for filling. But this proposed system will be identified special points of herbal plants/remedies. Then System will display the name of the category disease according to the provided details after going on an analyzing process. In order to that Using some specific details should be gathered each of herb plants. Then displaying some of prescriptions which are relevant to the displayed category will be shown under the each of categories. Among those categories, system will display most usable prescription also. Although users who knows some recipes or Ayurveda related posts then have a chance to upload to the wall. Either recipes of kadum bidum or any disease which predict in app related Ayurveda recipes can be uploaded. Hence, users can upload any type of Ayurveda posts/ texts/ recipes for knowledge base to outsiders as they wish.

In earlier decided to implement only "Analyzing the category of herb diseases and picture extraction of ingredients included in prescription medicine". After progress presentation-I (50%) decided to implement another two components that are mentioned

in previous paragraph and clearly applied in system overview diagram below figure 2.1 1 also.

As the technologies we used,

- Android studio/java

  This IDE is used for frontend development of the application.

- Spyder/Command prompt

  This IDE is used for back-end development

- Python 3.7
- TensorFlow
- Visual studio code
- OpenCV
- Pytesseract
- Base64
- Tesseract

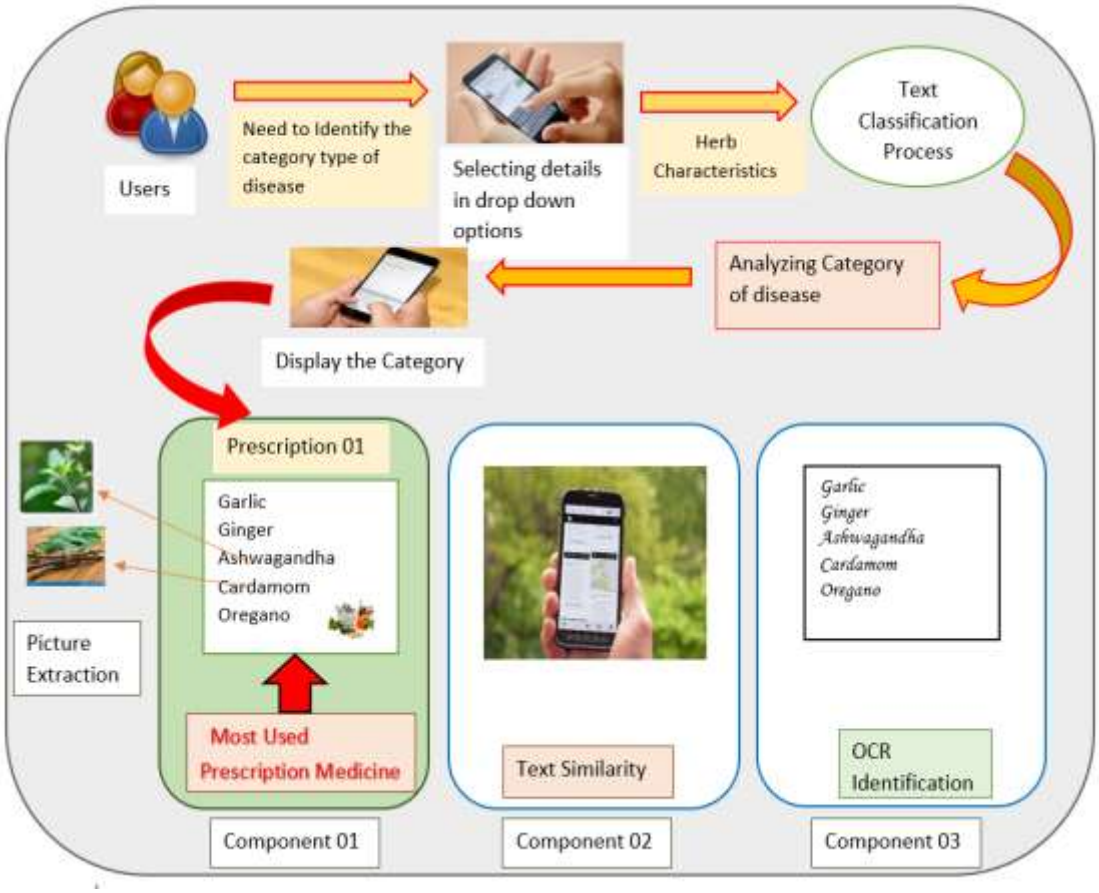


Figure 2.1 1: System overview diagram

2.2 Flow of Project

2.2.1 Identify the Category type of Disease

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.

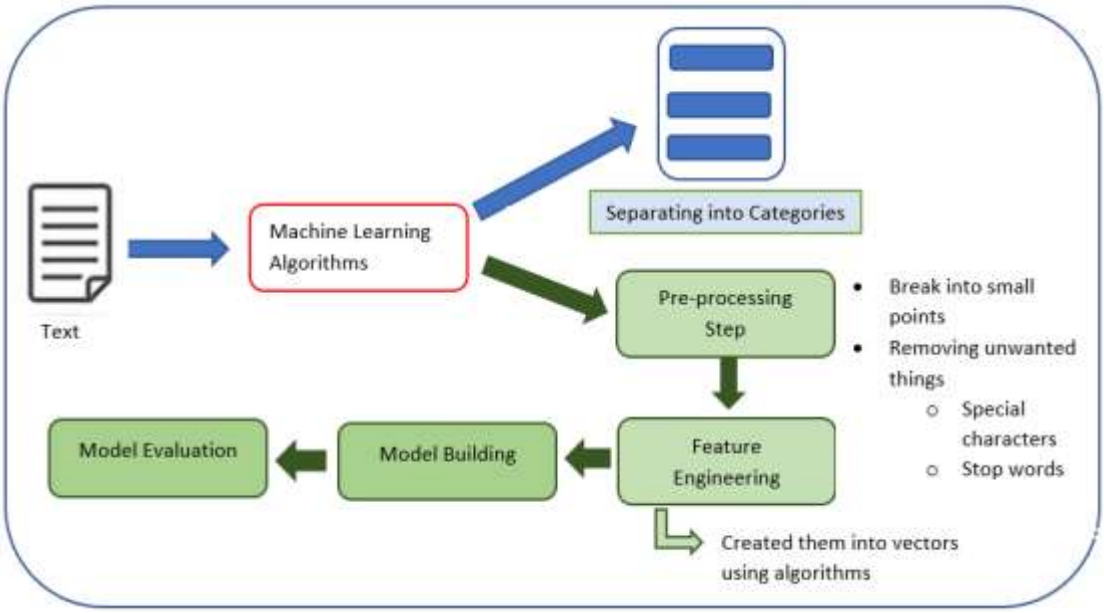


Figure 2.2.1 1: System Architecture

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. There end to end text classification which is a pipeline consists three main components. According to the Figure 2.2.1.1. 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.

2.2.1.1 Data collection

For this disease prediction component, this project needs different data that needs to be collected from external entities. Creating a dataset with 87 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.

2.2.1.2 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 models 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, different ideas can be identified. 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.

2.2.1.3 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, 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 display 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.

2.2.1.4 Text Classification Using Tools

Both NLTK and the TextBlob performs well in Text Classification processing.

NLTK	TextBlob

NLTK is a very big library holding 1.5GB and has been trained on a huge data with proving different dataset in multiple languages which can deploy according to the functionality its be 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.

Table 2.2.1.4 1 The features of text classification tools

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

2.2.1.5 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					

Table 2.2.1.5 1: Comparison of text classification tools

2.2.1.6 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 analysis, pre-process, and understand the written text. In This part, only use for the typing field and this field is consist with limited words as key words of herbal characteristics. Because if we give a large description there should have more stuffs further.

Text Classification Steps for typing field:

1. Loading data and Creating classifiers:

   First create custom classifiers using 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.

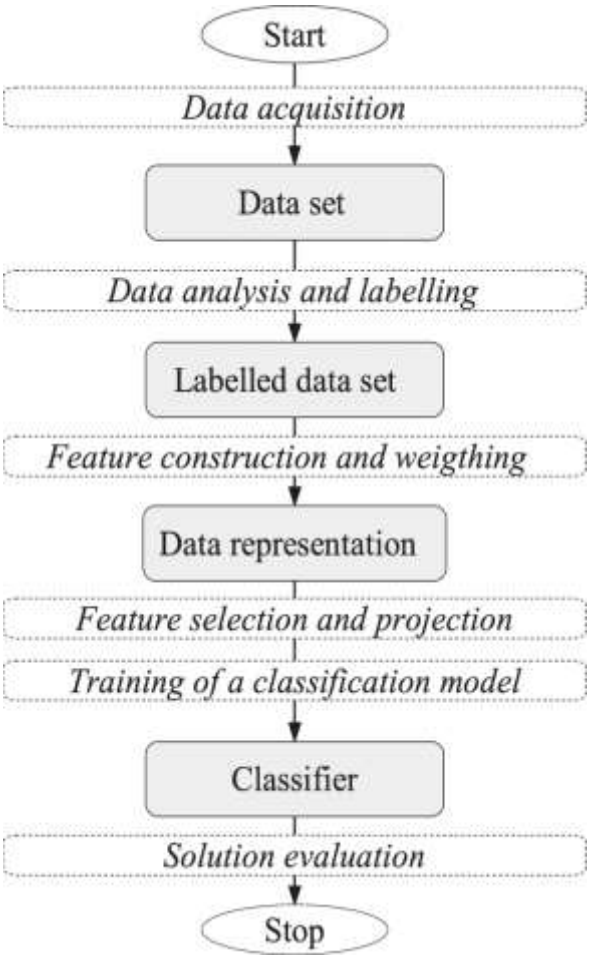
We can use our own extractors to identify each specialty.

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 model evaluation step.



