MACHINE LEARNING APPLICATION FOR CROP PREDICTION AND EFFICIENT FERTILIZER RECOMMENDATION USING PYTHON

A PROJECT REPORT

Submitted to

SRI VENKATESWARA UNIVERSITY

COLLEGE OF ENGINEERING

In partial fulfilment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

MUKTESWAR REDDY B	**(11716082)**
SAI SAHITH REDDY M	**(11716090)**
SWETHA K	**(11716097)**

Under the guidance of

Dr. P. Venkata Subba Reddy

HOD, Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI VENKATESWARA UNIVERSITY

COLLEGE OF ENGINEERING, TIRUPATI – 517502

2020-2021

SRI VENKATESWARA UNIVERSITY
COLLEGE OF ENGINEERING, TIRUPATI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"Machine Learning Application For Crop prediction and Efficient Fertilizer Recommendation Using Python"** was genuine and has been carried out under my supervision in the **Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering.** The work is comprehensive, complete and fit for evaluation carried out in partial fulfilment of the requirements for the award of **Bachelor of Technology** in **Computer Science and Engineering** during the academic year 2020-21.

To the best of our knowledge matter embodied in the project has not been submitted to any other University/Institution for the award of any Degree or Diploma.

Guide:	Head of the Department:
Dr. P. VENKATA SUBBA REDDY	Dr. P. VENKATA SUBBA REDDY
HOD, Professor,	Professor,
Department of CSE,	Department of CSE,
SVUCE, Tirupati.	SVUCE, Tirupati.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI VENKATESWARA UNIVERSITY

COLLEGE OF ENGINEERING, TIRUPATI – 517502.

Declaration

The project entitled **“Machine Learning Application For Crop prediction and Efficient Fertilizer Recommendation Using Python”** is a bonafide work performed by us, for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** from **Sri Venkateswara University, Tirupati.**

To the best of our knowledge matter embodied in the project has not been submitted to any other University/Institution for the award of any Degree or Diploma.

[MUKTESWAR REDDY B	**(11716082)]**
[SAI SAHITH REDDY M	**(11716090)]**
[SWETHA K	**(11716097)]**

ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude and indebtedness to our guide **Dr. P. VENKATA SUBBA REDDY**, HOD, Professor of Department of Computer Science and Engineering, for his valuable suggestions and encouragement throughout the project and also for providing necessary facilities for enabling timely completion of our project design.

We are deeply obliged to **Dr. P. VENKATA SUBBA REDDY**, Head of the Department, CSE, for providing valuable suggestions and critical reviews on the project, after project work and helping us in improving and providing the necessary administrative help for providing required facilities in order to complete the project.

We extend our thanks to our Teaching staff's and friends, for insightful ideas and valuable criticism of our work. We thank all of them for helping us a lot throughout the course of our project.

Yours Sincerely,

MUKTESWAR REDDY B	**(11716082)**
SAI SAHITH REDDY M	**(11716090)**
SWETHA K	**(11716097)**

Abstract

As we all know, India is the world's second most populous nation, and agriculture is the primary source of income for the vast majority of Indians. Agriculture plays a significant role in the country's economic development. Climate change and other environmental developments have posed a serious threat to agriculture. Farmers cultivate the same crops year after year without experimenting with new varieties, and they apply fertilisers in an ad hoc manner without understanding the deficient content or quantity. As a result, this has a direct impact on crop production, as well as causing soil acidification and damage to the top layer.

By looking at the past few years, there have been significant developments in how machine learning can be used in various industries and research.

So, we have designed the system using machine learning algorithms and flask for betterment of farmers. Our system will suggest the best suitable crop for particular land based on content and weather parameters like temperature, humidity, pH, rainfall.

And also, the system provides information about the required content and quantity of fertilizers like calcium, magnesium, potassium, sulphur, nitrogen, lime, carbon, phosphorous, moisture. Using all these data the system will predict the most suitable crop and most suitable fertilizer for farmer. This system will be an extra hand to the farmers. Hence by utilizing our system farmers can cultivate a new variety of crop, may increase in profit margin and can avoid soil pollution.

CONTENTS

LIST OF FIGURES

Page no:

CHAPTER I

INTRODUCTION

1.1 – Objective:

Crop yield expectations are a major agrarian concern. Climate and pesticides have a significant impact on agricultural yield. Exact data on past harvest yields is critical for making decisions related to farming dangers and future expectations. The analysis of teaching machines to learn and build models for future forecasts is widely used, and for good reason. Agribusiness assumes a basic job in the worldwide economy. Agribusiness plays a vital role in the global economy. Understanding total harvest yield is critical to addressing food security issues and mitigating the impact of environmental change as the human population continues to grow. With the impact of environmental change in India, the majority share of agrarian yields has been negatively impacted in terms of presentation over the last two decades.

Predicting the harvest yield well ahead of time would aid in the promotion and capacity-building steps. Such standards would also assist related agreement producers and ranchers in fitting their livestock. Enterprises for managing their business's teamwork. Harvesting is a mind-boggling feat that is influenced by climatically input parameters.

The input parameters for agribusiness vary from field to field and rancher to rancher. Obtaining such information for a larger area is a daunting task. Nonetheless, the Indian Meteorological Department gathered climatic data at each square metre territory in various parts of the district. Furthermore, the yield of each harvest in each state is consistently gathered and distributed by the agribusiness and partnership division. Such data sets are currently being used to forecast the effect on significant harvests and, as a result, their yield in a future year.

1.2 – Motivation:

Prior crop prediction and yield prediction was performed on the basis of farmers experience on a particular location. They will prefer the prior or neighbourhood or more trend crop in the surrounding region only for their land and they don't have enough of knowledge about soil nutrients content such as nitrogen, phosphorus, potassium in the land.

Being this as the current situation without the rotation of the crop and apply an inadequate amount of nutrients to soil it leads to reduce in the yield and soil pollution (soil acidification) and damages the top layer. And also, many people not have proper knowledge on using fertilizers, using more fertilizers will damage land.

Considering all these problems takes into the account we designed the system using a machine learning for betterment of the farmer. Machine learning (ML) is a game changer for agriculture sector. Machine learning is the part of artificial intelligence, has emerged together with bigdata technologies and high-performance computing to create new opportunities for data intensive science in the multi-disciplinary agrotechnology domain.

1.3- Chapters organization:

Chapter 1: Introduction

This chapter explains the objectives and motivation of the crop and fertilizer prediction.

Chapter 2: Literature survey

This chapter gives a detailed explanation of previous works done by other researchers. It provides inference and open problems for our proposed system.

Chapter 3: System Analysis and Design

This chapter gives detailed description about the hardware and software requirements for the proposed project. It tells why those components are chosen and explains about the design diagrams.

Chapter 4: Methodology

This chapter describes about how the system is implemented using machine Learning which has some important steps, they are: Pre-preocessing, Training model.

Chapter 5: Results

This chapter provides results of the machine learning models and the best model is chosen based on the results.

Chapter 6: Flask Web App

This chapter shows about the user interface of the system using flask. Code snippets are included along with screenshots of the user interface.

Chapter 7: Conclusion and Future Scope

This chapter provides a final conclusion of the project and also the applications of this project. Also, it shows the future scope of the project.

.

CHAPTER II

LITERATURE SURVEY

2.1- Background and Existing Research Work:

Anil Suat Terliksiz et.al., concentrated on soybean yield forecast of Lauderdale County, Alabama, USA utilizing 3D CNN model that use the spatiotemporal highlights [1]. The yield is given from USDA NASS Quick Stat apparatus for a considerable length of time 2003-2016. The expectation of harvest yield has direct effect on national and worldwide economies and assume significant job in the nourishment the executives and nourishment security.

Niketa Gandhi et.al. [2] Proposed a choice emotionally supportive network model for rice crop yield forecast for Maharashtra state, India. A GUI has been made in Java utilizing NetBeans apparatus and Microsoft Office Access database for the simplicity of ranchers and leaders. The interface takes into account the determination of the scope of precipitation, least temperature, normal temperature, most extreme temperature and reference crop evapotranspiration and predicts the normal class of yield viz., low, moderate or high.

Ranjini B Guruprasad et.al.,[3] introduced a contextual analysis of climate and soil information-based yield estimation demonstrating for paddy crop at various spatial goals (SR) levels, to be specific, at the area and taluk levels in India. We give a point by point investigation of precision of the yield estimation models across changed arrangements of highlights and diverse AI systems. Nilima et.al., [4] introduced a thought for example to how to send WSN on field and how Machine learning model is fitted for forecast of bug/ailments utilizing Naive Bayes Kernel Algorithm.

Remote Sensor Network is new innovation to world and nation like India where it can utilize in Agriculture Sector in India for expanding yield by giving early expectation of plant sicknesses and bug. This can be occurred by taking crude information from field where WSN organize is introduce and with fitting proper AI model for this information to get anticipated yield.

Shruti Kulkarni et.al., presents a model for example an information driven model that learns by notable soil just as precipitation information to break down and anticipate crop yield over seasons in a few locales, has been created [5]. For this investigation, a specific yield, Rice is considered.

The planned half breed neural system model distinguishes ideal mixes of soil parameters and mixes it with the precipitation design in a chose locale to develop the expectable harvest yield. The spine for the prescient investigation model regarding the precipitation depends on the Time-Series approach in Supervised Learning.

T. Mhudchuay et.al. [8] Concentrated on downpour took care of rice where the fundamental activities are when to begin development and when to collect. The objective is to locate the ideal development and collect period to such an extent that ranchers' salary is amplified. This paper speaks to a use of a Deep Q-learning in the rice crop development practice, where the ideal activities are resolved.

Shivi Sharma et.al., [9] proposed a technique utilized, in that dirt and condition highlights for example normal temperature, normal stickiness, all out precipitation and creation yield are utilized in anticipating two classes in particular: great yield and awful yield.

Suhas S Athani et.al. [10] Presents the data relating to the harm of harvests as of late because of the development of weeds. Weeds are one of the significant hazards to the genuine home and mankind. Right now, thought, Support Vector Machine (SVM) Classifier is used to make out whether plant is harvest or weed. The maize crops are consistently observed by catching pictures utilizing camera. So as to group a plant as a yield or weed, different highlights are removed which among them are shape, surface, shading.

CHAPTER III

SYSTEM ANALYSIS AND DESIGN

3.1- Problem Statement:

To The works done till now only concentrated on crop prediction using different soil properties and Data Mining Techniques. Fertilizer Recommendation is not taken into consideration. So, it is necessary to develop crop prediction and fertilizer recommendation system which predicts crop based on weather features like temperature, humidity, etc. and recommend fertilizer based on chemicals such as nitrogen, sulphur, etc.

3.2- Existing System:

An agro-based country depends on agriculture for its economic growth. When a population of the country increases dependency on agriculture also increases and subsequent economic growth of the country is affected. In this situation, the crop yield rate plays a significant role in the economic growth of the country. So, there is a need to increase crop yield rate. Some biological approaches (e.g. seed quality of the crop, crop hybridization, strong pesticides) and some chemical approaches (e.g. use of fertilizer, urea, potash) are carried out to solve this issue. In addition to these approaches, a crop sequencing technique is required to improve the net yield rate of the crop over the season. One of existing system we identified is Crop Selection Method (CSM) to achieve a net yield rate of crops over the season.

3.3- Proposed system:

The Proposed system will predict the most suitable crop and fertilizer for particular land using decision tree regression, random forest, neural network models and based on weather parameters such as Temperature, Humidity, soil PH, Rainfall and required content and quantity of fertilizers like calcium,

magnesium, potassium, sulphur, nitrogen, lime, carbon, phosphorous, moisture.

