# Farmer Assistant Application

A project report submitted in partial fulfillment of the requirements for the degree of

**Bachelor of Computer Engineering (Sem -Vll)**

by

**Pratik Harde (Roll No. 8871)**

**Ibin Babu (Roll No. 8872)**

**Ananya Sharma (Roll No. 8908)**

Under the guidance of

**Prof. Prachi Patil**



DEPARTMENT OF COMPUTER ENGINEERING

## Fr. Conceicao Rodrigues College of Engineering, Bandra (W), Mumbai – 400050

University of Mumbai

(2022-2023)

*This work is dedicated to my family.*

*I am very thankful for their motivation and support.*

**Internal Approval Sheet**

# CERTIFICATE

This is to certify that the project entitled **" Farmer Assistant App "** is a bonafide work of **Pratik Harde (8871), Ibin** **Babu (8872), Ananya Sharma (8908)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering (Sem VII).**

**Prof. Prachi Patil**

Supervisor/Guide

**Dr. Sujata Deshmukh Dr. Surendra Rathod**

Head of Department Principal

**Approval Sheet**

# Project Report Approval

This project report entitled by **Farmer Assistant App** by **Pratik Harde (8871), Ibin Babu (8872), Ananya Sharma (8908)** is approved for the degree of Bachelor of Computer Engineering.

Examiners

1.————————————–

2.————————————–

Date: 5 Nov 2022

Place:

# Declaration

We declare that this written submission represents our ideas in our own words and where others’ ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Pratik Harde (Roll No. 8871) **(sign) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Ibin Babu (Roll No. 8872) **(sign) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Ananya Sharma (Roll No. 8908) **(sign) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date: 5 November , 2022

iii

## Abstract

## In India, agriculture is a significant source of both income and employment. The most frequent issue that Indian farmers encounter is that they choose the incorrect crop for their soil, use the incorrect fertiliser, and are unable to identify plant diseases that are caused by their plants. As a result, they will see a major decline in productivity. Farmers' problems will be resolved via the Farmer Assistant App. In order to suggest the optimum crop to farmers, as well as fertiliser suggestions based on site-specific characteristics, precision agriculture uses research data on soil qualities, soil types, and crop production statistics. By doing this, crop selection errors become less frequent and productivity rises. The proposed recommendation and prediction system in this study uses ML models as learners to highly accurately and effectively select a crop for the site-specific factors. The mechanism for recommending fertiliser is also entirely based on Python logic. Here, we compare the data (the crop's ideal nutrients) with the user-entered information. The most variable nutrient is then classified as HIGH or LOW, and recommendations are then retrieved in accordance with those results.

**Acknowledgements**

We have great pleasure in presenting the report on **"Farmer Assistant App"**. I take this opportunity to express my sincere thanks towards the guide Prof. Prachi Patil, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with him during our visits to the department.

We thank Dr. Sujata Deshmukh, Head of Computer Engineering department, Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

Pratik Harde (8871)

Ibin Babu (8872)

Ananya Sharma (8908)

Date: 5 November , 2022

v

## Contents

[**Abstract iv**](#_1fob9te)

[**List of Figures ix**](#_3znysh7)

[**List of Tables x**](#_2et92p0)

1. **Introduction . . 1**
   1. [Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2](#_tyjcwt)
   2. [Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2](#_3dy6vkm)
2. **Literature Review 3**
   1. [Smart Farming ……………………………………………………………………………………...3](#_1t3h5sf)
   2. [High Resolution Mapping of Soil Properties Using Remote Sensing Variables in South-Western Burkina Faso……………………………………………. 3](#_4d34og8)
   3. [Internet of things (IoT) applications to fight against COVID-19 pandemic 3](#_2s8eyo1)
   4. [Classification of Soil and Crop Suggestion…………………………………………… 4](#_17dp8vu)
   5. Smart Agriculture Using WSN………………………………………………………………...4
   6. Intelligent Agriculture System To Assist Farmers In Smart Decision………..4
   7. Literature Review table…………………………………………………………………………5
3. **Problem Statement 7**
   1. [Drawbacks …………………………………………………………………………………………...7](#_3rdcrjn)
   2. [Solution To Above Problem…………………………………………………………………..7](#_26in1rg)
4. **Project Designs 8**
   1. Block Diagram […………………………………………………………………………………….8](#_lnxbz9)
   2. [Algorithms…………………………………………………………………………………………..9](#_35nkun2)
   3. [UML Diagram …………………………………………………………………………………13](#_1ksv4uv)

4.3.1 Architecture ……………………………………………………………………………...14

* 1. [Dataset………………………………………………………………………………………………15](#_44sinio)
  2. Hardware and Software requirements………………………………………………….17

1. **Implementation Details 18**
   1. [Methodology……………………………………………………………………………………….18](#_2jxsxqh)
2. **Results**……………………………………………………………………………………………………..20
3. **Conclusion and Future Enhancments**……………………………………………………..28

**References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .**

**Appendix -1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .**

**Appendix -2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .**

vii

**List of Figures**

4.1. Block Diagram

4.2. Disease Prediction Diagram

4.3. Crop Recommendation Diagram

4.4. DFD Level 0 and DFD Level 1

4.5. UML Diagram

4.6. Sequence Diagram

4.7 Architecture

**List of Tables**

6.1 Algorithm and Accuracy

viii

## Glossary

SVM Support Vector Machines

RF Random Forest

DFD Data Flow Diagrams

ML Machine Learning

KNN K-Nearest Neighbor

ix

# Chapter 1

## Introduction

A farmer's decision regarding which crop to cultivate is typically influenced by his knowledge as well as unimportant variables like the desire to make quick money, ignorance of market demand, exaggeration of a soil's ability to support a particular crop, and so forth. The farmer's family's financial situation could be severely strained if he makes the wrong choice. Maybe this is one of the numerous factors contributing to the innumerable farmer suicide cases that the media reports on every day. Such a wrong judgement would have detrimental effects on not only the farmer's family but the entire economy of an area in a country like India, where agriculture and associated sectors contribute to about 20.4% of its Gross Value Added (GVA). Because of this, we have determined that a farmer's decision on which crop to cultivate during a specific season is a very serious one. The urgent requirement is to create a system that might offer Indian farmers predictive information so they could choose which crop to produce with knowledge.

In light of this, we suggest a system, an intelligent system, that would evaluate soil characteristics (N, P, K, pH value, soil type, and nutrients concentration) as well as environmental factors (temperature, rainfall, geographic location in terms of state), before advising the user on the crop that would grow best. Additionally, a fertiliser recommendation based on the ideal nutrients of the produced crops is also made.

### 1.1 Motivation

### As about 70% of the population depends on agriculture, it serves as the economic foundation for developing nations like India. India's economy and employment are heavily reliant on agriculture. Indian farmers frequently struggle with choosing the right crop to support their soil's needs and which fertiliser to utilise for their particular crop. This problem of the farmers has been addressed through precision agriculture.

### 1.2 Objectives

* To develop a reliable model that can accurately estimate crop sustainability in a given state under specific climatic and soil conditions
* Give advice on the best crops to grow in the area so that the farmer doesn't suffer any losses.
* Suggest fertilisers for crops with supported chemical properties. **Chapter 2**

## Literature Review

### Smart Farming [1]

An affordable IOT + ML design for smart farming with several applications, a system for water management systems, and an improvement to the current irrigation techniques are all included. An IoT and ML-based farming system constantly keeps farmers informed of the potential weather patterns and offers the best recommendations for crops and irrigation techniques, resulting in increased production.

**Smart Management of Crop Cultivation using IoT and Machine Learning [2]**

The most suited crop to grow in that climate is predicted by a smart system that takes into account measured parameters (temperature, humidity), as well as other parameters (soil type, farm location, rainfall), into account.

### High Resolution Mapping of Soil Properties Using Remote Sensing Variable in South-Western Burkina Faso [3]

The four modules of MODIFIED SUPPORT VECTOR REGRESSION, a well-known machine learning technique, are used to determine real-time sampling of soil parameters. The modules consist of a sensor connected to an IoT device, an agricultural cloud, an analysis of real-time sensor data, and an agricultural user interface (AUI). The first module consists of a NodeMCU portable IoT device with environmental sensors, including pH and soil moisture sensors. Storage is part of the agri cloud module. Using a modified support vector machine method, the real-time data module is analysed to process the various types of crops and small plants. It is a simple online interface called Agri-user interface. Thus, using a modified support vector machine algorithm, a farmer can determine the types of crops and small plants that can be cultivated on their land using the help of the soil's qualities.

**Internet of things (IoT) applications to fight against COVID-19 pandemic [4]**

Internet of Things (IOT) and machine learning are two of the proposed new technologies. Using an IOT system, real-time data from the field may be gathered. The trained model receives input from the field area's collected data. Then, using the data, the trained model makes the predictions. The model's output significantly aids in planting the appropriate crops in the targeted field area.

**Classification of Soil and Crop Suggestion [5]**

Determines a model is proposed for predicting the kind of soil and suggests a suitable crop that can be cultivated in that soil. Several machine learning techniques, including KNN, SVM, and logistic regression, have been used to test the model. Compared to other models, the accuracy of the current model is highest.

**Smart Agriculture Using WSN [6]**

Proposed a smart agriculture system based on IOT and deep learning. With the aid of a wireless sensor network, this system keeps an eye on and gathers data on the soil characteristics from the field. After then, the data is uploaded to the cloud. Finally, by anticipating the crop that will be grown for the following crop cycle, the systems advise farmers on the appropriate irrigation strategies. The farmers will receive this information via SMS. Temperatures of the soil, the atmosphere, and the humidity are among the characteristics. This technique advises enhancing the efficiency even further by forecasting when it will be appropriate to apply pesticides, fertiliser, and manures.

**Intelligent Agriculture System To Assist Farmers In Smart Decision [7]**

The suggested approach will help farmers choose the best crop to cultivate based on a range of geographical and environmental parameters. By substituting intuition and inherited knowledge with more trustworthy data-driven ML models, the ML and IoT based recommendations would considerably enlighten the farmer and assist them in minimising expenses and making strategic decisions. This enables a scalable, trustworthy solution to a significant issue affecting hundreds of millions of people.