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

Figure 2.2.1.6 1 : Text classification flow chart

2.2.2 OpenCV OCR and text recognition with Tesseract

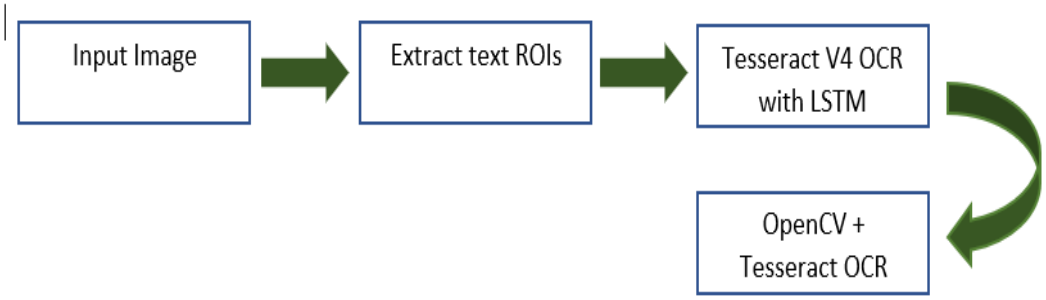


Figure 2.2.2 1: The OpenCV OCR pipeline

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, we 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.

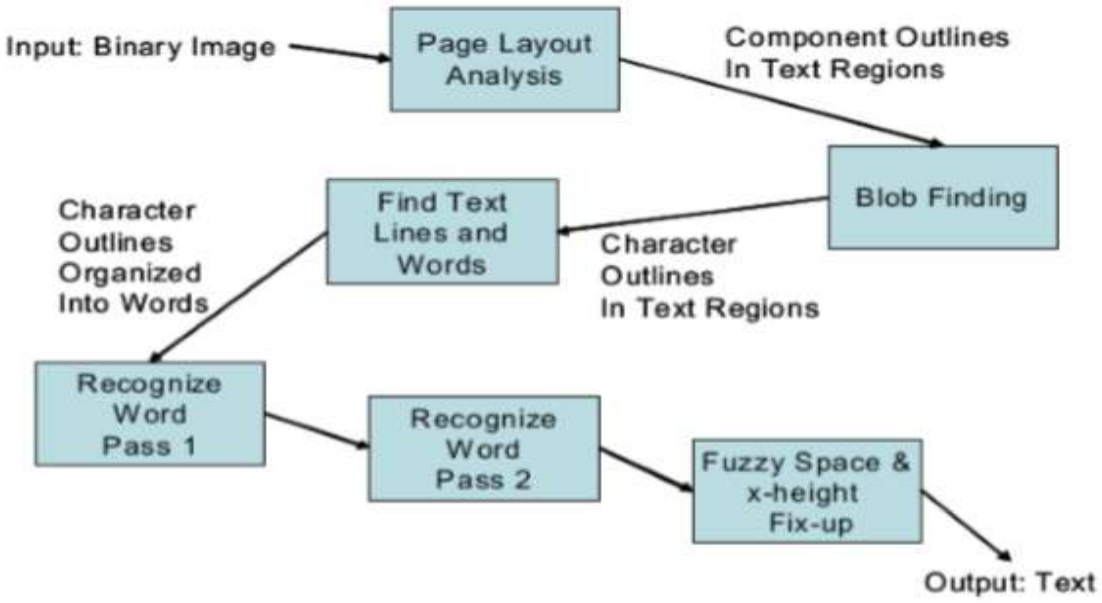


Figure 2.2.2 2: Block diagram of basic components of Tesseract

Extracting text from the image,

In 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 implement tesseract-OCR to convert the image into text. There implementation is simple to carry out because the module and call the defined method 'image_to_string' is converts the given image to string. Tesseract converts and returns the text available in an image.

2.2.2.1 Installing packages

- Install Tesseract + Python bindings

Need to install the Tesseract + Python bindings to created python scripts. Then it could be communicated with Tesseract and perform OCR on image 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.

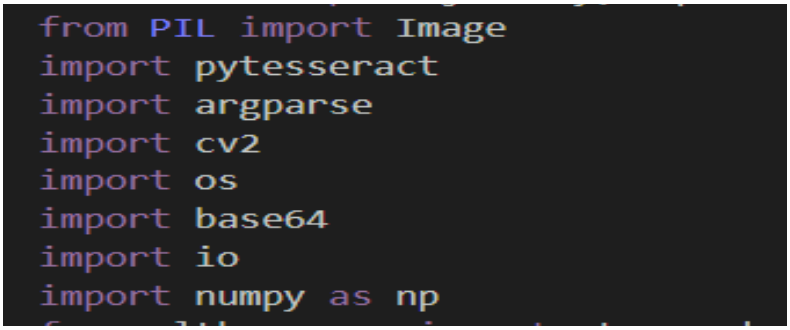


Figure 2.2.2.1 1: Import Packages

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 using 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 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 easier for Tesseract to correctly OCR the image.

After pre-processing 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.

2.2.3 Text similarity using Cosine 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.

2.2.3.1 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.

$$\text{Similarity} = \cos(\text{Ø}) = \frac{A.B}{|A||B|} = \frac{\sum AB}{\sqrt{\sum A_i^2}.\sqrt{\sum B_i^2}}.$$

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)
```

Figure 2.2.3.1 1: Apply the Cosine formula

2.2.3.2 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 deserializing 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.

