**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.

**Introduction**

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.

**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.

**Literature Survey**

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.

**System Analysis and Design**

**Problem Statement:**

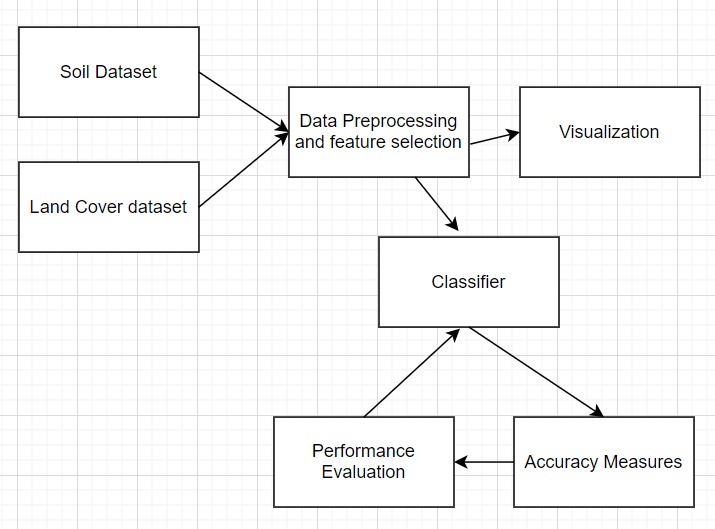
The works done till now only concentrated on crop prediction using different soil properties and Data Mining Techniques. Fertilizer Recommendation is not taken into consideration. So, it is necessary to develop crop prediction and fertilizer recommendation system which predicts crop based on weather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

**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.

**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.



Architecture Diagram

**System Requirements**

**Software Specifications:**

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

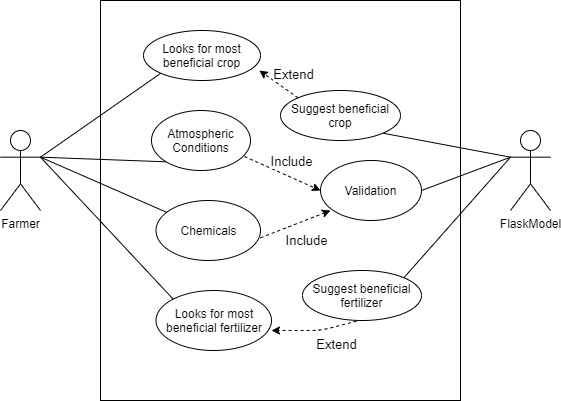
**Hardware Specifications:**

* RAM: Recommended Minimum 4GB
* Hard Disk: 500GB
* Processor: Recommended Minimum Intel HD Graphics 620
* Screen Size: Preferable above 13.5 inches

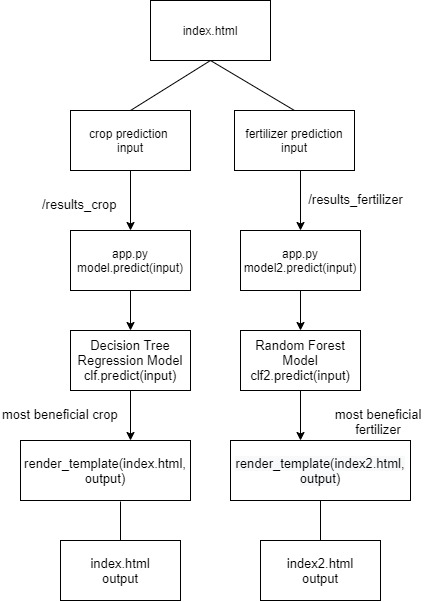
**Packages:**

* Pandas
* Pickle
* Numpy
* Tensorflow
* Sklearn
* Keras
* Matplotlib
* Imutils
* cv2
* Time
* os

**Use Case Diagram:**



**Process Flow Diagram:**

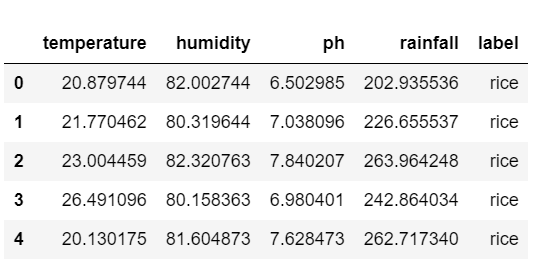


**Methodology**

1. **Dataset:**

There are two datasets. One for crop yield and another for fertilizers.

