SOIL CLASSIFICATION AND BEST CROP PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

V.SUSHANTHREDDY (38110655)
T.VIVEKANANDA (38110654)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL

ENGINEERING SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

SATHYABAMA

ISO 9001:2015

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE OF ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work **of V.SUSHANTH REDDY (38110655)** and **T.VIVEKANANDA(38110654)** who carried out the project entitled “SOIL CLASSIFICATION AND BEST CROP PREDICTIION under my supervision from November 2020 to March 2021.

Internal Guide

Dr. A. JESSUDOSS M.E., Ph.D.

Head of the Department

Dr. S. Vigneshwari M.E., Ph.D., and Dr. L. Lakshmanan M.E., Ph.D.,

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks the o **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E.,Ph.D.**, **Dean**, School of Computing **Dr.S.Vigneshwari M.E., Ph.D.** and **Dr.L.Lakshmanan M.E., Ph.D. ,** Heads of the Department of Computer Science and Engineering for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincerend a deep sense of gratitude to my Project Guide **Dr. A. JESSUDOSS M.E., Ph.D.,** for her valuable guidance, suggestions aconstant tha at encouragemethat nt paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of The **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

ABSTRACT

Agriculture is the backbone of Indian economy and livelihood to many people. The use of computer science in the field of agriculture will potentially solve many problems faced by farmers. Farmers often choose crops for their field based on their own experience and instinct. This sometimes leads to loss and less yield. If the selection of crops is done with productivity data of the entire region, it may lead to better results.However all the crops cannot be cultivated in a particular soil. So the soil must be analysed and crops must be suggested based on the type of soil. Many soil classification techniques involve testing in laboratories whichmight not be affordable and available to all the farmers.

This work suggests an idea that is useful and easily accessible to all the farmers in India without any need of hardware. A list of crops with their success rate will be suggested to the farmer when the region of agriculture and soil image (used for agriculture) are given as inputs. This list of crops are both profitable and produce more yield in that region.

The results obtained are promising. An accuracy of 94% is achieved in the soil classification module.The success rate for the crops obtained are realistic with the agricultural practices in the region. The web application developed is extremely user friendly and easy to use by the farmers.

.

.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
GPS	Global Positioning System
K-NN	K-Nearest Neighbors
PH	Power of Hydrogen
RGB	Red Green Blue
SRDI	Soil Resources Development Institute
SVM	Support Vector Machine

VGG 16

Visual Geometry Group 16 **xii**

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Agriculture is the primary source of livelihood for about 58% of the population of India. Continuous efforts have been taken to develop this sector as the whole nation depends on it for food. For thousands of years, we have been practicing agriculture but still, it remained underdeveloped for a long time. After the green revolution, we became self-sufficient and started exporting our surplus to other countries.

Earlier we used to depend completely on monsoon for the cultivation of food grains but now we have constructed dams, canals, tube-wells, and pump-sets. Also, we now have a better variety of fertilizers, pesticides, and seeds, which help us to grow more food in comparison to what we produce during old times. With the advancement of technology, advanced equipment, better irrigation facilities agriculture started improving. Furthermore, our agriculture sector has grown stronger than many countries and we are the largest exporter of many food grains.

In recent years, farmers are suffering financially and are facing many hardships. This is due to various reasons such as urbanisation, globalisation, pollution, water scarcity, less rainfall, low fertility of soil, drastic climatic changes, political and economic reasons, poverty, lack of technological assistance etc.

Addressing their needs through technology is the need of the hour.

Though we have very less to contribute to improvise the natural factors to help agriculture, we have a lot to contribute to this sector through computer science and technology. Internet of Things(IoT), Artificial Intelligence, smart agriculture, Agricultural Engineering, Irrigation Engineering are some of the fields that contributed to the development of agriculture in recent years.

With large scale increase in the availability of data, machine learning, deep learning, big data analytics can help in solving various problems. Machine learning has emerged with big data technologies and high-performance computing to create new opportunities for data intensive science in the multi-disciplinary agritechnologies domain. The works can be categorized as (a) crop management, including applications on yield prediction, disease detection, weed detection crop quality, and species recognition; (b) livestock management, including applications on animal welfare and livestock production; (c) water management; and (d) soil management. By applying machine learning to sensor data, farm management systems are evolving into real time artificial intelligence enabled programs that provide rich recommendations and insights for farmer decision support and action.

There are many ways to suggest crops suitable for a farm land. It can be based on the climate or soil or the crop that produces high profit in that region. We want to suggest crops considering all these factors. We also want soil classification to be done easily with android camera images so that the laboratory tests can be avoided to identify the type.

1.2 PROBLEM STATEMENT

While analysing the various problems faced by farmers,choosing crops for their land appears to be a concerning problem.Crops must be chosen not only based on the soil and climate but also on various other factors like usage of the crop in the particular area, cost, revenue, how much the crop is exported or imported. In this project, our aim is to suggest crops to farmers such that it leads to maximum production and profit.The problem statement is to provide a user friendly application that classifies the soil into four types (Alluvial,Black,Clay,Red) with a simple camera image and suggests the best crops which will give higher yield and profit.

1.3 OBJECTIVES

- To classify soil image into one of the four categories precisely (red, alluvial, black, clay).
- To implement different models and find the best suitable model for soil image classification.
- To suggest crops for a region considering weather and past production and profit rate.
- To give success rate for each crop cultivable in that soil and region

1.4 OVERVIEW OF THESIS

Chapter 2 elaborates on the related work in soil classification and crop suggestion domain. The issues in the previous work are analysed and it has been rectified. The requirements of the system (both functional and non-functional) are identified and specified in Chapter 3. The architecture diagram, use case diagram and activity diagram for the entire project is given in Chapter 4. In chapter 5, all the three modules in the project are discussed elaborately. The implementation of the system, the intermediate outcomes, evaluation are specified in Chapter 6. Results, screenshots and test cases are given in Chapter 7. In chapter 8, conclusion and future work of our project are specified.

CHAPTER 2

LITERATURE SURVEY

2.1 SOIL CLASSIFICATION

Srunitha K, S.Padmavathi created a soil classification model that uses Support Vector Machine based classification.Almost all countries export their products, Countries which export agricultural products depend on soil characteristics.Hence classifying soil, based on their characteristics is very important to reduce the product quantity loss.The nature of soil is influenced by many factors. Some of them are power of hydrogen (PH), Exchangeable sodium percentage, moisture content etc. depending on their amount in soil they show different characteristics and that varies for different region. The manual segmentation and classification methods are time consuming, require efficient people and expensive also .With the emerging of image processing and machine learning we can efficiently classify the soil sample in to groups and hence we can automate the classification process.Soil classification includes steps like image acquisition, image pre processing, feature extraction and classification.

2.1.1 SVM(Support Vector Machine)

SVM models are mainly used for analyzing the data for regression and classification. For a set of training examples it belongs to either one of the two categories, a support vector machine algorithm for training generates a model which tells the new thing falls into which category by a non-probabilistic binary classifier.

The SVM model is the depiction of points in space which are mapped.

Thus, the data of different types are separated by as wide as possible.

2.1.2 BASIC SEGMENTATION METHOD

The segmentation process splits the region of interest from that of non-interest regions.A two class classifier is required for classifying pixels in feature space considering segmentation as a two class problem. Method of segmentation includes,

1) Training data with one or a few images having objects. Traditional segmentation or by manually foreground and background regions are splitted. Pixels in objects are marked using I and I-which produces RGB color histogram. Color values are also marked.

2) Prepare for SVM the training data,

(xi,yi) , + + +l,if Xi€[. xi is -1,if X,EI. +- Xi€[VI , yi= a color vector.

2.1.3 TRANSFORMATION

The transformation phase includes color quantization,low pass filter and gabor filter techniques.In color quantization they create a new image visually similar to that of the original image. Thus, it reduces the distinct colors used in the original image. Then a low-pass filter passes frequency below the cutoff frequency and attenuates the higher frequency. The attenuated frequency depends on the filter design.For the extraction of features from an image Gabor filter with different frequencies are useful. In image processing a 2-D Gabor filter is used for feature extraction especially while doing segmentation and analyzing texture.

2.1.4 STATISTICAL PARAMETERS

1) Mean = Neighboring/Total
2) Std=√Mean

2.1.5 WORKING OF THE SYSTEM

1. Applying the transformation (includes low mask filter, color quantization , histogram) to the original image.
2. Using statistical measures to analyse the color ,texture and shape.
3. Finding the distance with Euclidean distance formula.

The classifications of non-sandy soils are better classified with SVM.

2.2 BEST CROP PREDICTION

Sk Al Rahman, Kaushik Mitra, S.M. et al used dataset, collected from 500 soil series in Bangladesh which is identified by Soil Resources Development Institute (SRDI).Soil series means group of soils which is formed from the same kind of parent materials and remains under the similar conditions of drainage, vegetation time and climate. It also has the same patterns of soil horizons with differentiating properties.Each type of soil can have different kinds of features and different kinds of crops grow on different types of soils. We need to know the features and characteristics of various soil types to understand which crops grow better in certain soil types.The main purpose of the proposed work is to create a suitable model for classifying various kinds of soil series data along with suitable crops suggestions for certain 5 areas of certain Upazila of Bangladesh. Here, Several machine learning algorithms such as weighted k-Nearest Neighbor (k-NN), Bagged Trees and Gaussian kernel based Support Vector Machines (SVM) are used for soil classification.The method involves two phases: training phase and testing phase. Two datasets are used: Soil dataset and crop dataset. Soil dataset contains class labeled chemical features of soil which include salinity,pH values and iron,magnesium content etc. This system mainly uses three methods namely, Weighted K-NN, Gaussian Kernel based SVM, and Bagged Tree.

2.2.1 WEIGHTED K-NN

It is a refinement of the k-NN classification algorithm.It weighs the contribution of each of the k neighbors according to their distance to the query point, giving greater weight wi to closer neighbors. It makes use of all training examples not just k if weighting is used.The algorithm then becomes a global one.

The only disadvantage is that the algorithm will run more slowly.

2.2.2 SVM

SVM is a supervised machine learning algorithm which works based on the concept of decision planes that defines decision boundaries. A decision boundary separates the objects of one class from the object of another class. Kernel function is used to separate non-linear data by transforming input to a higher dimensional space.

The Gaussian radial basis function kernel is used in this method.

2.2.3 BAGGED TREE

Here they have used a bagged decision tree ensemble classifier which consists of 30 trees. Bagging generates a set of models each trained on a random sampling of the data. The predictions from those models are aggregated to produce the final prediction using averaging.

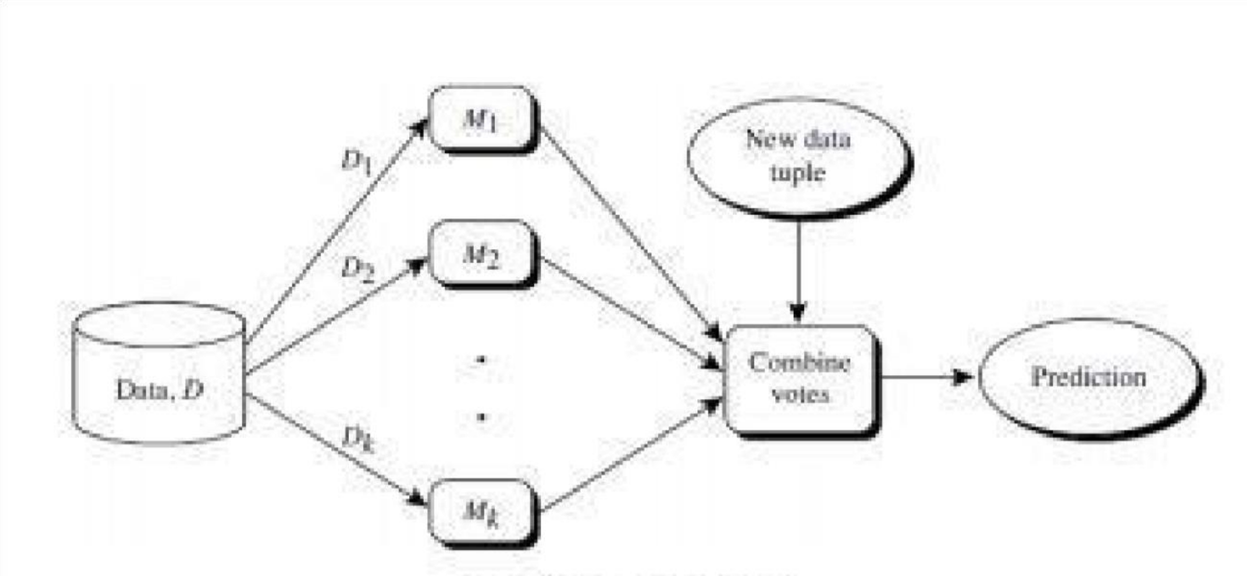


Fig 2.1.Bagged Tree

They used two-third of the samples collected for training the model and the rest are used for testing.In their research,they worked with soils series of six upazillas of Khulna district, Bangladesh. Upazillas are:‘Rupsha’, ‘Dighalia’,

‘Fultola’, ‘Koyra’, ‘Dakop’,‘Terokhada’. There are a total of 15 soil series in this 6 Upazillas.In our work, we have worked with 4 soil classes;they are Alluvial,Black,Clay and Red.

The soil classification accuracy and also the recommendation of crops for specific soil provided by this model is more appropriate than many existing methods.One of the drawbacks of the model is they have restricted it to soil types only to few districts.

CHAPTER 3
REQUIREMENTS ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

Below are the functional requirements of this project.

- The system must enable the user to upload images of any type and quality.

- Time taken to load the website must be less.
- Website must be user-friendly.
- The flow of the process must be well defined and clear for naive users.
- The system must be able to classify soil images taken even from simple android mobile phones.
- Soil images must be classified more accurately.
- Time taken for image classification must be less.
- System must be able to differentiate crops present/absent in the dataset.
- Crops that are not present in the dataset can be suggested to the users as other options. ● The suggestion and success rate calculation must be fast.