```
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
import json
import pickle
```

Figure 2.2.3.2 1 Installed packages for text similarity

2.2.3.3 Process of Text similarity

When considering about the demonstrate of this process, the important one is to consider about 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.

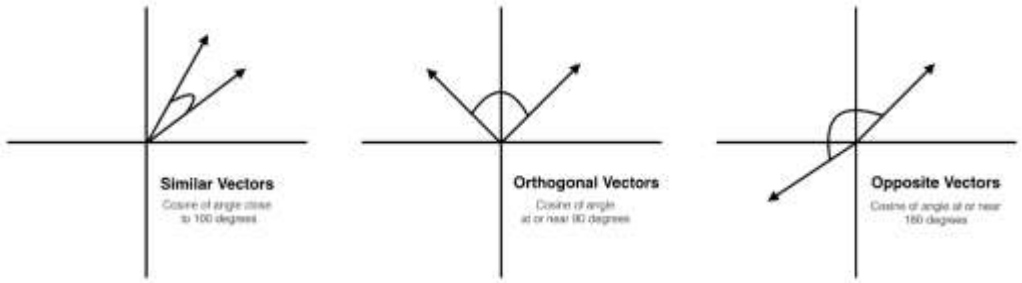


Figure 2.2.3.3 1: Stages of Text similarity

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

nltk.tokenization is the process by which 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.

After tokenization, remove stop words from the string, then from a set containing keywords of both strings.

Create a vector and assign the relevant values to the Cosine formula described in sub-topic 2.2.3.1.

2.3 Commercialization of the product

People who can access to Arogya in offline or online with friendly user interfaces and Giving better solutions for the critical health-related issues in Ayurveda. Considering the function related text classification and feature extraction that can be commercialized for saving time of people who are spending a tedious lifestyle.

2.3.1 Budget plan and Justification

Arogya App is a mobile application for an Intelligent Ayurvedic Management Platform (IAMP) providing specific features to users who are willing to giving Ayurveda treatments.

- Finding specific medicine plants and their detailed- information
- Meet Ayurveda doctors and discuss with Ayurveda medicine treatments
- Finding rare plants in diseases categories and analyzing its characteristics

	Sri Lankan rupees
Ayurvedic Doctor Charges (per appointment)	Rs. 3000/=
Data collection travelling chargers	Rs. 4000/=
Documentation printout cost	Rs. 3000/=
Total	***Rs 10000/=***

Table 2.3.1 1: Budget related stuff

2.3.2 Target Audience

- People who use ayurvedic treatment
- Ayurvedic plant sellers
- Doctors, Students, locals, and foreigners

2.3.3 Market Space

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

2.4 Testing and Implementation

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.

- Frontend – Android application

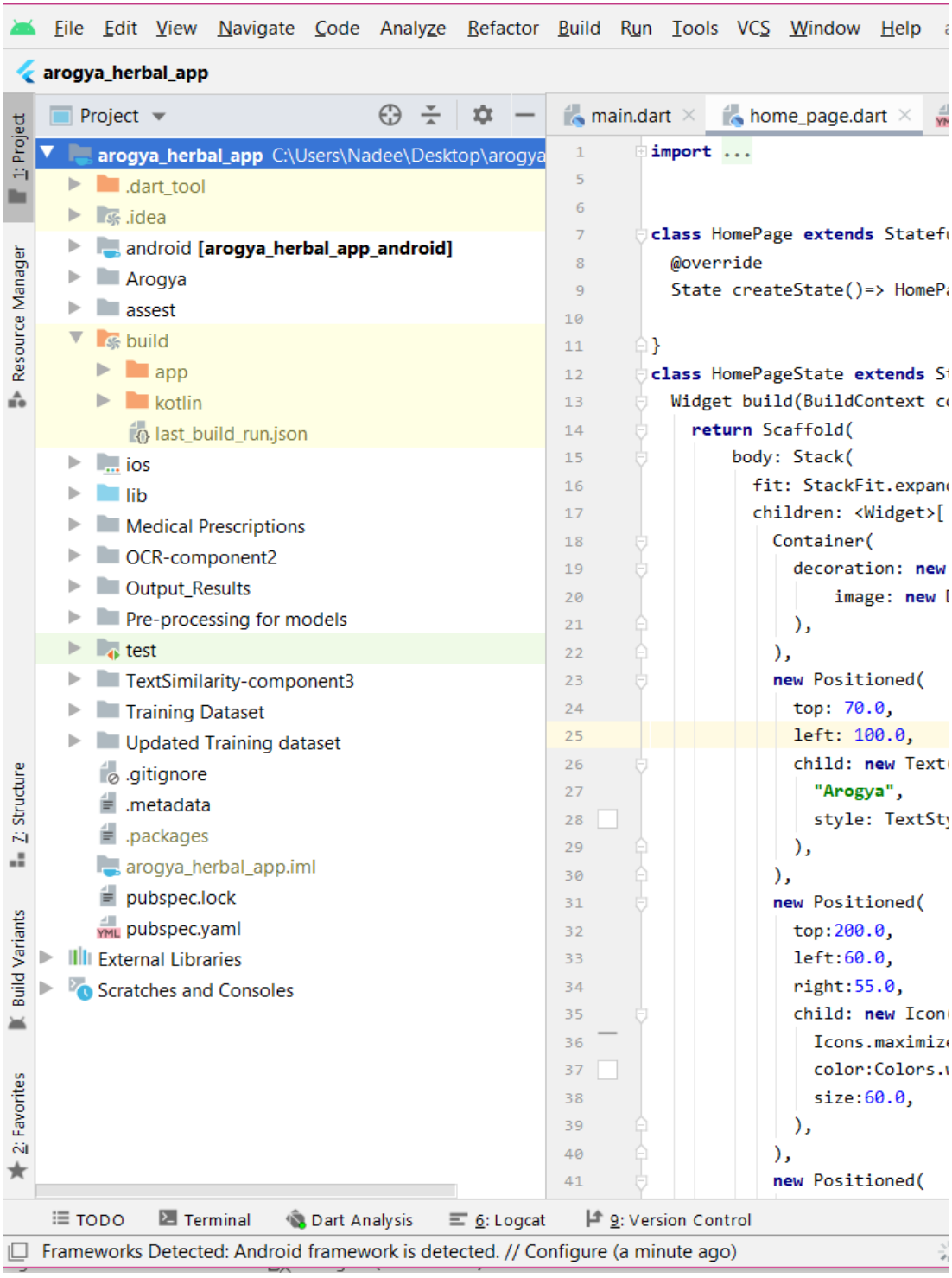


Figure 2.4 1 : Frontend folder structure

- Backend – Machine learning model

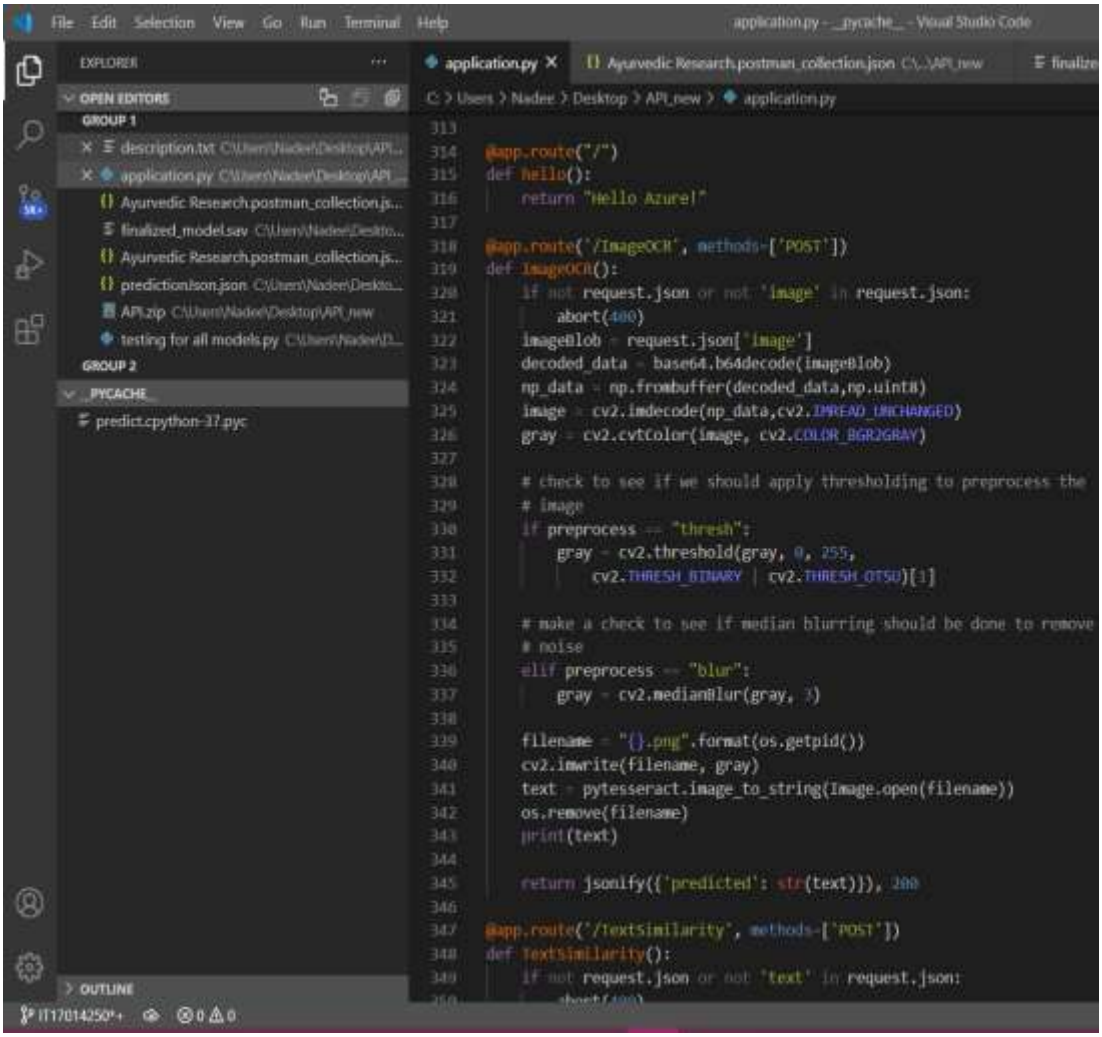


Figure 2.4 2 : Backend folder structure

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

2.4.1 Identifying the category type of Disease – Implementation & Testcase Implementation:

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
```

Figure 2.4.1 1 : Created new records and assign values

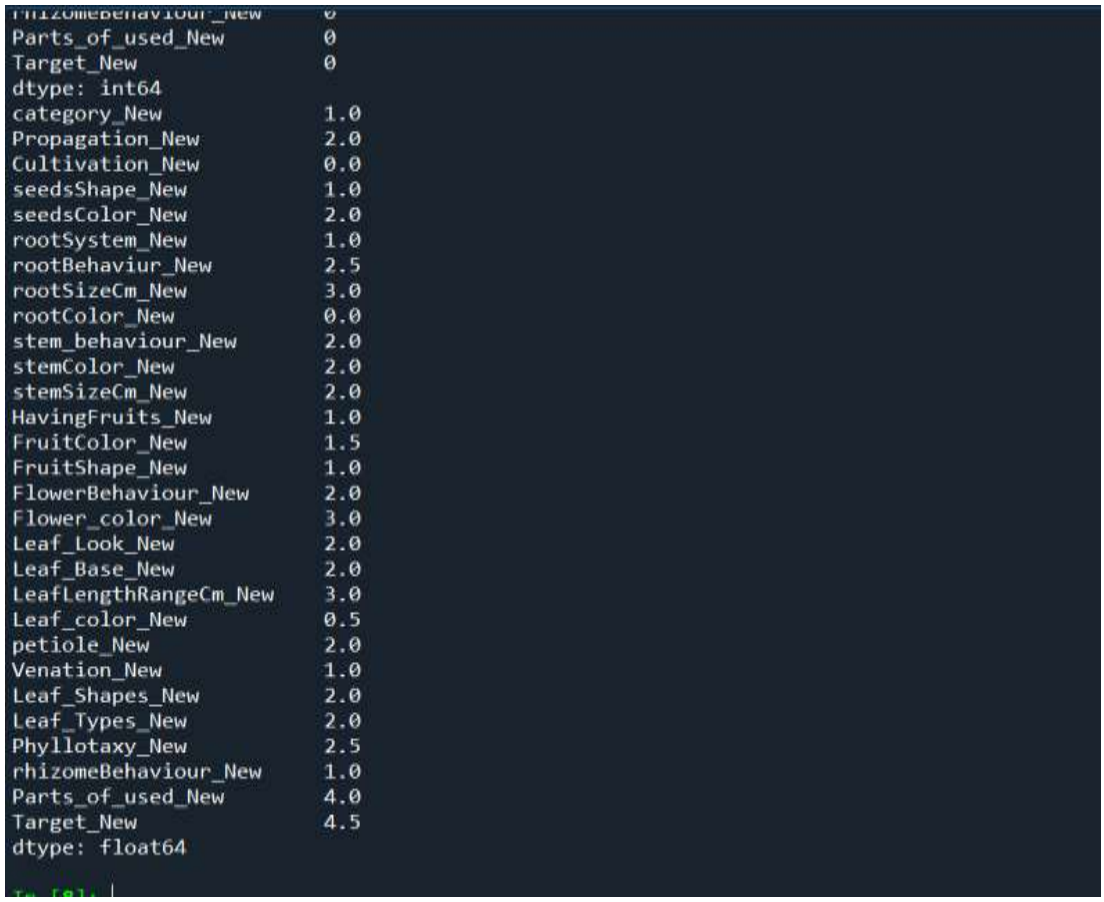


Figure 2.4.1 2: Display the assigned values included to new columns

```
>>> sns.heatmap(c,cmap="BrBG",annot=True)
<matplotlib.axes._subplots.AxesSubplot object at 0x0000026785042188>
>>> c
                        category_New  Propagation_New  ...  Parts_of_used_New  Target_New
category_New                1.000000         0.412814  ...           0.616417    0.116454
Propagation_New             0.412814         1.000000  ...           0.503046    0.879469
Cultivation_New             0.489898        -0.343303  ...           0.199091   -0.446610
seedsShape_New             -0.155556         0.566275  ...           0.190734    0.427392
seedsColor_New              0.225711        -0.704576  ...           0.022015   -0.883634
rootSystem_New             -0.105529         0.092199  ...          -0.094105    0.340483
rootBehaviur_New            0.020145        -0.098104  ...          -0.060254   -0.187202
rootSizeCm_New              0.139951        -0.380404  ...           0.147255   -0.674965
rootColor_New              -0.584237        -0.651339  ...          -0.598323   -0.283488
stem_behaviour_New         -0.145790        -0.427232  ...          -0.284390   -0.396579
stemColor_New               0.162816         0.433334  ...           0.183504    0.549029
stemSizeCm_New              0.469959        -0.315456  ...           0.123723   -0.486074
HavingFruits_New            0.105529        -0.092199  ...           0.094105   -0.340483
FruitColor_New             -0.131426         0.200042  ...          -0.051274    0.139137
FruitShape_New             -0.657855         0.326566  ...          -0.269987    0.575858
FlowerBehaviour_New         0.412814         1.000000  ...           0.503046    0.879469
Flower_color_New           -0.100862        -0.858221  ...          -0.229256   -0.875905
Leaf_Look_New               0.295481         0.968086  ...           0.397592    0.946969
Leaf_Base_New               0.042563        -0.225597  ...          -0.232475    0.042056
LeafLengthRangeCm_New      -0.420137        -0.829100  ...          -0.348788   -0.867168
Leaf_color_New              0.447932        -0.180907  ...           0.059351   -0.384741
petiole_New                -0.119176        -0.720175  ...          -0.106274   -0.871166
Venation_New                0.342864         0.896490  ...           0.337318    0.878522
Leaf_Shapes_New             0.113333         0.832424  ...           0.221078    0.943399
Leaf_Types_New              0.119176         0.720175  ...           0.106274    0.871166
Phyllotaxy_New              0.062639        -0.625179  ...           0.000109   -0.757096
rhizomeBehaviour_New       -0.359024        -0.272118  ...          -0.359150    0.064073
Parts_of_used_New           0.616417         0.503046  ...           1.000000    0.266404
Target_New                  0.116454         0.879469  ...           0.266404    1.000000
```