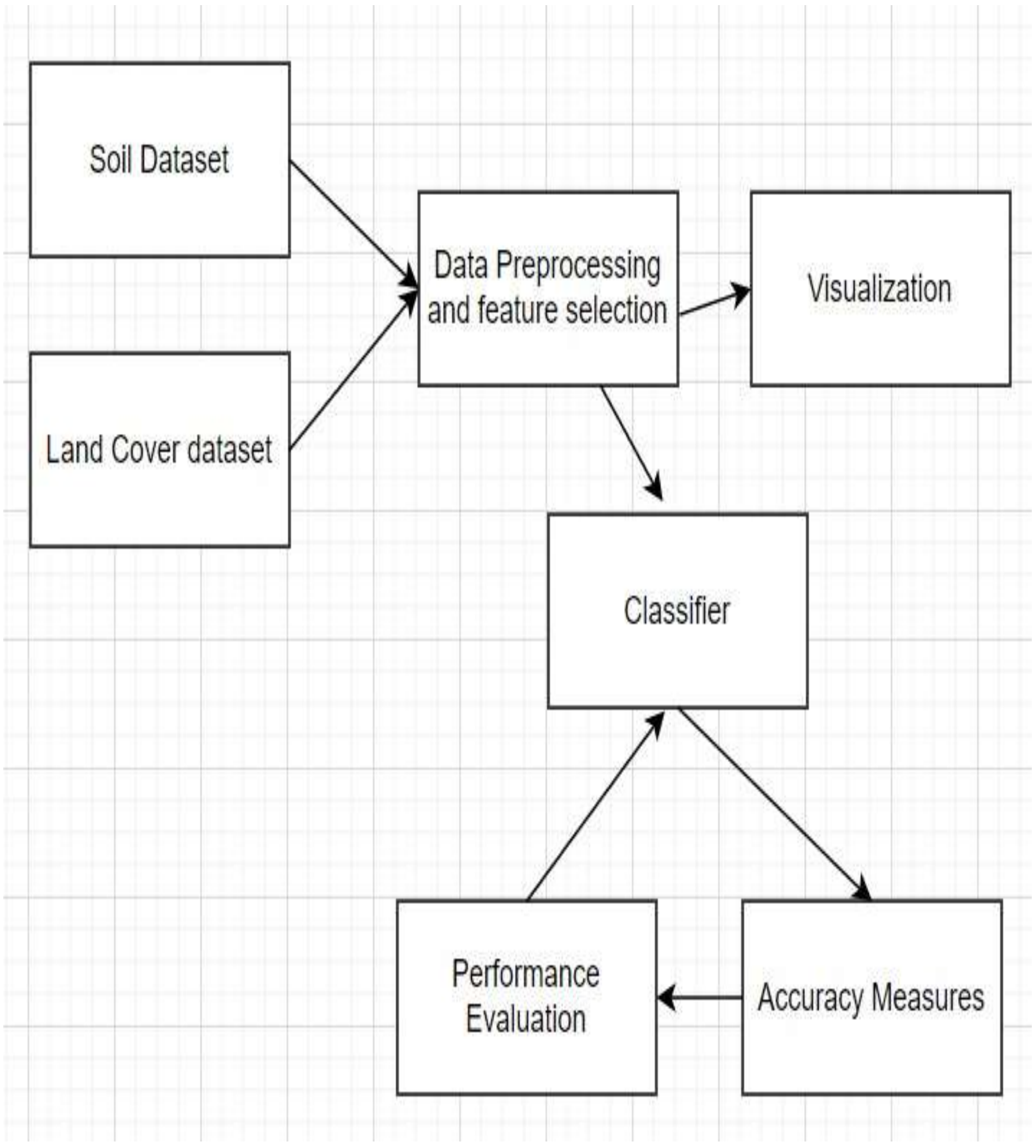


Figure 3.3 Architecture Diagram

In the above architecture diagram the first phase is getting the datasets for both crop and fertilizer. Next, the collected data is pre-processed by removing null values, reshaping the data and applying one hot encoding. Visualization is done to get an overall overview of the datasets.

The pre-processed training data is sent into the classifier models. The actual performance is tested by passing testing data to the classifiers. Based on the accuracy measures like accuracy, recall, etc. the performance evaluation is done and the appropriate classifier model is selected based on the appropriate performance results.

3.4 - System Requirements:

3.4.1- Hardware Requirements:

- Python 3.6 or higher
- Operating System: Windows/Ubuntu
- Front End: Chrome/Firefox/Opera
- Jupyter Notebook
- Visual Studio
- Flask

3.4.2- Software Requirements:

- RAM: Recommended Minimum 4GB
- Hard Disk: 500GB

- Processor: Recommended Minimum Intel HD Graphics 620
- Screen Size: Preferable above 13.5 inches

3.4.3- Packages:

• Pandas - Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

• Pickle - Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

• Numpy - NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

• Tensorflow - TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.

• Sklearn - The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

• Keras - Keras is a Python-based framework that makes it easy to debug and explore. Highly modular neural networks library written in Python.

• Matplotlib - Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

• Imutils - Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

• cv2 - OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imread() method loads an image from the specified file.

• Time – Time module helps in calculating the time taken for the model to predict.

• os - The OS module in Python provides functions for interacting with

the operating system. This module provides a portable way of using operating system-dependent functionality. The *os* and *os. path* modules include many functions to interact with the file system.

3.5 – Introduction to UML:

The UML stands for Unified Modeling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

3.5.1 - GOALS OF UML:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual Modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the Modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.6 - UML Diagrams:

3.6.1 <u>USE CASE DIAGRAM:</u>

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

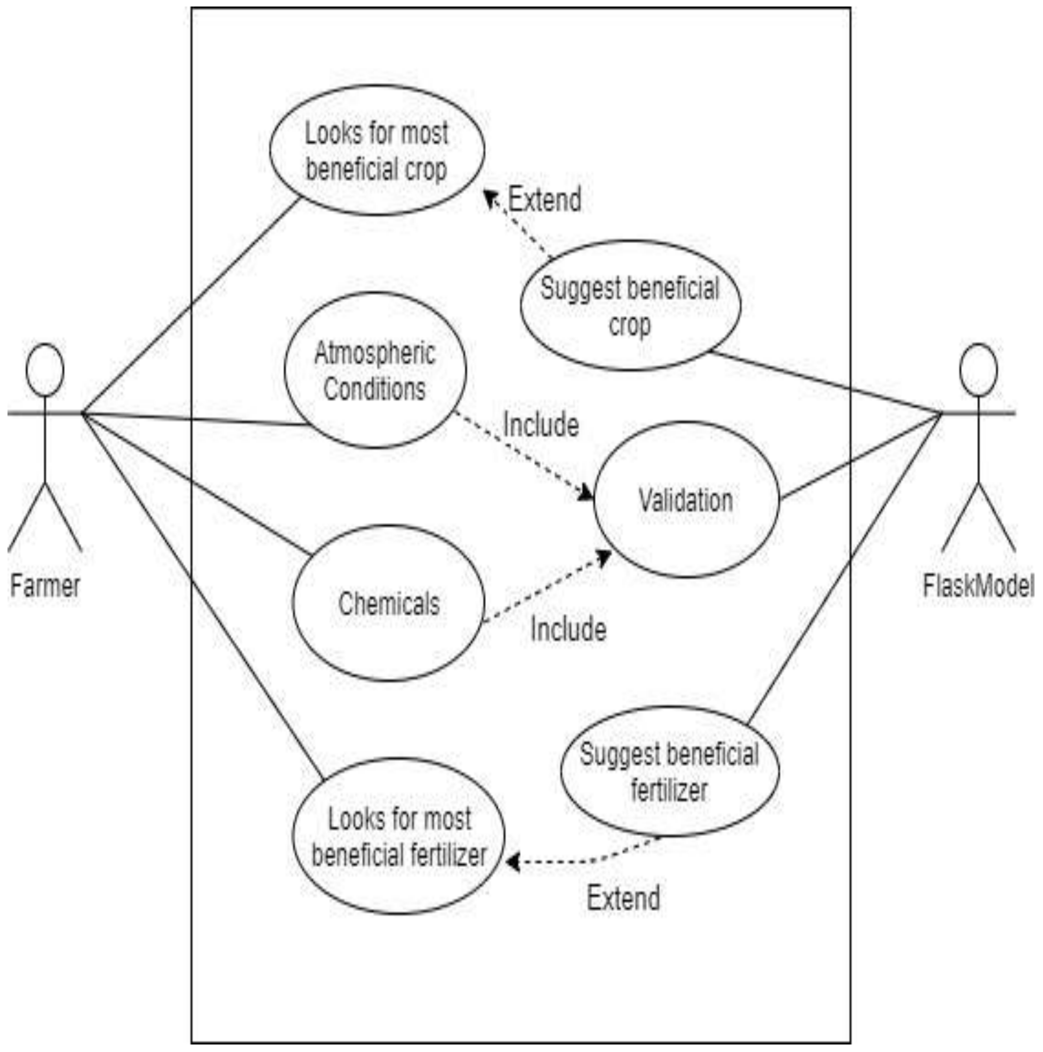


Figure 3.6.1 Use Case Diagram

The farmer looks for the most appropriate beneficial crop by providing the inputs such as temperature, rainfall, pH, humidity. The flask model validates the inputs and based on the input it provides the beneficial crop. Similarly, if the farmer is looking for most appropriate beneficial fertilizers by providing Ca, Mg, K, S, N, Lime, C, P, Moisture values. There values are passed to the flask model and it validates the appropriate units, measurements and gives the most suitable fertilizer. In the diagram, the both inputs are included in the validation.

3.6.2 PROCESS FLOW DIAGRAM:

In the process flow diagram, the index.html flask ui can take two inputs, crop prediction input and fertilizer prediction input. After submitting the inputs respective app.routes will be called they are /results_crop and /results_fertilizer . In app.py the inputs are sent into the models. Crop prediction input is sent into decision tree regression model to predict, similarly fertilizer input is sent into random forest model. After prediction the output is rendered through app.py separately as two files and rendered as index.html and index2.html.

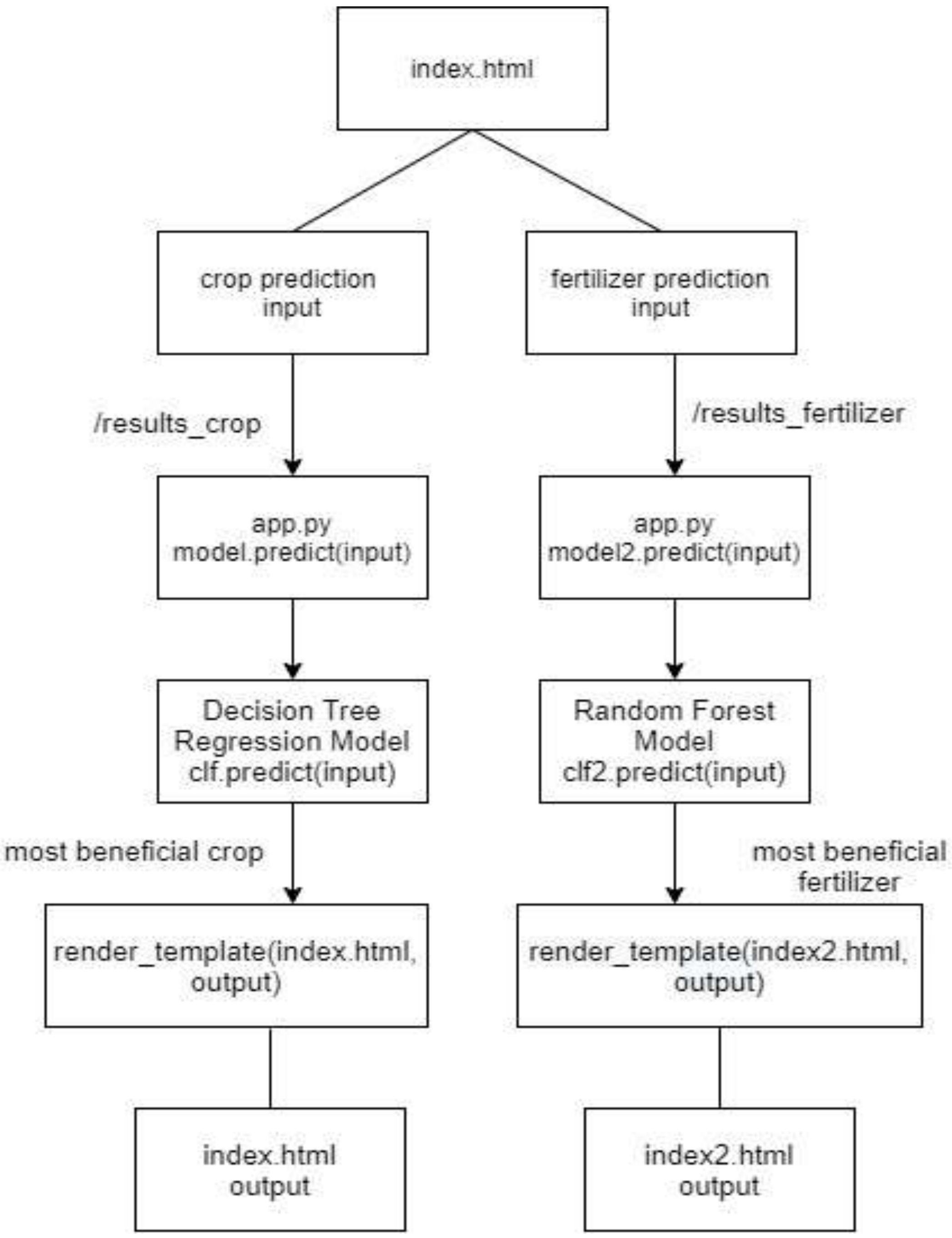

index.html
crop prediction input
fertilizer prediction input
/results_crop
/results_fertilizer
app.py model.predict(input)
app.py model2.predict(input)
Decision Tree Regression Model clf.predict(input)
Random Forest Model clf2.predict(input)
most beneficial crop
most beneficial fertilizer
render_template(index.html, output)
render_template(index2.html, output)
index.html output
index2.html output