3.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements are the hardware and software that a user needs to have in order to effectively use this system for his/her advantage.

3.2.1 Hardware Requirements

The hardware requirements for this project for user includes

- Intel i5 or i7 processor
- 4-8 GB RAM
- For big dataset, 16GB RAM is required
- At Least 20GB of free space in hard disk

3.2.2 Software Requirements

The software requirements includes :

- 64 bit Window 8 or 10
- Python 3.x
- Django
- Tensorflow 3.x

- Other python libraries like numpy, keras, seaborn etc

CHAPTER 4

SYSTEM DESIGN

4.1 OVERALL ARCHITECTURE DIAGRAM

The overall Architecture diagram for the proposed system is shown in Fig 4.1. The proposed work is split into different processing phases namely Soil Classification, Suitable Crop Suggestion and Best Crop Prediction. These working phases execute in the depicted flow to produce the list of crops with success rate as output from the input soil image and region.

The Soil Classification module is designed to classify the different types of soil using a deep learning model.This model inputs soil images from the user and states the type of the soil as output. The output is one of the following: Alluvial soil, Red soil, Black soil, Clay soil.

The Suitable Crop Suggestion module gets the type of soil from the previous model as input and gives the suitable crops cultivable in that area. This module provides a list of suitable crops for the soil type fetched from the database or local storage.

The Best Crop Prediction module aims to find the crops that are best for their region,so that the farmers can get a maximum profit by cultivating these crops. This model is fed with the list of crops from the previous model, and it will output a list of best crops and success rate of those crops. The model is trained using data for the past 10 years collected from various trusted sources. The success rate of the crop is predicted based on the following parameters present in the dataset which include Imports & exports ,Production,Production per unit area and Gross production value of the crop in the past 10 years.