**Fig 2.1 Literature Survey Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Reference | Dataset | Techniques | Results |
| 1. | [An IoT Based Smart Farming System Using Machine Learning (researchgate.net)](https://www.researchgate.net/publication/347979426_An_IoT_Based_Smart_Farming_System_Using_Machine_Learning) | The data have been collected from 30 US and Canadian Cities, as well as 6 Palestine cities. The dataset contains ~5 years of high temporal resolution (hourly measurements) data of various weather attributes, such as temperature, humidity | i) data collection phase using sensors deployed in an  agricultural field, ii) data cleaning and storage phase, and iii)  predictive processing using some AI methods.  Data collection phase using sensors deploying in an agricultural field.  Data cleaning and storage phase and predicting processing using some AI methods. | Showed that AI techniques play a pivotal role in agriculture of precision by using machine learning and open sources technologies. |
| 2. | [Smart Management of Crop Cultivation using IOT and Machine Learning | IRJET Journal - Academia.edu](https://www.academia.edu/37955255/Smart_Management_of_Crop_Cultivation_using_IOT_and_Machine_Learning) | - | uses random forest algorithm to predict appropriate crop based on current NPK value of soil.  Machine Learning Algorithm(KNN ) is used to calculate the crop which is best to grow in the particular field based on the values received at real time. | Using the real time values obtained from the field and running the algorithm on them, the most suitable crop for aparticular land at a given time is predicted |
| 3. | [High Resolution Mapping of Soil Properties Using Remote Sensing Variables in South-Western Burkina Faso: A Comparison of Machine Learning and Multiple Linear Regression Models (researchgate.net)](https://www.researchgate.net/publication/312646633_High_Resolution_Mapping_of_Soil_Properties_Using_Remote_Sensing_Variables_in_South-Western_Burkina_Faso_A_Comparison_of_Machine_Learning_and_Multiple_Linear_Regression_Models) | A total of 1104 soil samples (1002 in sub-watershed and 102 out-side) | Four statistical prediction models–multiple linear regression (MLR), random for-  est regression (RFR), support vector machine (SVM), stochastic gradient boosting (SGB)–  were tested and compared.  Four statistical prediction models–multiple linear regression (MLR), random forest regression (RFR), support vector machine (SVM), stochastic gradient boosting (SGB)–were tested and compared. | Internal validation was conducted by cross validation while the predictions were validated against an independent set of soil samples considering the modelling area and an extrapolation area. |
| 4. | [Internet of things (IoT) applications to fight against COVID-19 pandemic - PMC (nih.gov)](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7198990/) | No Dataset information is provided | IoT is used to capture health data from various locations of the infected patient and manage all the data using the virtual management system  contact tracing, cluster identification and compliance of quarantine. | By using a statistical-based method, IoT gets helpful to predict an upcoming situation of this disease. |
| 5. | [Soil Classification and Crop Suggestion using Machine Learning Techniques by IJRASET - Issuu](https://issuu.com/ijraset/docs/soil_classification_and_crop_suggestion_using_mach#:~:text=The%20algorithm%20such%20as%20SVM%20can%20be%20used,crop%20dataset%20with%20geographical%20attributes%20as%20its%20features.) | They have used a large dataset extracted from the Australian Department of Agriculture and Food(AGRIC) to conduct the research | The model has been tested using various machine learning algorithms such as KNN, SVM and logistic regression. | The accuracy of the present model is maximum than the existing models. |
| 6. | [Smart Agriculture Using WSN and IoT: Environment & Agriculture Book Chapter | IGI Global (igi-global.com)](https://www.igi-global.com/chapter/smart-agriculture-using-wsn-and-iot/268036) | - | This system monitors and collects the soil parameters from the field with the help of a wireless sensor network. The collected data is then uploaded in the cloud. Finally, the systems suggest best irrigation practices to the farmers by predicting the crop to be sown for next crop rotation. | This system suggests further improving the effectiveness by predicting the suitable time for applying pesticides, fertilizer, and manures. |
| 7. | [Intelligent Agriculture System To Assist Farmers In Smart Decision (quickcompany.in)](https://www.quickcompany.in/patents/intelligent-agriculture-system-to-assist-farmers-in-smart-decision-making-using-iot-data-analytics-and-machine-learning) | A total of 9 types of soil present in dataset. | The ML and IoT based suggestions will significantly educate the farmer and help them minimize costs and make strategic decisions by replacing intuition and passed-down knowledge with far more reliable data-driven ML models. | provide intelligent agriculture system to assist farmers in smart decision making using IoT data analytics and machine learning. |

# Chapter 3

## Problem Statement

### 3.1 Drawbacks of Current Solutions

The fact that each author of each publication concentrated on one factor (weather or soil) for predicting the appropriateness of crop development was one weakness we found in all of these important published papers. But in our opinion, the best and most accurate prediction should be made by concurrently taking into account both of these aspects. This happens frequently because even if a particular soil type may be ideal for supporting a certain crop variety, the yield will suffer if the local climate doesn't appear to be favourable for that crop kind.

### 3.2 Solution to the Problem

By proposing an effective crop recommendation system that accounts for all relevant factors, such as temperature, rainfall, location, and soil quality, we hope to eliminate the above shortcomings. Giving farmers crop suggestions is the main task that this system is primarily concerned with carrying out. In order to give the user a simple and trustworthy knowledge of the insight to determine and plant the crops, we also supply the fertilisers to be used for crops grown in various states.

# Chapter 4

## Project Design

### 4.1 Block Diagram

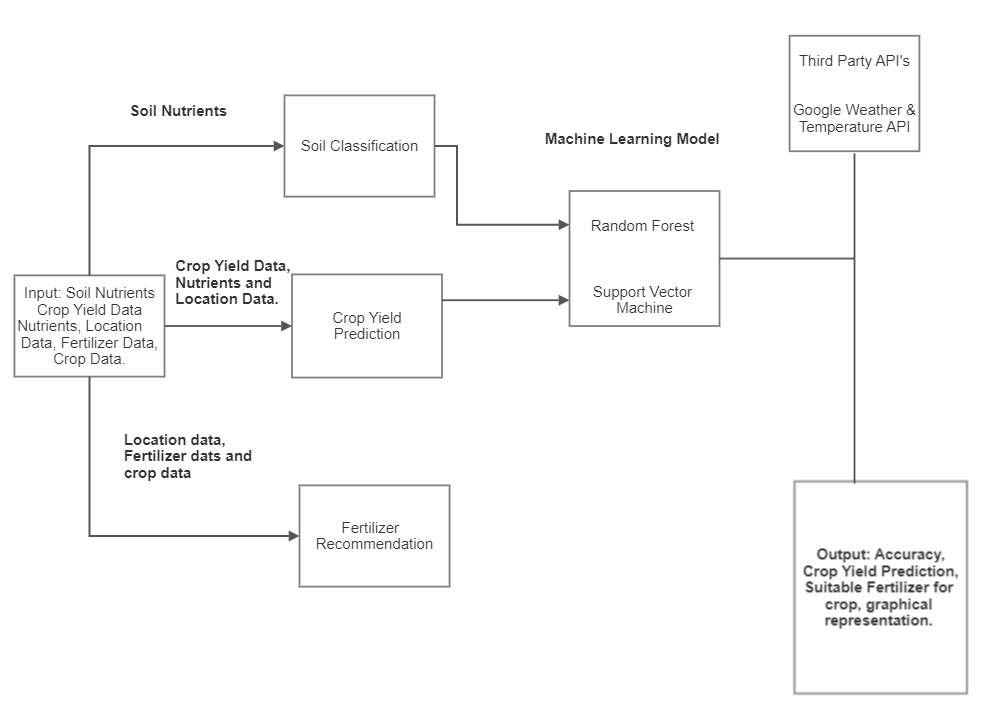


Fig 4.1 Block Diagram for Crop and Fertilizer Recommendation

Crop recommendation, fertiliser recommendation, and disease prediction are the three goals of our method. We are utilising an API key to retrieve the data for the fertiliser recommendation. The system will provide recommendations and predictions based on the nutrients dataset.

### 4.2 Algorithms

**4.2.1 Decision Tree**

A non-parametric supervised learning technique for classification and regression is called a decision tree (DT). The objective is to learn straightforward decision rules derived from the data features in order to build a model that predicts the value of a target variable. A fractional approximation of a tree can be thought of. For instance, in the example below, using a series of if-then-else decision rules, decision trees learn from data to approximate a sine curve.

**4.2.2 Support Vector Machine**

In an N-dimensional space (N being the number of features), the support vector machine algorithm seeks to locate a hyperplane that clearly categorises the data points.

There are a variety of different hyperplanes that might be used to split the two classes of data points. Finding a plane with the greatest margin—that is, the greatest separation between data points from both classes—is our goal. Maximizing the margin distance adds some support, increasing the confidence with which future data points can be categorised.

**4.2.3 Logistic Regression**

When classifying observations into a collection of discrete classes, the classification algorithm logistic regression is utilised. Email spam or not spam, online fraud or not fraud, and malignant or benign tumours are a few instances of classification issues. To convert its output into a probability value, logistic regression uses the logistic sigmoid function.

What are the types of logistic regression

1. Binary (eg. Tumor Malignant or Benign)

2. Multi-linear functions failsClass (eg. Cats, dogs or Sheep's)

The machine learning technique known as logistic regression, which is based on the probability notion and used to solve classification problems, is used to analyse data in a predicted manner.

**4.2.4 Random Forest**

Supervised machine learning algorithms like random forest are frequently employed in classification and regression issues. On various samples, it constructs decision trees and uses their average for classification and majority vote for regression.

The Random Forest Algorithm's ability to handle data sets with both continuous variables, as in regression, and categorical variables, as in classification, is one of its most crucial qualities. In terms of classification issues, it delivers superior outcomes.

To precisely address the issue of high-variance in Decision Trees, Random Forests was created. You're not just training one Decision Tree, as the name implies—you're training an entire forest! A forest of Bagged Decision Trees in this instance.

Random Forests algorithm follows these steps:

1. Starting with the original dataset, generate N bagged samples of size n, where n is the number of samples to be created.

2. With each of the N bagged datasets as input, train a decision tree. Don't, however, look at every feature in the dataset while doing a node split. Choose M features at random from the entire training set of features. Then, using impurity metrics like Gini Impurity or Entropy, choose the optimal split.

3. Combine the outcomes of each decision tree into a single output.

4. If you're working on a regression problem, average the results for each observation, as produced by each tree.

5. If you're working on a regression problem, make a majority decision for each observation across all trees.

**Disease Prediction Diagram**

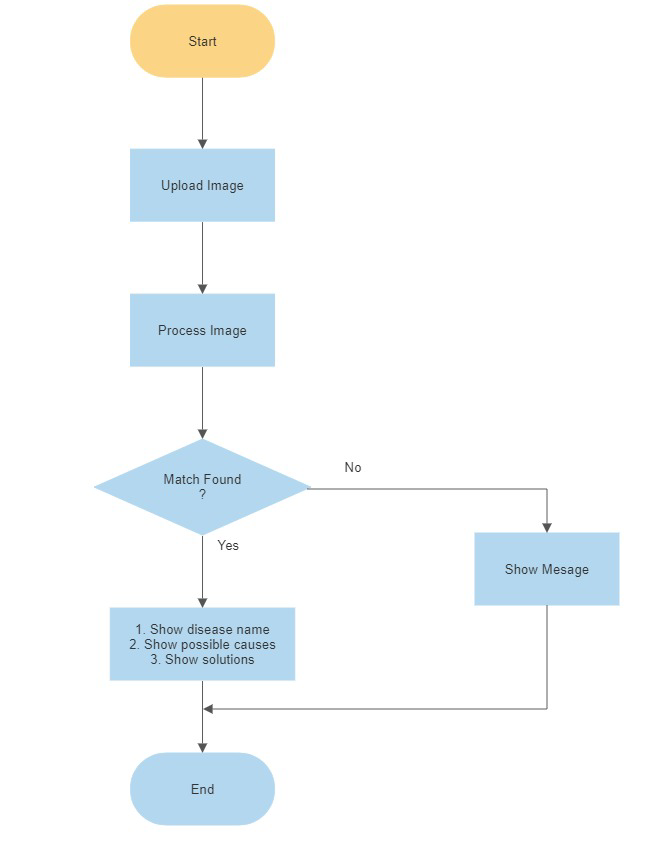


Fig. 4.2 Disease Prediction

**Crop Recommendation Diagram**

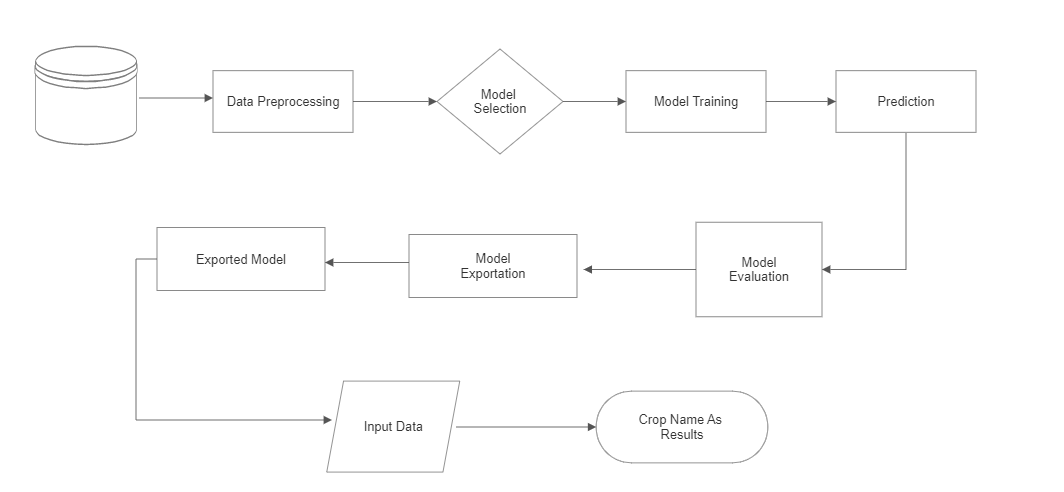
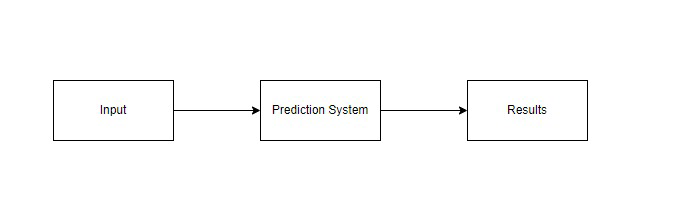