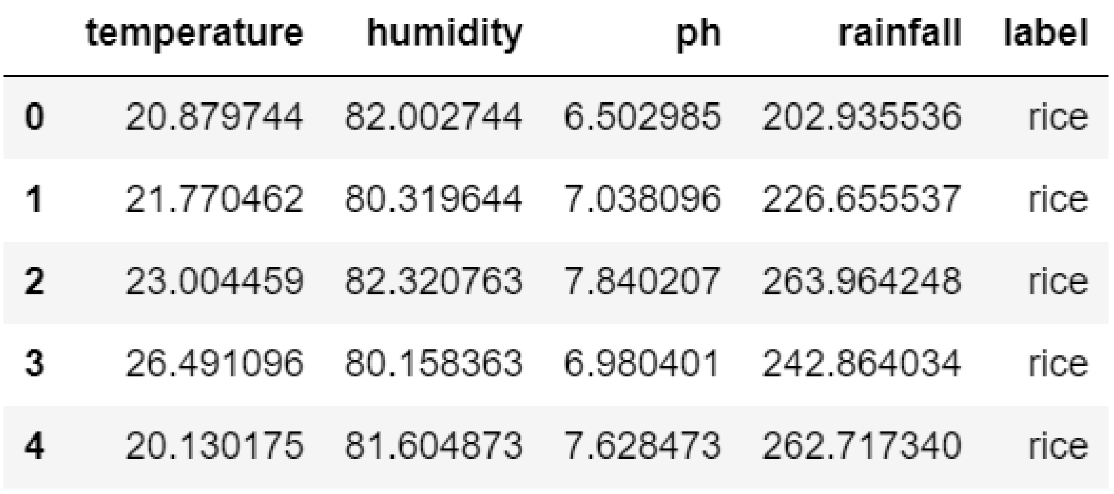
Figure 3.6.2 Process Flow Diagram

CHAPTER IV

METHODOLOGY

4.1 – Dataset:

In the There are two datasets. One for crop yield and another for fertilizers. Below figure shows the sample data of crop yield dataset.



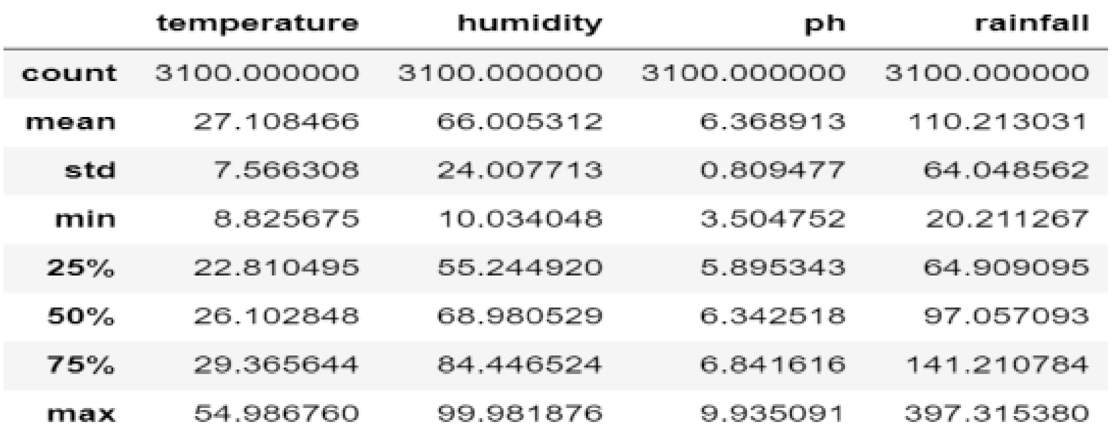
	temperature	humidity	ph	rainfall	label
0	20.879744	82.002744	6.502985	202.935536	rice
1	21.770462	80.319644	7.038096	226.655537	rice
2	23.004459	82.320763	7.840207	263.964248	rice
3	26.491096	80.158363	6.980401	242.864034	rice
4	20.130175	81.604873	7.628473	262.717340	rice

Figure 4.1.1 Sample Data of Crop Yield Dataset

Code: data=pd.read_csv('cpdata.csv')

data.head()

The above code reads the dataset and displays the first few rows.



	temperature	humidity	ph	rainfall
count	3100.000000	3100.000000	3100.000000	3100.000000
mean	27.108466	66.005312	6.368913	110.213031
std	7.566308	24.007713	0.809477	64.048562
min	8.825675	10.034048	3.504752	20.211267
25%	22.810495	55.244920	5.895343	64.909095
50%	26.102848	68.980529	6.342518	97.057093
75%	29.365644	84.446524	6.841616	141.210784
max	54.986760	99.981876	9.935091	397.315380

Figure 4.1.2 Crop Yield Dataset Description

Code: data.describe()

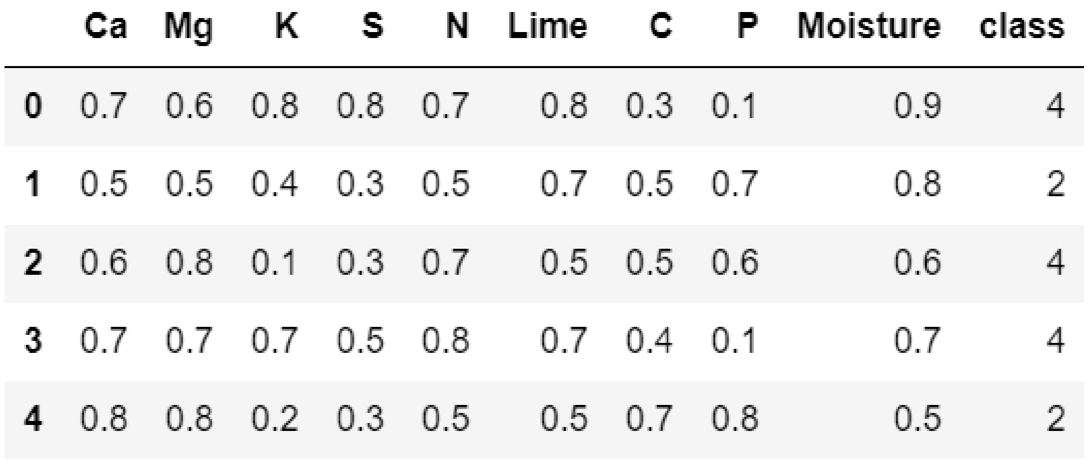
Describe is the function which gives all the statistical information about the dataset.

Below figure shows the sample data of fertilizer yield dataset.

Code: fert_data = pd.read_csv('fertestimate.csv')

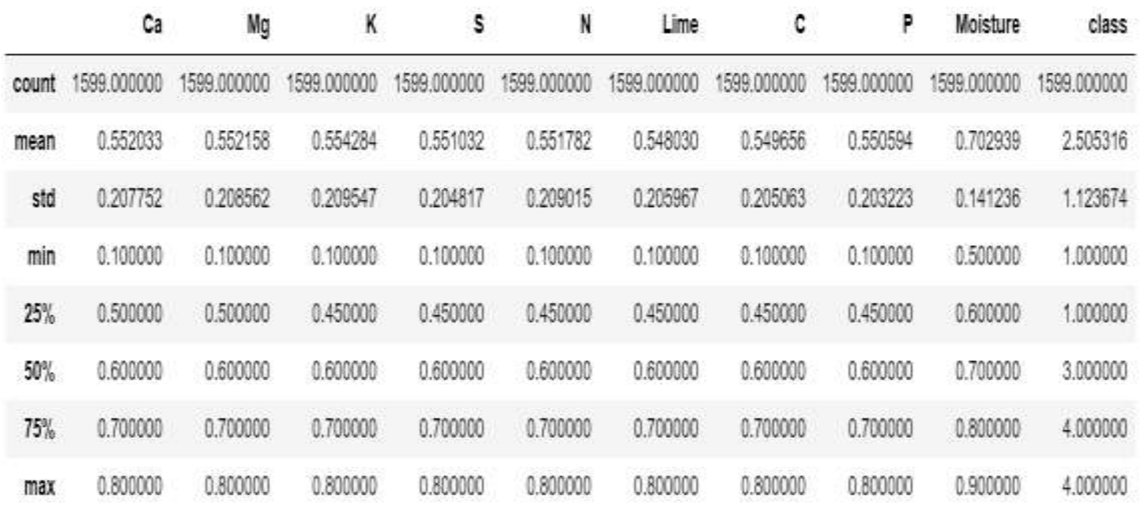
fert_data.head()

The above code reads the dataset and displays the first few rows.



	Ca	Mg	K	S	N	Lime	C	P	Moisture	class
0	0.7	0.6	0.8	0.8	0.7	0.8	0.3	0.1	0.9	4
1	0.5	0.5	0.4	0.3	0.5	0.7	0.5	0.7	0.8	2
2	0.6	0.8	0.1	0.3	0.7	0.5	0.5	0.6	0.6	4
3	0.7	0.7	0.7	0.5	0.8	0.7	0.4	0.1	0.7	4
4	0.8	0.8	0.2	0.3	0.5	0.5	0.7	0.8	0.5	2

Figure 4.1.3 Sample Data of Fertilizer Yield Dataset



	Ca	Mg	K	S	N	Lime	C	P	Moisture	class
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	0.552033	0.552158	0.554284	0.551032	0.551782	0.548030	0.549656	0.550594	0.702939	2.505316
std	0.207752	0.208562	0.209547	0.204817	0.209015	0.205967	0.205063	0.203223	0.141236	1.123674
min	0.100000	0.100000	0.100000	0.100000	0.100000	0.100000	0.100000	0.100000	0.500000	1.000000
25%	0.500000	0.500000	0.450000	0.450000	0.450000	0.450000	0.450000	0.450000	0.600000	1.000000
50%	0.600000	0.600000	0.600000	0.600000	0.600000	0.600000	0.600000	0.600000	0.700000	3.000000
75%	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000	0.800000	4.000000
max	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.800000	0.900000	4.000000

Figure 4.1.4 Fertilizer Yield Dataset Description

Code: fert_data.describe()

Describe is the function which gives all the statistical information about the dataset.

4.2 – Visualization:

Histogram Plots of the Crop Yield Dataset:

1) counts, bins = np.histogram(data.iloc[:,0])

   plt.hist(bins[:-1], bins, weights=counts)

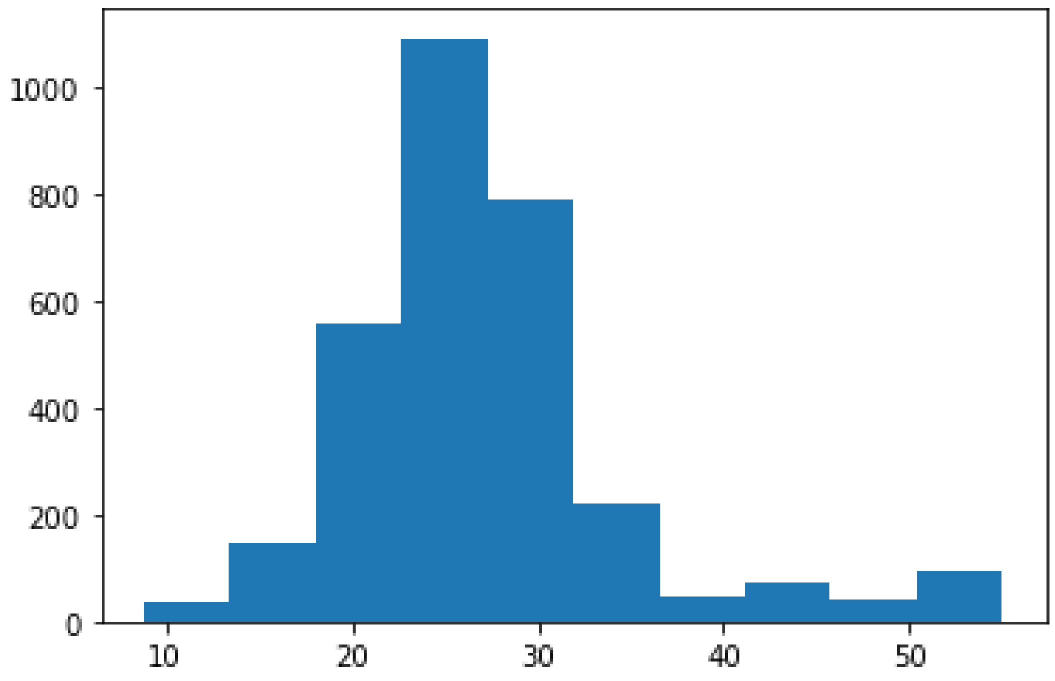


Figure 4.2.1 Histogram Plot of Temperature

The 1st histogram plot shows the temperature feature values where it can be inferred that many values lies between 20 and 30.