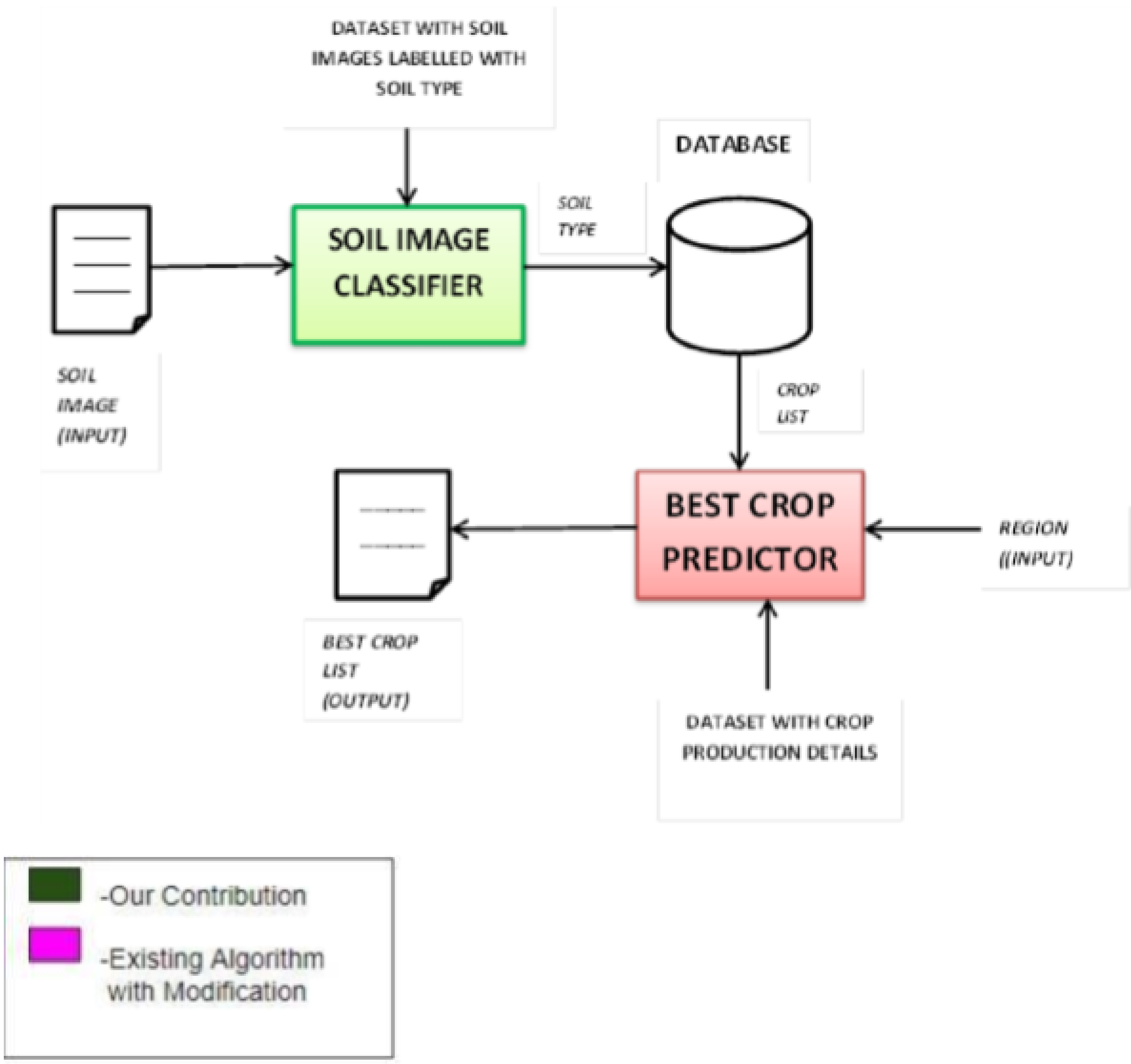


Fig 4.1. Overall Architecture Diagram

4.2 USE CASE DIAGRAM

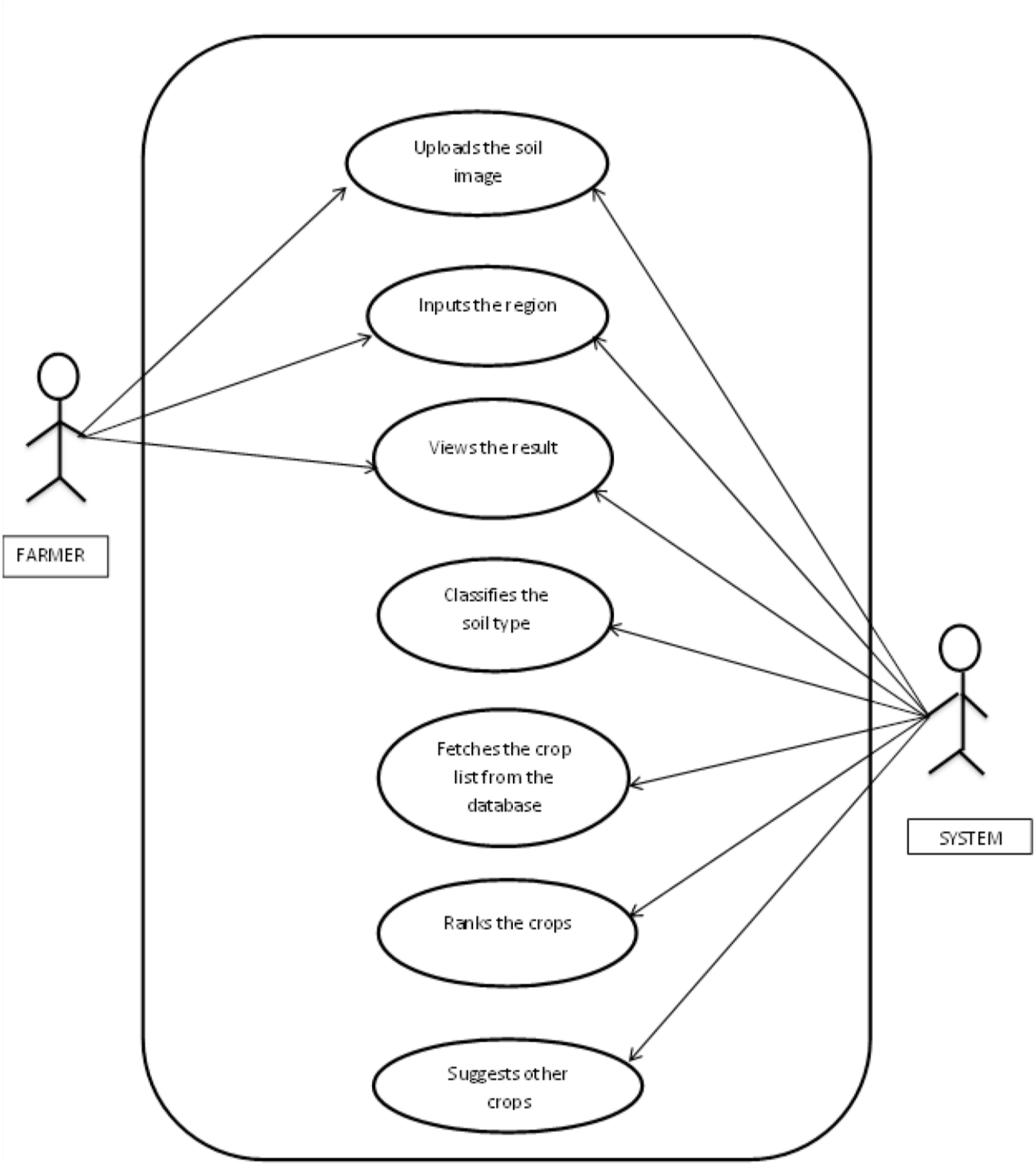


Fig 4.2.Use case diagram of web app

4.3 FLOW DIAGRAM

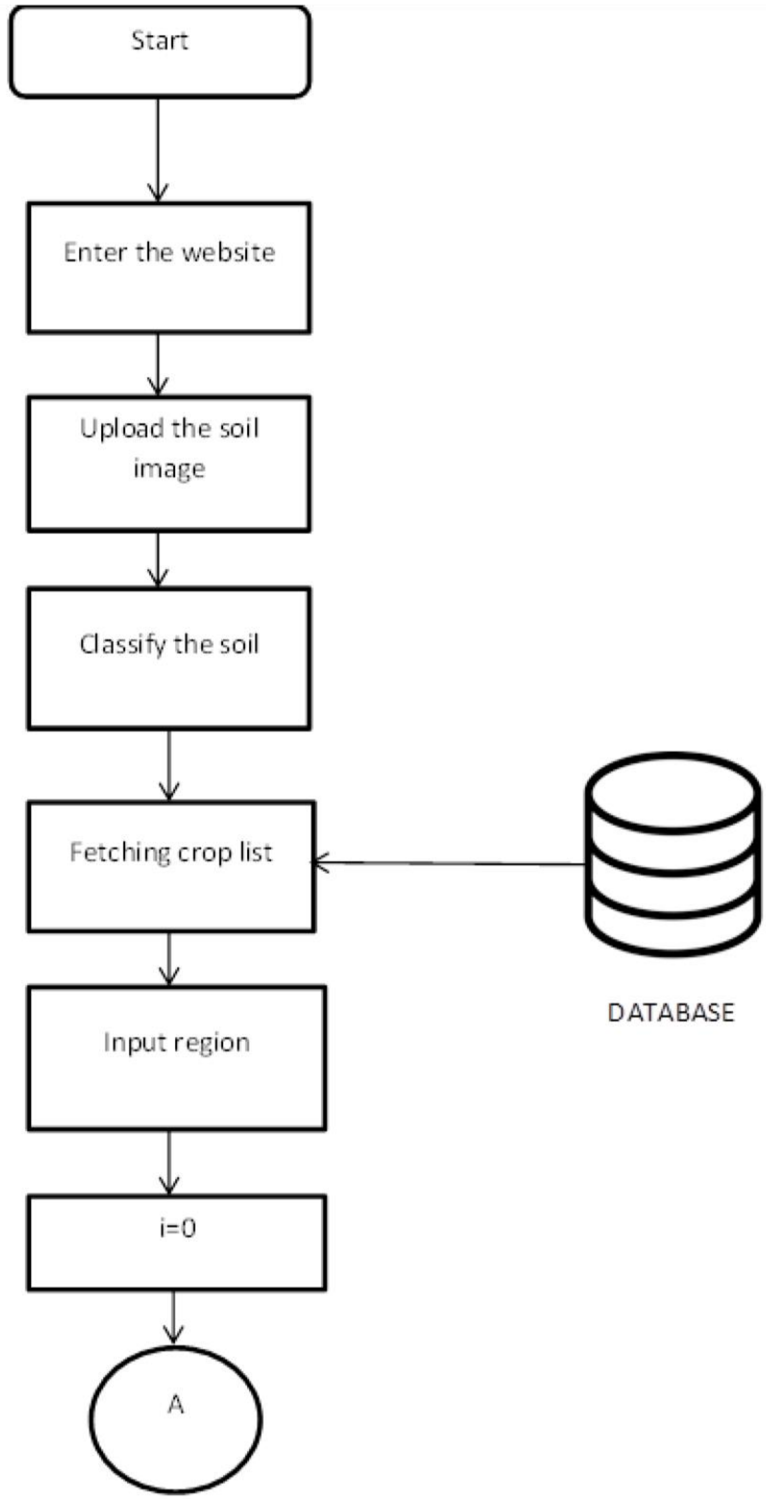


Fig 4.3. First part of Flow diagram of the web app

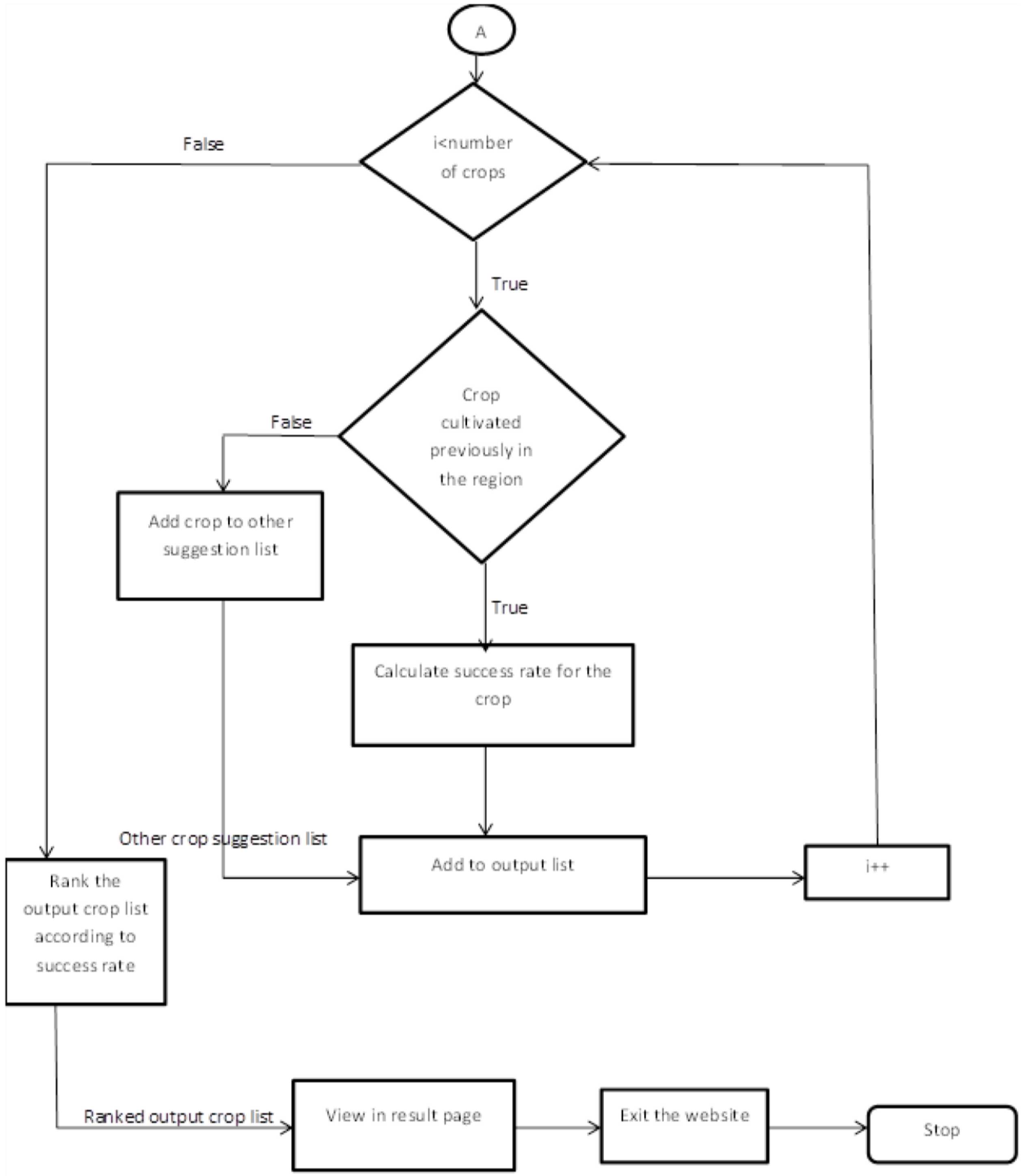


Fig 4.4. Second part of Flow diagram of the web app

CHAPTER 5

MODULE DESIGN

5.1 SOIL CLASSIFICATION

In this model, the aim is to classify the different types of soil using a deep learning model. This model inputs soil images from the user and states the type of the soil as output. We used SVM and CNN architectures like LeNet, AlexNet, VGG 16, ResNet for soil image classification and evaluated the accuracy of each of the classifiers. The CNN model that produced the highest accuracy was chosen for the soil classification. The models are trained with the Kaggle soilnet dataset that includes 903 images of four soil types, namely red, alluvial, black and clay.

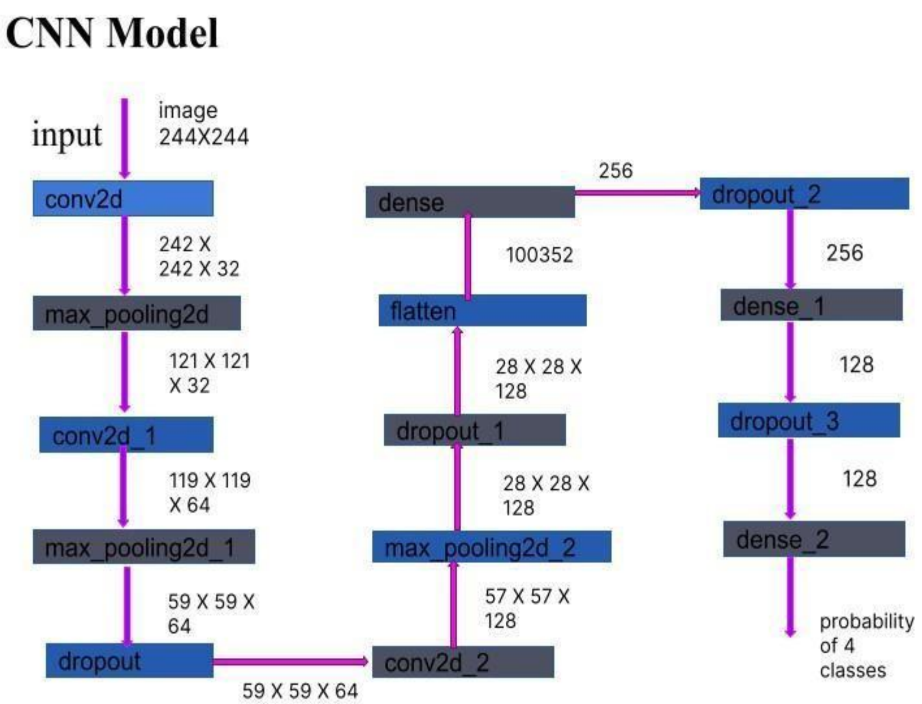


Fig 5.1. Architecture of Custom CNN model

The CNN architecture depicted in Fig 5.1 is built with conventional layer by layer feature extraction techniques.There are three convolutional layers with ReLU activation function followed by max pooling. Then the feature map is flattened. Finally there are three fully connected layers with ReLU activation function. Dropouts are added to avoid overfitting. The final dense layer has softmax activation function.

5.2 SUITABLE CROP SUGGESTION

As shown in Fig 5.2, the type of soil from the previous model is used to decide the suitable crops cultivable in that area. This module provides a list of suitable crops for the soil type fetched from local storage. This list was collected from authorised sources. Table 5.1 shows the list of crops for the four types of soil.

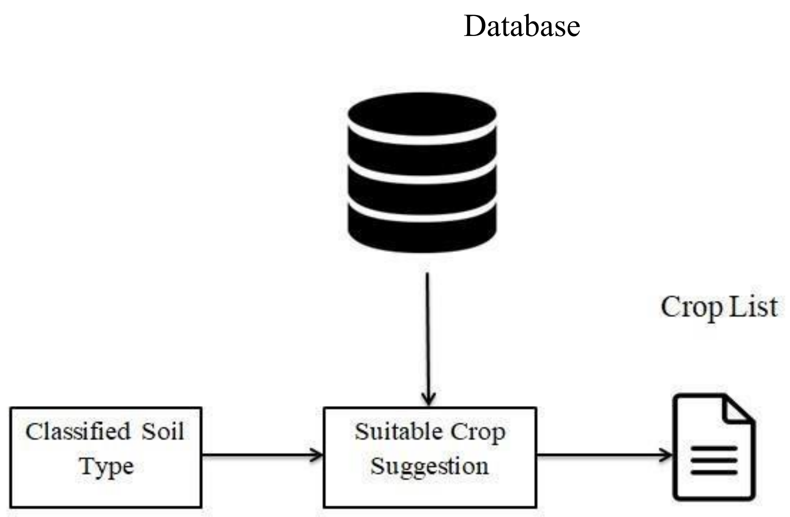


Fig 5.2. Crop suggestion module

Table 5.1. Crops suitable for each soil type

TYPE OF SOIL	CROPS
Alluvial soil	Wheat, Rice, Jute, Coconut, Sugarcane, Pulses, Oilseed, Groundnut.
Red soil	Wheat, Cotton, Pulses, Coconut, Tobacco, Millets, Oilseed, Potato, Groundnut, Rice, Orchards.
Black soil	Cotton, Pulses, Soyabean, Millets, Linseed, Tobacco, Barley, Sugarcane, Rice.

Clay soil	Wheat, Rice, Sorghum, Jowar, Groundnut.

5.3 BEST CROP PREDICTION

The aim of this model is to find the crops that are best for their region, so that the farmers can get a maximum profit by cultivating these crops. This model is fed with the list of crops from the previous model, the region as input and it will output a list of best crops and success rate of those crops. The model is trained using data for the past 10 years collected from various trusted sources. Algorithms used will be customized using multiple linear regression and customized K fold method.

The success rate of the crop is predicted based on the following parameters present in the dataset:

- Imports & exports of the crop in the past 10 years
- Production of the crop in the past 10 years
- Production per unit area of the crop in the past 10 years, for the concerned area
- Gross production value of the crop in the past 10 years

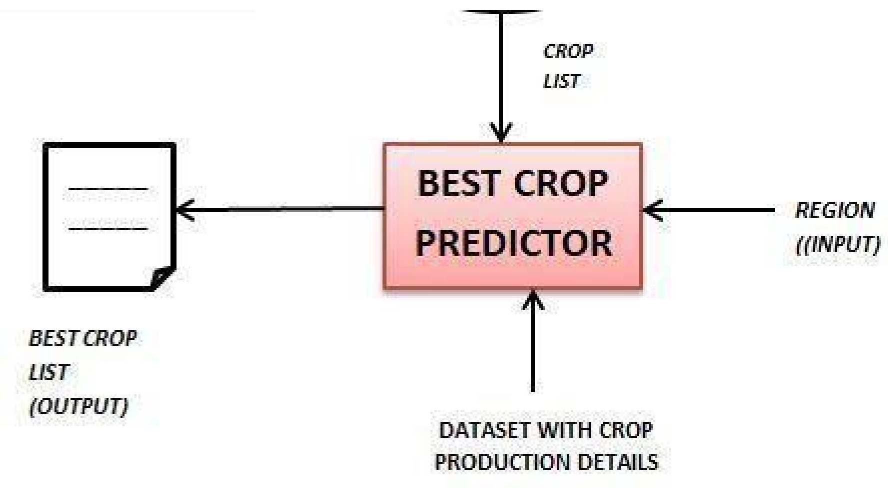


Fig 5.3. Best crop prediction Architecture

Ten different multiple linear regressions are done to predict various parameters like imports, exports, gross production, production per unit area and production. These regressions are done beforehand and the predicted values are stored in separate csv files. These values are used in future calculations of success rate.

CHAPTER 6

IMPLEMENTATION DETAILS

6.1 SOIL CLASSIFICATION

6.1.1 Dataset Description

The dataset is obtained from the following kaggle URL.

https://www.kaggle.com/omkargurav/soil-classification-image-data

The data set consists of 903 RGB images labelled as "Alluvial Soil", "Red Soil", "Clay Soil", "Black Soil".

The number of images in each category are given in table 6.1.

Table 6.1. Soil image dataset split up

	Alluvial	Black	Red	Clay	Total
Train	175	212	184	144	715
Test	48	47	46	47	188

6.1.2 CNN Model

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. While there are many predefined CNN models available, a custom CNN model has been developed to accommodate for the soil images dataset.

Due to the low number of images in the dataset, data augmentation is done. Then the CNN model is created and trained using the training dataset. The number of epochs for training is varying since callback early function is used. Hence if there

```
#to avoid overfitting
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
```

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
## fit model
FH=model.fit(train_data,validation_data= test_data,batch_size=32,epochs = 100,callbacks=[ear
ly])
```

is no more improvement in the loss parameter, the epochs are terminated. Maximum number of epochs is fixed as 100.

Images are converted to 244 X 244 size with RGB color values. The colors are retained as they are important in soil classification. Adam optimisers are used to adjust the weight parameters. Accuracy is monitored with Sparse categorical cross entropy function. The training code is shown in Fig 6.1.

Fig 6.1. Training code

While testing, the images are converted to 244 X 244 size and predicted with the model created.

The CNN architecture is as follows:

1) First Convolutional layer is added with 32 filters of 3X3 size with Relu activation function.

2) 32 feature maps are generated from this layer each of size 242X242. 3) Max pooling layer is added with pool size 2X2

4) The above layer generates 32 feature maps of size 121X121.

5) Second Convolutional layer with 64 filters of 3X3 size with Relu activation function.

6) 64 feature maps are generated from this layer each of size 119X119.

7) Max pooling layer is added with pool size 2X2

8) The above layer generates 64 feature maps of size 59X59.

9) 30% of the above connections are dropped out to avoid overfitting.

10) Third Convolutional layer with 128 filters of 3X3 size with Relu activation function.

11) 128 feature maps are generated from this layer each of size 57X57.

12) Max pooling layer is added with pool size 2X2

13) The above layer generates 128 feature maps of size 28X28.

14) 20% of the above connections are dropped out to avoid overfitting.

15) The above 28X28X128 output is flattened into a one dimensional array of size 100352.

16) A fully connected layer of output size 256 and Relu activation function is added.

17) 15% of the above connections are dropped out.

18) A fully connected layer of output size 128 and Relu activation function is added.

19) 1% of the above connections are dropped out.

20) A final fully connected layer with four output classes(4 soil types) is added with softmax activation function.

The summary of our CNN model is given in Fig 6.2.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 242, 242, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 121, 121, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 119, 119, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 59, 59, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 59, 59, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 57, 57, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 28, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 256)               25690368
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 516
=================================================================
Total params: 25,817,028
Trainable params: 25,817,028
Non-trainable params: 0
_________________________________________________________________
```