Fig. 4.3 Crop Recommendation

### DFD Level 0 and DFD Level 1

****

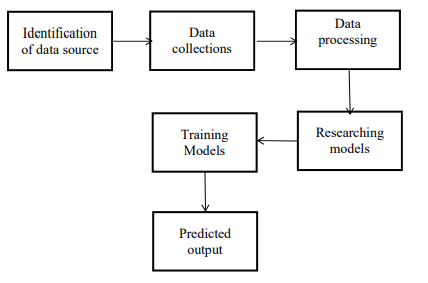


Fig. 4.4 DFD Level 0 and DFD Level 1

### 4.3 UML Diagram

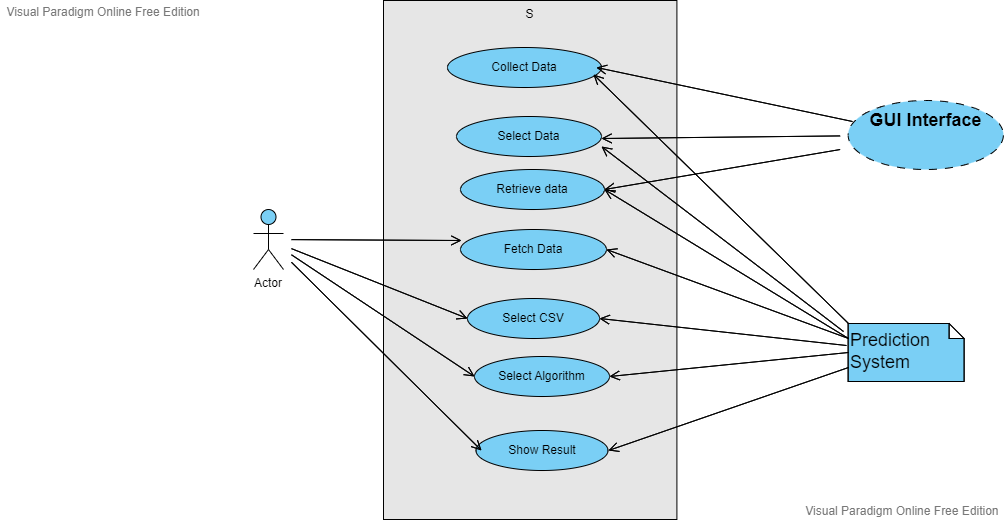


Fig 4.5 UML Diagram

### Architecture

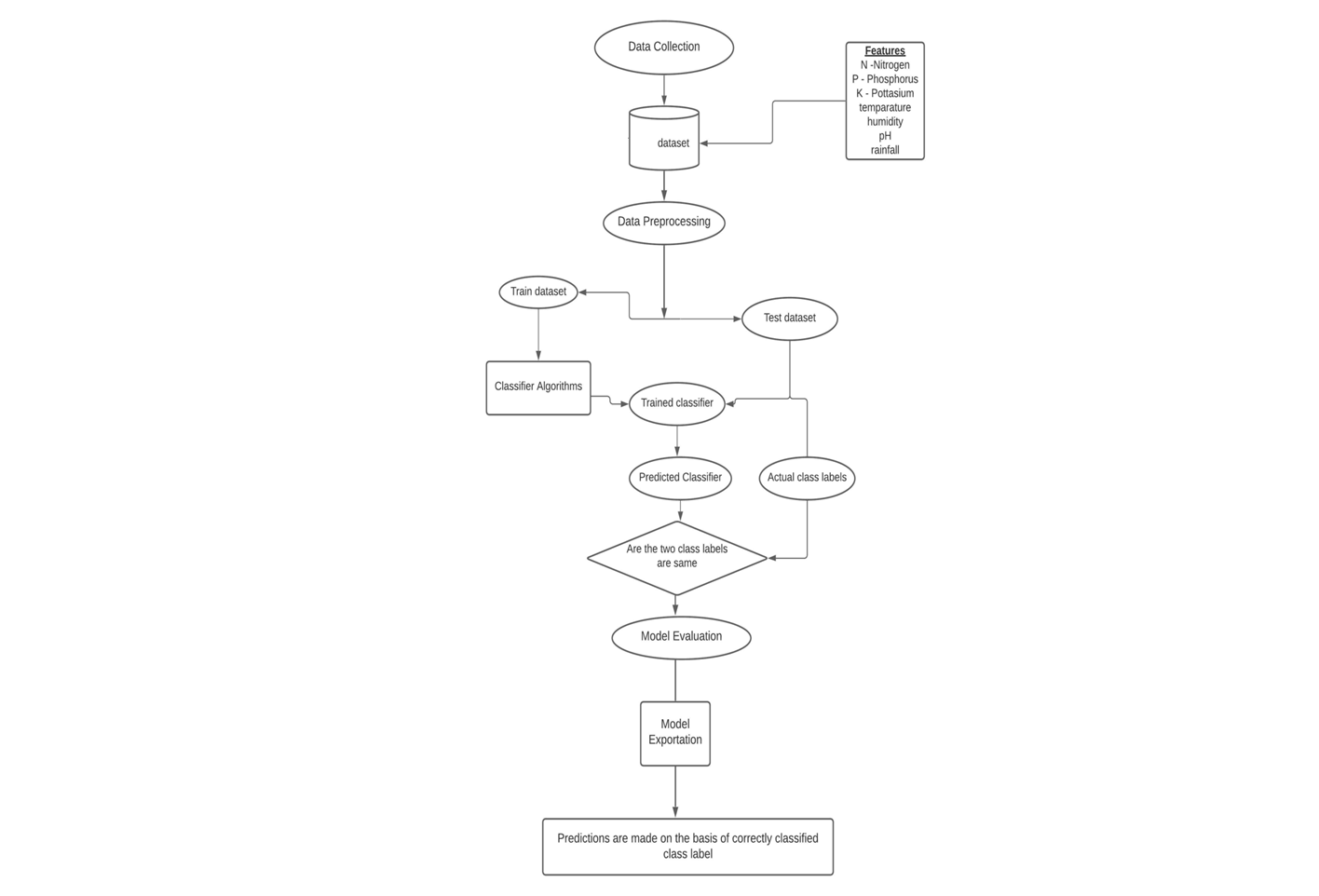


Figure 4.7 : Architecture

### 4.3 Dataset

**Datasets include:-**

∙ Crop Recommendation dataset

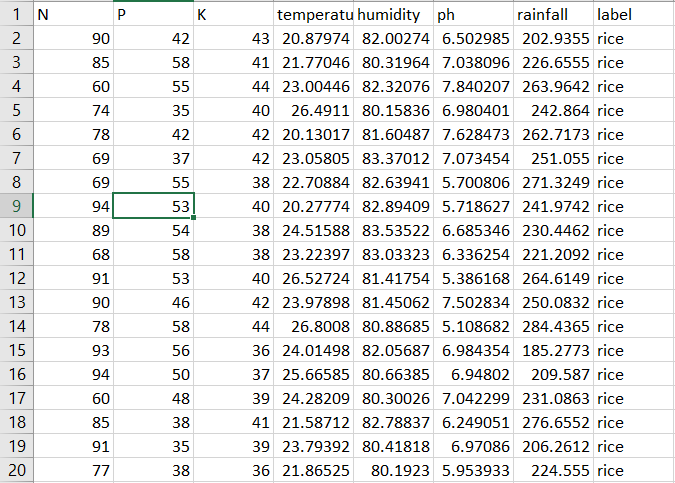
∙ Fertilizer dataset

∙ Plant Disease dataset

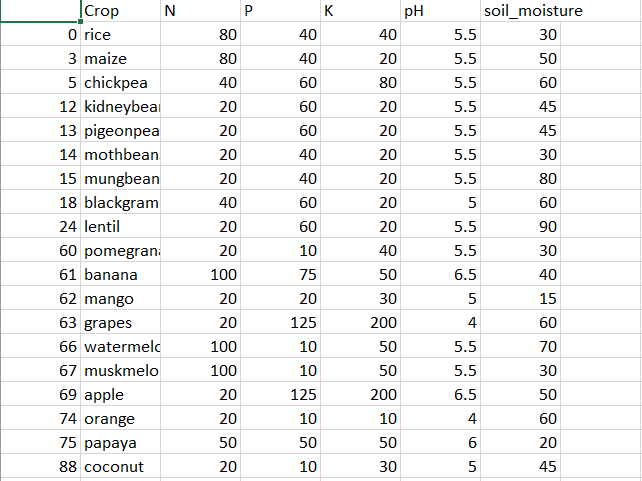
∙ State -District Wise Crop dataset

A brief description of the datasets:

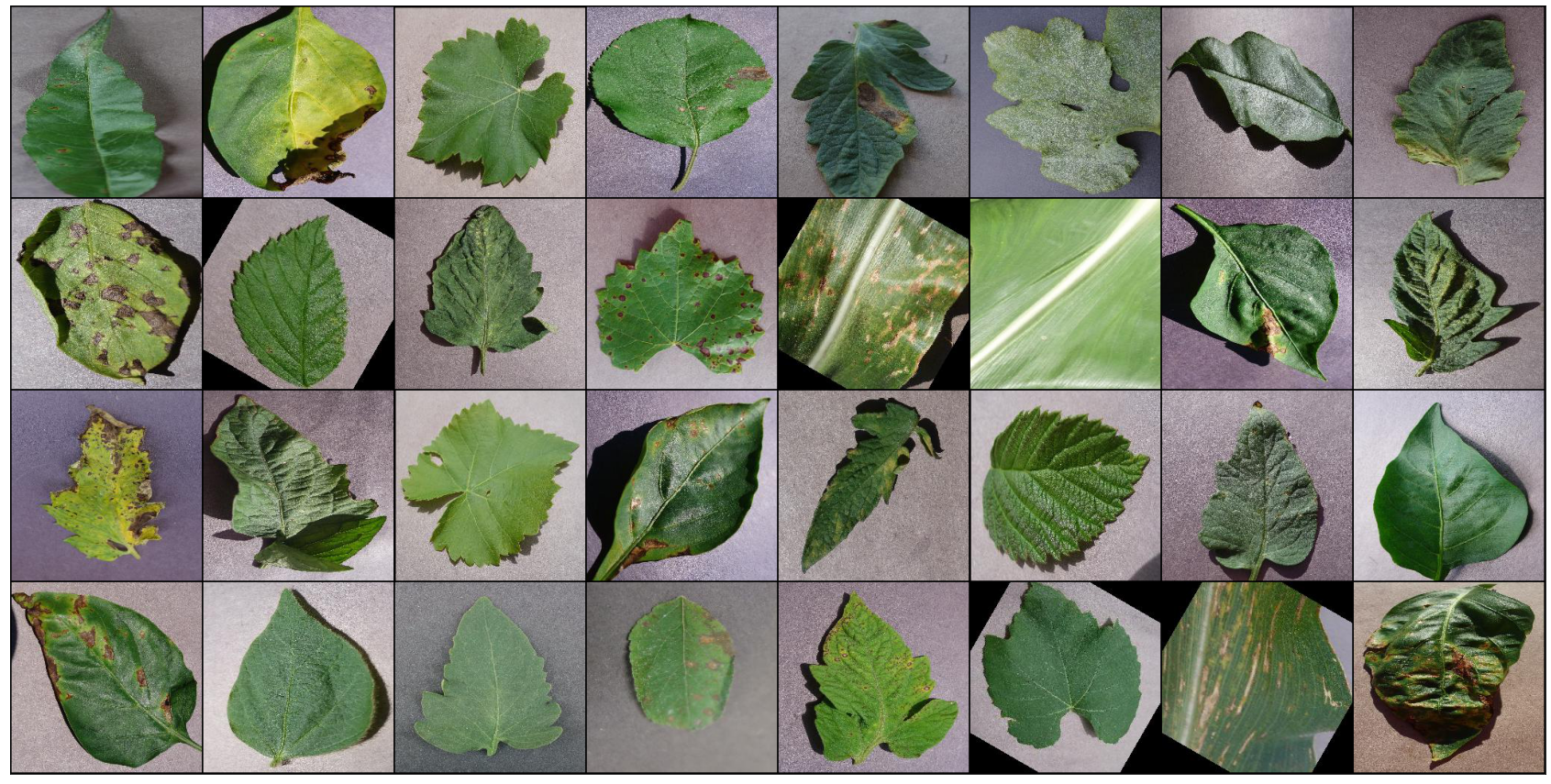
* **Crop Recommendation dataset :** In kg per hectare, this information shows the yield for 18 important crops farmed in every state. When the yield is 0, it means that the crop is not grown in that stage. containing a total of 2201 items in the dataset.



* **Fertilizer dataset** : The columns in this dataset have the following attributes: State, Nitrogen, Phosphorous, Potassium, Average Ph Level, and Soil Moisture Content. There are 25 total records in the collection.



* **Plant Disease dataset:** 76100 items in this dataset include photos of illness leaves and 38 distinct classes.

For instance, Apple->Apple scab, Apple black rot, and Apple healthy

* 1. **Hardware and Software**

    **•** Hardware System Configuration:

* Processor: 2 gigahertz (GHz) or faster processor.
* RAM: 4 gigabyte (GB) for 32-bit or 4 GB for 64-bit.
* Hard disk space: =>16GB.

**•** Software Configuration:

* Operating System: Windows XP/7/8/8.1/10, Linux and Mac
* Coding Language: Python.
* Tools:  Pandas, Numpy ,Seaborn , Pickle,Scikit-learn, Pytorch & ResNet.
* Framework:  Flask.
* Other Tools :  HTML, CSS, Bootstrap.

**•** Tools and Library used

* Library-torch, Pandas, Numpy, Matplotlib, Seaborn, Pickle, Scikit-learn, Pytorch and resnet.
* Tool- Flask

# Chapter 5

## Implementation Details

### 5.1. Methodology

### Data Analysis

An analysis of the data is one of the first tasks the system carries out during implementation. We did this in an effort to determine whether there were any connections between the different attributes that were available in the dataset. Acquisition of Training Dataset: Any machine learning algorithm's accuracy is based on the training dataset's precision and the amount of parameters it uses. In this project, we carefully chose the parameters that would produce the best results after analysing different datasets obtained from the government website and Kaggle. Environmental indicators have been taken into account in numerous studies in this sector to predict agricultural sustainability, while yield has also been a prominent component in some and just economic variables in others. To give the farmer the most accurate and trustworthy advice on which crop will be best for his land, we have attempted to combine environmental criteria like rainfall, temperature, ph, soil nutrients, soil type, and location with economic parameters like production and yield.

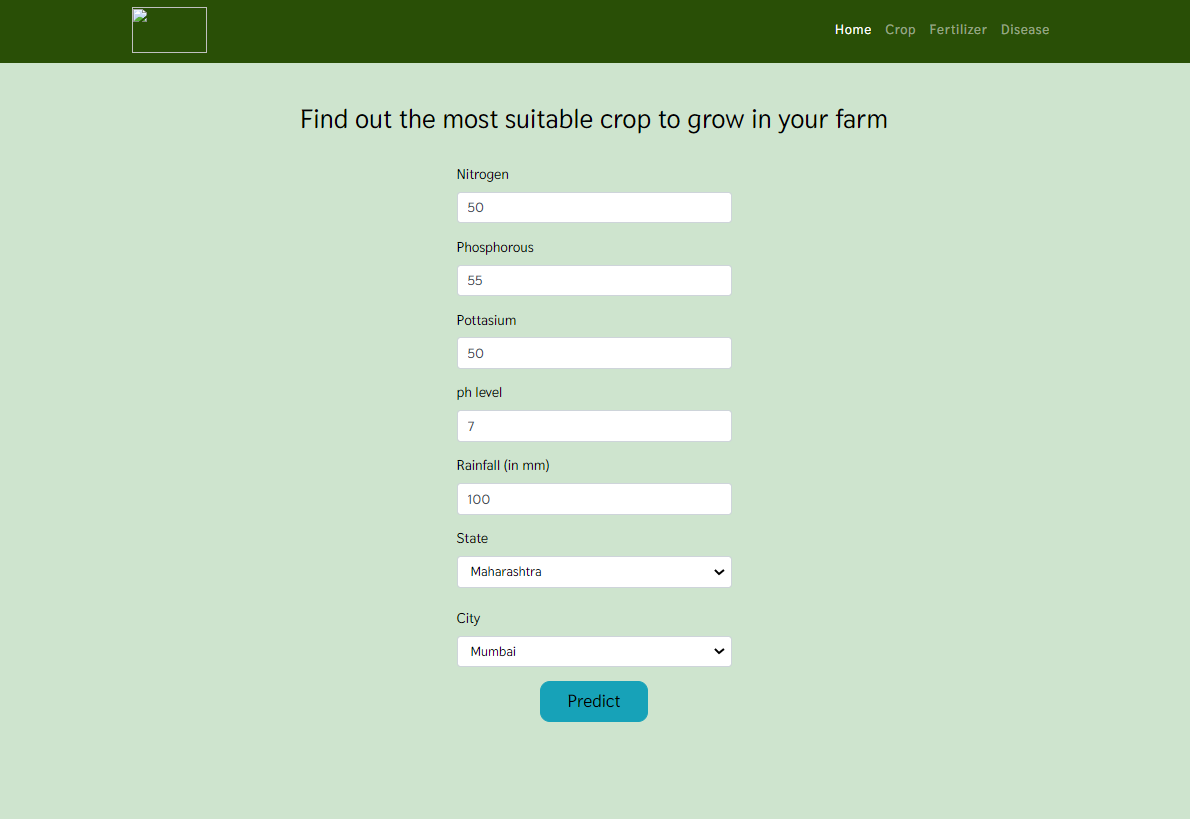
**Data Preprocessing**

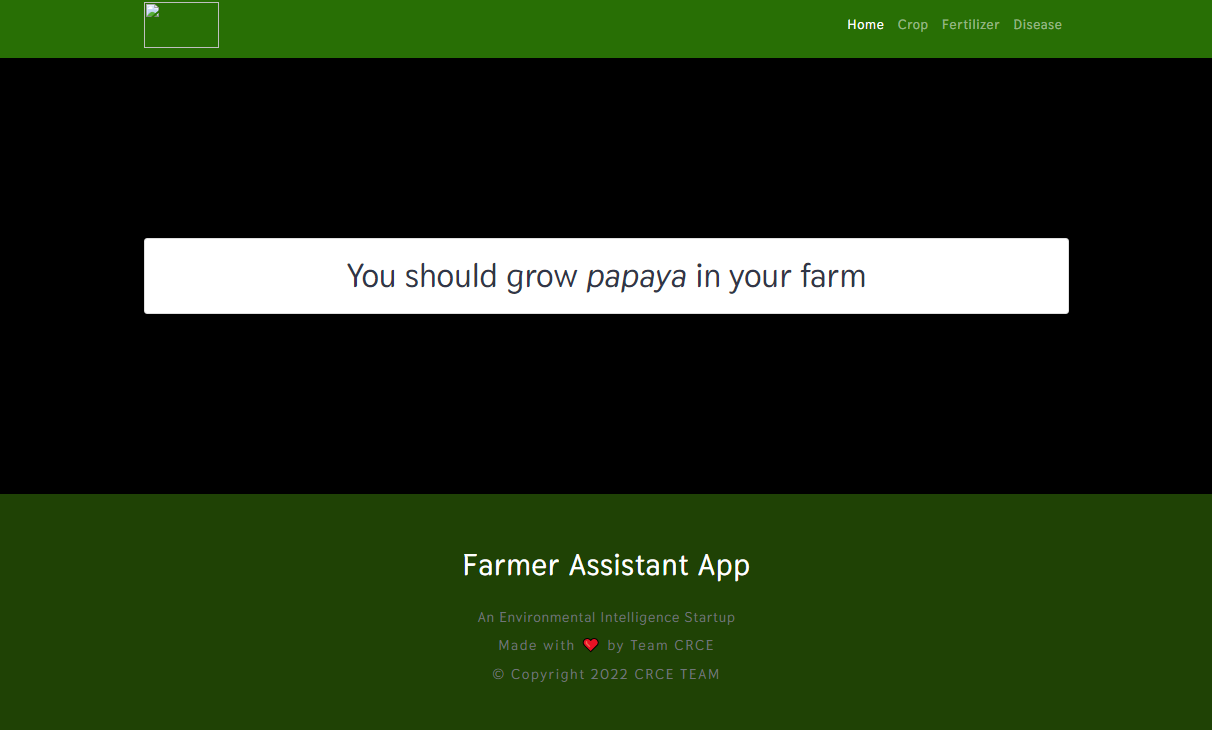
Preprocessing comes after data analysis and visualisation. Data cleaning and preparation for usage in machine learning algorithms are accomplished through the crucial step of data preprocessing. Preprocessing primarily focuses on addressing any missing values and outliers as well as inaccurate or outlier-containing data. There are two approaches to deal with missing data. The first approach is to just delete the entire row that contains the incorrect or missing information. This method is simple to apply, however it should only be applied to huge datasets. If there are a lot of missing values, using this strategy to tiny datasets may result in an excessive reduction in dataset size. The accuracy of the outcome may be seriously impacted by this. We won't be employing this strategy because our dataset is not that large. Since the values in the dataset we utilised were in string format, we had to transform and encode them into integer values before feeding them to the neural network. We first transformed the data into pandas categorical data, made codes for the various states and crops, attached these, and built distinct datasets.