Figure 2.4.1 3: Details of the values of correlation with data variables

Testcase:

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

Table 2.4.1 1 : Testcase of predicting the disease type

2.4.2 Analyzing the Image using OCR – Implementation & Testcase

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)
```

```
@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
```

Figure 2.4.2 1: Implementation of Image OCR

When upload an image which should have to convert image to string type using base64. This is for pass the API through the image to display on the frontend.

This is the link used for converting as per in below.

https://codebeautify.org/image-to-base64-converter

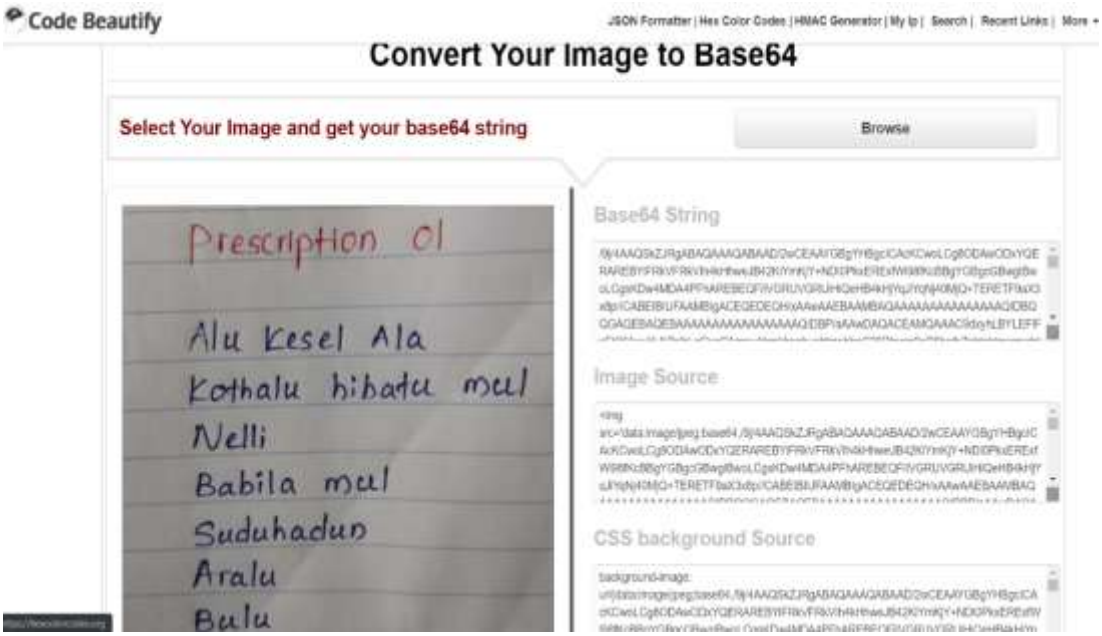


Figure 2.4.2 2: Convert image to Base64

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

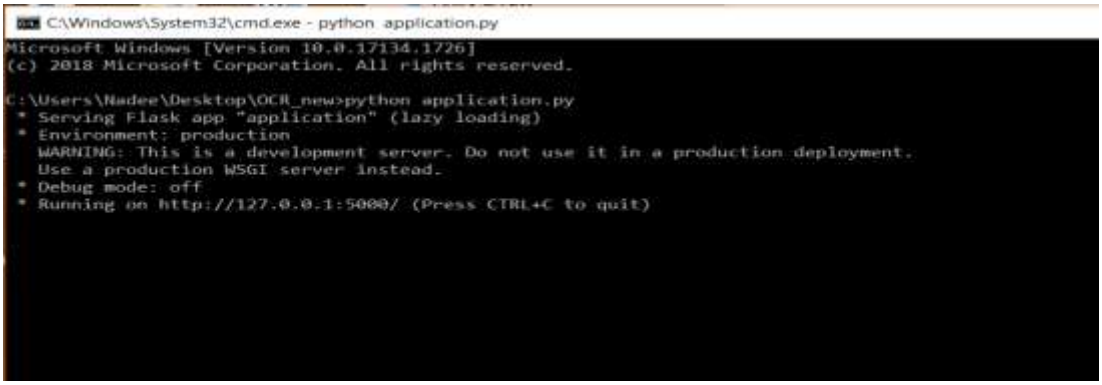


Figure 2.4.2 3 : Checking the results through the postman

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

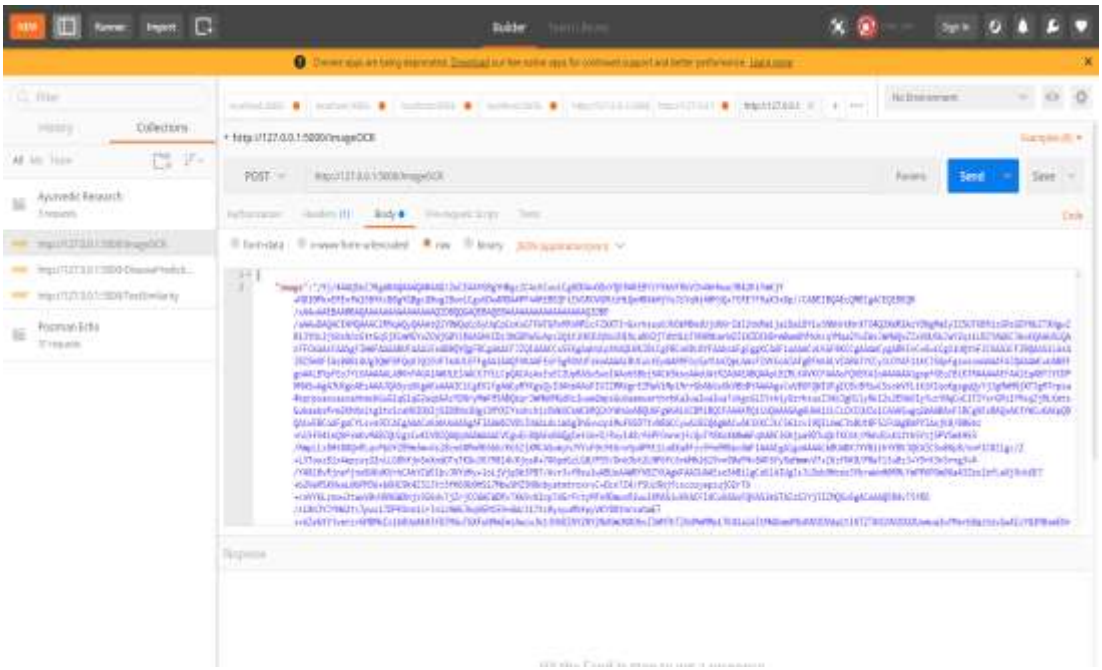


Figure 2.4.2 4: Apply the string type of image txt to check the postman

Testcase:

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

Table 2.4.2 1: Test case of Identify the uploaded image text

2.4.3 Identifying Ayurvedic related posts using Text similarity – Implementation & Test case

Implementation:

```
@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
```

Figure 2.4.3 1 : Implementation of text similarity

Using postman for checking the results,

Applying the text to check though postman

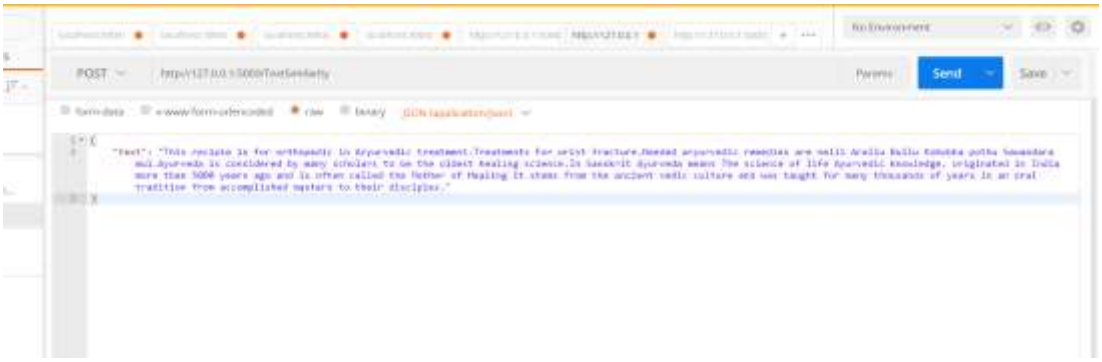


Figure 2.4.3 2: Checking the text similarity results through the postman

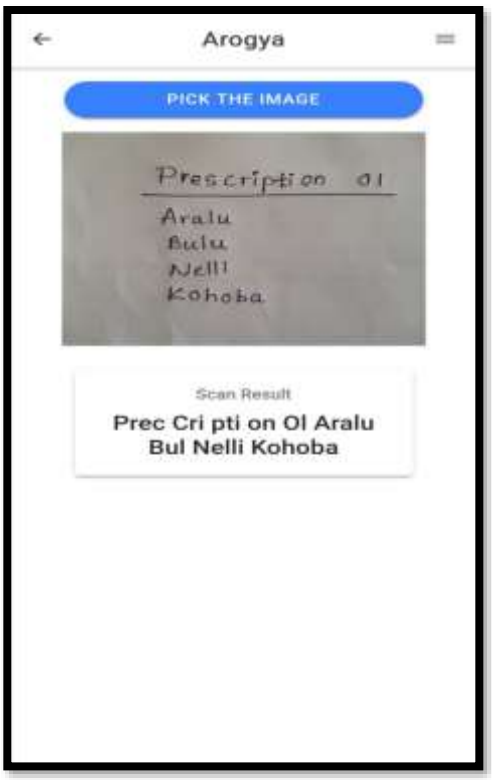
Test case ID	001
Test case scenario	Upload Ayurveda related post/recipe
Test steps	1. Login to the Arogya app with correct credentials. 2. Navigate to the post/recipe upload interface. 3. Select a post from the gallery 4. Upload the post/recipe
Test Data	1. Username = nadee@gmail.com Password= nadee@321 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

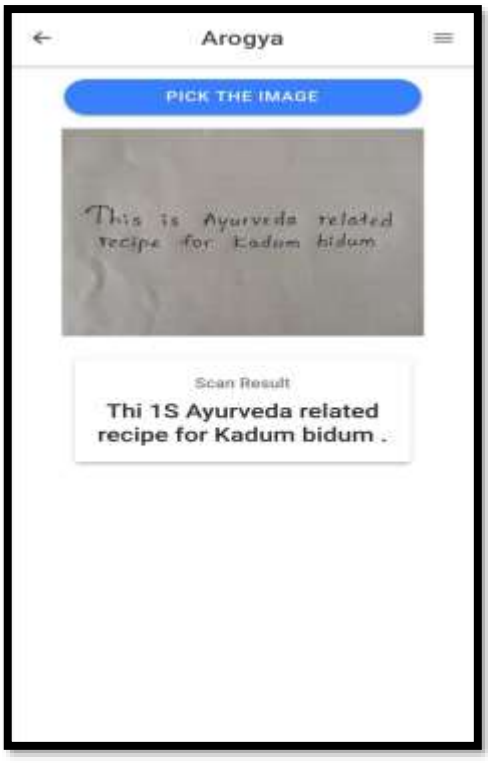
	was taught for many thousands of years in an oral tradition from accomplished masters to their disciples.
Expected Results	Identify the ayurvedic related or not. If it ayurvedic related, then upload to the wall. If not post will be blocked.
Actual Results	Upload to the wall
Pass/fail	Pass

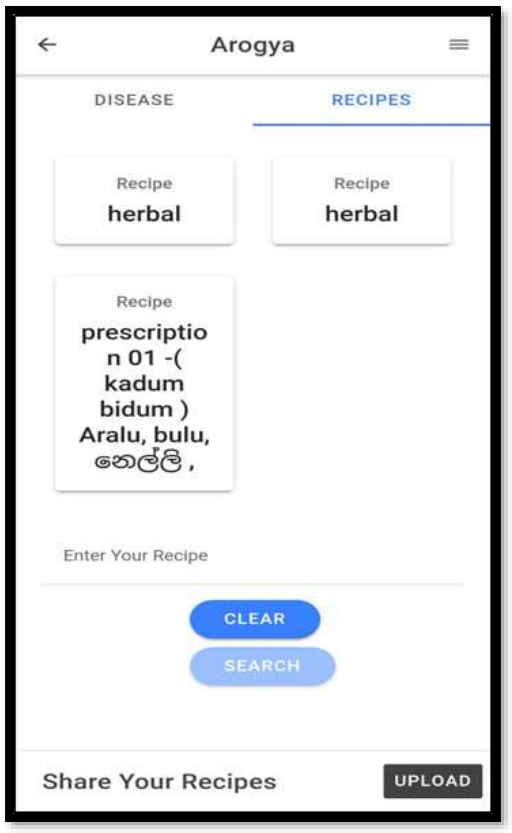
Table 2.4.3 1: Test case of Testing the post whether its Ayurveda or not

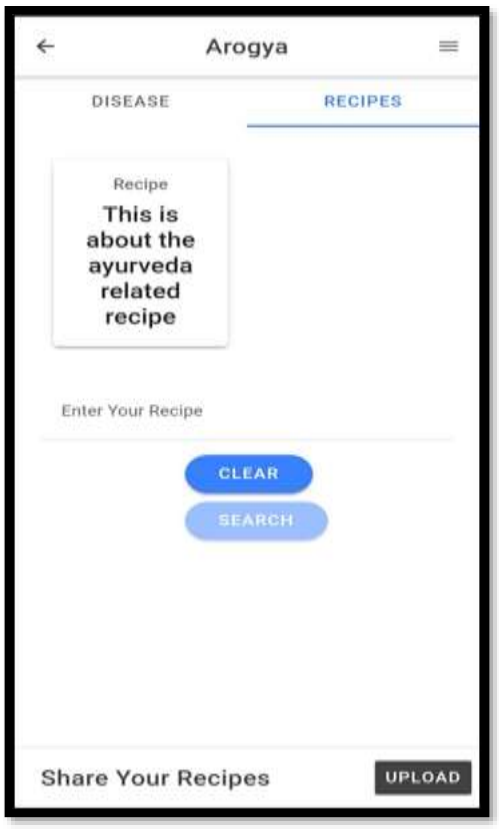
User Interfaces of the Application:


Arogya
PICK THE IMAGE
INTRODUCTION
RESEARCH PROBLEM
BACKGROUND/RESEARCH GAP
SPECIFIC/SUB OBJECTIVES
Scan Result
\ INTRODUCTION \ _) RESEARCH PROBLEM _| BACKGROUND/RESEARC H GAP J SPECIFIC/SUB OBJECTIVES


Arogya
PICK THE IMAGE
Prescription 01
Aralu
Bulu
Nelli
Kohoba
Scan Result
Prec Cri pti on Ol Aralu Bul Nelli Kohoba


Arogya
PICK THE IMAGE
This is Ayurveda related recipe for Kadum bidum
Scan Result
Thi 1S Ayurveda related recipe for Kadum bidum .


Arogya
DISEASE
RECIPES
Recipe
herbal
Recipe
herbal
Recipe
prescriptio
n 01 -(
kadum
bidum)
Aralu, bulu,
නෙල්ලි ,
Enter Your Recipe
CLEAR
SEARCH
Share Your Recipes
UPLOAD


Arogya
DISEASE
RECIPES
Recipe
This is
about the
ayurveda
related
recipe
Enter Your Recipe
CLEAR
SEARCH
Share Your Recipes
UPLOAD

3. RESULTS AND DISCUSSION

3.1 Results

3.1.1 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]:
```

Figure 3.1.1 1: Predicting result of the component of the identify the disease type

3.1.1.1 Visualization graphs

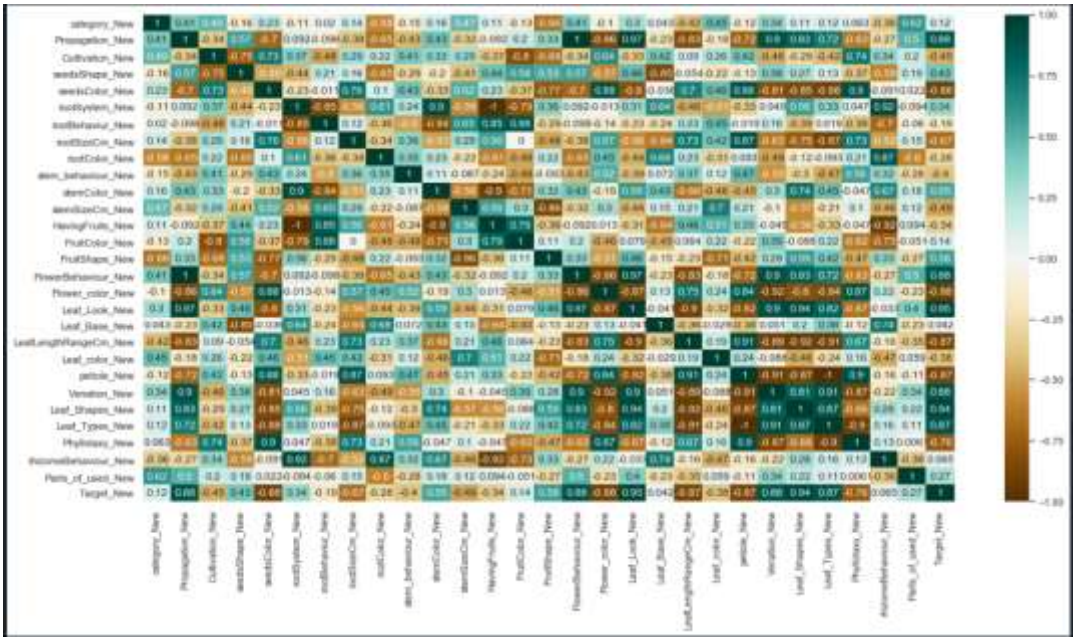
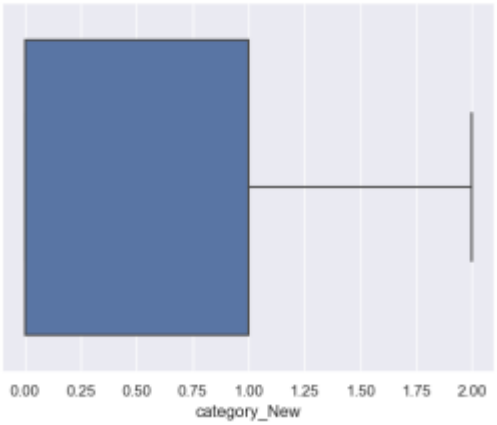
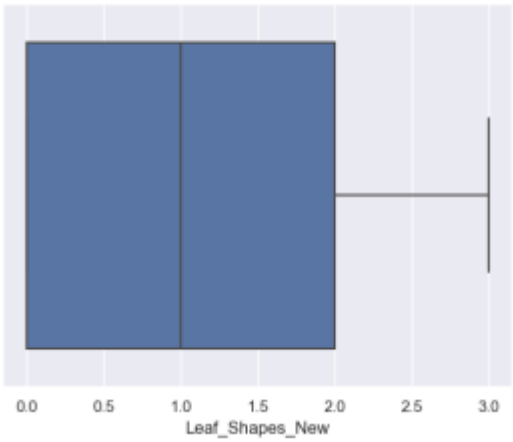
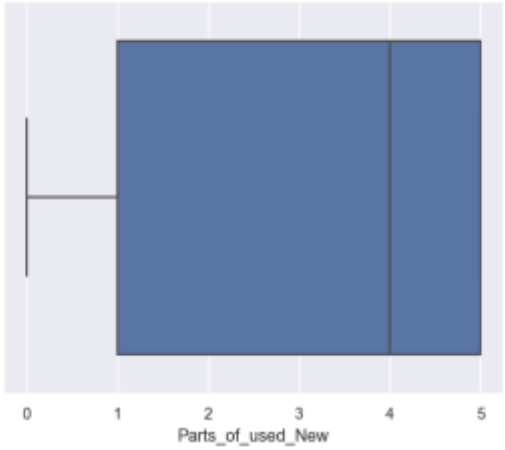
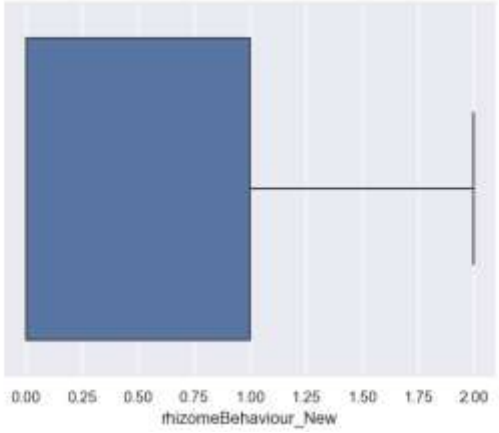