2) counts, bins = np.histogram(data.iloc[:,1])

plt.hist(bins[:-1], bins, weights=counts)

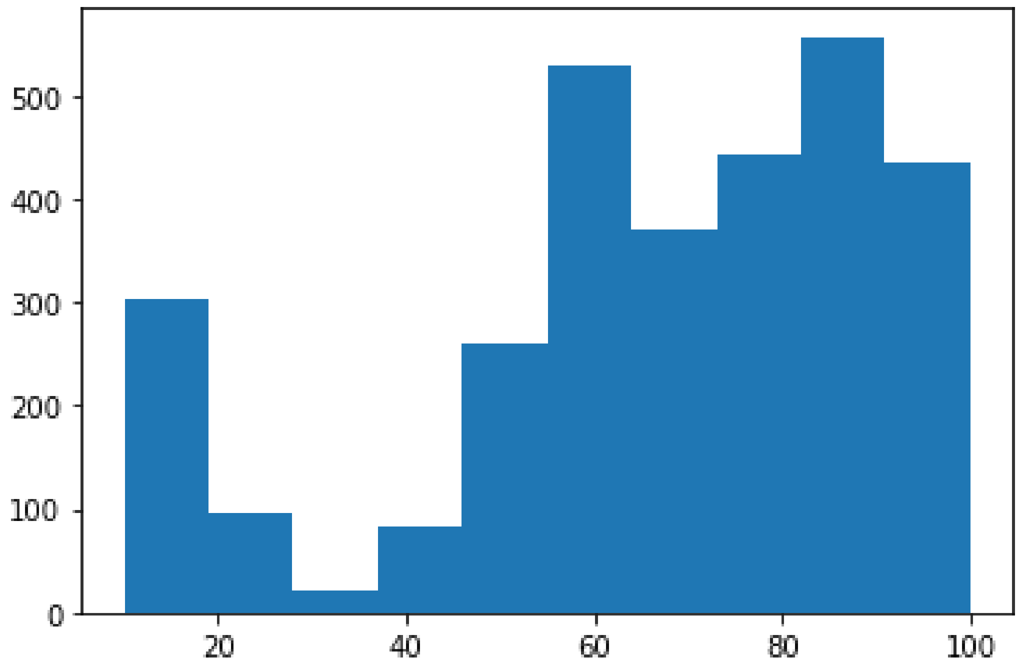


Figure 4.2.2 Histogram Plot of Humidity

The 2nd histogram plot shows the humidity feature values where it can be inferred that many values lies between 40 and 100.

```
3) counts, bins = np.histogram(data.iloc[:,2])
plt.hist(bins[:-1], bins, weights=counts)
```

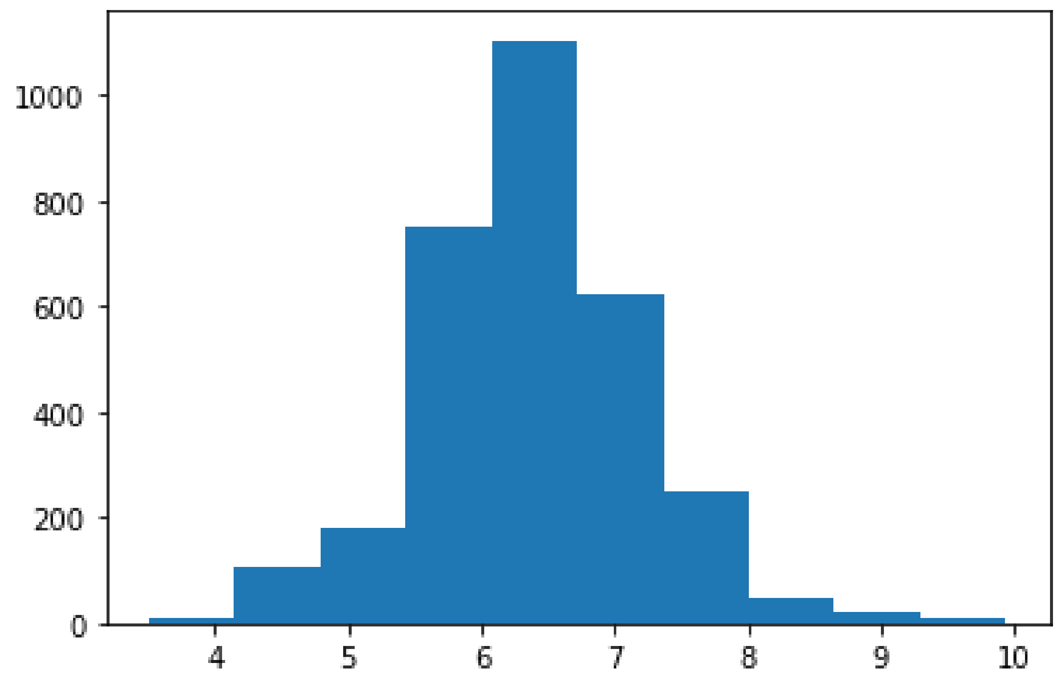


Figure 4.2.3 Histogram Plot of pH

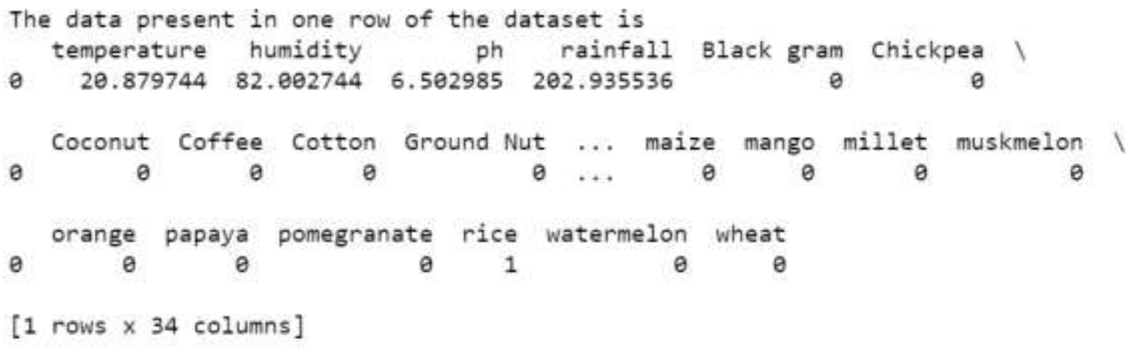
The 3rd histogram plot shows the pH feature values where it can be inferred that many values lie between 5 and 8.

4.3 – Pre-processing:

Handling Null Values in both datasets:

There are no null values in crop yield dataset and fertilizer yield dataset.

Creating variable for target column in crop yield dataset:



```
The data present in one row of the dataset is
   temperature   humidity        ph    rainfall  Black gram  Chickpea  \
0    20.879744  82.002744  6.502985  202.935536           0         0

   Coconut  Coffee  Cotton  Ground Nut  ...  maize  mango  millet  muskmelon  \
0        0       0       0           0  ...      0      0       0          0

   orange  papaya  pomegranate  rice  watermelon  wheat
0       0       0            0     1           0      0

[1 rows x 34 columns]
```

Code:

label= pd.get_dummies(data.label).iloc[: , 1:]

data= pd.concat([data,label],axis=1)

data.drop('label', axis=1,inplace=True)

print('The data present in one row of the dataset is')

print(data.head(1))

train=data.iloc[:, 0:4].values

test=data.iloc[: ,4:].values

In the above code the features and values are getting reshaped. All the target variables are made into columns. Now the new columns are 34.

Label Encoding in fertilizer yield dataset:

LabelEncoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier. The categorical values have been converted into numeric values.

Approach:

Create an instance of LabelEncoder() and store it in labelencoder variable/object.

Apply fit and transform which does the trick to assign numerical value to categorical value and the same is stored in new column called “State_N”.

The below code is used to label encode the values which are present in the crop yield dataset. LabelEncoder() and fit_transform() functions are used.

Code:

```
from numpy import array
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
values = array(y)

# integer encode

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
```

One Hot Encoding in fertilizer yield dataset:

One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns.

Each integer value is represented as a binary vector.

The below code represents the one hot encoding for the fertilizer dataset. Functions like OneHotEncoder() and fit_transform() are used to transform the data values between 0 and 1.

Code:

```
# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
y = onehot_encoder.fit_transform(integer_encoded)

x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,shuffle='false')
```

Standard Scaler in crop yield dataset:

StandardScaler removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. StandardScaler can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature.

Code:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

Standard Scalar is used for training and testing set. Methods used are StandardScaler(), fit_transform(), transform().

4.4 – Data Splitting:

Data Split:

Data splitting is commonly used in machine learning to split data into a train, test, or validation set. Each algorithm divided the data into two subset, training/validation. The training set was used to fit the model and validation for the evaluation.

Data splitting is the act of partitioning available data into two portions, usually for cross-validatory purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance.

The dataset should be divided into two parts. One will be the training set and other will be the testing set.

Crop yield dataset split:

Code:

```
X_train,X_test,y_train,y_test=train_test_split(train,test,test_size=0.3)
```

The above code splits the dataset into 70 to 30 ratios. 70 percent of the data is allocated to the training set and remaining 30 percent is testing data.

Fertilizer yield dataset split:

Code:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(fert_data.iloc[:,:], y, test_size =
0.250)
```

The above code splits the dataset into 75 to 25 ratios. 75 percent of the data is allocated to the training set and remaining 25 percent is testing data.

4.5 Model Selection and Training:

Most Suitable Crop Yield:

Three models are implemented for crop yield. They are: Decision Tree, SVM, Neural Network.

Decision Tree Classifier:

The decision tree classifier creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute.

There are several steps in decision tree:

1) Splitting:

The process of partitioning the data set into subsets. Splits are formed on a particular variable.

2) Pruning:

The shortening of branches of the tree. Pruning is the process of reducing the size of the tree by turning some branch nodes into leaf nodes, and removing the leaf nodes under the original branch. Pruning is useful because classification trees may fit the training data well, but may do a poor job of classifying new values. A simpler tree often avoids over-fitting.

3) Tree Selection:

The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest cross-validated error.

Training the decision tree model:

We import the DecisionTreeRegressor class from sklearn.tree and assign it to the variable 'regressor'. Then we fit the X_train and the y_train to the model by using theregressor.fit function. We use the reshape(-1,1) to reshape our variables to a single column vector.

Implementation:

#Importing Decision Tree classifier

```
from sklearn.tree import DecisionTreeRegressor
clf=DecisionTreeRegressor()

#Fitting the classifier into training set
clf.fit(X_train,y_train)
pred=clf.predict(X_test)

from sklearn.metrics import accuracy_score
# Finding the accuracy of the model
a=accuracy_score(y_test,pred)
print("The accuracy of this model is: ", a*100)

import pickle
pickle.dump(clf,open('decision_tree_regression_model.pkl','wb'))

np.argmax(pred[0])
pred[0]
```

In the above code we predict the results of the test set with the model trained on the training set values using the regressor.predict function and assign it to ‘y_pred’.

SVM:

SVM or Support Vector Machine is a linear model for classification problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

Kernel is linear. The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labelled as one class and all the points that fall on the other side will be labelled as the second.

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Here, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

The training set will be fitted to the SVM classifier. To create the SVM classifier, we will import SVC class from Sklearn.svm library. Below is the code for it:

Implementation:

```
# importing necessary libraries
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

#Reading the csv file
data=pd.read_csv('cpdata.csv')

# X -> features, y -> label
```

```
X = data.iloc[:,0:4]
y = data.iloc[:,4]