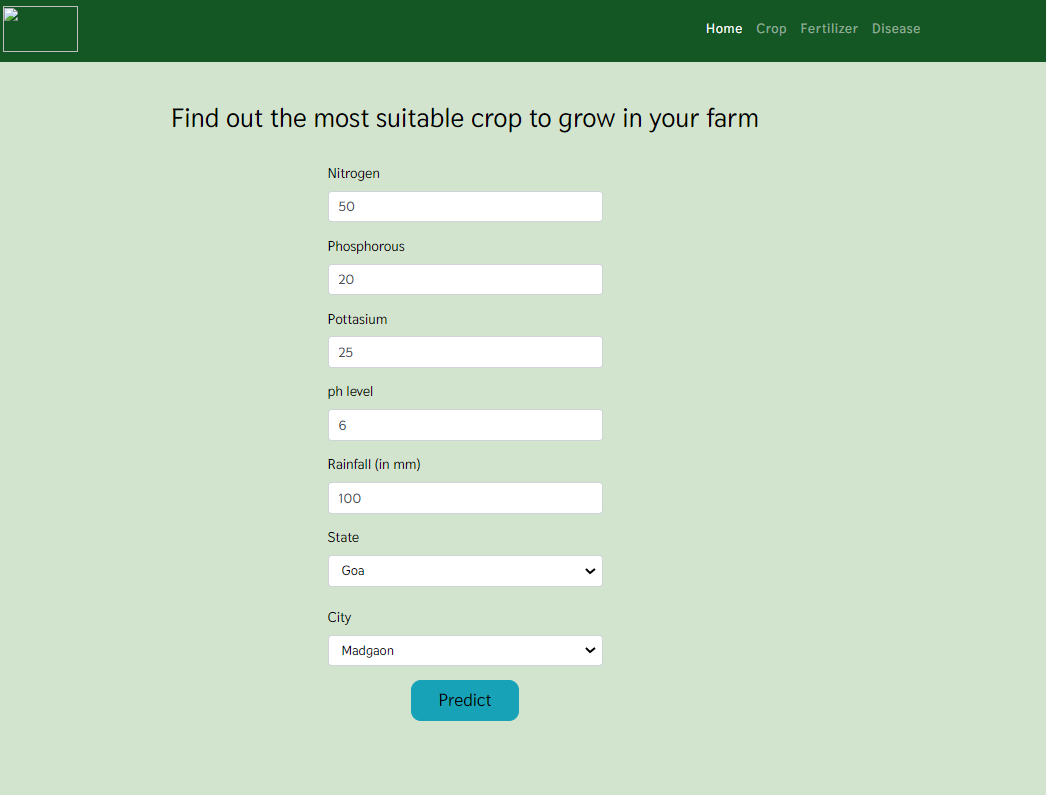
**Chapter 6**

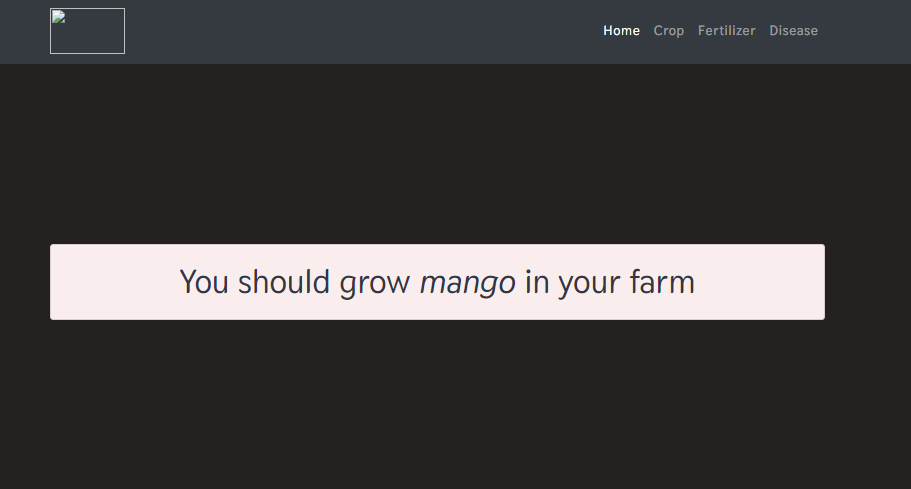
**Result**

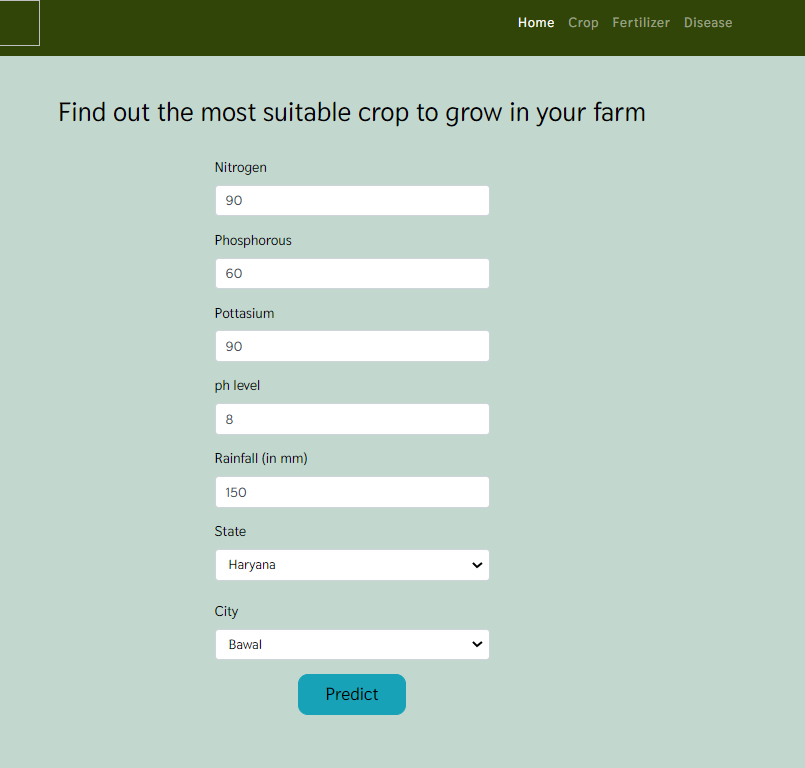
**Crop Recommendation (Input and Output)**