Fig 6.2. CNN model summary

```
#evaluate model
model.evaluate(test_data)
```

```
6/6 [==============================] - 3s 503ms/step - loss: 0.3086 - accuracy: 0.8989
```

```
[0.3086017966270447, 0.8989361524581909]
```

Fig 6.3. Training of CNN model

	precision	recall	f1-score	support
0.0	1.00	0.71	0.83	7
1.0	0.67	1.00	0.80	6
2.0	1.00	0.90	0.95	10
3.0	1.00	1.00	1.00	9
accuracy			0.91	32
macro avg	0.92	0.90	0.90	32
eighted avg	0.94	0.91	0.91	32

Fig 6.4. Confusion matrix of the model

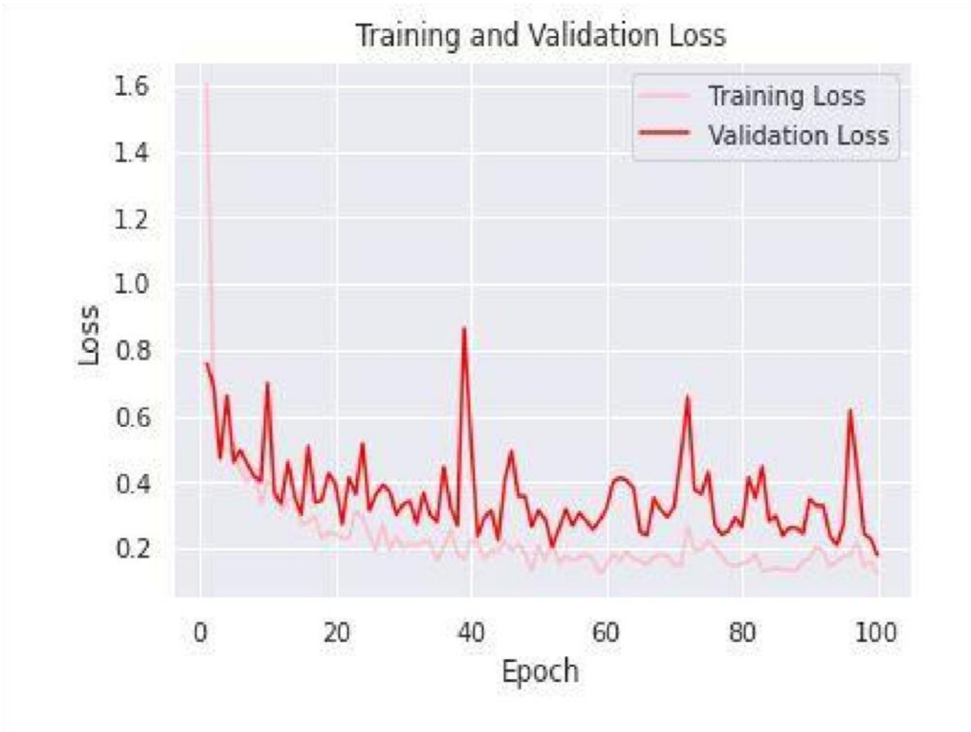


Fig 6.5. Training and validation loss graph

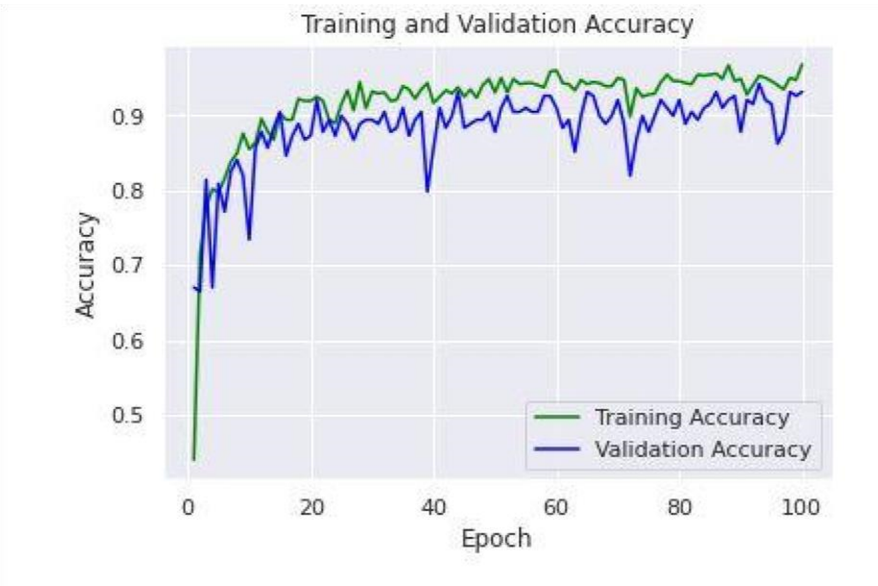


Fig 6.6. Training and validation accuracy graph

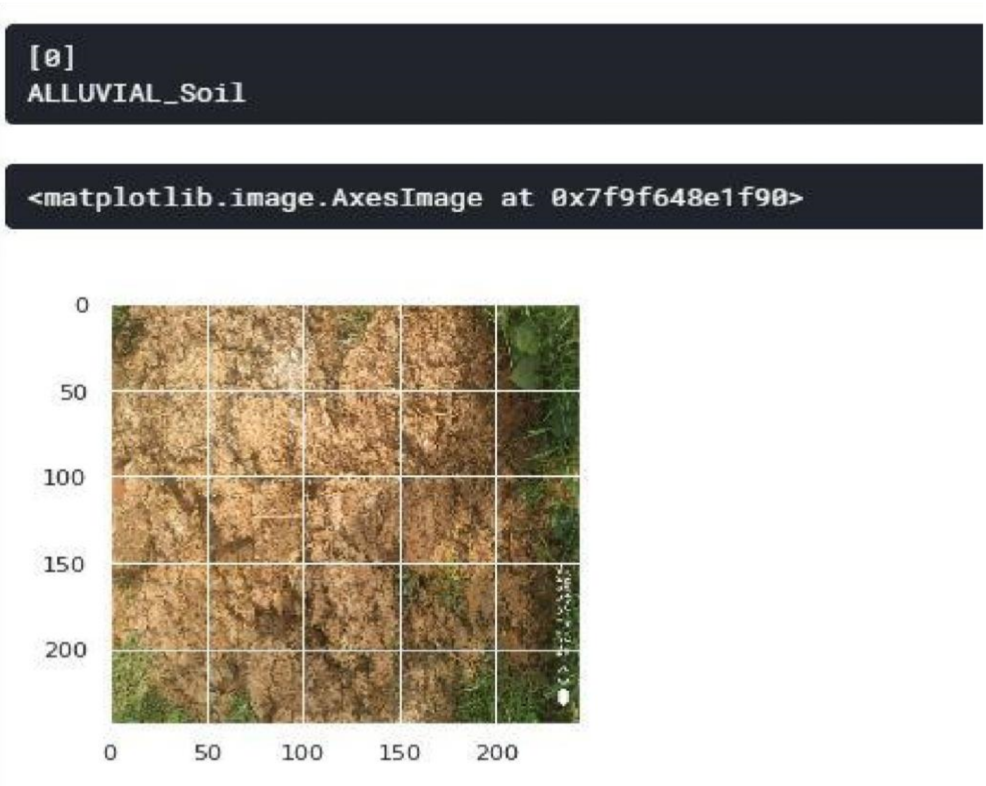


Fig 6.7. Testing the model with alluvial soil image

6.1.3 Explanation of the layers

2D convolutional layer

The input of the 2D convolutional layer is 3 dimensional. A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input. A convolution is a linear operation that involves the multiplication of a set of weights with the input.

Dense layer

Dense layer is the regular deeply connected neural network layer. Dense layer does the below operation on the input and returns the output.

output = activation(dot(input, kernel) + bias) where,

- input represent the input data
- kernel represent the weight data
- dot represent numpy dot product of all input and its corresponding weights ● bias represent a biased value used in machine learning to optimize the model ● activation represents the activation function.

Dropout

It refers to dropping out units (hidden and visible) in a neural network. It is a simple way to prevent neural networks from overfitting.

Flatten layer

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer.

Maximum Pooling layer

It is a pooling operation that calculates the maximum value in each patch of each feature map.

ReLU activation function

It stands for Rectified Linear Unit.

$$f(x) = \begin{cases} x & if\ x > o \\ 0 & otherwise. \end{cases}$$

It returns the modulus value.

Softmax activation function

Softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

6.1.4 Details of other models

Table 6.2. Details of other algorithms implemented

Algorithm	Details
SVM	Supervised learning model in which images are converted into 2 arrays for processing.

VGG16	It consists of two 2D convolutional layers followed by average pooling layers. Finally there are three fully connected dense layers. The first two have tanh activation function and last one has softmax activation function.
Alexnet	AlexNet contains eight layers. The first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers.
Lenet5	It is a network with 16 layers which have the trainable parameters. There are also other layers like Max pool layer which do not have the trainable parameters. Similar to AlexNet, it has only 3x3 convolutions.

6.1.5 Outputs of other models

```
     original  predicted
0           0          0
1           0          1
2           0          0
3           0          2
4           0          2
..        ...        ...
183         3          3
184         3          3
185         3          3
186         3          3
187         3          3

[188 rows x 2 columns]
Accuracy of data is 0.8351063829787234
```

Fig 6.8. SVM soil classification

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 240, 240, 6)       456
_________________________________________________________________
average_pooling2d (AveragePo (None, 120, 120, 6)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 116, 116, 16)      2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 58, 58, 16)        0
_________________________________________________________________
flatten (Flatten)            (None, 53824)             0
_________________________________________________________________
dense (Dense)                (None, 120)               6459000
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 340
=================================================================
Total params: 6,472,376
Trainable params: 6,472,376
Non-trainable params: 0
_________________________________________________________________
```

Fig 6.9. Lenet5 model summary

```
#evaluate model
model.evaluate(test_data)
```

```
6/6 [==============================] - 2s 361ms/step - loss: 1.2078 - accuracy: 0.5851
```

```
[1.2078230381011963, 0.585106372833252]
```

+ Code   + Markdown

Fig 6.10. Alexnet model training

```
Train on 500 samples, validate on 150 samples
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32
(from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
500/500 [==============================] - 211s 421ms/sample - loss: 2.2959 - acc: 0.3660 - val_loss: 1.1761 - va
l_acc: 0.5000
Epoch 2/5
500/500 [==============================] - 206s 412ms/sample - loss: 0.9819 - acc: 0.5800 - val_loss: 0.6980 - va
l_acc: 0.5867
Epoch 3/5
500/500 [==============================] - 210s 421ms/sample - loss: 0.5589 - acc: 0.7720 - val_loss: 0.6686 - va
l_acc: 0.7400
Epoch 4/5
500/500 [==============================] - 211s 423ms/sample - loss: 0.4565 - acc: 0.8260 - val_loss: 0.4699 - va
l_acc: 0.8400
Epoch 5/5
500/500 [==============================] - 206s 412ms/sample - loss: 0.4210 - acc: 0.8320 - val_loss: 0.4022 - va
l_acc: 0.8200
150/150 [==============================] - 16s 109ms/sample - loss: 0.4022 - acc: 0.8200
Test Loss 0.4022025998433431
Test Accuracy 0.82
```

Fig 6.11. VGG16 soil classification

6.1.6 Comparison of the results

Table 6.3. Comparison of the implemented algorithms

ALGORITHM	**ACCURACY**	**FINDINGS**
CNN	0.94	Conventional layer by layer feature extraction resulted in high accuracy.
SVM	0.83	SVM for image classification is better than CNN only when there is less data
VGG 16	0.82	It is quite slow and occupies more space.
AlexNet	0.58	Due to the large number of parameters, it suffered from overfitting.

Lenet5	0.73	It is the oldest CNN architecture and the performance is not better than new models

6.2 BEST CROP PREDICTION

6.2.1 Dataset description

The dataset is obtained from the following Github link:

https://github.com/amanparmar17/Cultivo/tree/master/datasets

The data set consists of:

- Details of the 11 crops selected.
- District wise crops total to 309 in number.
- Information about area harvested, yield and production of 10-11 crops(required for the main analysis and prediction) for 10 years.
- Details of import, export, production quantity of 10 crops in India.
- Data about production, import quantity, export quantity, domestic supply, feed, seed, food, food supply quantity, protein supply,fat supply about 10-11 crops for 10 yrs.
- Information about gross production value,net production value for 10-11 crops for 10 years.

6.2.2 Multiple linear regression

Multiple linear regression refers to a statistical technique that is used to predict the outcome of a variable based on the value of two or more variables.The variable that we want to predict is known as the dependent variable, while the variables we use to predict the value of the dependent variable are known as independent.

Multiple Linear Regression for our dataset

1) Gross production and Net production dataset:

It consists of the following 6 variables:

- Gross_Production_Value_current_million_SLC

- Gross_Production_Value_constant_2004_2006_1000_dollar
- Net_Production_Value_constant_2004_2006_1000_dollar
- Gross_Production_Value_constant_2004_2006_million_SLC
- Gross_Production_Value_current_million_US_dollar
- Gross_Production_Value_constant_2004_2006_million_US_dollar We will keep one variable as a dependent variable and other 5 variables as independent variables. We will repeat the following process for all the six combinations of independent and dependent variables.

For each crop:

dependent variable is predicted by linear regression

mean is calculated.

All the predicted and calculated means are stored for each crop in a separate csv file.

Scatter plots :

Dependent variable is scattered (dots) and independent variables are used to draw the plots (lines) in the graph. The following are multi-dimensional graphs(6 dimensions). Hence only the dependent and independent variables are specified for easier understanding.

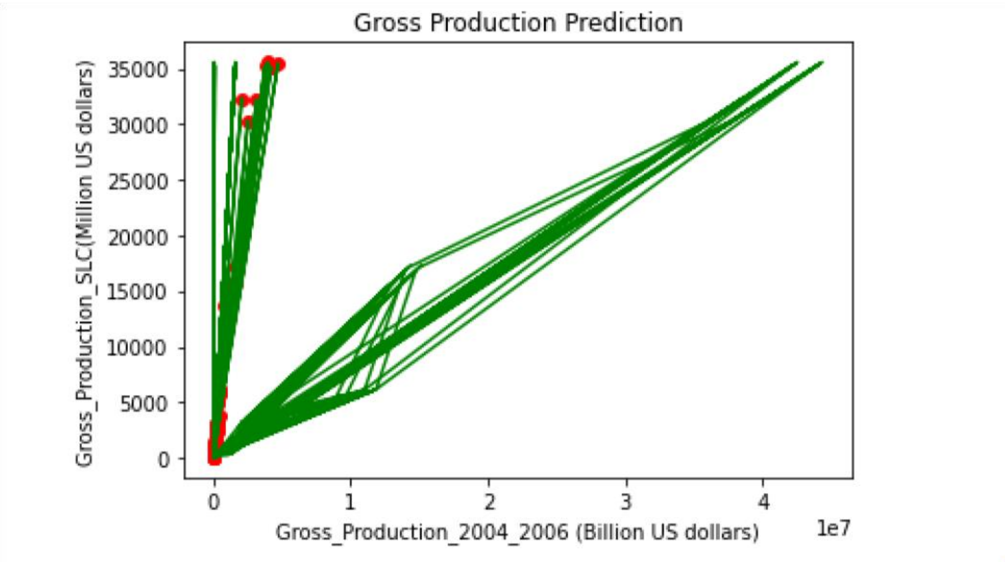


Fig 6.12 Multiple Linear Regression(1)

Scatter plot of Gross production Prediction