# dividing X, y into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)

# training a linear SVM classifier
from sklearn.svm import SVC
svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)

# model accuracy for X_test
accuracy = svm_model_linear.score(X_test, y_test)

# creating a confusion matrix
cm = confusion_matrix(y_test, svm_predictions)

print(accuracy)
```

In the above code, we have used kernel='linear', as here we are creating SVM for linearly separable data. However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train).

After getting the y_pred vector, we can compare the result of y_pred and y_test to check the difference between the actual value and predicted value.

Neural Network:

A simple neural network includes an input layer, an output (or target) layer and, in between, a hidden layer. The layers are connected via nodes, and these connections form a “network”.

Neural network is made up of many perceptron layers; that's why it has the name 'multi-layer perceptron. ' These layers are also called hidden layers of dense layers. They are the primary unit that works together to form a perceptron layer. These neurons receive information in the set of inputs.

A neural network has many layers. Each layer performs a specific function, and the complex the network is, the more the layers are. That’s why a neural network is also called a multi-layer perceptron.

The purest form of a neural network has three layers:

- The input layer
- The hidden layer
- The output layer

The input layer picks up the input signals and transfers them to the next layer. It gathers the data from the outside world.

The hidden layer performs all the back-end tasks of calculation. A network can even have zero hidden layers. However, a neural network has at least one hidden layer. The output layer transmits the final result of the hidden layer’s calculation.

Steps of how neural network works:

1. Information is fed into the input layer which transfers it to the hidden layer
2. The interconnections between the two layers assign weights to each input randomly
3. A bias added to every input after weights are multiplied with them individually
4. The weighted sum is transferred to the activation function
5. The activation function determines which nodes it should fire for feature extraction
6. The model applies an application function to the output layer to deliver the output
7. Weights are adjusted, and the output is back-propagated to minimize error

The model uses a cost function to reduce the error rate. We have to change the weights with different training models.

- The model compares the output with the original result
- It repeats the process to improve accuracy

Implementation:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
```

Encoding for neural network:

```
# encode class values as integers
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)
```

Model:

```
# define baseline model
def baseline_model():
  # create model
  model = Sequential()
  model.add(Dense(64, input_dim=4, activation='relu'))
  model.add(Dense(128,activation='relu'))
  model.add(Dense(256,activation='relu'))
  model.add(Dense(31, activation='softmax'))
  # Compile model
  model.compile(loss='categorical_crossentropy',
optimizer='adam'
metrics=['accuracy'])
return model

estimator = KerasClassifier(build_fn=baseline_model, epochs=20, batch_size=5, verbose=1)
kfold = KFold(n_splits=2, shuffle=True)
results = cross_val_score(estimator, X, dummy_y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

The model used is sequential where several layers were added to it. Four dense layers were added with different values 64,128,256,31. The activation functions are relu and softmax.

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The softmax activation will output one value for each node in the output layer.

Adam optimizer is used. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

Finally the epochs used are 20 and the batch size is 5.

Most Beneficial Fertilizer Recommendation:

Two models are used for most beneficial fertilizer recommendation. They are: Neural Network and Random Forest Classifier.

Neural Network:

There are 3 hidden layers in this model and an output layer to predict fertilizer. The nodes in hidden layer 1 is 64, hidden layer 2 is 128 and in the hidden layer 3 is 64. The number of classes for the output layer is 4. Adam optimizer is used same as the crop prediction neural network model. Same 20 epochs are taken for this model. Different batch sizes are taken to check the performance.

Parameters:

```
# neural network parameters

n_nodes_hl1 = 64
n_nodes_hl2 = 128
n_nodes_hl3 = 64
n_classes = 4
batch_size = 100
data_index = 0
```

Generating Batch:

```
# generate batch

def generate_batch(batch_size):
    global data_index
    batch = np.ndarray(shape=(batch_size, 9), dtype=np.float32)    #the same
shapes
```

```
as train data
    labels = np.ndarray(shape=(batch_size, 4), dtype=np.float32)
    for i in range(batch_size):
        batch[i] = np.array(x_train)[data_index]
        labels[i] = y_train[data_index]
        data_index = (data_index + 1) % len(x_train)
    return batch, labels
```

Model:

```
# define the model

def neural_network_model(data):
    # input data* weights + bias
    hidden_1_layer={'weights': tf.Variable(tf.random_normal([9, n_nodes_hl1])),
                    'biases': tf.Variable(tf.random_normal([n_nodes_hl1]))}
    hidden_2_layer={'weights':tf.Variable(tf.random_normal([n_nodes_hl1,
n_nodes_hl2])),
                    'biases': tf.Variable(tf.random_normal([n_nodes_hl2]))}

    hidden_3_layer={'weights':tf.Variable(tf.random_normal([n_nodes_hl2,
n_nodes_hl3])),
                    'biases': tf.Variable(tf.random_normal([n_nodes_hl3]))}
    output_layer={'weights':tf.Variable(tf.random_normal([n_nodes_hl3,
n_classes])),
                        'biases': tf.Variable(tf.random_normal([n_classes]))}

    l1=tf.add(tf.matmul(data,hidden_1_layer['weights'])                                    ,
hidden_1_layer['biases'])
    l1 = tf.nn.relu(l1) # rectified linear --> activation function

    l2= tf.add(tf.matmul(l1, hidden_2_layer['weights']) , hidden_2_layer['biases'])
    l2 = tf.nn.relu(l2)

    l3= tf.add(tf.matmul(l2, hidden_3_layer['weights']) , hidden_3_layer['biases'])
    l3 = tf.nn.relu(l3)

    output = tf.matmul(l3, output_layer['weights']) + output_layer['biases']

    return output
```

Training neural network:

```
# train neural network

def train_neural_network(xin):
    prediction = neural_network_model(xin)
    cost                                                                              =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=yin,logits=pred
iction))

optimizer = tf.train.AdamOptimizer(0.001).minimize(cost) #learning rate = 
0.001

    hm_epochs = 20
    eploss = []
    with tf.Session() as sess:
        sess.run(tf.initialize_all_variables())

        for epoch in range(hm_epochs):
            epoch_loss = 0
            for _ in range(int(len(x_train)/batch_size)) :
                epoch_x,epoch_y = generate_batch(batch_size)
                    _,c = sess.run([optimizer,cost], feed_dict={xin:epoch_x,
   yin:epoch_y})
                epoch_loss += c
            print('Epoch',epoch,'completed out of', hm_epochs, 'loss: ', epoch_loss)
            eploss.append(epoch_loss)

        correct = tf.equal(tf.argmax(prediction,1), tf.argmax(yin,1))

        accuracy = tf.reduce_mean(tf.cast(correct,'float'))
        a = float(accuracy.eval({xin:x_test, yin:y_test}))
        print('accuracy: ', a*100,'%')
        # predict an output
        predict = tf.argmax(prediction,1)
        example = np.array([0.05,0.01,0.01,0.01,0.02,0.01,0.03,0.01,0.01])
        example = example.reshape(-1,len(example))
        predict = predict.eval({xin:example})
        print("prediction : Fertilizer", label_encoder.inverse_transform(predict))

        #plot loss vs no. of epochs


        plt.figure()
        plt.subplot(2,2,1)
        plt.plot(eploss)
        plt.ylabel('Loss')
        plt.xlabel('Number of epochs')
        plt.subplot(2,2,1)
        plt.plot()
        plt.title('Loss vs Number of epochs')
        plt.ylabel('Loss')
        plt.xlabel('Number of epochs')
        plt.show()

train_neural_network(xin)
```

Random Forest Classifier:

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Random forest adds additional randomness to the model, while growing the trees.

For this classification problem Random Forest gives you probability of belonging to class. SVM gives you distance to the boundary, you still need to convert it to probability somehow if you need probability. SVM gives you "support vectors", that is points in each class closest to the boundary between classes.

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Important hyperparameters:

The hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster. Let's look at the hyperparameters of sklearns built-in random forest function.

1. Increasing the predictive power

Firstly, there is the n_estimators hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is max_features, which is the maximum number of features random forest considers to split a node. Sklearn provides several options, all described in the documentation.

The last important hyperparameter is min_sample_leaf. This determines the minimum number of leafs required to split an internal node.

2. Increasing the model's speed

The n_jobs hyperparameter tells the engine how many processors it is allowed to use. If it has a value of one, it can only use one processor. A value of “-1” means that there is no limit.

The random_state hyperparameter makes the model’s output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.

Lastly, there is the oob_score (also called oob sampling), which is a random forest cross-validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out-of-bag samples. It's very similar to the leave-one-out-cross-validation method, but almost no additional computational burden goes along with it.

Implementation:

```
from sklearn.svm import SVC

# creating a RF classifier
clf2 = SVC()

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf2.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = clf2.predict(X_test)

# metrics are used to find accuracy or error
from sklearn import metrics
```

```
# using metrics module for accuracy calculation

print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_train, clf.predict(X_train)))

import pickle
pickle.dump(clf2,open('random_forest_model.pkl','wb'))
```

Random classifier model is imported from the sklearn. The X_train and y_train is passed to the model. The X_test is used to predict the model by using predict() function. Importing metrics from sklearn is used to get the accuracy of the model with test data.

4.6 Input and Output:

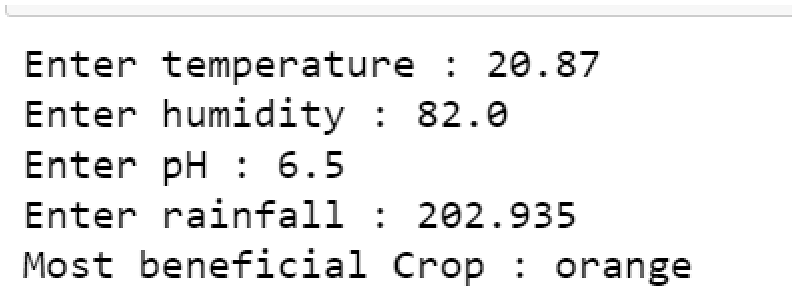
Input for most suitable crop prediction:

```
#temperature   humidity       ph    rainfall
temp = float(input('Enter temperature : '))
hum = float(input('Enter humidity : '))
ph = float(input('Enter pH : '))
rainfall = float(input('Enter rainfall : '))
```

crop = data_copy.columns[np.argmax(clf.predict([[temp,hum,ph,rainfall]])[0]) + 4]

print('Most beneficial Crop :', crop)

Output for most suitable crop prediction:



```
Enter temperature : 20.87
Enter humidity : 82.0
Enter pH : 6.5
Enter rainfall : 202.935
Most beneficial Crop : orange
```

Figure 4.6.1 Output for Most Suitable Crop

Here, the values for temperature, humidity, pH, rainfall are given as 20.87, 82.0, 6.5, 202.935. These values are sent to the trained decision tree model. The model predicts the orange as the most beneficial crop.

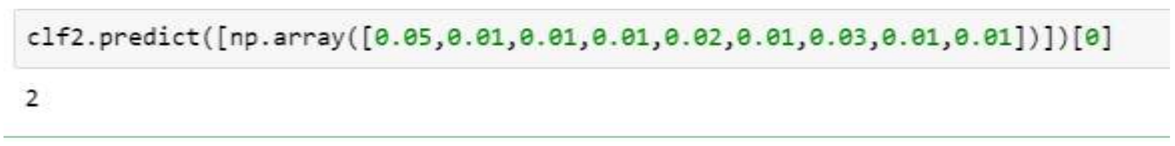
Input for most beneficial fertilizer prediction:

import pickle

pickle.dump(clf2,open('random_forest_model.pkl','wb'))

clf2.predict([np.array([0.05,0.01,0.01,0.01,0.02,0.01,0.03,0.01,0.01])])[0]

Output for most beneficial fertilizer prediction:



```
clf2.predict([np.array([0.05,0.01,0.01,0.01,0.02,0.01,0.03,0.01,0.01])])[0]

2
```

Figure 4.6.2 Output for Most Suitable Fertilizer

Values of calcium, magnesium, potassium, sulphur, nitrogen, lime, carbon, phosphorous, moisture are sent into the trained random forest model and the output is given as “2”. Here “2” represents the encoded value of one of the fertilizers.

CHAPTER V

RESULTS

5.1- Performance Evaluation:

Most Suitable Crop Yield:

1) SVM:

Accuracy of SVM model is 86.32 percent.

2) Decision Tree Classifier:

Accuracy of this model is 90.645 percent which is higher than SVM model.

3) Neural Network:

20 epochs are taken in this neural network model with batch size as 5 and with verbose of 1.

From epoch 1 accuracy started with 20percent which is very low. Later in upcoming epochs accuracy has increased. In 20th epoch the accuracy is of 71.94 which is very less than SVM and Decision Tree Classifier models.

Result:

```
Epoch 1/20
1550/1550 [==============================] - 0s 296us/step - loss: 2.9093 - accuracy: 0.2819
```