****

****

****

****

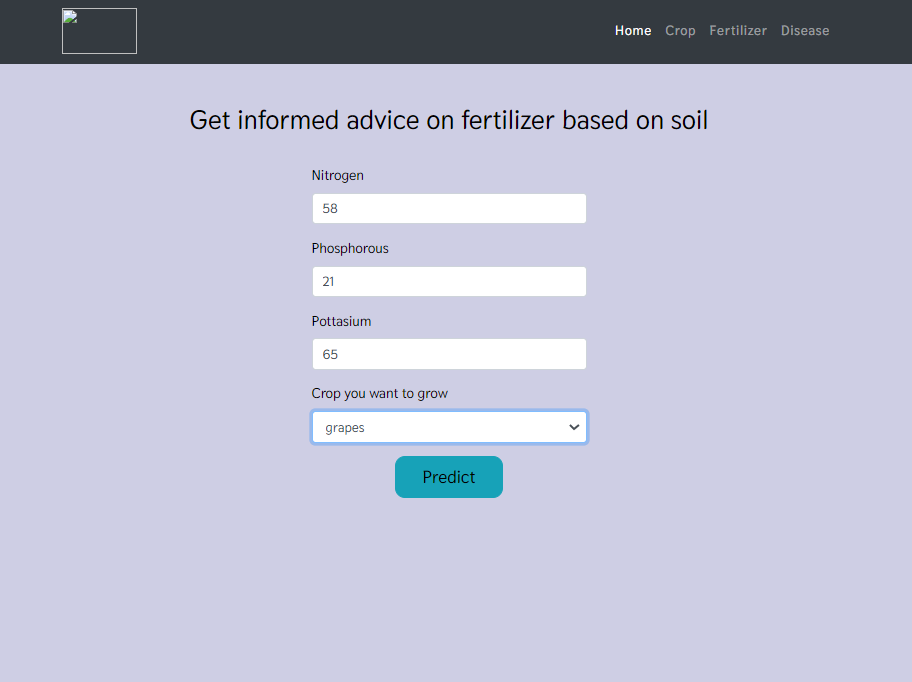
****

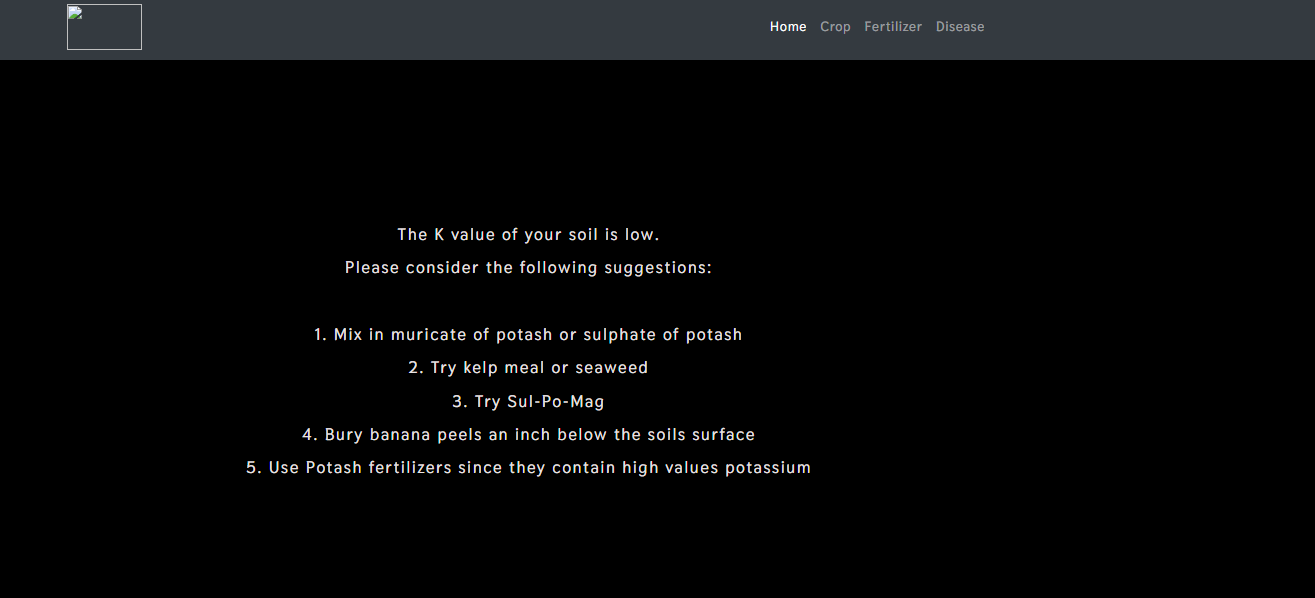
****

**Fertilizer Recommendation(Input and Output)**

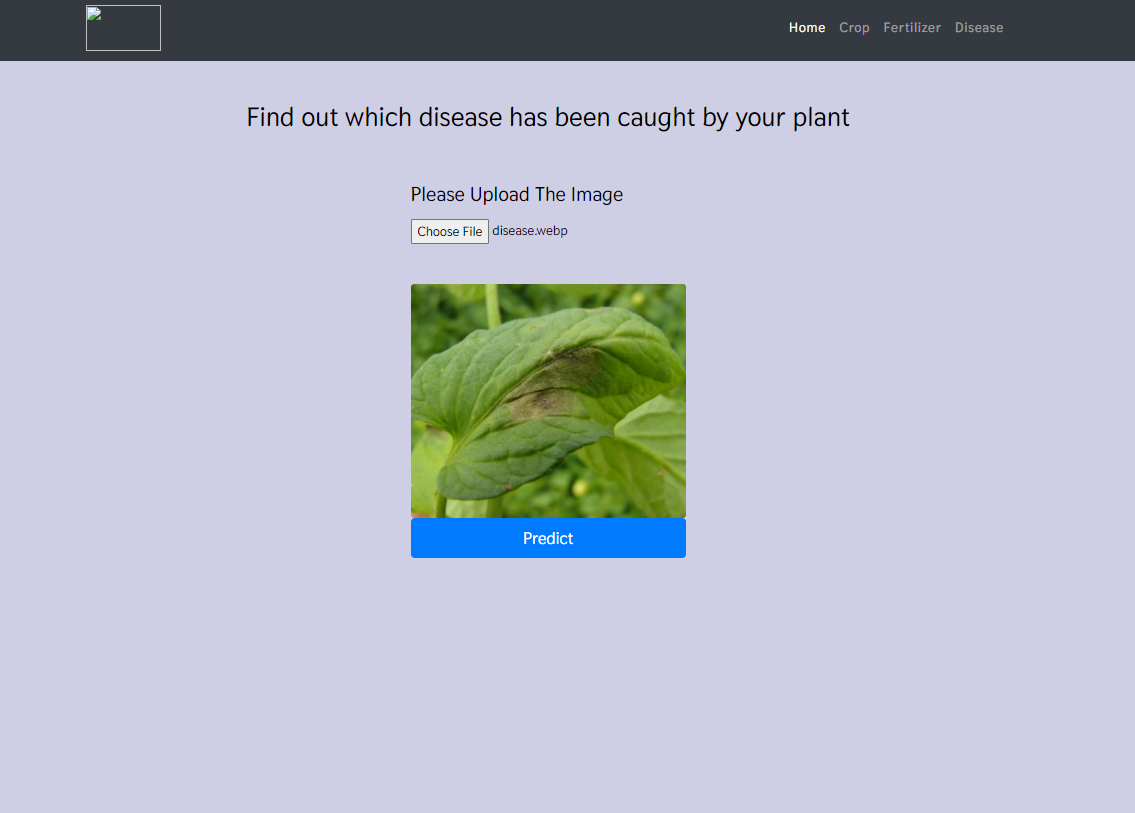
****

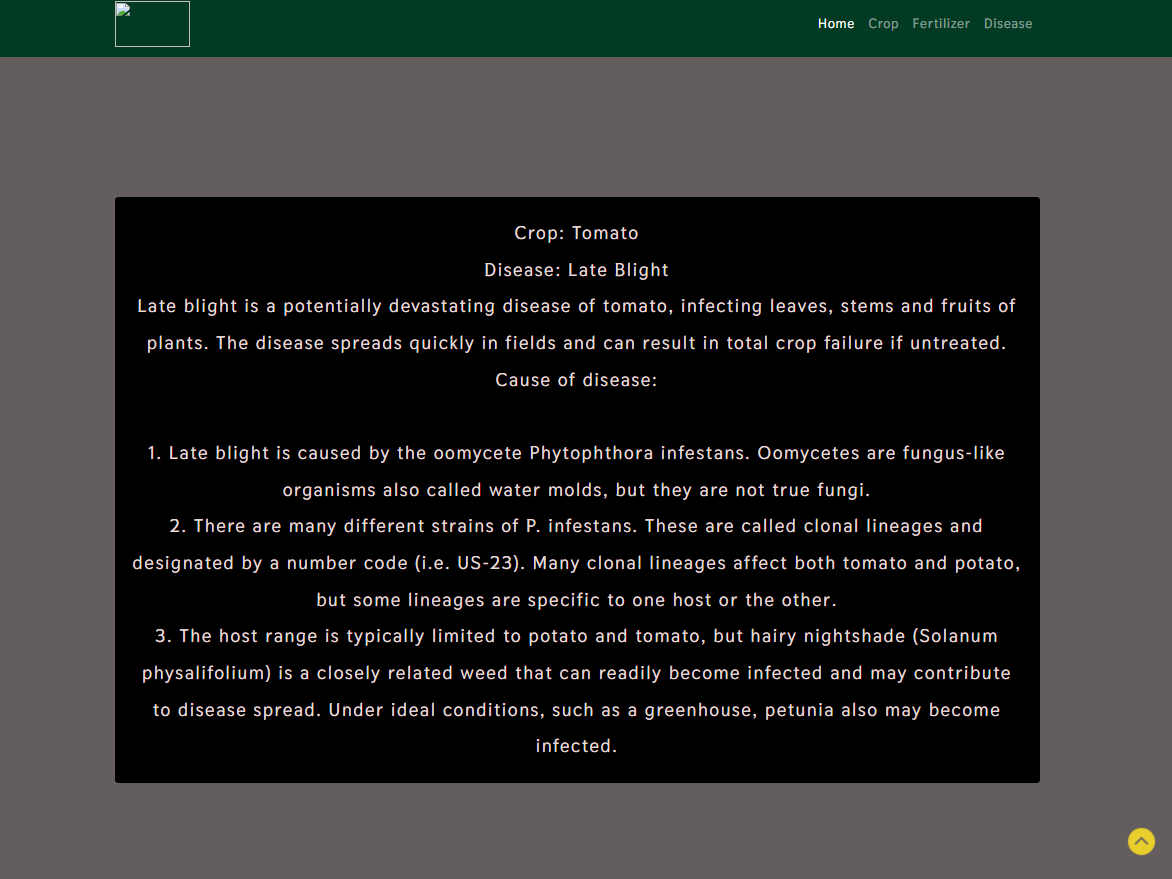
****

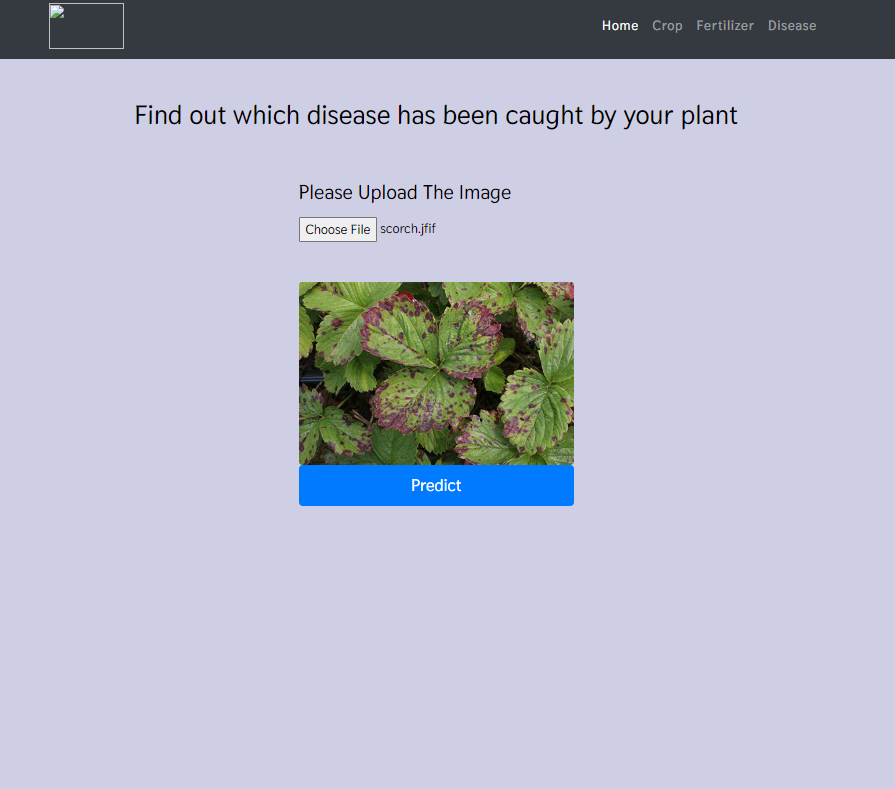
****

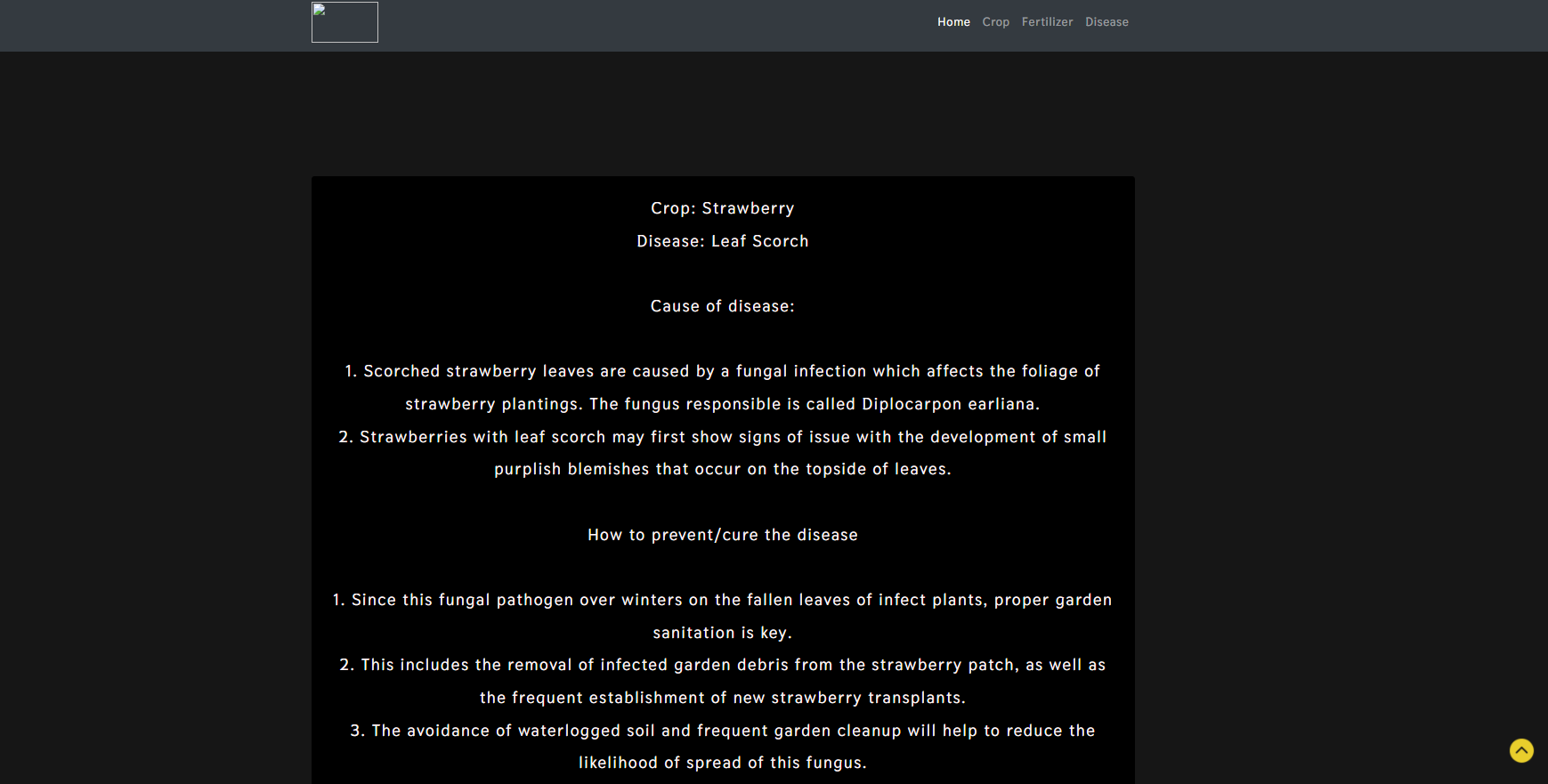
****

**Disease Prediction**

****

****

****

****

**Accuracy**

The four models we employed for our crop recommendation system, as shown in table no. 2.1, which have the highest accuracy.

Table No. 2.1 Crop Recommendation

|  |  |
| --- | --- |
| **Algorithm** | **Accuracy (%)** |
| Logistic Regression | 95 |
| Decision Tree | 90 |
| Random Forest | 99 |
| SVM | 97 |

In order to predict plant diseases, we now only used one model i.e the ResNet model which able to predict diseases with an accuracy of 98.03 percent.

Table no. 2.2 Disease Prediction

|  |  |
| --- | --- |
| **Algorithm** | **Accuracy (%)** |
| ResNets | 98.03 |

# Chapter 7

## Conclusion And Future Enhancements

### 7.1 Conclusion

The entire system aids in crop selection by offering information that most farmers would not be aware of, lowering the likelihood of crop failure and raising output. Additionally, it stops them from suffering losses. Millions of farmers around the nation may access the system, which can be expanded to the online. The crop recommendation system will be further developed to connect with a yield predictor, another subsystem that would also give the farmer an estimate of production if he plants the recommended crop. The outcomes are beneficial because they allow farmers to operate more effectively.