```
(80, 5)
(80,)
[3.22530732e+03 3.51189022e+04 6.03035681e+03 1.38021990e+03
 1.11034959e+03 1.68169231e+03 1.45780794e+04 1.44473980e+03
 3.42712710e+01 1.68686843e+04 3.56011326e+02 3.79130248e+02
 2.51232298e+02 3.43587768e+02 1.12561980e+03 3.05178553e+02
 2.49488318e+03 2.69260928e+02 1.58247454e+03 1.68704885e+04]
[ 3227.81947863 35121.92483289  6024.55940941  1380.70035464
  1111.08739713  1681.64098207 14571.31448312  1445.11565354
    35.749643   16863.01731306   356.3415752    379.44703284
   252.63639783   343.93029838  1127.43378392   305.72071754
  2494.28102229   270.66054851  1582.31176928 16865.40163489]
Final R2 score :
0.999999892325306
```

Fig 6.13 R2 score of production related multiple linear regression

2) Production and area dataset:

For each district:

For each crop:

Dependent variable is production per unit area

Independent variable is area

Predicted production/area's mean is calculated(pred_mean)

Production/area' mean is calculated(org_mean)

Pred_mean and org_mean are stored in a separate csv file.

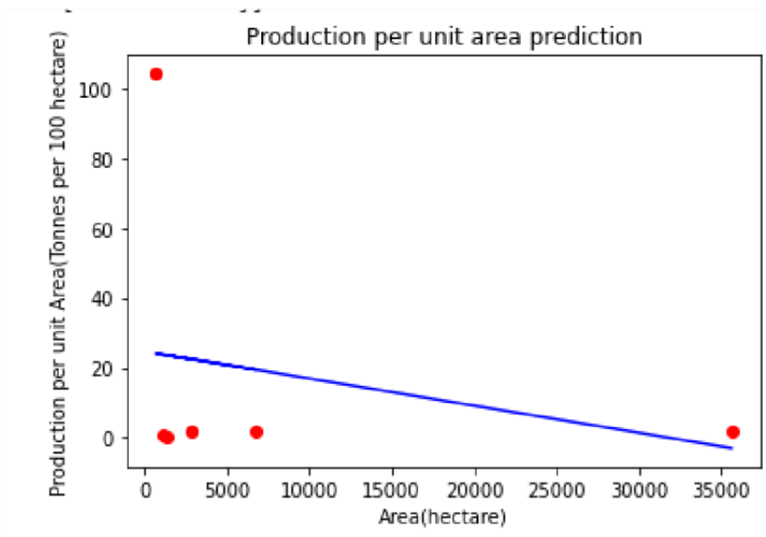


Fig 6.14. Multiple Regression Graph(2)

Dependent variable : Production per unit area

Independent variable : Area

3) Import, export and production dataset:

For each crop:

Independent variable is seed

Dependent variable is production

Production(pred_prod) is predicted and mean is calculated.

For each crop:

Independent variables are stock, export, domestic, production

Dependent variable is import

Import(pred_import) is predicted and mean is calculated.

For each crop:

Independent variables are production, imports, stock, seed

Dependent variable is export

Export(pred_export) is predicted and mean is calculated.

For each crop, mean is calculated for import, export and production(original means).

The original means and predicted means are stored in a separate csv file.

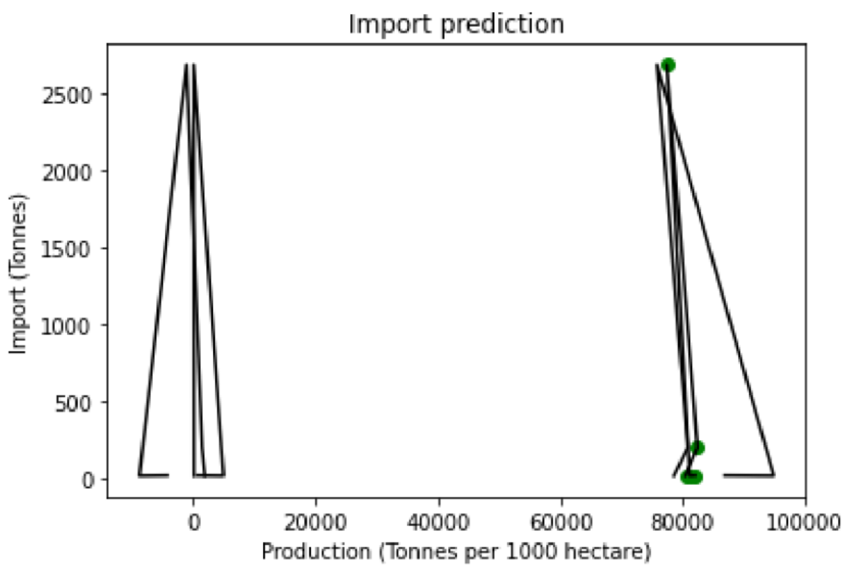


Fig 6.15. Multiple Regression Graph(3)

Dependent variable : Import

Independent variables : stock, export, domestic, production

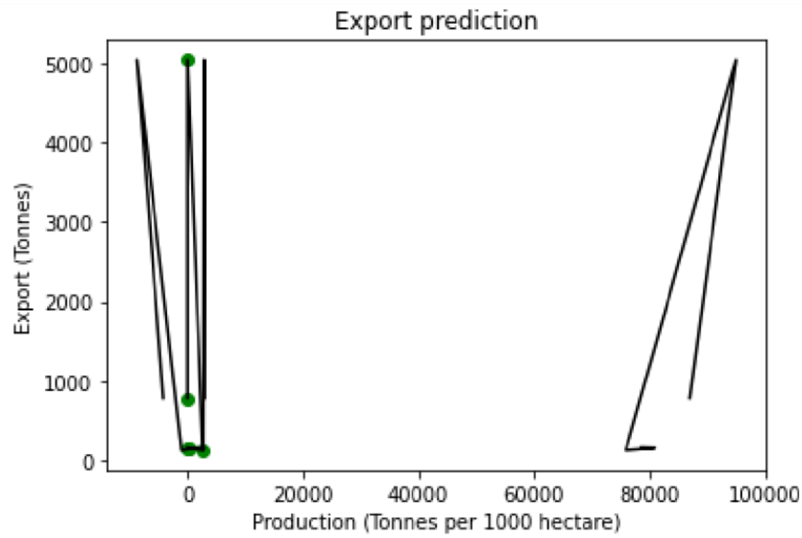


Fig 6.16. Multiple Regression Graph(4)

Dependent variable : Export

Independent variables : production, imports, stock, seed

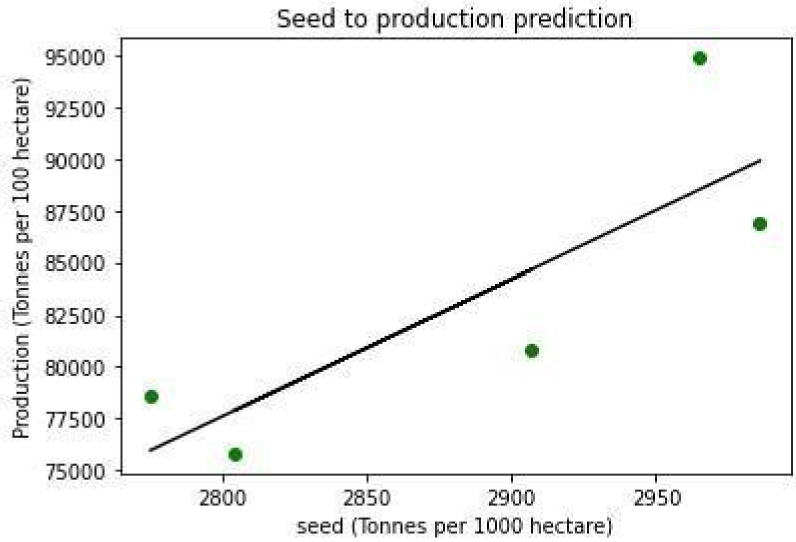


Fig 6.17. Multiple Regression Graph(5)

Dependent variable : production

Independent variables : seed

```
    Crop  Export  Imports  Stock  Production
63   NaN      17     1139      1           6
64   NaN       1      699      8           6
65   NaN       2     1101      3           6
66   NaN       1      924    -48           6
67   NaN      10      939    -37           6
68   NaN       1     1094      5           6
69   NaN       0     1108    -29           6
[[1755]
 [1585]
 [1594]
 [2038]
 [1954]
 [2262]
 [1690]]
Final R2 score :
0.4790698666052339
```

Fig 6.18 R2 score of import prediction

```
    Crop  Domestic  Imports  Production  Seed
63     6       NaN     1139        1755  2878
64     6       NaN      699        1585  2291
65     6       NaN     1101        1594  2696
66     6       NaN      924        2038  2913
67     6       NaN      939        1954  2846
68     6       NaN     1094        2262  3360
69     6       NaN     1108        1690  2768
[[  1]
 [  8]
 [  3]
 [-48]
 [-37]
 [  5]
 [-29]]
Final R2 score :
0.46833102478175165
```

Fig 6.19 R2 score of export prediction

```
{9: [9.0], 4: [4.0], 0: [0.0], 2: [2.0], 3: [3.0], 5: [5.0], 8: [8.0], 7: [7.0], 1: [1.0], 6: [6.0]}
Final R2 score :
0.7341655123908758
```

Fig 6.20 R2 score of production prediction

Final calculation:

p1 = predicted production per unit area mean / original production per unit area mean

p2 = predicted gross production mean / original gross production mean p3 = predicted exports mean / original exports mean p4 = predicted imports mean / original imports mean p5 = predicted production mean / original production mean mean final = (p1 + p2 + p3 + p4 + p5)/5 success rate % = mean final X 100

6.3 WEB APPLICATION IN DJANGO

Django is an open source web development framework in python. Django is useful to create ML based web applications easily because of its effectiveness in running python scripts. It consists of 3 main components: model, view and templates.

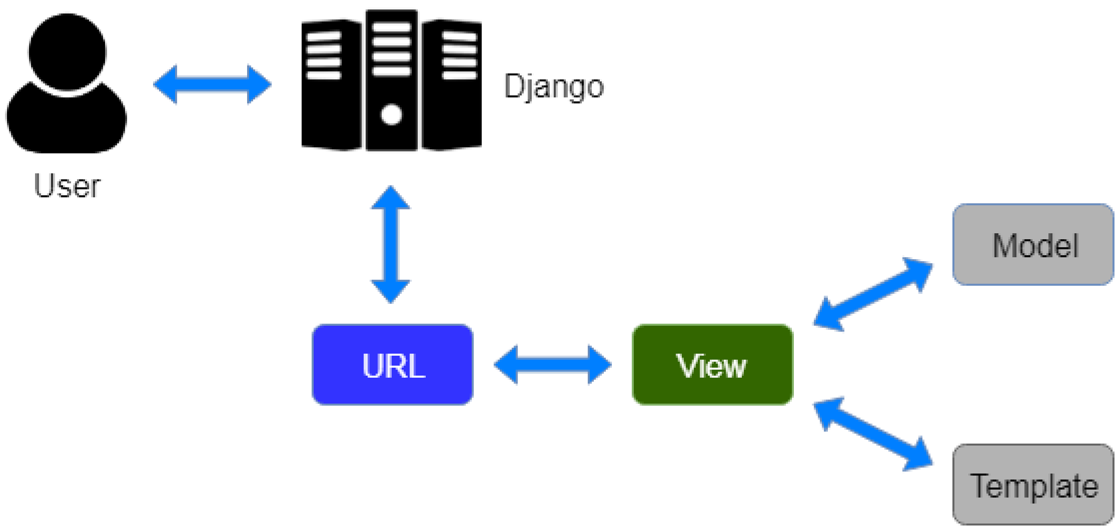


Fig 6.21. Django architecture

Template

The Template is a presentation layer which handles the User Interface part completely. In this web application, there are 3 main web pages namely input soil image, input region and result page. Result page displays 3 output components such as classified soil type, ranked crop list and other crop suggestions.

View

The View is used to execute the business logic and interact with a model to carry data and render a template. In this web app, view links the output from the soil classification module, fetches the crop list accordingly and sends it to the crop prediction module. It also executes machine learning(Multiple linear regression) and deep learning(CNN) for crop prediction and soil classification respectively.

Other settings in Django

These are the urls created for the website.