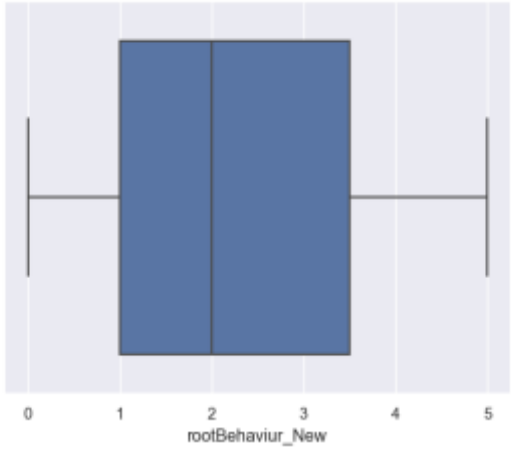
Figure 3.1.1.1 1 : Correlation between the variables in the dataset

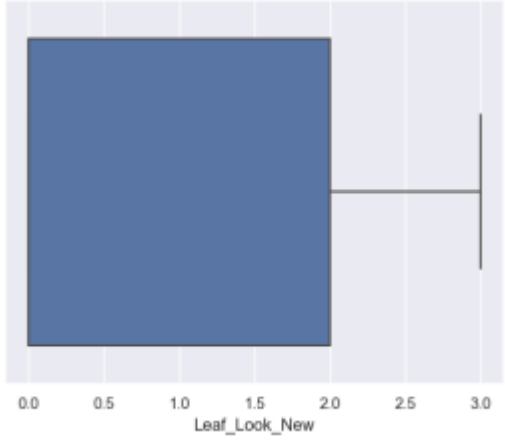

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
category_New


0.0 0.5 1.0 1.5 2.0 2.5 3.0
Leaf_Shapes_New

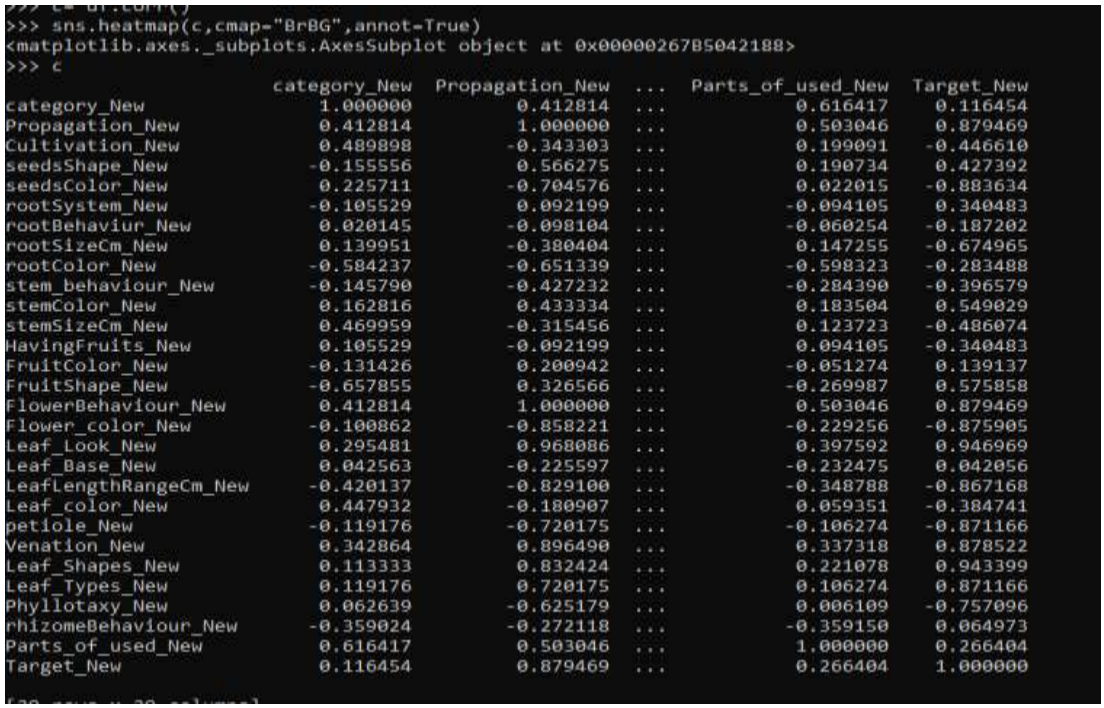

0 1 2 3 4 5
Parts_of_used_New


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

This is the entire implementation part of the disease identification component. All the parts are mentioned there as figure



```
>>> sns.heatmap(c,cmap="BrBG",annot=True)
<matplotlib.axes._subplots.AxesSubplot object at 0x0000026785042188>
>>> c
                     category_New  Propagation_New  ...  Parts_of_used_New  Target_New
category_New             1.000000         0.412814  ...           0.616417    0.116454
Propagation_New          0.412814         1.000000  ...           0.503046    0.879469
Cultivation_New          0.489898        -0.343303  ...           0.199091   -0.446610
seedsShape_New          -0.155556         0.566275  ...           0.190734    0.427392
seedsColor_New           0.225711        -0.704576  ...           0.022015   -0.883634
rootSystem_New          -0.105529         0.092199  ...          -0.094105    0.340483
rootBehaviur_New         0.020145        -0.098104  ...          -0.060254   -0.187202
rootSizeCm_New           0.139951        -0.380404  ...           0.147255   -0.674965
rootColor_New           -0.584237        -0.651339  ...          -0.598323   -0.283488
stem_behaviour_New      -0.145790        -0.427232  ...          -0.284390   -0.396579
stemColor_New            0.162816         0.433334  ...           0.183504    0.549029
stemSizeCm_New           0.469959        -0.315456  ...           0.123723   -0.486074
HavingFruits_New         0.105529        -0.092199  ...           0.094105   -0.340483
FruitColor_New          -0.131426         0.200942  ...          -0.051274    0.139137
FruitShape_New          -0.657855         0.326566  ...          -0.269987    0.575858
FlowerBehaviour_New      0.412814         1.000000  ...           0.503046    0.879469
Flower_color_New        -0.100862        -0.858221  ...          -0.229256   -0.875905
Leaf_Look_New            0.295481         0.968086  ...           0.397592    0.946969
Leaf_Base_New            0.042563        -0.225597  ...          -0.232475    0.042056
LeafLengthRangeCm_New   -0.420137        -0.829100  ...          -0.348788   -0.867168
Leaf_color_New           0.447932        -0.180907  ...           0.059351   -0.384741
petiole_New             -0.119176        -0.720175  ...          -0.106274   -0.871166
Venation_New             0.342864         0.896490  ...           0.337318    0.878522
Leaf_Shapes_New          0.113333         0.832424  ...           0.221078    0.943399
Leaf_Types_New           0.119176         0.720175  ...           0.106274    0.871166
Phyllotaxy_New           0.062639        -0.625179  ...           0.006109   -0.757096
rhizomeBehaviour_New    -0.359024        -0.272118  ...          -0.359150    0.064973
Parts_of_used_New        0.616417         0.503046  ...           1.000000    0.266404
Target_New               0.116454         0.879469  ...           0.266404    1.000000
```