### 7.2 Future Enhancements

* We'll include new machine learning models.
* Additional plant photos can be gathered to strengthen and generalise the disease detection component.
* In order to create a large data set for fertiliser and improve accuracy, we will gather more information.

**References**

[1] 2019, 10th International Conference on Computing, Communication and Networking Technologies, “Low-cost IOT+ML design for smart farming with multiple applications”, Fahad Kamraan Syed, Agniswar Paul, Ajay Kumar, Jaideep Cherukuri.

[2] 2019 IEEE “ Smart Management of Crop Cultivation using IoT and Machine Learning” Archana Gupta, Dharmil Nagda, Pratiksha Nikhare, Atharva Sandbhor

[3] Radhika, Narendiran, “Kind of Crops and Small Plants Prediction using IoT with Machine earning,” International Journal of Computer & Mathematical Sciences, 2018.

[4] “Crop Recommendation on Analyzing Soil Using Machine Learning” Anguraj.Ka, Thiyaneswaran.Bb, Megashree.Gc, Preetha Shri.J.Gd, Navya.Se, Jayanthi. Jf, 2020.

[5] “Classification of Soil and Crop Suggestion using Machine Learning Techniques”, A. Mythili IEEE 2019.

[6] Mehta, P., Shah, H., Kori, V., Vikani, V., Shukla, S., & Shenoy, M.,2018. “Survey of unsupervised machine learning algorithms on precision agricultural data”, IEEE

[7] “IOT based Crop Recommendation, Crop Disease Prediction and Its Solution” Rani Holambe, Pooja Patil, Padmaja Pawar, Saurabh Salunkhe , Mr. Hrushikesh Joshi, 2019 IRJET

[8] Tom M. Mitchell, Machine Learning, India Edition 2013, McGrawHill Education.

[9] <https://data.gov.in/>

[10][https://www.javatpoint.com/machine-learning-support-vector-v machine-algorithm](https://www.javatpoint.com/machine-learning-support-vector-v%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20machine-algorithm)

[11]<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

[12] <https://www.geeksforgeeks.org/decision-tree/>

[13][https://www.geeksforgeeks.org/deploy-machine-learning-model- using-flask/](https://www.geeksforgeeks.org/deploy-machine-learning-model-%20%20%20%20using-flask/)

[14][https://www.analyticsvidhya.com/blog/2020/09/integrating- machine-learning- into- web-applications-with-flask/](https://www.analyticsvidhya.com/blog/2020/09/integrating-%20%20%20%20%20%20machine-learning-%20%20into-%20%20%20%20%20%20%20%20web-applications-with-flask/)

[15]<https://www.geeksforgeeks.org/xgboost/>

**Appendix – 1**

**Crop Recommendation Code:**

|  |
| --- |
| pip install numpy  # Importing libraries  from \_\_future\_\_ import print\_function  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.metrics import classification\_report  from sklearn import metrics  from sklearn import tree  import warnings  warnings.filterwarnings('ignore')  PATH = 'Crop\_recommendation.csv'  df = pd.read\_csv(PATH)  df.head()  df.tail()  df.size  df.shape  df['label'].unique()  df.dtypes  df['label'].value\_counts()  sns.heatmap(df.corr(),annot=True)  features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]  target = df['label']  labels = df['label']  # Initializing empty lists to append all model's name and corresponding name  acc = []  model = []  # Splitting into train and test data  from sklearn.model\_selection import train\_test\_split  Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(features,target,test\_size = 0.2,random\_state =2)  from sklearn.tree import DecisionTreeClassifier  DecisionTree = DecisionTreeClassifier(criterion="entropy",random\_state=2,max\_depth=5)  DecisionTree.fit(Xtrain,Ytrain)  predicted\_values = DecisionTree.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('Decision Tree')  print("DecisionTrees's Accuracy is: ", x\*100)  print(classification\_report(Ytest,predicted\_values))  from sklearn.model\_selection import cross\_val\_score  # Cross validation score (Decision Tree)  score = cross\_val\_score(DecisionTree, features, target,cv=5)  score  import pickle  # Dump the trained Naive Bayes classifier with Pickle  DT\_pkl\_filename = 'DecisionTree.pkl'  # Open the file to save as pkl file  DT\_Model\_pkl = open(DT\_pkl\_filename, 'wb')  pickle.dump(DecisionTree, DT\_Model\_pkl)  # Close the pickle instances  DT\_Model\_pkl.close()  from sklearn.naive\_bayes import GaussianNB  NaiveBayes = GaussianNB()  NaiveBayes.fit(Xtrain,Ytrain)  predicted\_values = NaiveBayes.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('Naive Bayes')  print("Naive Bayes's Accuracy is: ", x)  print(classification\_report(Ytest,predicted\_values))  # Cross validation score (NaiveBayes)  score = cross\_val\_score(NaiveBayes,features,target,cv=5)  score  import pickle  # Dump the trained Naive Bayes classifier with Pickle  NB\_pkl\_filename = 'NBClassifier.pkl'  # Open the file to save as pkl file  NB\_Model\_pkl = open(NB\_pkl\_filename, 'wb')  pickle.dump(NaiveBayes, NB\_Model\_pkl)  # Close the pickle instances  NB\_Model\_pkl.close()  from sklearn.svm import SVC  SVM = SVC(gamma='auto')  SVM.fit(Xtrain,Ytrain)  predicted\_values = SVM.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('SVM')  print("SVM's Accuracy is: ", x)  print(classification\_report(Ytest,predicted\_values))  # Cross validation score (SVM)  score = cross\_val\_score(SVM,features,target,cv=5)  score  from sklearn.linear\_model import LogisticRegression  LogReg = LogisticRegression(random\_state=2)  LogReg.fit(Xtrain,Ytrain)  predicted\_values = LogReg.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('Logistic Regression')  print("Logistic Regression's Accuracy is: ", x)  print(classification\_report(Ytest,predicted\_values))  # Cross validation score (Logistic Regression)  score = cross\_val\_score(LogReg,features,target,cv=5)  score  import pickle  # Dump the trained Naive Bayes classifier with Pickle  LR\_pkl\_filename = 'LogisticRegression.pkl'  # Open the file to save as pkl file  LR\_Model\_pkl = open(DT\_pkl\_filename, 'wb')  pickle.dump(LogReg, LR\_Model\_pkl)  # Close the pickle instances  LR\_Model\_pkl.close()  from sklearn.ensemble import RandomForestClassifier  RF = RandomForestClassifier(n\_estimators=20, random\_state=0)  RF.fit(Xtrain,Ytrain)  predicted\_values = RF.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('RF')  print("RF's Accuracy is: ", x)  print(classification\_report(Ytest,predicted\_values))  # Cross validation score (Random Forest)  score = cross\_val\_score(RF,features,target,cv=5)  score  import pickle  # Dump the trained Naive Bayes classifier with Pickle  RF\_pkl\_filename = 'RandomForest.pkl'  # Open the file to save as pkl file  RF\_Model\_pkl = open(RF\_pkl\_filename, 'wb')  pickle.dump(RF, RF\_Model\_pkl)  # Close the pickle instances  RF\_Model\_pkl.close()  pip install xgboost  import xgboost as xgb  XB = xgb.XGBClassifier()  XB.fit(Xtrain,Ytrain)  predicted\_values = XB.predict(Xtest)  x = metrics.accuracy\_score(Ytest, predicted\_values)  acc.append(x)  model.append('XGBoost')  print("XGBoost's Accuracy is: ", x)  print(classification\_report(Ytest,predicted\_values))  # Cross validation score (XGBoost)  score = cross\_val\_score(XB,features,target,cv=5)  score  import pickle  # Dump the trained Naive Bayes classifier with Pickle  XB\_pkl\_filename = 'XGBoost.pkl'  # Open the file to save as pkl file  XB\_Model\_pkl = open(XB\_pkl\_filename, 'wb')  pickle.dump(XB, XB\_Model\_pkl)  # Close the pickle instances  XB\_Model\_pkl.close()  plt.figure(figsize=[10,5],dpi = 100)  plt.title('Accuracy Comparison')  plt.xlabel('Accuracy')  plt.ylabel('Algorithm')  sns.barplot(x = acc,y = model,palette='dark')  accuracy\_models = dict(zip(model, acc))  for k, v in accuracy\_models.items():      print (k, '-->', v)  data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])  prediction = RF.predict(data)  print(prediction)  data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])  prediction = RF.predict(data)  print(prediction) |

**Fertilizer Recommendation:**

|  |
| --- |
| ## Notebook for transforming raw cpdata to Mergable data  ### Filter cpdata.csv to MergeFileCrop.cv  ### Filter fertilizer.csv to MergerFileFert.csv  import pandas as pd  # Reading the data  crop\_data\_path = '../Data-raw/cpdata.csv'  fertilizer\_data\_path = '../Data-raw/Fertilizer.csv'  crop = pd.read\_csv(crop\_data\_path)  fert = pd.read\_csv(fertilizer\_data\_path)  crop.head()  fert.head()  # Function for lowering the cases  def change\_case(i):  i = i.replace(" ", "")  i = i.lower()  return i  fert['Crop'] = fert['Crop'].apply(change\_case)  crop['label'] = crop['label'].apply(change\_case)  #make some changes in ferttilizer dataset  fert['Crop'] = fert['Crop'].replace('mungbeans','mungbean')  fert['Crop'] = fert['Crop'].replace('lentils(masoordal)','lentil')  fert['Crop'] = fert['Crop'].replace('pigeonpeas(toordal)','pigeonpeas')  fert['Crop'] = fert['Crop'].replace('mothbean(matki)','mothbeans')  fert['Crop'] = fert['Crop'].replace('chickpeas(channa)','chickpea')  crop.head()  crop.tail()  crop\_names = crop['label'].unique()  crop\_names  fert.head()  del fert['Unnamed: 0']  crop\_names\_from\_fert = fert['Crop'].unique()  crop\_names\_from\_fert  for i in crop\_names\_from\_fert:  print(crop[crop['label'] == i])  crop['label']  extract\_labels = []  for i in crop\_names\_from\_fert:  if i in crop\_names:  extract\_labels.append(i)  # using extract labesl on crop to get all the data related to those labels  new\_crop = pd.DataFrame(columns = crop.columns)  new\_fert = pd.DataFrame(columns = fert.columns)  for label in extract\_labels:  new\_crop = new\_crop.append(crop[crop['label'] == label])  for label in extract\_labels:  new\_fert = new\_fert.append(fert[fert['Crop'] == label].iloc[0])  new\_crop  new\_fert  new\_crop.to\_csv('../Data-raw/MergeFileCrop.csv')  new\_fert.to\_csv('../Data-raw/FertilizerData.csv') |