```
app_name="cultivo_main"    #called namespacing of urls,,,so as
urlpatterns=[
    path(r'',views.TemplateView.as_view(),name="home"),
    path(r'input_soil',views.upload_file,name="input_soil"),
    path(r'result',views.work,name="result"),
]
```

Fig 6.22 Urls.py

The web application runs without any issues and the server is initiated.

```
System check identified no issues (0 silenced).
March 18, 2021 - 10:19:47
Django version 3.1.7, using settings 'cultivo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{False Positives} + \text{False Negatives} + \text{True Negatives}}$$

- Precision

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- F-measure

$$\text{F1 Score} = \frac{2 \text{ x } (\text{Precision x Recall})}{\text{Precision} + \text{Recall}}$$

Table 7.1. Evaluation scores of implemented models

	CNN	**Alexnet**	**Lenet5**	**VGG16**	**SVM**
Accuracy	0.94	0.58	0.73	0.82	0.83
Precision	0.91	0.53	0.78	0.81	0.85
Recall	0.93	0.52	0.75	0.86	0.84
F-measure	0.92	0.49	0.75	0.83	0.84

- R-Squared

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

Table 7.2. R2 scores of Multiple linear regression

Linear Regression	**R2 score**
Gross production	0.999
Production per unit area	0.614
Imports	0.479
Exports	0.468
Production	0.734

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Soil images are classified accurately. Soil image classification works well for real time images. Crops with success rate are calculated taking all the mentioned parameters like export, import, production per unit area etc into account. The developed website is extremely user-friendly with simple and clear migrations. Most of the calculations are done beforehand to reduce the latency to the users. We strongly believe that the developed system solves the problem of choosing suitable crops for their fields by farmers.

8.2 FUTURE WORK

While the developed system takes only soil type to determine the crops suitable, it might be more realistic if the weather and climatic conditions are also considered to make the decision. Instead of manual entry of a region, GPS technology can be used to determine the location. With the availability of the type of soil in a particular region, the usage of images to find the type of soil can be eliminated. The website can be extended as a complete guide to farmers including the fertilizers, pesticides to be used etc.

REFERENCES

[1] Abimala, T., Sashya, S.F. and Sripriya, K., (2020), 'Soil Classification and Crop Suggestion using Image Processing', International Journal on Computing, pp. 3544.

[2] Barman, U. and Choudhury, R.D., (2020), 'Soil texture classification using multi class support vector machines'. Information Processing in Agriculture, *7*(2), pp.318-332.

[3] Bhattacharya, Biswanath, and Dimitri P. Solomatine., (2006), 'Machine learning in soil classification', IEEE International Joint Conference on Neural Networks ,pp.186-195.

[4] Devi, M. P. K., Anthiyur, U., and Shenbagavadivu, M. S., (2016).' Enhanced Crop Yield Prediction and Soil Data Analysis Using Data Mining'.

International Journal of Modern Computer Science, 4(6).

[5] Rahman, S.A.Z., Mitra, K.C. and Islam, S.M., (2018), 'Soil classification using machine learning methods and crop suggestions based on soil series'.

21st IEEE International Conference of Computer and Information Technology, pp. 1-4.

[6] Ramesh, D., Vardhan, B. V., (2013), 'Data mining techniques and applications to agricultural yield data'. International journal of advanced research in computer and communication engineering, 2(9), pp.3477-3480.

[7] Saranya, N., Mythili, A., (2020), 'Classification of Soil and Crop Suggestion Using Machine Learning Techniques'.International Journal of Engineering Research & Technology, pp.671-673.

[8] Srunitha.k, Dr.S.Padmavathi, (2016), 'Performance of SVM Classifier For Image Based Soil Classification'. International conference on Signal Processing,Communication,Power and Embedded System, pp.411-414.

[9] Suresh, G., (2021), 'Efficient Crop Yield Recommendation System Using Machine Learning For Digital Farming'. International Journal of Modern Agriculture, 10(1), pp. 906-914.

[10] Zhang, X., Younan, N.H., and O'Hara, C.G., (2005), 'Wavelet domain statistical hyperspectral soil texture classification'. IEEE Transactions on geoscience and remote sensing, 43(3), pp.615-618.

CODE

```
import os
from re import template
import MySQLdb
from flask import Flask, session, url_for, redirect, render_template, request, abort, flash
from database import db_connect,user_reg,user_loginact,user_upload,user_viewimages
from database import db_connect,image_info,view_pred
from database import db_connect
from werkzeug.utils import secure_filename


app = Flask(__name__)
app.secret_key = os.urandom(24)


@app.route("/")
def FUN_root():
    return render_template("index.html")


@app.route("/index.html")
def logout():
    return render_template("index.html")


@app.route("/register.html")
```

```
def reg():
    return render_template("register.html")

@app.route("/login.html")
def login():
    return render_template("login.html")

@app.route("/upload.html")
def up():
    return render_template("upload.html")

@app.route("/viewdata.html")
def up1():
    return render_template("viewdata.html")

# -------------------------------------------register-------------------------------------------------------
@app.route("/regact", methods = ['GET','POST'])
def registeract():
  if request.method == 'POST':
    id="0"
    status = user_reg(id,request.form['username'],request.form['password'],request.form['mobile'],request.form['address'])
    if status == 1:
     return render_template("login.html",m1="sucess")
    else:
     return render_template("register.html",m1="failed")
#-------------------------------------------Login-----------------------------------------------------
```

```
@app.route("/loginact", methods=['GET', 'POST'])
def useract():
    if request.method == 'POST':
        status = user_loginact(request.form['username'], request.form['p
assword'])
        print(status)
        if status == 1:
            session['username'] = request.form['username']

            return render_template("userhome.html", m1="sucess")
        else:
            return render_template("login.html", m1="Login Failed")
#-------------------------------------------Upload Image---------------------
-------------
@app.route("/upload", methods = ['GET','POST'])
def upload():
   if request.method == 'POST':
      id="0"
      status = user_upload(id,request.form['name'],request.form['image
'])
      if status == 1:
       return render_template("upload.html",m1="sucess")
      else:
       return render_template("upload.html",m2="failed")
#--------------------------------------View Images---------------------------
--------------
@app.route("/viewimage.html")
def viewimages():
    data = user_viewimages(session['username'])
```

```
    print(data)
    return render_template("viewimage.html",user = data)

#-------------------------------------Track---------------------------------
-----------
@app.route("/track")
def track():
    name = request.args.get('name')
    iname = request.args.get('iname')

    data = image_info(iname)
    print("dddddddddddddddddddddddddddd")
    print(data)
    # data = v_image(data)
    # print("dddddddddddddddddddddddddddd")
    # print(data)
    return render_template("viewdata.html",m1="sucess",data=data)


#----------------------------------Predict-----------------------------------
-------------
@app.route("/predict", methods = ['GET','POST'])
def predict1():
  if request.method == 'POST':
    Soiltype = request.form['Soiltype']
    n = int(request.form['n'])
    p = int(request.form['p'])
    k = int(request.form['k'])
    ph = float(request.form['ph'])
```

```
temp = int(request.form['temp'])
import pandas as pd
import numpy as np
optimum = pd.read_excel("optimum2.xlsx", 'newData')
#price = pd.read_excel("optimum2.xlsx", 'pricePerhr')
optimum['N'] = optimum.N.astype(float)
optimum['P'] = optimum.P.astype(float)
optimum['K'] = optimum.K.astype(float)
optimum['TEMPERATURE'] = optimum.TEMPERATURE.asty
pe(float)
X = optimum.drop("CLASS",axis=1)
y = optimum.CLASS
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X,y)
columns = ['N','P','K','pH','TEMPERATURE']
values = np.array([ n ,p ,k, ph , temp])
pred = pd.DataFrame(values.reshape(-1, len(values)),columns=c
olumns)
# print(pred.dtype)
print(pred)
prediction = clf.predict(pred)
print(prediction)


#prediction=1
data=view_pred(prediction[0])
return render_template('crops.html',data=data)
```

```
# ---------------------------------------------Update Item-------------------------------------------

if __name__ == "__main__":
    app.run(debug=True, host='127.0.0.1', port=5000)
```

Database.py:

```
import sqlite3
import hashlib
import datetime
import MySQLdb
from flask import session
from datetime import datetime
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
```

```
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import os
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os
import numpy as np
import cv2
import natsort
import xlwt
import datetime
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
def db_connect():
    _conn = MySQLdb.connect(host="localhost", user="root",
                            passwd="root", db="logo")
    c = _conn.cursor()

    return c, _conn

# -------------------------------register-----------------------------------------
-------------------------
def user_reg(id,username, password, mobile, address,):
    try:
        c, conn = db_connect()
        print(id,username, password,
              mobile, address)
        j = c.execute("insert into register (id,username,password,mobile
,address) values ('"+id+"','"+username +
                      "','"+password+"','"+mobile+"','"+address+"')")
        conn.commit()
        conn.close()
        print(j)
        return j
    except Exception as e:
        print(e)
        return(str(e))
# -------------------------------------Login -----------------------------------
---
def user_loginact(username, password):
```

```
        try:
            c, conn = db_connect()
            j = c.execute("select * from register where username='" +
                          username+"' and password='"+password+"'")
            data = c.fetchall()
            print(data)
            for a in data:
              session['uname'] = a[0]

            c.fetchall()
            conn.close()
            return j
        except Exception as e:
            return(str(e))
    #-----------------------------------Upload Image--------------------------
---------------
    def user_upload(id,name, image):
        try:
            c, conn = db_connect()
            print(name,image)
            username = session['username']
            j = c.execute("insert into upload (id,name,image,username) valu
es ('"+id+"','"+name+"','"+image +"','"+username +"')")
            conn.commit()
            conn.close()
            print(j)
            return j
        except Exception as e:
            print(e)
```

```
        return(str(e))

    #--------------------------------------View Images-------------------------
-------------

    def user_viewimages(username):
        c, conn = db_connect()
        c.execute("select * from upload where  username='"+username +"'
")
        result = c.fetchall()
        conn.close()
        print("result")
        return result

    #-----------------------------------Track-------------------------------------
--------------

    def view_pred(prediction):
        c, conn = db_connect()
        c.execute("Select * From crop where id='"+str(prediction)+"'")
        result = c.fetchall()
        conn.close()
        print("result")
        return result
    # ---------------------------------------------Update Items-----------------
------------------------


    def image_info(image_path):
        classes = {0:"Alluvial",1:"Black",2:"Clay",3:"Red"}
```

```
# dimensions of our images
    img_width, img_height = 224,224

# load the model we saved
    model = load_model('soilnew.h5')
# predicting images
#img = image.load_img('MRICOVID/Train/covid/1.jpg', target_size
=(img_width, img_height))
    image = load_img(image_path,target_size=(224,224))
    image = img_to_array(image)
    image = image/255
    image = np.expand_dims(image,axis=0)
#model = load_model('soilnew.h5')
    result = np.argmax(model.predict(image))
    prediction = classes[result]
    print(prediction)



    print("ddddddddddddddddddddddddddddddddd")
    print(image_path)
    #result="Alluvial"
    c, conn = db_connect()
    c.execute("SELECT * FROM soilcrop WHERE soiltype ='"+predi
ction+"' ORDER BY RAND() LIMIT 1")
    result = c.fetchall()
    conn.close()
    print("result")
    return result
```

```
if __name__ == "__main__":
    print(db_connect())
```

Image_Search.py:

```
from tensorflow.keras.layers import Input,Lambda,Dense,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt


IMAGE_SIZE = [224,224]

train_path = "Landmark/Train/"
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,horizontal_flip=True,zoom_range=0.2,validation_split=0.15)

training_set = train_datagen.flow_from_directory(
```

```
        train_path,target_size=(224,224), batch_size=32,class_mode='categorical',
        subset='training')

validation_set = train_datagen.flow_from_directory(
        train_path,target_size=(224,224), batch_size=32,class_mode='categorical',shuffle = True,
        subset='validation')

from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import GlobalAveragePooling2D,Dropout

## We are initialising the input shape with 3 channels rgb and weights as imagenet and include_top as False will make to use our own custom inputs

mv = VGG19(input_shape=IMAGE_SIZE+[3],weights='imagenet',include_top=False)

for layers in mv.layers:
    layers.trainable = False

# if u want to add more folders and train then change number 4 to 5 or 6 based on folders u have to train
x = Flatten()(mv.output)
prediction = Dense(4,activation='softmax')(x)
```