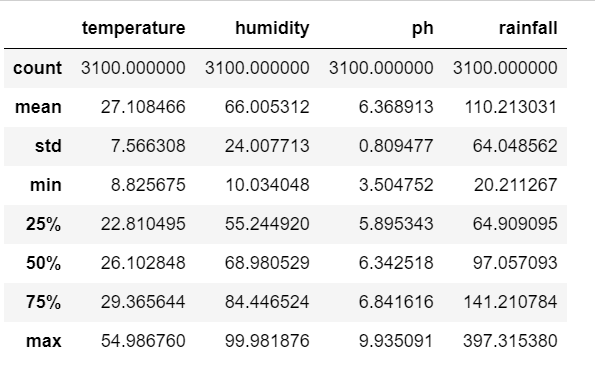
Below figure shows the sample data of crop yield dataset.



Sample Data of Crop Yield Dataset

Code: data=pd.read\_csv('cpdata.csv')

data.head()



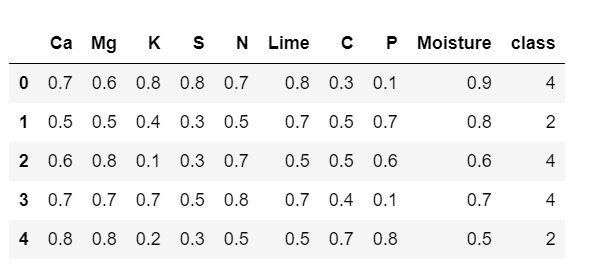
Crop Yield Dataset Description

Code: data.describe()

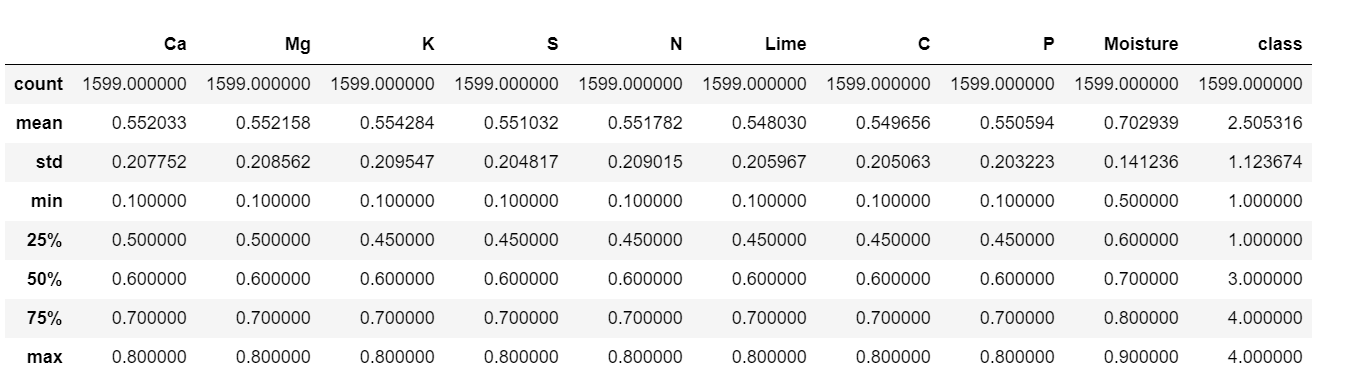
Below figure shows the sample data of fertilizer yield dataset.