```
Epoch 2/20

1550/1550 [==============================] - 0s 227us/step - loss:
1.6678 - accuracy: 0.4297

Epoch 3/20

1550/1550 [==============================] - 0s 232us/step - loss:
1.4637 - accuracy: 0.4871

Epoch 4/20

1550/1550 [==============================] - 0s 195us/step - loss:
1.3157 - accuracy: 0.5071

Epoch 5/20

1550/1550 [==============================] - 0s 206us/step - loss:
1.2308 - accuracy: 0.5361

Epoch 6/20

1550/1550 [==============================] - ETA: 0s - loss: 1.1928
- accuracy: 0.55 - 0s 224us/step - loss: 1.1796 - accuracy: 0.5581

Epoch 7/20

1550/1550 [==============================] - 0s 195us/step - loss:
1.1240 - accuracy: 0.5703

Epoch 8/20

1550/1550 [==============================] - 0s 204us/step - loss:
1.0385 - accuracy: 0.6077

Epoch 9/20

1550/1550 [==============================] - 0s 223us/step - loss:
0.9925 - accuracy: 0.6323

Epoch 10/20

1550/1550 [==============================] - 0s 201us/step - loss:
0.9385 - accuracy: 0.6426
```

```
Epoch 11/20
1550/1550 [==============================] - 0s 213us/step - loss:
0.9058 - accuracy: 0.6516
Epoch 12/20
1550/1550 [==============================] - 0s 243us/step - loss:
0.8955 - accuracy: 0.67030s - loss: 0.7861 - accu
Epoch 13/20
1550/1550 [==============================] - 0s 192us/step - loss:
0.8005 - accuracy: 0.6794
Epoch 14/20
1550/1550 [==============================] - 0s 200us/step - loss:
0.7698 - accuracy: 0.6929
Epoch 15/20
1550/1550 [==============================] - 0s 233us/step - loss:
0.7895 - accuracy: 0.6916
Epoch 16/20
1550/1550 [==============================] - 0s 200us/step - loss:
0.7586 - accuracy: 0.6929
Epoch 17/20
1550/1550 [==============================] - 0s 199us/step - loss:
0.7302 - accuracy: 0.7148
Epoch 18/20
1550/1550 [==============================] - 0s 243us/step - loss:
0.7065 - accuracy: 0.7297
Epoch 19/20
1550/1550 [==============================] - 0s 193us/step - loss:
0.6986 - accuracy: 0.7194
```

```
Epoch 20/20
1550/1550 [==============================] - 0s 200us/step - loss:
0.7032 - accuracy: 0.7219
1550/1550 [==============================] - 0s 117us/step
Epoch 1/20
1550/1550 [==============================] - 0s 320us/step - loss:
2.9460 - accuracy: 0.2613
Epoch 2/20
1550/1550 [==============================] - 0s 193us/step - loss:
1.6648 - accuracy: 0.4381
Epoch 3/20
1550/1550 [==============================] - 0s 194us/step - loss:
1.4285 - accuracy: 0.4845
Epoch 4/20
1550/1550 [==============================] - 0s 253us/step - loss:
1.3296 - accuracy: 0.5058
Epoch 5/20
1550/1550 [==============================] - 0s 196us/step - loss:
1.2368 - accuracy: 0.5477
Epoch 6/20
1550/1550 [==============================] - 0s 196us/step - loss:
1.1755 - accuracy: 0.5606
Epoch 7/20
1550/1550 [==============================] - 0s 257us/step - loss:
1.1317 - accuracy: 0.5735
Epoch 8/20
1550/1550 [==============================] - 0s 197us/step - loss:
```

```
1.0615 - accuracy: 0.6039
Epoch 9/20
1550/1550 [==============================] - 0s 199us/step - loss:
1.0085 - accuracy: 0.6232
Epoch 10/20
1550/1550 [==============================] - 0s 262us/step - loss:
0.9494 - accuracy: 0.65940s - loss: 0.9499 - accura
Epoch 11/20
1550/1550 [==============================] - 0s 195us/step - loss:
0.9021 - accuracy: 0.6626
Epoch 12/20
1550/1550 [==============================] - 0s 204us/step - loss:
0.8530 - accuracy: 0.6755
Epoch 13/20
1550/1550 [==============================] - 0s 254us/step - loss:
0.8337 - accuracy: 0.6910
Epoch 14/20
1550/1550 [==============================] - 0s 196us/step - loss:
0.8016 - accuracy: 0.6974
Epoch 15/20
1550/1550 [==============================] - 0s 200us/step - loss:
0.7541 - accuracy: 0.7071
Epoch 16/20
1550/1550 [==============================] - 0s 246us/step - loss:
0.7675 - accuracy: 0.7077
Epoch 17/20
1550/1550 [==============================] - 0s 201us/step - loss:
```

```
0.7526 - accuracy: 0.71480s - loss: 0.7531 - accuracy: 0.71

Epoch 18/20

1550/1550 [==============================] - 0s 215us/step - loss:
0.7470 - accuracy: 0.71230s - loss: 0.7875 - accura

Epoch 19/20

1550/1550 [==============================] - 0s 246us/step - loss:
0.7790 - accuracy: 0.7052

Epoch 20/20

1550/1550 [==============================] - 0s 218us/step - loss:
0.6932 - accuracy: 0.7194

1550/1550 [==============================] - 0s 99us/step

Baseline: 73.10% (1.23%)
```

Most Beneficial Fertilizer Recommendation:

1) Neural Network:

After training for 20 epochs the accuracy was 89.58 percent.

Result:

```
Epoch 0 completed out of 20 loss:  1659.4805755615234

Epoch 1 completed out of 20 loss:  441.3720483779907

Epoch 2 completed out of 20 loss:  193.34368991851807

Epoch 3 completed out of 20 loss:  126.6965103149414
Epoch 4 completed out of 20 loss:  112.05246210098267
Epoch 5 completed out of 20 loss:  103.72900462150574
Epoch 6 completed out of 20 loss:  91.89268016815186
Epoch 7 completed out of 20 loss:  85.90711784362793
Epoch 8 completed out of 20 loss:  79.64293336868286
Epoch 9 completed out of 20 loss:  72.33458936214447
Epoch 10 completed out of 20 loss:  67.17194044589996
Epoch 11 completed out of 20 loss:  64.94717049598694
Epoch 12 completed out of 20 loss:  58.9133198261261
Epoch 13 completed out of 20 loss:  58.017539620399475
Epoch 14 completed out of 20 loss:  54.12311366200447
Epoch 15 completed out of 20 loss:  52.099132657051086
Epoch 16 completed out of 20 loss:  48.95827567577362
Epoch 17 completed out of 20 loss:  43.81526052951813
Epoch 18 completed out of 20 loss:  44.6748993396759
Epoch 19 completed out of 20 loss:  41.683428049087524
accuracy:  89.58333134651184 %
```

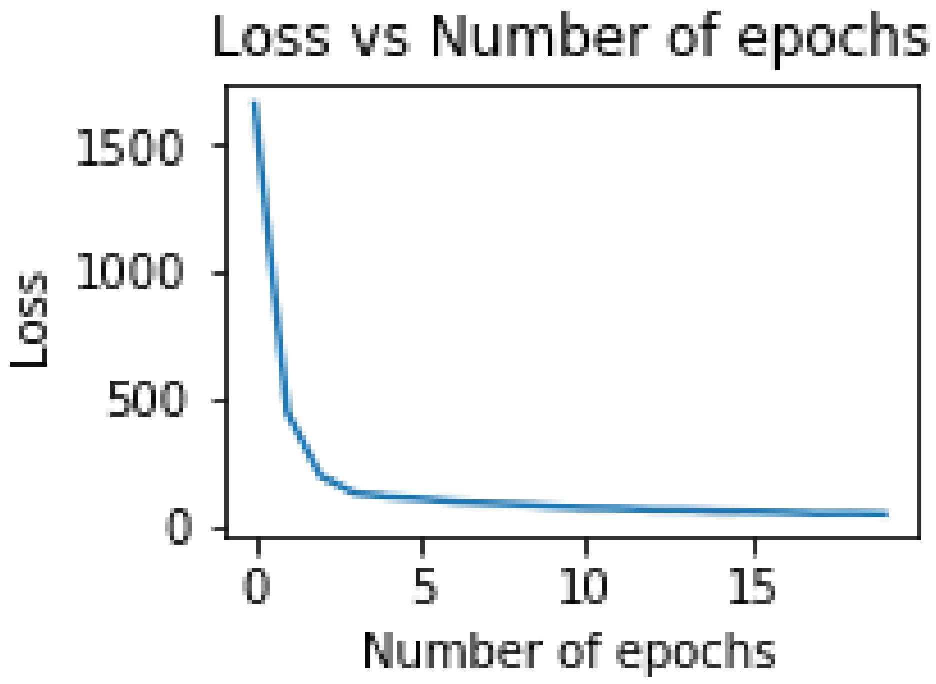
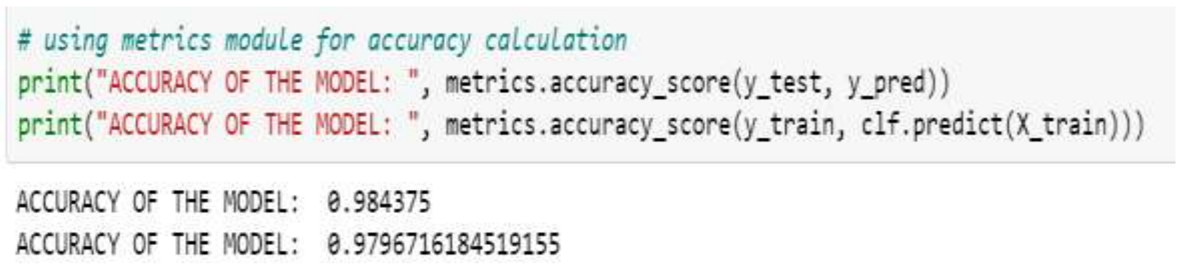


Figure 5.1 Graph between Loss and Number of epochs

2) Random Forest:

For training data, the accuracy was 97.96 percent and for testing data it is 98.43 percent which is higher and better than neural network model.



```
# using metrics module for accuracy calculation
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_train, clf.predict(X_train)))