Figure 3.1.1.1 2: Values of data variables which are corelated

3.1.1.2 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)
```

Figure 3.1.1.2 1 : Testing results of the all the models

3.1.1.3 Accuracy Graph for the comparison between testing models

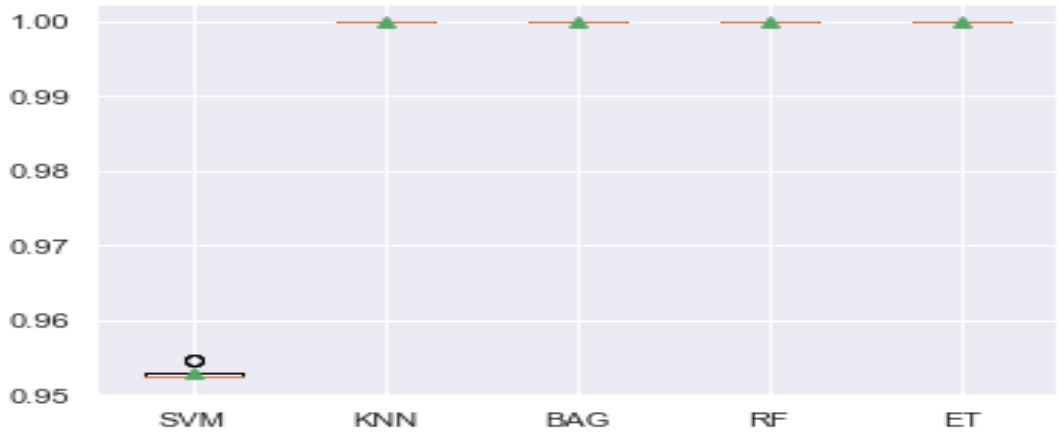


Figure 3.1.1.3 1: Accuracy graphs for the comparison between testing models

The best model is SVM (Support vector machine). Because it predicted the accuracy of 95%. Other models were predicted the 100% with overfitting issues.

3.1.2 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.

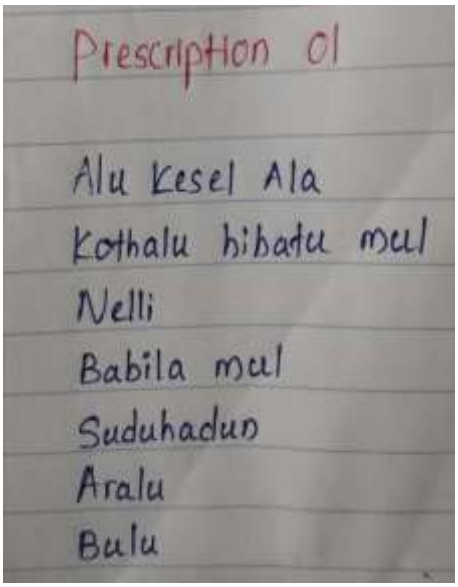


Figure 3.1.2 1: Handwriting image

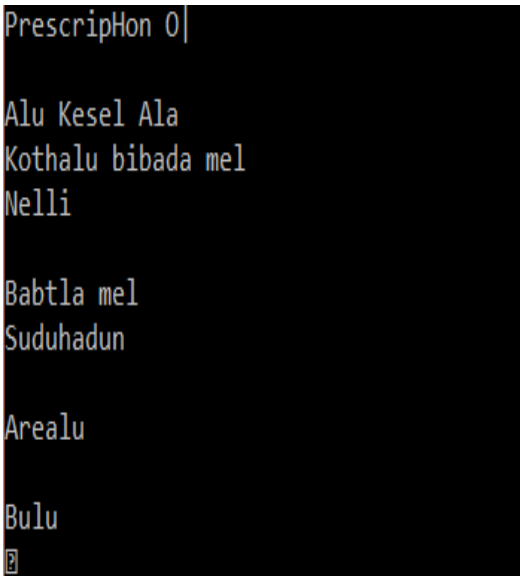


Figure 3.1.2 2: OCR output results

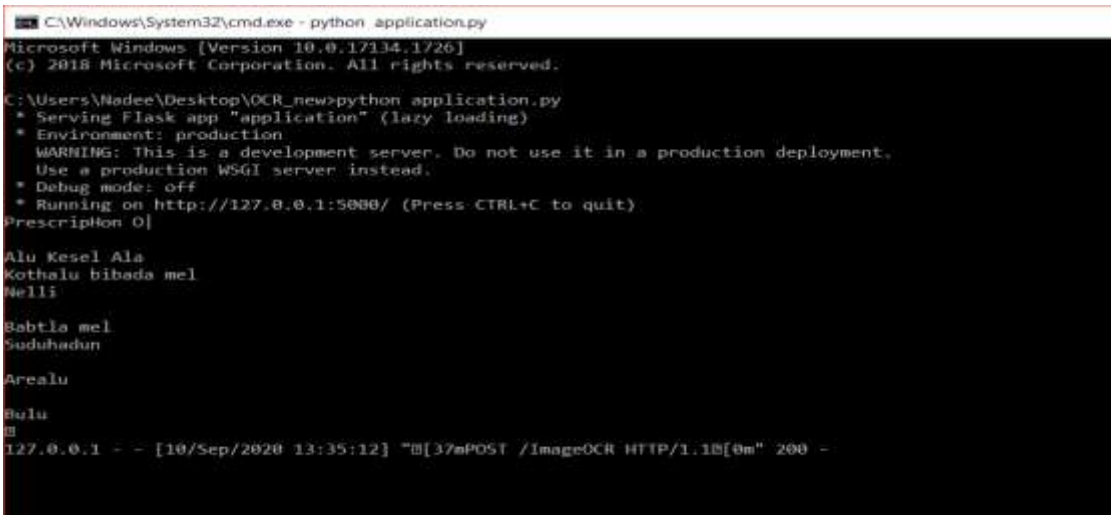


Figure 3.1.2 3 : Results display in the Command prompt

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

Figure 3.1.2 4: Postman checking the results of OCR

3.1.3 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."

}

Figure 3.1.3 1: The output results of the text including Ayurveda related text

3.2 Research Findings and Discussion

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. SUMMARY

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 sub-components 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.

5. CONCLUSION

Arogya is an app developed centralized on single base social platform consider as IAMP. It consists of related to sri Lankan herbal plants and other related information of Ayurveda. The proposed system has been consisting of providing different features rather than the existing apps were introduced. This system was capable of given the new technological tricks using the computer vision and Machine learning things. Analyzing the category type of disease using its characteristics and Identifying the Ayurveda related post/recipes using cosine text similarity and OCR techniques are mainly focus on here to implement under the machine learning aspects with computer vison. There are some issues when implementing the application related the accuracy and other usability issues. 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.

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.

6. REFERENCES

[1]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.

[2]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.

[3] 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

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

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

[6] "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

[7] "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/

[8] 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/

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

7. APPENDIXES

```
from flask import Flask
from flask import jsonify,request
from PIL import Image
import pytesseract
import argparse
import cv2
import os
import base64
import io
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
import json
import pickle

app = Flask(__name__)

preprocess = "thresh"
pytesseract.pytesseract.tesseract_cmd = r'C:\\Program Files\\Tesseract-
OCR\\tesseract.exe'

# Text cosine detection usage
filename ="description.txt"
with open(filename) as f:
    text = f.read()
sentences = re.split(r' *[\.\?!][\'"\)\]]* *', text)
# sw contains the list of stopwords
sw = stopwords.words('english')
# End text cosine detection usage

categorys = ['plant','vine','shurb']
def categorize_category(value):
    value = (value.rstrip()).lstrip()
    categoryList = [item.lower() for item in categorys]
    if value.lower() in (item for item in categoryList):
        return categoryList.index(value.lower())
    else:
```

```
        return None

propagation = ['rhizome','seeds and tubers','seeds and cuttings','seeds'
,'root and seeds']
def categorize_propagation(value):
    value = (value.rstrip()).lstrip()
    propagationList = [item.lower() for item in propagation]
    if value.lower() in (item for item in propagationList):
        return propagationList.index(value.lower())
    else:
        return None

cultivation = ['any','wet zone']
def categorize_cultivation(value):
    value = (value.rstrip()).lstrip()
    cultivationList = [item.lower() for item in cultivation]
    if value.lower() in (item for item in cultivationList):
        return cultivationList.index(value.lower())
    else:
        return None

seedsShape = ['no','round','kidney shaped']
def categorize_seedShape(value):
    value = (value.rstrip()).lstrip()
    seedsShapeList = [item.lower() for item in seedsShape]
    if value.lower() in (item for item in seedsShapeList):
        return seedsShapeList.index(value.lower())
    else:
        return None

seedsColor = ['shine black','brownish green','No','black']
def categorize_seedsColor(value):
    value = (value.rstrip()).lstrip()
    seedsColorList = [item.lower() for item in seedsColor]
    if value.lower() in (item for item in seedsColorList):
        return seedsColorList.index(value.lower())
    else:
        return None

rootSystem = ['tap root','fibrous root']
def categorize_rootSystem(value):
```

```
    value = (value.rstrip()).lstrip()
    rootSystemList = [item.lower() for item in rootSystem]
    if value.lower() in (item for item in rootSystemList):
        return rootSystemList.index(value.lower())
    else:
        return None

rootBehaviur = ['thin and expanded','thinner and shallow with soft and f
ragrant','long and deep','tuberous','camphor-
like aroma','thick long and deep']
def categorize_rootBehaviur(value):
    value = (value.rstrip()).lstrip()
    rootBehaviurList = [item.lower() for item in rootBehaviur]
    if value.lower() in (item for item in rootBehaviurList):
        return rootBehaviurList.index(value.lower())
    else:
        return None

rootColor = ['brown','beige']
def categorize_rootColor(value):
    value = (value.rstrip()).lstrip()
    rootColorList = [item.lower() for item in rootColor]
    if value.lower() in (item for item in rootColorList):
        return rootColorList.index(value.lower())
    else:
        return None

stem = ['woody cylindrical','thin,cylindrical,fluuted,stoloniferous with
 long internodes','leafless stem','delicate,smooth,brittle','straight sp
ines']
def categorize_stem(value):
    value = (value.rstrip()).lstrip()
    stemList = [item.lower() for item in stem]
    if value.lower() in (item for item in stemList):
        return stemList.index(value.lower())
    else:
        return None

stemColor = ['green','milk white','pink','purple green']
def categorize_stemColor(value):
    value = (value.rstrip()).lstrip()
```