**Disease Prediction Code:**

|  |
| --- |
| !pip install torchsummary  pip install torchvision  import os                       # for working with files  import numpy as np              # for numerical computationss  import pandas as pd             # for working with dataframes  import torch                    # Pytorch module  import matplotlib.pyplot as plt # for plotting informations on graph and images using tensors  import torch.nn as nn           # for creating  neural networks  from torch.utils.data import DataLoader # for dataloaders  from PIL import Image           # for checking images  import torch.nn.functional as F # for functions for calculating loss  import torchvision.transforms as transforms   # for transforming images into tensors  from torchvision.utils import make\_grid       # for data checking  from torchvision.datasets import ImageFolder  # for working with classes and images  from torchsummary import summary              # for getting the summary of our model  %matplotlib inline  data\_dir = 'C:\\Users\\PRATIK\\Desktop\\BE Project\\input\\New Plant Diseases Dataset(Augmented)\\New Plant Diseases Dataset(Augmented)'  train\_dir = data\_dir + '\\train'  valid\_dir = data\_dir + '\\valid'  diseases = os.listdir(train\_dir)  # printing the disease names  print(diseases)  print("Total disease classes are: {}".format(len(diseases)))  plants = []  NumberOfDiseases = 0  for plant in diseases:      if plant.split('\_\_\_')[0] not in plants:          plants.append(plant.split('\_\_\_')[0])      if plant.split('\_\_\_')[1] != 'healthy':          NumberOfDiseases += 1  # unique plants in the dataset  print(f"Unique Plants are: \n{plants}")  # number of unique plants  print("Number of plants: {}".format(len(plants)))  # number of unique diseases  print("Number of diseases: {}".format(NumberOfDiseases))  # Number of images for each disease  nums = {}  for disease in diseases:      nums[disease] = len(os.listdir(train\_dir + '/' + disease))    # converting the nums dictionary to pandas dataframe passing index as plant name and number of images as column  img\_per\_class = pd.DataFrame(nums.values(), index=nums.keys(), columns=["no. of images"])  img\_per\_class  #pip install -U numpy  # plotting number of images available for each disease  index = [n for n in range(38)]  plt.figure(figsize=(20, 5))  plt.bar(index, [n for n in nums.values()], width=0.3)  plt.xlabel('Plants/Diseases', fontsize=10)  plt.ylabel('No of images available', fontsize=10)  plt.xticks(index, diseases, fontsize=5, rotation=90)  plt.title('Images per each class of plant disease')  #Images available for training  n\_train = 0  for value in nums.values():      n\_train += value  print(f"There are {n\_train} images for training")  #Data Preparation for training  # datasets for validation and training  train = ImageFolder(train\_dir, transform=transforms.ToTensor())  valid = ImageFolder(valid\_dir, transform=transforms.ToTensor())  #Image shape  img, label = train[0]  print(img.shape, label)  # total number of classes in train set  len(train.classes)  # for checking some images from training dataset  def show\_image(image, label):      print("Label :" + train.classes[label] + "(" + str(label) + ")")      plt.imshow(image.permute(1, 2, 0))   #Some Images from training dataset  show\_image(\*train[0])  show\_image(\*train[70])  show\_image(\*train[30000])  # Setting the seed value  random\_seed = 7  torch.manual\_seed(random\_seed)  # setting the batch size  batch\_size = 10  # DataLoaders for training and validation  train\_dl = DataLoader(train, batch\_size, shuffle=True, num\_workers=2, pin\_memory=True)  valid\_dl = DataLoader(valid, batch\_size, num\_workers=2, pin\_memory=True)  # helper function to show a batch of training instances  def show\_batch(data):      for images, labels in data:          fig, ax = plt.subplots(figsize=(30, 30))          ax.set\_xticks([]); ax.set\_yticks([])          ax.imshow(make\_grid(images, nrow=8).permute(1, 2, 0))          break  # Images for first batch of training  #show\_batch(train\_dl)  # 🏗️ Modelling 🏗️  # for moving data into GPU (if available)  def get\_default\_device():      """Pick GPU if available, else CPU"""      if torch.cuda.is\_available:          return torch.device("cuda")      else:          return torch.device("cpu")  # for moving data to device (CPU or GPU)  def to\_device(data, device):      """Move tensor(s) to chosen device"""      if isinstance(data, (list,tuple)):          return [to\_device(x, device) for x in data]      return data.to(device, non\_blocking=True)  # for loading in the device (GPU if available else CPU)  class DeviceDataLoader():      """Wrap a dataloader to move data to a device"""      def \_\_init\_\_(self, dl, device):          self.dl = dl          self.device = device        def \_\_iter\_\_(self):          """Yield a batch of data after moving it to device"""          for b in self.dl:              yield to\_device(b, self.device)        def \_\_len\_\_(self):          """Number of batches"""          return len(self.dl)  device = get\_default\_device()  device  # Moving data into GPU  train\_dl = DeviceDataLoader(train\_dl, device)  valid\_dl = DeviceDataLoader(valid\_dl, device)  #👷 Building the model architecture 👷  #Residual Block code implementation  class SimpleResidualBlock(nn.Module):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.conv1 = nn.Conv2d(in\_channels=3, out\_channels=3, kernel\_size=3, stride=1, padding=1)          self.relu1 = nn.ReLU()          self.conv2 = nn.Conv2d(in\_channels=3, out\_channels=3, kernel\_size=3, stride=1, padding=1)          self.relu2 = nn.ReLU()        def forward(self, x):          out = self.conv1(x)          out = self.relu1(out)          out = self.conv2(out)          return self.relu2(out) + x # ReLU can be applied before or after adding the input  # for calculating the accuracy  def accuracy(outputs, labels):      \_, preds = torch.max(outputs, dim=1)      return torch.tensor(torch.sum(preds == labels).item() / len(preds))  # base class for the model  class ImageClassificationBase(nn.Module):        def training\_step(self, batch):          images, labels = batch          out = self(images)                  # Generate predictions          loss = F.cross\_entropy(out, labels) # Calculate loss          return loss        def validation\_step(self, batch):          images, labels = batch          out = self(images)                   # Generate prediction          loss = F.cross\_entropy(out, labels)  # Calculate loss          acc = accuracy(out, labels)          # Calculate accuracy          return {"val\_loss": loss.detach(), "val\_accuracy": acc}        def validation\_epoch\_end(self, outputs):          batch\_losses = [x["val\_loss"] for x in outputs]          batch\_accuracy = [x["val\_accuracy"] for x in outputs]          epoch\_loss = torch.stack(batch\_losses).mean()       # Combine loss          epoch\_accuracy = torch.stack(batch\_accuracy).mean()          return {"val\_loss": epoch\_loss, "val\_accuracy": epoch\_accuracy} # Combine accuracies        def epoch\_end(self, epoch, result):          print("Epoch [{}], last\_lr: {:.5f}, train\_loss: {:.4f}, val\_loss: {:.4f}, val\_acc: {:.4f}".format(              epoch, result['lrs'][-1], result['train\_loss'], result['val\_loss'], result['val\_accuracy']))    #👷 Defining the final architecture of our model 👷  # Architecture for training  # convolution block with BatchNormalization  def ConvBlock(in\_channels, out\_channels, pool=False):      layers = [nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, padding=1),               nn.BatchNorm2d(out\_channels),               nn.ReLU(inplace=True)]      if pool:          layers.append(nn.MaxPool2d(4))      return nn.Sequential(\*layers)  # resnet architecture  class ResNet9(ImageClassificationBase):      def \_\_init\_\_(self, in\_channels, num\_diseases):          super().\_\_init\_\_()            self.conv1 = ConvBlock(in\_channels, 64)          self.conv2 = ConvBlock(64, 128, pool=True) # out\_dim : 128 x 64 x 64          self.res1 = nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))            self.conv3 = ConvBlock(128, 256, pool=True) # out\_dim : 256 x 16 x 16          self.conv4 = ConvBlock(256, 512, pool=True) # out\_dim : 512 x 4 x 44          self.res2 = nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))            self.classifier = nn.Sequential(nn.MaxPool2d(4),                                         nn.Flatten(),                                         nn.Linear(512, num\_diseases))        def forward(self, xb): # xb is the loaded batch          out = self.conv1(xb)          out = self.conv2(out)          out = self.res1(out) + out          out = self.conv3(out)          out = self.conv4(out)          out = self.res2(out) + out          out = self.classifier(out)          return out  # defining the model and moving it to the GPU  model = to\_device(ResNet9(3, len(train.classes)), device)  model  # getting summary of the model  INPUT\_SHAPE = (3, 256, 256)  print(summary(model.cuda(), (INPUT\_SHAPE)))  # 🏋️ Training the model 🏋️  # for training  @torch.no\_grad()  def evaluate(model, val\_loader):      model.eval()      outputs = [model.validation\_step(batch) for batch in val\_loader]      return model.validation\_epoch\_end(outputs)  def get\_lr(optimizer):      for param\_group in optimizer.param\_groups:          return param\_group['lr']    def fit\_OneCycle(epochs, max\_lr, model, train\_loader, val\_loader, weight\_decay=0,                  grad\_clip=None, opt\_func=torch.optim.SGD):      torch.cuda.empty\_cache()      history = []        optimizer = opt\_func(model.parameters(), max\_lr, weight\_decay=weight\_decay)      # scheduler for one cycle learniing rate      sched = torch.optim.lr\_scheduler.OneCycleLR(optimizer, max\_lr, epochs=epochs, steps\_per\_epoch=len(train\_loader))          for epoch in range(epochs):          # Training          model.train()          train\_losses = []          lrs = []          for batch in train\_loader:              loss = model.training\_step(batch)              train\_losses.append(loss)              loss.backward()                # gradient clipping              if grad\_clip:                  nn.utils.clip\_grad\_value\_(model.parameters(), grad\_clip)                optimizer.step()              optimizer.zero\_grad()                # recording and updating learning rates              lrs.append(get\_lr(optimizer))              sched.step()              # validation          result = evaluate(model, val\_loader)          result['train\_loss'] = torch.stack(train\_losses).mean().item()          result['lrs'] = lrs          model.epoch\_end(epoch, result)          history.append(result)        return history  %%time  history = [evaluate(model, valid\_dl)]  history  epochs = 2  max\_lr = 0.01  grad\_clip = 0.1  weight\_decay = 1e-4  opt\_func = torch.optim.Adam  %%time  history += fit\_OneCycle(epochs, max\_lr, model, train\_dl, valid\_dl,                               grad\_clip=grad\_clip,                               weight\_decay=1e-4,                               opt\_func=opt\_func)  # 📈 Plotting 📈  #Helper functions for plotting  def plot\_accuracies(history):      accuracies = [x['val\_accuracy'] for x in history]      plt.plot(accuracies, '-x')      plt.xlabel('epoch')      plt.ylabel('accuracy')      plt.title('Accuracy vs. No. of epochs');  def plot\_losses(history):      train\_losses = [x.get('train\_loss') for x in history]      val\_losses = [x['val\_loss'] for x in history]      plt.plot(train\_losses, '-bx')      plt.plot(val\_losses, '-rx')      plt.xlabel('epoch')      plt.ylabel('loss')      plt.legend(['Training', 'Validation'])      plt.title('Loss vs. No. of epochs');    def plot\_lrs(history):      lrs = np.concatenate([x.get('lrs', []) for x in history])      plt.plot(lrs)      plt.xlabel('Batch no.')      plt.ylabel('Learning rate')      plt.title('Learning Rate vs. Batch no.');  # 🧪 Testing model on test data 🧪  #We only have 33 images in test data, so let's check the model on all images  test\_dir = "C:\\Users\\PRATIK\\Desktop\\BE Project\\input\\test"  test = ImageFolder(test\_dir, transform=transforms.ToTensor())  test\_images = sorted(os.listdir(test\_dir + '/test')) # since images in test folder are in alphabetical order  test\_images  def predict\_image(img, model):      """Converts image to array and return the predicted class          with highest probability"""      # Convert to a batch of 1      xb = to\_device(img.unsqueeze(0), device)      # Get predictions from model      yb = model(xb)      # Pick index with highest probability      \_, preds  = torch.max(yb, dim=1)      # Retrieve the class label      return train.classes[preds[0].item()]  # predicting first image  img, label = test[4]  plt.imshow(img.permute(1, 2, 0))  print('Label:', test\_images[4], ', Predicted:', predict\_image(img, model))  # getting all predictions (actual label vs predicted)  for i, (img, label) in enumerate(test):      print('Label:', test\_images[i], ', Predicted:', predict\_image(img, model)) |

**Appendix - 2 (Plagarism Report)**