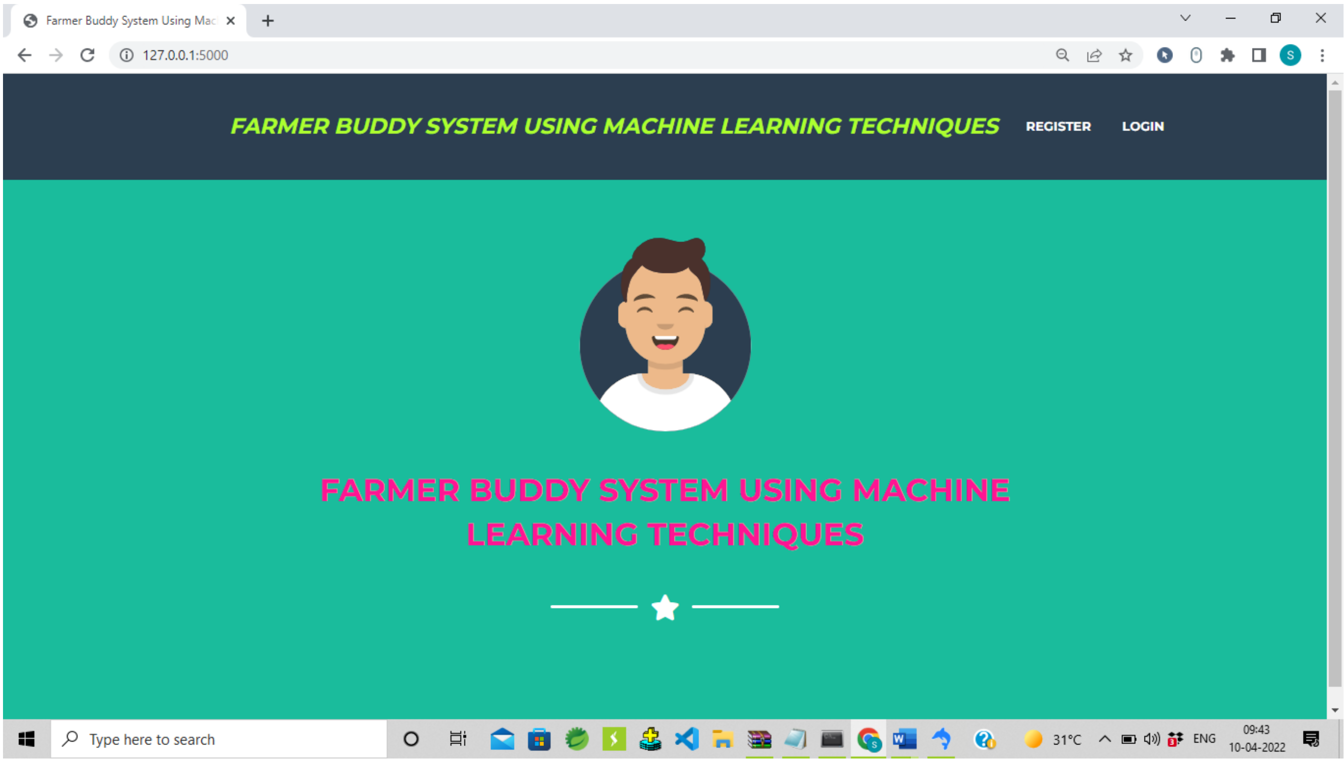
```
# In[7]:

model = Model(inputs=mv.input,outputs=prediction)

model.summary()

import tensorflow as tf


class myCallback(tf.keras.callbacks.Callback):

    def on_epoch_end(self,epoch,logs={}):

        if(logs.get('loss')<=0.05):

            print("\nEnding training")

            self.model.stop_training = True

# initiating the myCallback function

callbacks = myCallback()



## Let us compile the model with Adam optimizer and loss function categorical_crossentropy and metrics as categorical_accuracy

from tensorflow.keras.optimizers import Adam

model.compile(optimizer=Adam(lr=0.0001),loss='categorical_crossentropy',metrics=['categorical_accuracy'])

history = model.fit(training_set,

                    validation_data=validation_set,

                    epochs=50,

                    verbose=1,

                    steps_per_epoch=len(training_set),

                    validation_steps=len(validation_set),

                    callbacks = [callbacks]

                   )

acc = history.history['categorical_accuracy']

val_acc = history.history['val_categorical_accuracy']
```

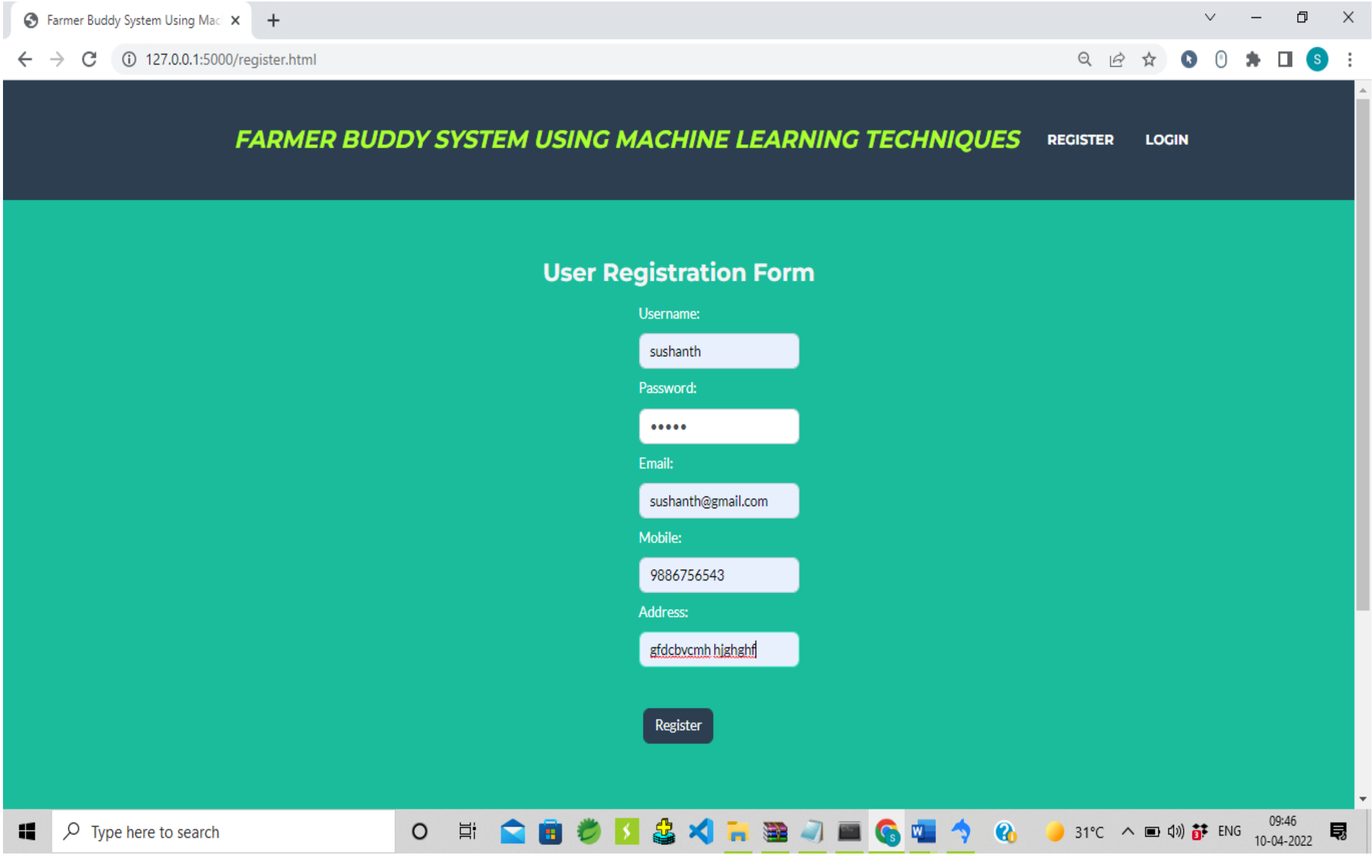
```
loss = history.history['loss']

val_loss = history.history['val_loss']

epochs = range(len(acc))

import matplotlib.pyplot as plt

plt.plot(epochs,acc)

plt.plot(epochs,val_acc)

plt.title("Training and validation Accuracy")

plt.figure()


plt.plot(epochs,loss)

plt.plot(epochs,val_loss)

plt.title("Training and validation Loss")

plt.figure()

model.save("VGG-19.h5")

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

import numpy as np


# dimensions of our images

img_width, img_height = 224,224


# load the model we saved

model = load_model('VGG-19.h5')

# predicting images

img = image.load_img('FruitsDB/Test/Low_quality_Apple/1.jpg', tar
get_size=(img_width, img_height))

x = image.img_to_array(img)
```

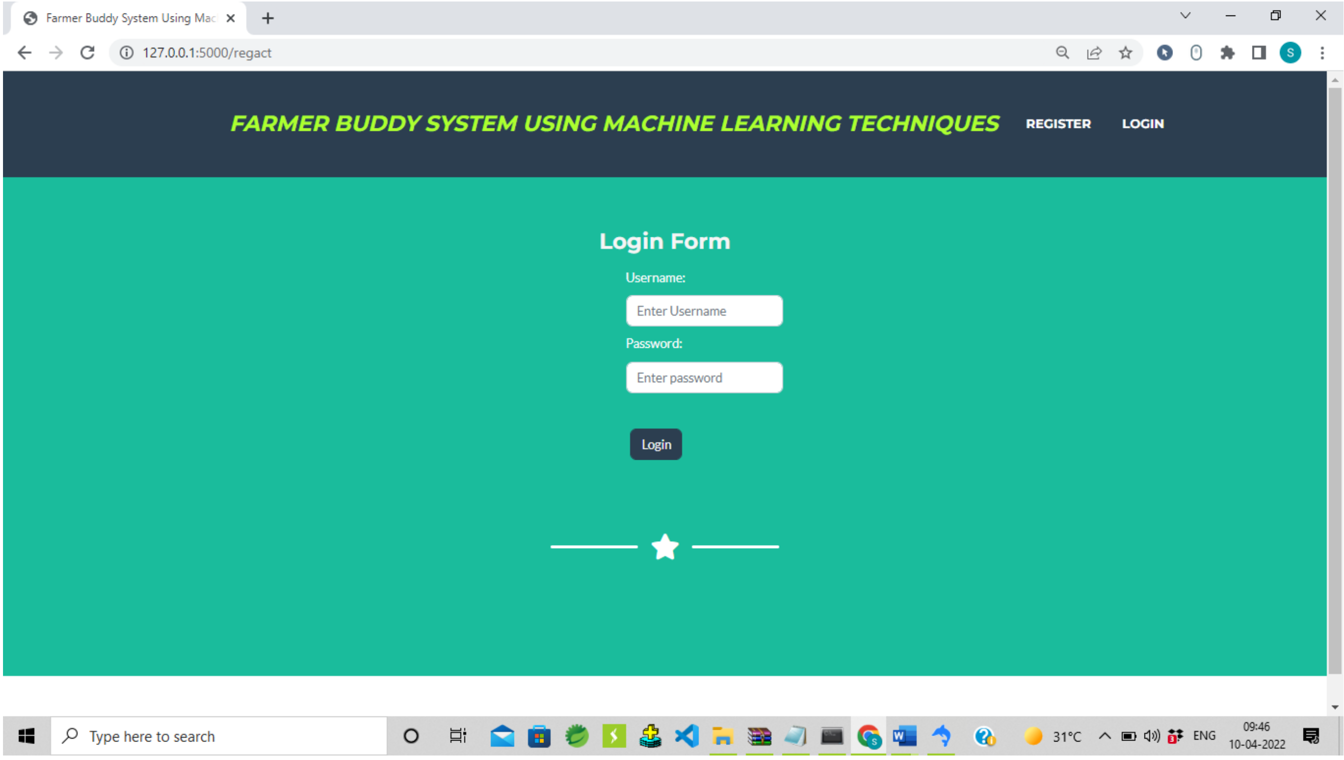
```
x = np.expand_dims(x, axis=0)

classes = model.predict(x)
print (classes)
```