ACCURACY OF THE MODEL:  0.984375
ACCURACY OF THE MODEL:  0.9796716184519155
```

Random forest is the suitable model for most beneficial fertilizer recommendation.

CHAPTER VI

FLASK WEB APP

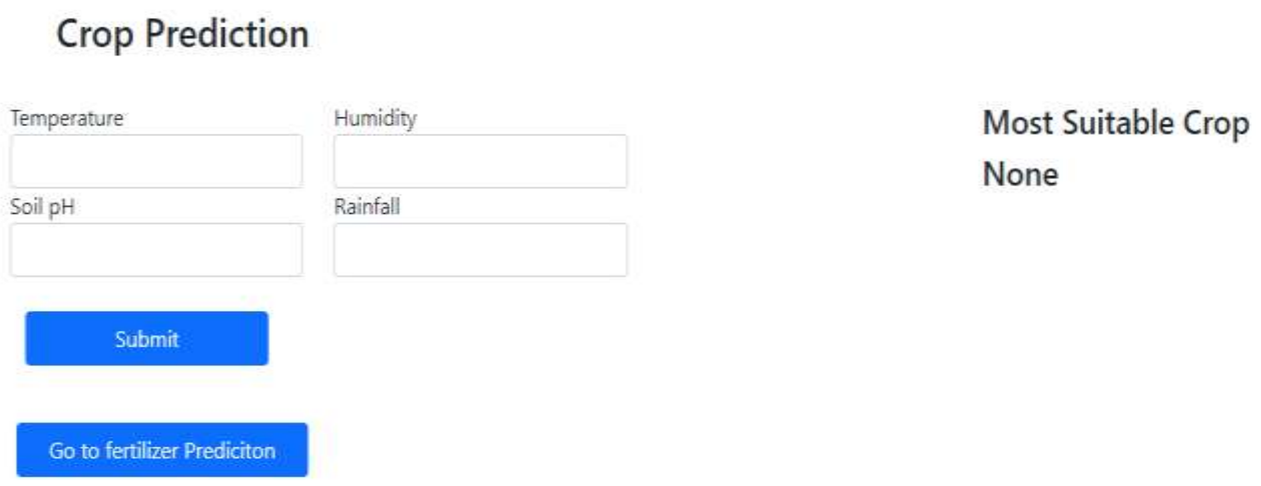
6.1- User Interface and Output:

Flask is used to create and user interface for this crop and fertilizer prediction.

Flask: Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application.

User should enter the temperature, humidity, pH, rainfall input fields for getting crop prediction.

And for fertilizer prediction calcium, magnesium, potassium, sulphur, nitrogen, lime, carbon, phosphorous, moisture input fields should be entered.



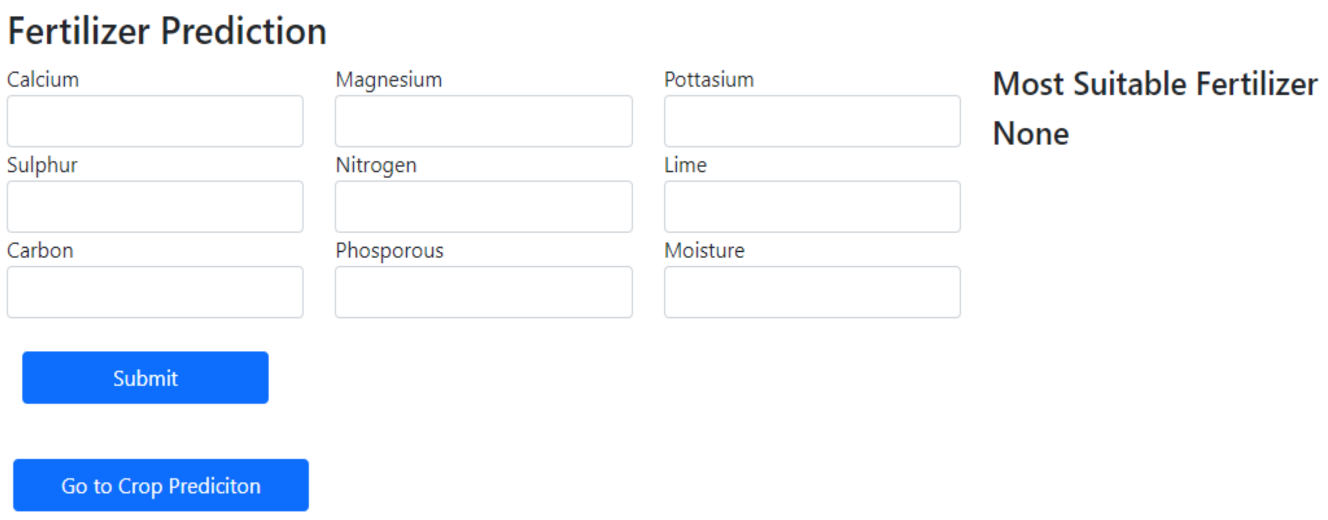
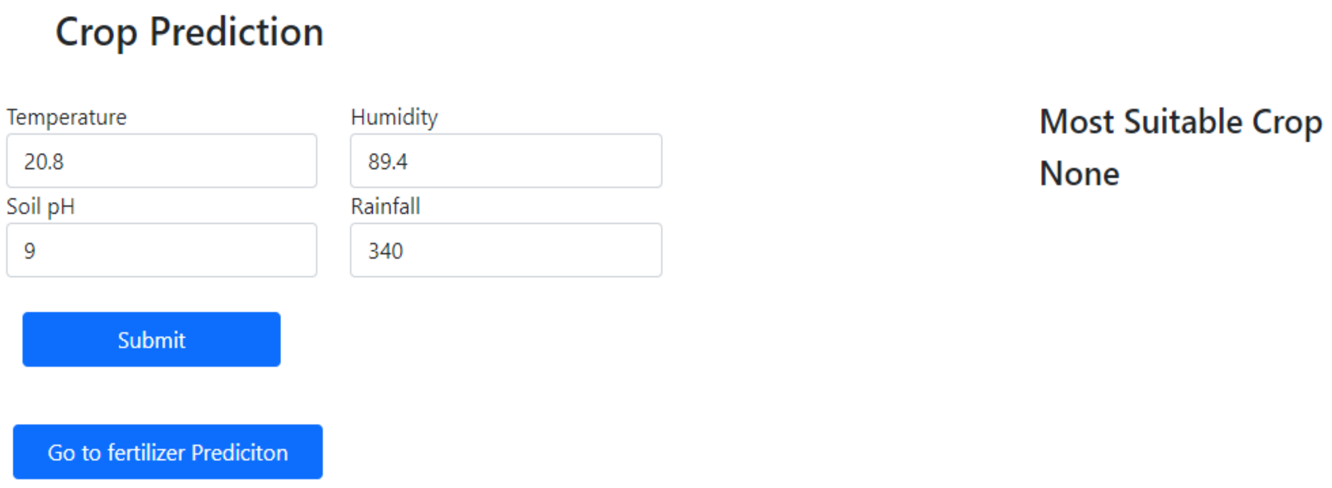


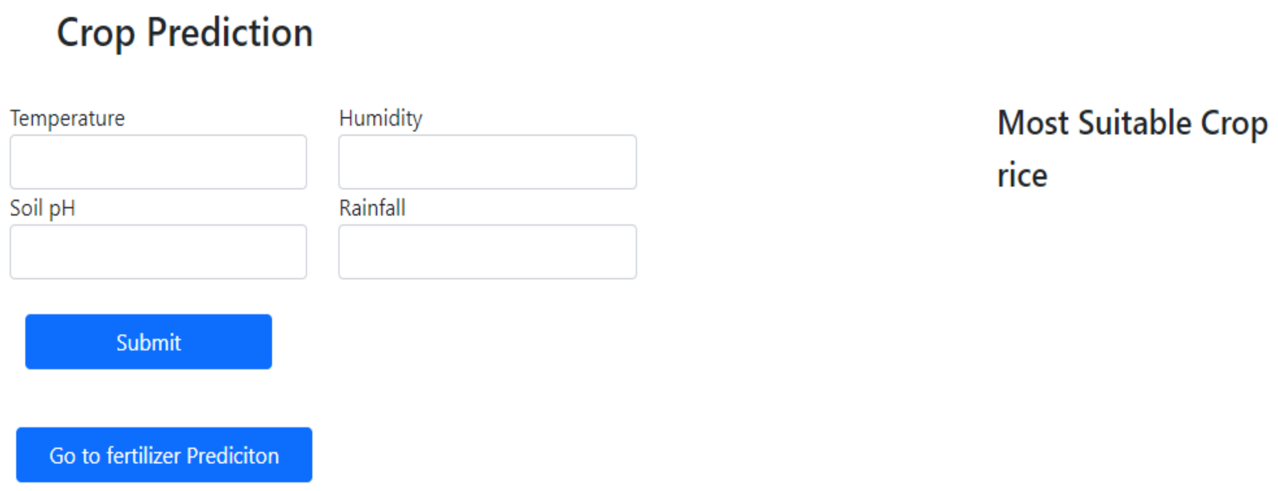
Figure 6.1.1 index.html & index2.html


Crop Prediction

Temperature	Humidity
20.8	89.4
Soil pH	Rainfall
9	340

Most Suitable Crop
None

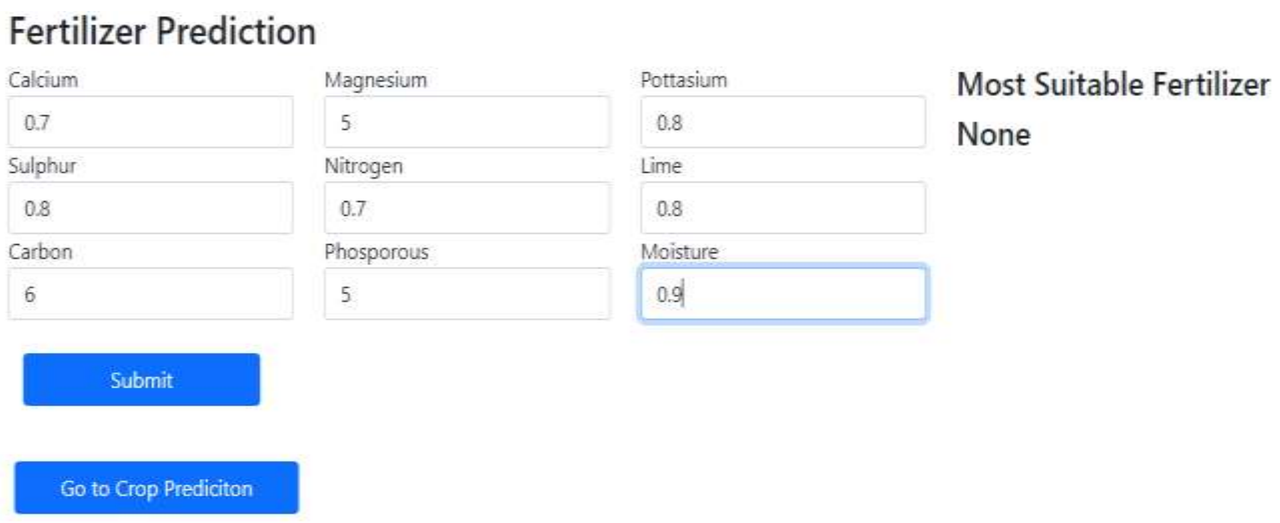
Figure 6.1.2 Filling inputs in index.html


Crop Prediction

Temperature	Humidity
Soil pH	Rainfall

Most Suitable Crop
rice

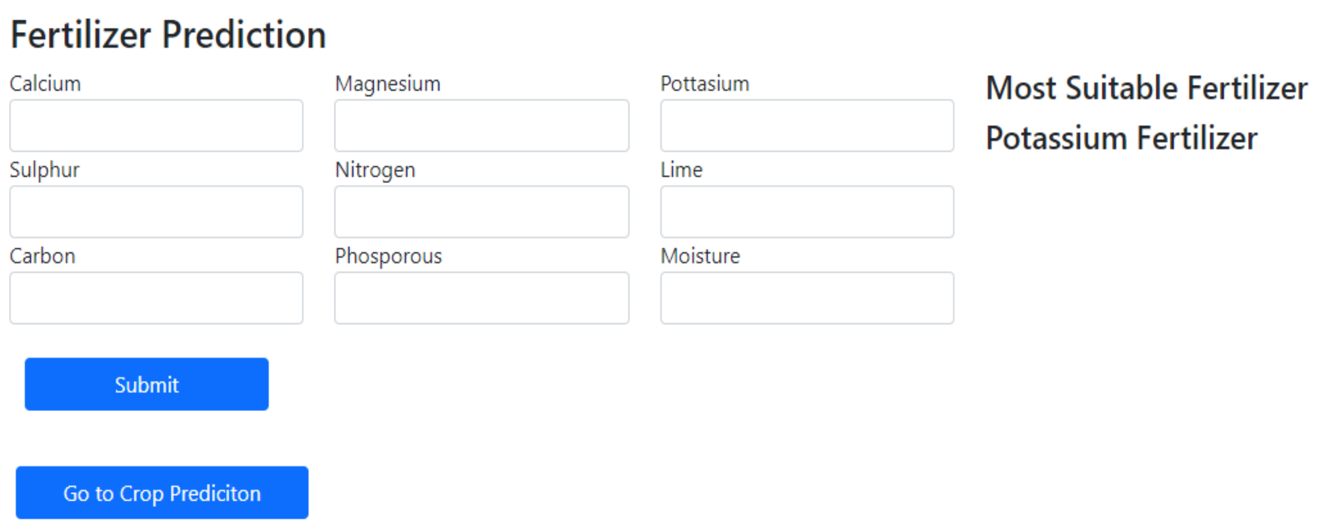
Figure 6.1.3 Most Suitable Crop Output


Fertilizer Prediction

Calcium	Magnesium	Pottasium
0.7	5	0.8
Sulphur	Nitrogen	Lime
0.8	0.7	0.8
Carbon	Phosporous	Moisture
6	5	0.9

Most Suitable Fertilizer
None

Figure 6.1.4 Filling inputs in index2.html



Fertilizer Prediction

Calcium	Magnesium	Pottasium
Sulphur	Nitrogen	Lime
Carbon	Phosporous	Moisture

Most Suitable Fertilizer

Potassium Fertilizer

Figure 6.1.5 Most Suitable Fertilizer Output

6.2- Code Snippets:

app.py

```
from flask import Flask, render_template, request, session

from sklearn.tree import DecisionTreeRegressor

import pickle

import numpy as np

import pandas as pd

app = Flask(__name__)


model = pickle.load(open("decision_tree_regression_model.pkl", "rb"))

model2 = pickle.load(open("random_forest_model.pkl", "rb"))

app.secret_key = "abfdhsuadhsaujc"