```
    stemColorList = [item.lower() for item in stemColor]
    if value.lower() in (item for item in stemColorList):
        return stemColorList.index(value.lower())
    else:
        return None

fruitColor = ['no','green or blackish purpule','greenish with brown','gr
eenish']
def categorize_fruitColor(value):
    value = (value.rstrip()).lstrip()
    fruitColorList = [item.lower() for item in fruitColor]
    if value.lower() in (item for item in fruitColorList):
        return fruitColorList.index(value.lower())
    else:
        return None

fruitshape = ['club-shaped capsules','oval','no','round']
def categorize_fruitshape(value):
    value = (value.rstrip()).lstrip()
    fruitshapeList = [item.lower() for item in fruitshape]
    if value.lower() in (item for item in fruitshapeList):
        return fruitshapeList.index(value.lower())
    else:
        return None

flowerBehaviour = ['small','small and spikes','large and axillary spikes
','minute and soft spikes','minute,small']
def categorize_flowerBehaviour(value):
    value = (value.rstrip()).lstrip()
    flowerBehaviourList = [item.lower() for item in flowerBehaviour]
    if value.lower() in (item for item in flowerBehaviourList):
        return flowerBehaviourList.index(value.lower())
    else:
        return None

flowerColor = ['ashe/ greenish white','white,crimson,rose-
tinged','white,pink,purple','pale yellow','Red berrish/white/ pinkis whi
te','yellow green']
def categorize_flowerColor(value):
    value = (value.rstrip()).lstrip()
    flowerColorList = [item.lower() for item in flowerColor]
```

```
    if value.lower() in (item for item in flowerColorList):
        return flowerColorList.index(value.lower())
    else:
        return None

leafLook = ['thin and long','large','Oval','kidney shaped']
def categorize_leafLook(value):
    value = (value.rstrip()).lstrip()
    leafLookList = [item.lower() for item in leafLook]
    if value.lower() in (item for item in leafLookList):
        return leafLookList.index(value.lower())
    else:
        return None

leafBase = ['Rounded','Heart-shaped or cordate','wedge-
shaped or cuneate']
def categorize_leafBase(value):
    value = (value.rstrip()).lstrip()
    leafBaseList = [item.lower() for item in leafBase]
    if value.lower() in (item for item in leafBaseList):
        return leafBaseList.index(value.lower())
    else:
        return None

leafLengthRange = ['0.5 -2.5','15-25','2.5-3.5','20-30','8.0-12.0']
def categorize_leafLengthRange(value):
    leafLengthRangeList = [item.lower() for item in leafLengthRange]
    if value.lower() in (item for item in leafLengthRangeList):
        return leafLengthRangeList.index(value.lower())
    else:
        return None

leafColor = ['green','shiny green','brownish green']
def categorize_leafColor(value):
    value = (value.rstrip()).lstrip()
    leafColorList = [item.lower() for item in leafColor]
    if value.lower() in (item for item in leafColorList):
        return leafColorList.index(value.lower())
    else:
        return None
```

```
petiole = ['normal','clasping','sessile']
def categorize_petiole(value):
    value = (value.rstrip()).lstrip()
    petioleList = [item.lower() for item in petiole]
    if value.lower() in (item for item in petioleList):
        return petioleList.index(value.lower())
    else:
        return None

venation = ['parrallel-veined','reticulate/net-veined']
def categorize_venation(value):
    value = (value.rstrip()).lstrip()
    venationList = [item.lower() for item in venation]
    if value.lower() in (item for item in venationList):
        return venationList.index(value.lower())
    else:
        return None

leafShape = ['Acicular','lanceolate','obovate','orbicular or peltate']
def categorize_leafShape(value):
    value = (value.rstrip()).lstrip()
    leafShapeList = [item.lower() for item in leafShape]
    if value.lower() in (item for item in leafShapeList):
        return leafShapeList.index(value.lower())
    else:
        return None

leafType = ['compound','narrow','simple']
def categorize_leafType(value):
    value = (value.rstrip()).lstrip()
    leafTypeList = [item.lower() for item in leafType]
    if value.lower() in (item for item in leafTypeList):
        return leafTypeList.index(value.lower())
    else:
        return None

phyllotaxy = ['Alternative','opposite','peltate or  unifoliate','spiral'
,'whorled']
def categorize_phyllotaxy(value):
    value = (value.rstrip()).lstrip()
    phyllotaxyList = [item.lower() for item in phyllotaxy]
```

```
    if value.lower() in (item for item in phyllotaxyList):
        return phyllotaxyList.index(value.lower())
    else:
        return None

rhizomeBehaviour = ['no','Growing vertically down','knotted and rough']
def categorize_rhizomeBehaviour(value):
    value = (value.rstrip()).lstrip()
    rhizomeBehaviourList = [item.lower() for item in rhizomeBehaviour]
    if value.lower() in (item for item in rhizomeBehaviourList):
        return rhizomeBehaviourList.index(value.lower())
    else:
        return None

partsUsed = ['leaves','rhizome and leaves','root','roots,leaves,bark,flo
wers','tuberous root','whole part']
def categorize_partsUsed(value):
    value = (value.rstrip()).lstrip()
    partsUsedList = [item.lower() for item in partsUsed]
    if value.lower() in (item for item in partsUsedList):
        return partsUsedList.index(value.lower())
    else:
        return None

Target = ['diabetics','diarrhorea','Gastritics','Kidney disease','pneumo
nia, typhoid','Skin diseases','Snake-bite, Stomach pains']
def categorize_Target(value):
    value = (value.rstrip()).lstrip()
    TargetList = [item.lower() for item in Target]
    if value.lower() in (item for item in TargetList):
        return TargetList.index(value.lower())
    else:
        return None

def categorize_stemSize(value):
    if float(value) <= 50:
        return 0
    elif float(value) <= 100:
        return 1
    elif float(value) <= 150:
        return 2
```

```
    elif float(value) <= 200:
        return 3
    elif float(value) <= 250:
        return 4
    elif float(value) <= 300:
        return 5
    elif float(value) <= 350:
        return 6
    elif float(value) <= 410:
        return 7
    else:
        return None

def categorize_rootSize(value):
    if float(value) <= 20:
        return 0
    elif float(value) <= 40:
        return 1
    elif float(value) <= 60:
        return 2
    elif float(value) <= 80:
        return 3
    elif float(value) <= 110:
        return 4
    else:
        return None

def categorize_ReturnTrueFalse(value):
    if value.lower() == 'yes':
        return 1
    else:
        return 0

def DetectDisease(category, propagation, cultivation, seedShape, seedCol
or, rootSystem, rootBehaviur, rootSize, rootColor, stem, stemColor, stem
Size, havingFruits, fruitColor, fruitshape, flowerBlooming, flowerBehavi
our, flowerColor, leafLook, leafBase, leafLengthRange, leafColor, petiol
e,venation, leafShape, leafType, phyllotaxy, rhizomeBehaviour, partsUsed
):
    filename = 'finalized_model.sav'
    loaded_model = pickle.load(open(filename, 'rb'))
```

```
    row = [categorize_category(category),categorize_propagation(propagat
ion),categorize_cultivation(cultivation),categorize_seedShape(seedShape)
, categorize_seedsColor(seedColor),categorize_rootSystem(rootSystem),cat
egorize_rootBehaviur(rootBehaviur),categorize_rootSize(rootSize),categor
ize_rootColor(rootColor),categorize_stem(stem),categorize_stemColor(stem
Color),categorize_stemSize(stemSize),categorize_ReturnTrueFalse(havingFr
uits),categorize_fruitColor(fruitColor),categorize_fruitshape(fruitshape
),categorize_flowerBehaviour(flowerBehaviour),categorize_flowerColor(flo
werColor),categorize_leafLook(leafLook),categorize_leafBase(leafBase),ca
tegorize_leafLengthRange(leafLengthRange),categorize_leafColor(leafColor
),categorize_petiole(petiole),categorize_venation(venation),categorize_l
eafShape(leafShape),categorize_leafType(leafType),categorize_phyllotaxy(
phyllotaxy),categorize_rhizomeBehaviour(rhizomeBehaviour),categorize_par
tsUsed(partsUsed)]
    result = loaded_model.predict([row])
    resultClass = Target[result[0]]
    print('>Predicted=' + resultClass)
    return resultClass


@app.route("/")
def hello():
    return "Hello Azure!"

@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)

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

@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

@app.route('/DiseasePrediction', methods=['POST'])
def DiseasePrediction():
    category = request.json['category']
    propagation = request.json['propagation']
    cultivation = request.json['cultivation']
    seedShape = request.json['seedShape']
    seedColor = request.json['seedColor']
    rootSystem = request.json['rootSystem']
    rootBehaviur = request.json['rootBehaviur']
    rootSize = request.json['rootSize']
    rootColor = request.json['rootColor']
    stem = request.json['stem']
    stemColor = request.json['stemColor']
    stemSize = request.json['stemSize']
    havingFruits = request.json['havingFruits']
    fruitColor = request.json['fruitColor']
    fruitshape = request.json['fruitshape']
    flowerBlooming = request.json['flowerBlooming']
    flowerBehaviour = request.json['flowerBehaviour']
    flowerColor = request.json['flowerColor']
    leafLook = request.json['leafLook']
    leafBase = request.json['leafBase']
    leafLengthRange = request.json['leafLengthRange']
    leafColor = request.json['leafColor']
    petiole = request.json['petiole']
    venation = request.json['venation']
    leafShape = request.json['leafShape']
    leafType = request.json['leafType']
    phyllotaxy = request.json['phyllotaxy']
    rhizomeBehaviour = request.json['rhizomeBehaviour']
    partsUsed = request.json['partsUsed']

    disease = DetectDisease(category, propagation, cultivation, seedShap
e, seedColor, rootSystem, rootBehaviur, rootSize, rootColor, stem, stemC
olor, stemSize, havingFruits, fruitColor, fruitshape, flowerBlooming, fl
owerBehaviour, flowerColor, leafLook, leafBase, leafLengthRange, leafCol
```

```
or, petiole,venation, leafShape, leafType, phyllotaxy, rhizomeBehaviour,
 partsUsed)

    return jsonify({'disease': str(disease)}), 200

app.run()
```