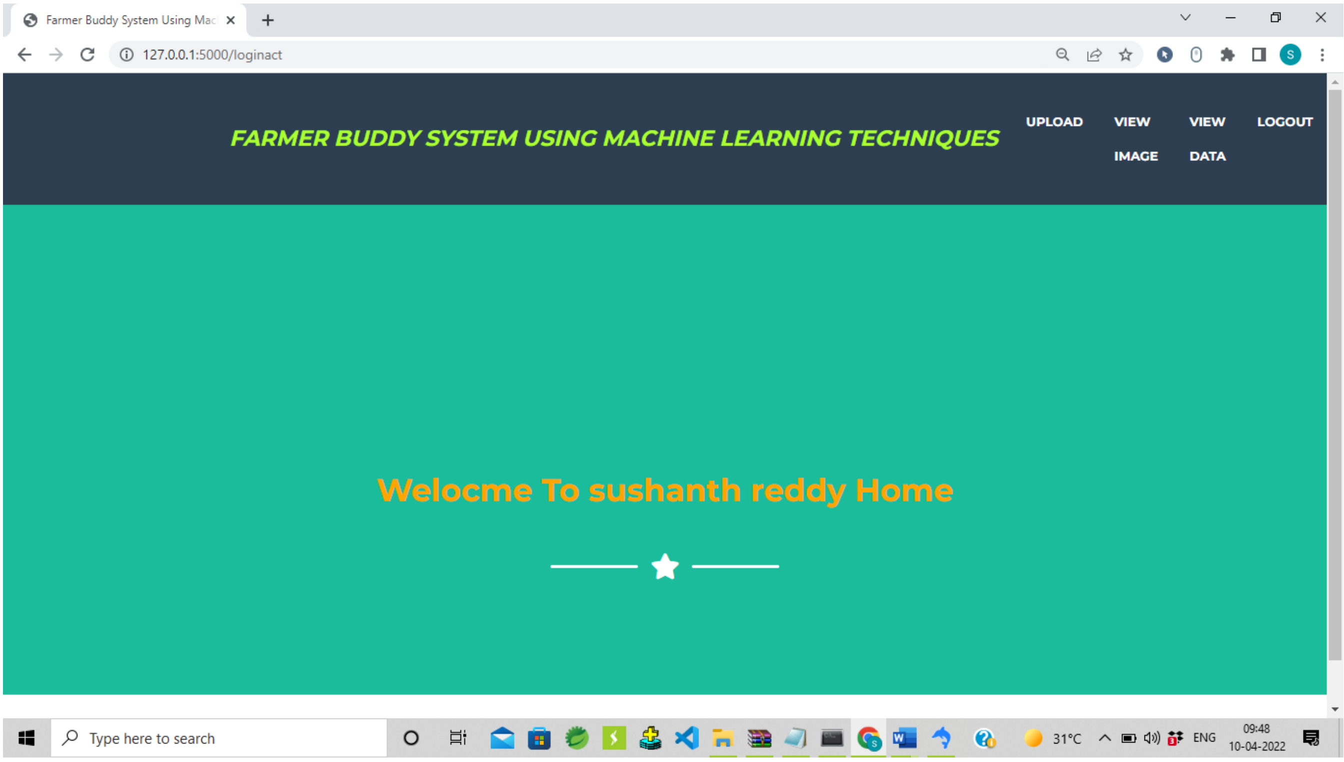
OUTPUT SCREEN SHOOTS

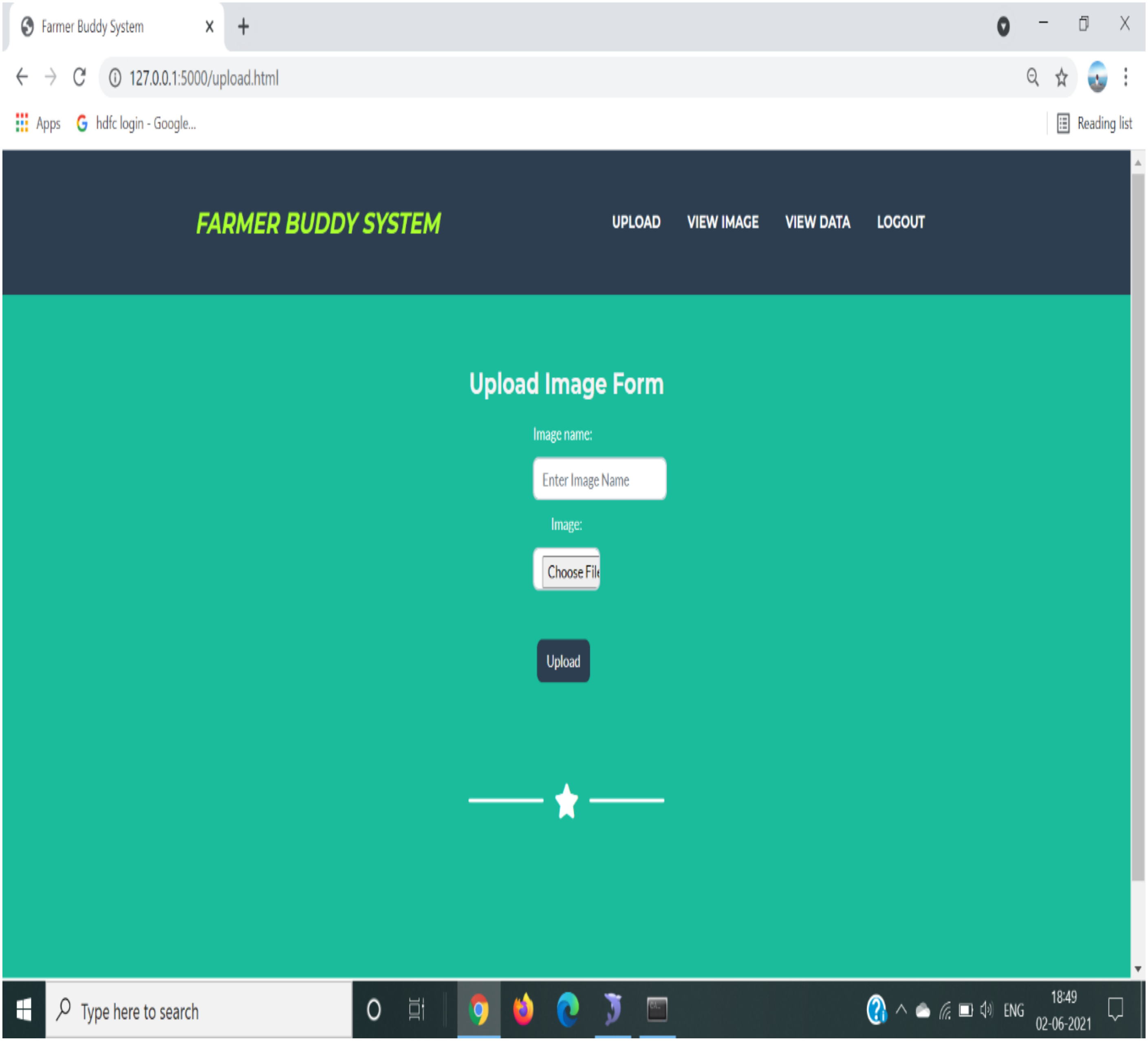
1.



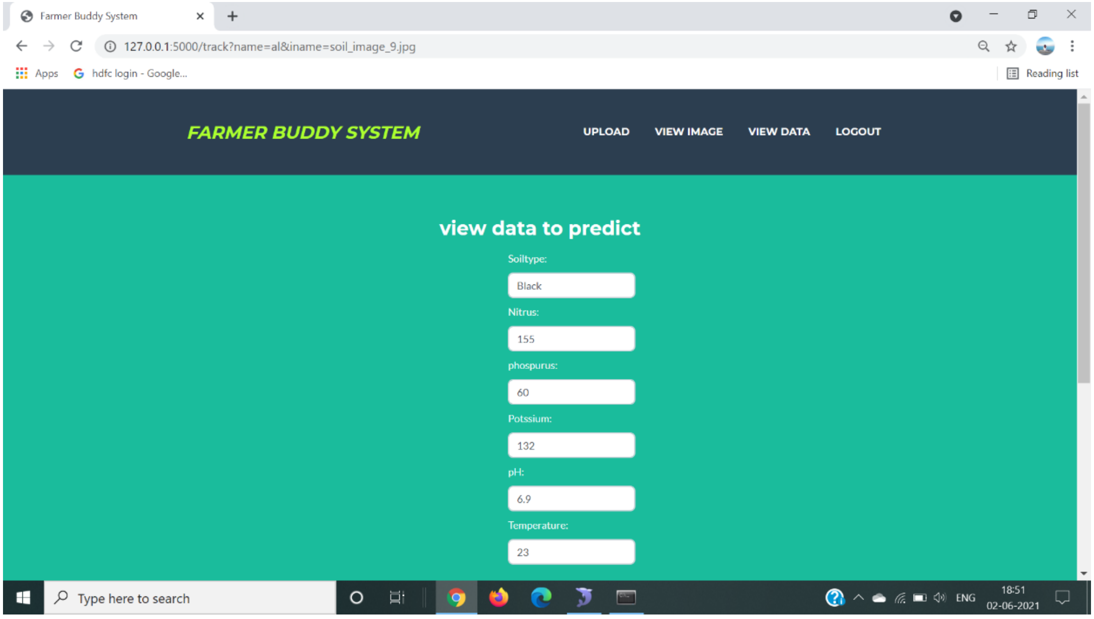

Farmer Buddy System Using Mac
127.0.0.1:5000/register.html
FARMER BUDDY SYSTEM USING MACHINE LEARNING TECHNIQUES
REGISTER
LOGIN
User Registration Form
Username:
sushanth
Password:
Email:
sushanth@gmail.com
Mobile:
9886756543
Address:
gfdcbvcmh hjghghf
Register
Type here to search
31°C
ENG
09:46
10-04-2022

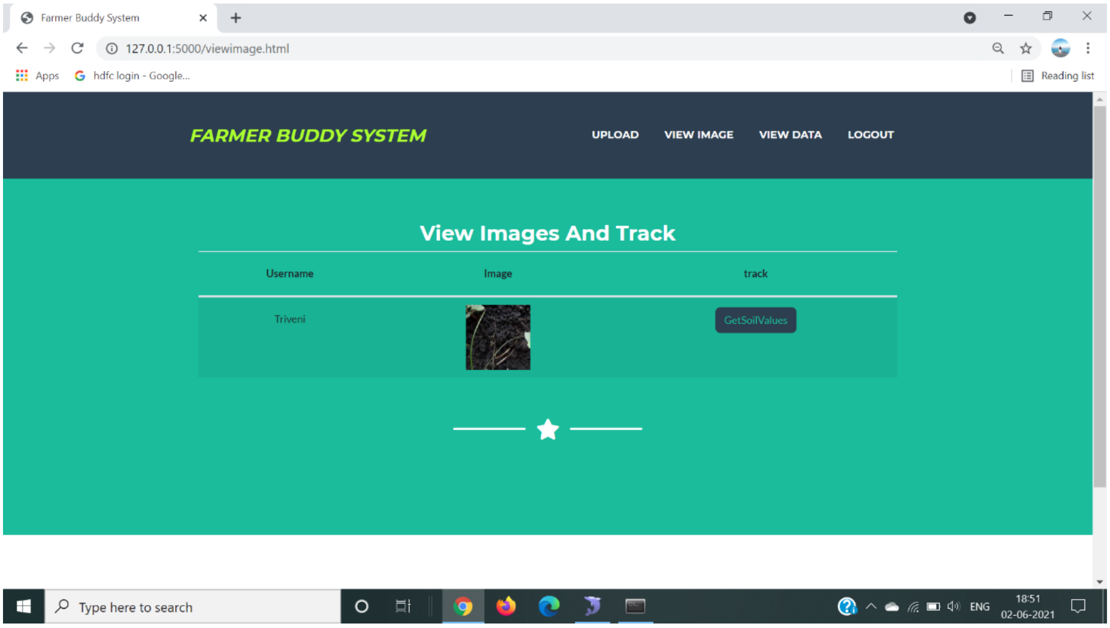

Farmer Buddy System Using Mac
127.0.0.1:5000/regact
FARMER BUDDY SYSTEM USING MACHINE LEARNING TECHNIQUES
REGISTER
LOGIN
Login Form
Username:
Enter Username
Password:
Enter password
Login
Type here to search
31°C
ENG
09:46
10-04-2022

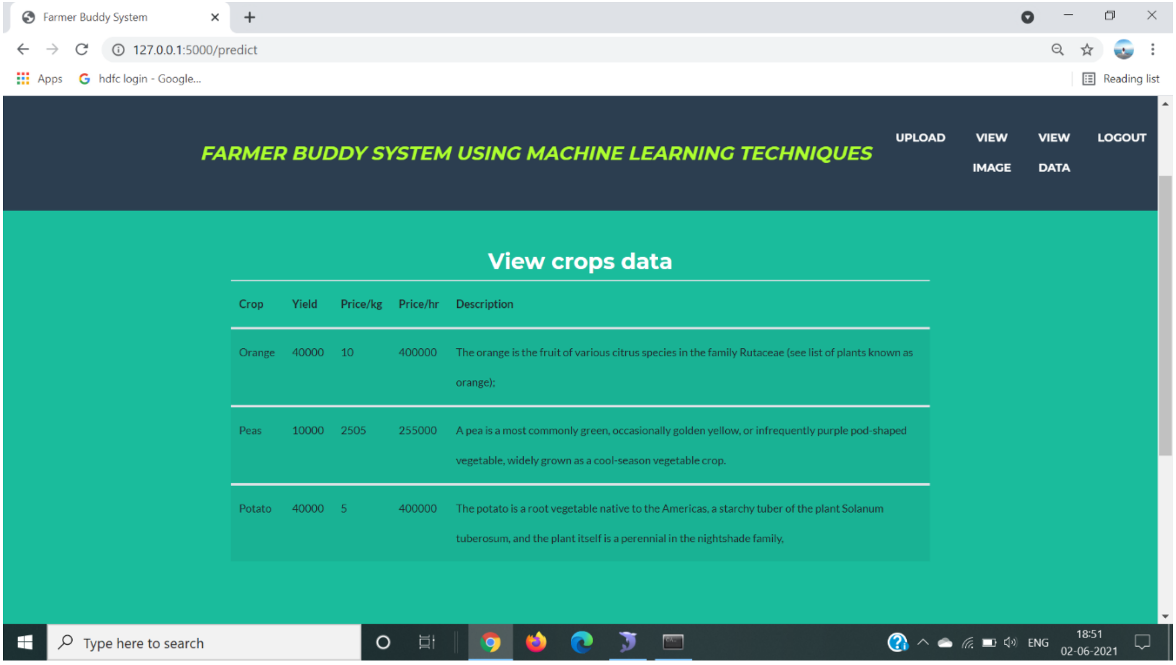

Farmer Buddy System Using Mac
127.0.0.1:5000/loginact
FARMER BUDDY SYSTEM USING MACHINE LEARNING TECHNIQUES
UPLOAD
VIEW IMAGE
VIEW DATA
LOGOUT
Welocme To sushanth reddy Home
Type here to search
31°C
ENG
09:48
10-04-2022



Farmer Buddy System
127.0.0.1:5000/upload.html
Apps
hdfc login - Google...
Reading list
FARMER BUDDY SYSTEM
UPLOAD
VIEW IMAGE
VIEW DATA
LOGOUT
Upload Image Form
Image name:
Enter Image Name
Image:
Choose File
Upload
Type here to search
ENG
18:49
02-06-2021


Farmer Buddy System
127.0.0.1:5000/track?name=al&iname=soil_image_9.jpg
Apps
hdfc login - Google...
Reading list
FARMER BUDDY SYSTEM
UPLOAD
VIEW IMAGE
VIEW DATA
LOGOUT
view data to predict
Soiltype:
Black
Nitrus:
155
phospurus:
60
Potssium:
132
pH:
6.9
Temperature:
23
Type here to search
ENG
18:51
02-06-2021


Farmer Buddy System
127.0.0.1:5000/viewimage.html
Apps
hdfc login - Google...
Reading list
FARMER BUDDY SYSTEM
UPLOAD
VIEW IMAGE
VIEW DATA
LOGOUT
View Images And Track
Username
Image
track
Triveni
GetSoilValues
Type here to search
ENG
18:51
02-06-2021


Farmer Buddy System
127.0.0.1:5000/predict
Apps
hdfc login - Google...
Reading list
FARMER BUDDY SYSTEM USING MACHINE LEARNING TECHNIQUES
UPLOAD
VIEW IMAGE
VIEW DATA
LOGOUT
View crops data
Crop Yield Price/kg Price/hr Description
Orange 40000 10 400000 The orange is the fruit of various citrus species in the family Rutaceae (see list of plants known as orange);
Peas 10000 2505 255000 A pea is a most commonly green, occasionally golden yellow, or infrequently purple pod-shaped vegetable, widely grown as a cool-season vegetable crop.
Potato 40000 5 400000 The potato is a root vegetable native to the Americas, a starchy tuber of the plant Solanum tuberosum, and the plant itself is a perennial in the nightshade family.
Type here to search
ENG
18:51
02-06-2021

THANK YOU