######################

data=pd.read_csv('cpdata.csv')

label= pd.get_dummies(data.label).iloc[: , 1:]

data= pd.concat([data,label],axis=1)

data.drop('label', axis=1,inplace=True)

data_copy = data

######################



@app.route("/")

def intial_route():
    session['crop'] = 'None'
    session['fert_type'] = 'None'
    return render_template('index.html',crop = session['crop'], fert_type = session['fert_type'])


@app.route("/results_crop", methods=["POST", "GET"])
def results_crop():
    result = request.form
    print(request.form)
    rainfall = float(result["rainfall"])
    humidity = float(result["humidity"])
    ph = float(result["ph"])
    temperature = float(result['temperature'])

    #check parameters order
    prediction = model.predict([[temperature,humidity,ph,rainfall]])
    session['crop'] = data_copy.columns[np.argmax(prediction[0]) + 4]
    return render_template('index.html',crop = session['crop'], fert_type = session['fert_type'])


@app.route("/results_fertilizer", methods=["POST", "GET"])
def results_fertilizer():
    result = request.form
    print(request.form)
    ca = float(result["ca"])
    mg = float(result["mg"])
    k = float(result["k"])
    s = float(result["s"])
    n = float(result["n"])
    lime = float(result["lime"])
    c = float(result["c"])
    p = float(result["p"])
    moisture = float(result["moisture"])

    #check parameters order
    prediction = model2.predict([[ca,mg,k,s,n,lime,c,p,moisture]])[0]
    if prediction == 1:
        session['fert_type'] = 'Organic and Inorganic Fertilizer'
    elif prediction == 2:
        session['fert_type'] = 'Nitrogen Fertilizer'
    elif prediction == 3:
        session['fert_type'] = 'Phosphate Fertilizer'
    elif prediction == 4:
        session['fert_type'] = 'Potassium Fertilizer'

    return render_template('index.html',crop = session['crop'], fert_type = session['fert_type'])



if __name__ == "__main__":
    app.run()
```

index.html

```
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
  crossorigin="anonymous"
/>

<body>
  <div class="container">
    <div class="row m-4">
      <h3>Crop Prediction</h3>
    </div>
    <div class="row">
      <div class="col-8">
        <form method="POST" action="http://localhost:5000/results_crop">
          <div class="row">
            <div class="form-group col-md-4">
              <label for="temperature">Temperature</label>
              <input
                type="text"
                class="form-control"
                id="temperature"
                name="temperature"
              />
            </div>
            <div class="form-group col-md-4">
              <label for="Humidity">Humidity</label>
              <input
                type="text"
                class="form-control"
                id="humidity"
                name="humidity"
              />
            </div>
          </div>
          <div class="row">
            <div class="form-group col-md-4">
              <label for="ph">Soil pH</label>
              <input type="text" class="form-control" id="ph" name="ph" />
            </div>
            <div class="form-group col-md-4">
              <label for="rainfall">Rainfall</label>
              <input
                type="text"
                class="form-control"
                id="rainfall"
                name="rainfall"
              />
            </div>
          </div>
          <div class="row">
            <button type="submit" class="col-md-3 m-4 btn btn-primary">
              Submit
            </button>
          </div>
        </form>
      </div>
      <div class="col-4">
        <div class="row">
          <h4>Most Suitable Crop</h4>
          <h4>{{crop}}</h4>
        </div>
      </div>
    </div>
  </div>

  <div>
    <form method="post" action="http://localhost:5000/index_fert">
      <button type="submit" style="margin-left: 130px;" class="col-md-2 btn btn-
primary">Go to fertilizer Prediciton</button>
    </form>
  </div>
 </body>
```

index2.html

```
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
  crossorigin="anonymous"
/>

<body>
  <div class="container">


    <class="row m-4">
      <h3>Fertilizer Prediction</h3>
    </class=>
    <div class="row">
      <div class="col-8">
        <form method="POST" action="http://localhost:5000/results_fertilizer">
          <div class="row">
            <div class="form-group col-md-4">
              <label for="ca">Calcium</label>
              <input type="text" class="form-control" id="ca" name="ca" />
            </div>
            <div class="form-group col-md-4">
              <label for="mg">Magnesium</label>
              <input type="text" class="form-control" id="mg" name="mg" />
            </div>
            <div class="form-group col-md-4">
              <label for="k">Pottasium</label>
              <input type="text" class="form-control" id="k" name="k" />
            </div>
          </div>
          <div class="row">
            <div class="form-group col-md-4">
              <label for="s">Sulphur</label>
              <input type="text" class="form-control" id="s" name="s" />
            </div>
            <div class="form-group col-md-4">
              <label for="n">Nitrogen</label>
              <input type="text" class="form-control" id="n" name="n" />
            </div>
            <div class="form-group col-md-4">
              <label for="lime">Lime</label>
              <input type="text" class="form-control" id="lime" name="lime" />
            </div>
          </div>
          <div class="row">
            <div class="form-group col-md-4">
              <label for="c">Carbon</label>
              <input type="text" class="form-control" id="c" name="c" />
            </div>
            <div class="form-group col-md-4">
              <label for="p">Phosporous</label>
              <input type="text" class="form-control" id="p" name="p" />
            </div>
            <div class="form-group col-md-4">
              <label for="moisture">Moisture</label>
              <input
                type="text"
                class="form-control"
                id="moisture"
                name="moisture"
              />
            </div>
          </div>
          <div class="row">
            <button type="submit" class="col-md-3 m-4 btn btn-primary">
              Submit
            </button>
          </div>
        </form>
      </div>
      <div class="col-4">
        <div class="row">
          <h4>Most Suitable Fertilizer</h4>
          <h4>{{fert_type}}</h4>
        </div>
      </div>
    </div>
  </div>
  <div >
    <form method="post" action="http://localhost:5000/">
       <button type="submit" style="margin-left: 130px;" class="col-md-2 btn btn-
primary">Go to Crop Prediciton</button>
    </form></div> </body>
```

CHAPTER VII
CONCLUSION AND FUTURE SCOPE

7.1 - Conclusion:

This application is a utility for hospital executives and helps them to review, monitor and analyze the hospital performance metrics on a daily basis such as daily appointments, number of patients admitted and appointments for a given day, patient details like patient name and mobile, email and appointment slot number with dates, diagnosis details like the least and the most common diagnosed diseases and patient demographic distribution like percent of patients coming from each county, without using paper-based generated reports and desktop reports. This system can improve the efforts of hospital executives by providing hospital operational data in a portable manner with ease of access and helps them in making effective decisions required for improving the quality of patient care services. Doctor can easily monitor the patients and their appointments for a given day.

7.2 – Application:

This flask app can be used by the farmers which helps them to decide which crops and fertilizers to be used based on the climate and chemicals data. This will increase the production of crops and also prevents soil pollution. The farmers will use fertilizers which are needed so that there won’t be any excess usage so, it prevents soil pollution. The government should encourage this type of apps so that there might not be any additional burden for them.

7.3 – Future Scope:

We have to collect all required data by giving GPS locations and information of a land and by taking access from Rain forecasting system of by the government, we can predict crops and fertilizers by just giving GPS location. Also, we can develop the model to avoid over and under crisis of the food. Complex neural network models like the CNN will be used to check for the higher accuracy and performance metrics.

REFERENCES

1. Gandhi N, Armstrong LJ, Petkar O. Proposed decision support system (DSS) for Indian rice crop yield prediction. In2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR) 2016 Jul 15 (pp. 13-18). IEEE.

2. Terliksiz AS, Altýlar DT. Use Of Deep Neural Networks For Crop Yield Prediction: A Case Study Of Soybean Yield in Lauderdale County, Alabama, USA. In2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics) 2019 Jul 16 (pp. 1-4). IEEE.

3. Guruprasad RB, Saurav K, Randhawa S. Machine Learning Methodologies for Paddy Yield Estimation in India: a Case Study. InIGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium 2019 Jul 28 (pp. 7254-7257). IEEE.

4. Wani H, Ashtankar N. An appropriate model predicting pest/diseases of crops using machine learning algorithms. In2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS) 2017 Jan 6 (pp. 1-4). IEEE.

5. Kulkarni S, Mandal SN, Sharma GS, Mundada MR. Predictive Analysis to Improve Crop Yield using a Neural Network Model. In2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2018 Sep 19 (pp. 74-79). IEEE.

6. Bhanumathi S, Vineeth M, Rohit N. Crop Yield Prediction and Efficient use of Fertilizers. In2019 International Conference on Communication and Signal Processing (ICCSP) 2019 Apr 4 (pp. 0769-0773). IEEE.

7. Rale N, Solanki R, Bein D, Andro-Vasko J, Bein W. Prediction of Crop Cultivation. In2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) 2019 Jan 7 (pp. 0227-0232). IEEE.

8. Mhudchuay T, Kasetkasem T, Attavanich W, Kumazawa I, Chanwimaluang T. Rice Cultivation Planning Using A Deep Learning Neural Network. In2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) 2019 Jul 10 (pp. 822-825). IEEE.

9. Sharma S, Rathee G, Saini H. Big data analytics for crop prediction mode using optimization technique. In2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) 2018 Dec 20 (pp. 760-764). IEEE.

10. Athani SS, Tejeshwar CH. Support vector machine-based classification scheme of maize crop. In2017 IEEE 7th International Advance Computing Conference (IACC) 2017 Jan 5 (pp. 84-88). IEEE.

11. Rushika G., Juilee K, Pooja M, Sachee N, and Priya R.L.(2018). Prediction of Crop Yield using Machine Learning, Issue 02 IRJET (pg 2337-2339).

12. Ruchita T, Shreya B, Prasanna D, and Anagha C (2017). Crop Yield Prediction using Big Data Analytics, Volume 6, Issue 11, IJCMS.

13. Monali P, Santosh K, Vishwakarma, and Ashok V (2015). Analysis of Soil Behaviour and Prediction of Crop Yield using Data Mining Approach, ICCICN 2015.

14. Mrs. N. Hemageetha, and Dr. G.M. Nasira (2016). Analysis of Soil Condition based on pH value using Classification Techniques, IOSR-JCE, Issue 6, (pp.50-54).

15. E. Manjula, and S. Djodiltachoumy (2017). Data Mining Technique to Analyze Soil Nutrients based on Hybrid Classification. IJARCS.

16. Rohit Kumar Rajak, Ankit Pawar, Mitalee Pendke, Pooja Shinde, Suresh Rathod, and Avinash Devare (2017). Crop Recommendation System to maximize Crop yield using Machine Learning. Issue 12 IRJET.

17. S. Veenadhari, Dr. Bharat Misra, Dr. CD singh (2014). Machine Learning Approach for Forecasting Crop Yield based on Climatic Parameters. ICCCI-2014

18. Sadia A, Abu Talha K, Mahrin Mahia, Wasit A, and Rashedur M.R.(2018). Analysis of Soil Properties and Climatic Data To Predict Crop Yields and Cluster Different Agricultural Regions of Bangladesh, IEEE ICIS 2018 (pp.80-85).

19. Arun K, Navin K, and Vishal V (2018). Efficient Crop Yield Prediction using Machine Learning, (pp.3151-3159), IRJET.

20. Subhadra M, Debahuti M, Gour, H. S.(2016). Applications of Machine Learning Techniques in Agricultural Crop Production: A Review Paper, Vol 9(38), DOI:10.17485/ijst/2016/v9i38/95032 IJST.

21. Mrs. Prajakta P.B., Yogesh S. P, and Dinesh D.P. (2017). Improved Crop Yield Prediction using Neural Network, IJARIIE, (pg.3094-3101).

22. Dhivya B H, Manjula R, Siva Bharathi S, and Madhumathi R (2017). A Survey on Crop Yield Prediction based on Agricultural Data, DOI:10.15680/IJIRSET.2017.0603053, IJIRSET.

23. Omkar B,Nilesh M,Shubham G,Chandan M, and D.S. Zingade (2017), Crop Prediction System using Machine Learning, Volume 4, Special Issue 5, IJAERD.

24. Sneha N, Dr. Jharna Majumdar (2017). Big Data Application in Agriculture to Maximize the Rice Yield Crop Production using Data Mining Technique DOI: 10.15680/IJIRCCE.2017. 0505045, IJIRCCE.

25. S.Bhanumathi, M.Vineeth and N.Rohit (2019). Crop Yield Prediction and Efficient use of Fertilizers, (pg.0769- 0773), ICCSP.

26. Niketa G, Leisa A.J., Owaiz Petkar, and Amiya K.T.(2016). Rice Crop Yield Prediction in India using Support Vector Machines 978-1-5090-2033-1/16/$31.00 ©2016 IEEE.