Code: fert\_data = pd.read\_csv('fertestimate.csv')

fert\_data.head()



Sample Data of Fertilizer Yield Dataset



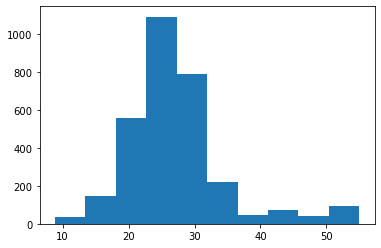
Fertilizer Yield Dataset Description

Code: fert\_data.describe()

Histogram Plots of the Crop Yield Dataset:

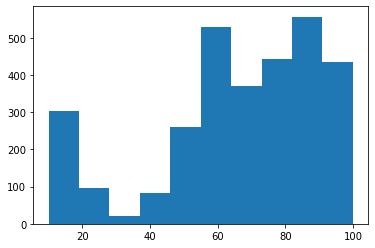
1. counts, bins = np.histogram(data.iloc[:,0])

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



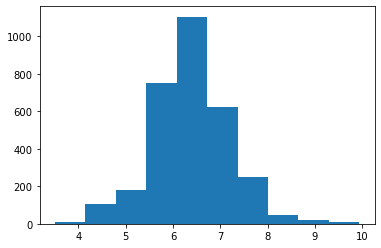
2) counts, bins = np.histogram(data.iloc[:,1])

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



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

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

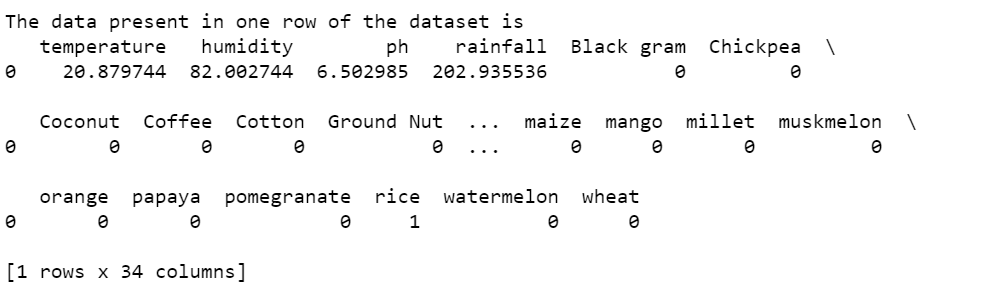


1. **Pre-processing:**

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:



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

Label Encoding in fertilizer yield dataset:

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:

# 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:

Code:

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

1. **Data Splitting:**

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)

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)

1. **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 decisiontree. 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.

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]

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.

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)

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.

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))

Most Beneficial Fertilizer Recommendation:

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

Neural Network:

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=prediction))

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.

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'))

1. **Input and Output:**

Input for most suitable crop prediction:

#temperature humidity ph rainfall

temp = float(input('Enter temperature : '))

hum = float(input('Enter humidity : '))

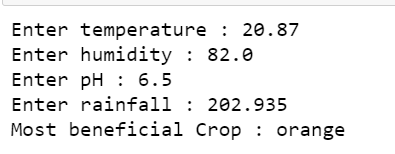
ph = float(input('Enter pH : '))

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:



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.

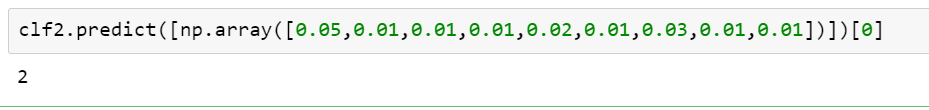
Input for most beneficial fertilizer prediction:

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:



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.

**Results**

**Performance Evaluation:**

Most Suitable Crop Yield:

1. SVM:

Accuracy of SVM model is 86.32 percent.

1. Decision Tree Classifier:

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

1. 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

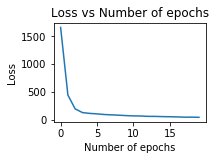
Epoch 16 completed out of 20 loss: 48.95827567577362

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

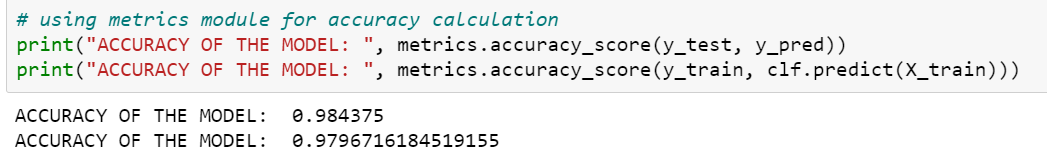
accuracy: 89.58333134651184 %



Graph between Loss and Number of epochs

1. 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.



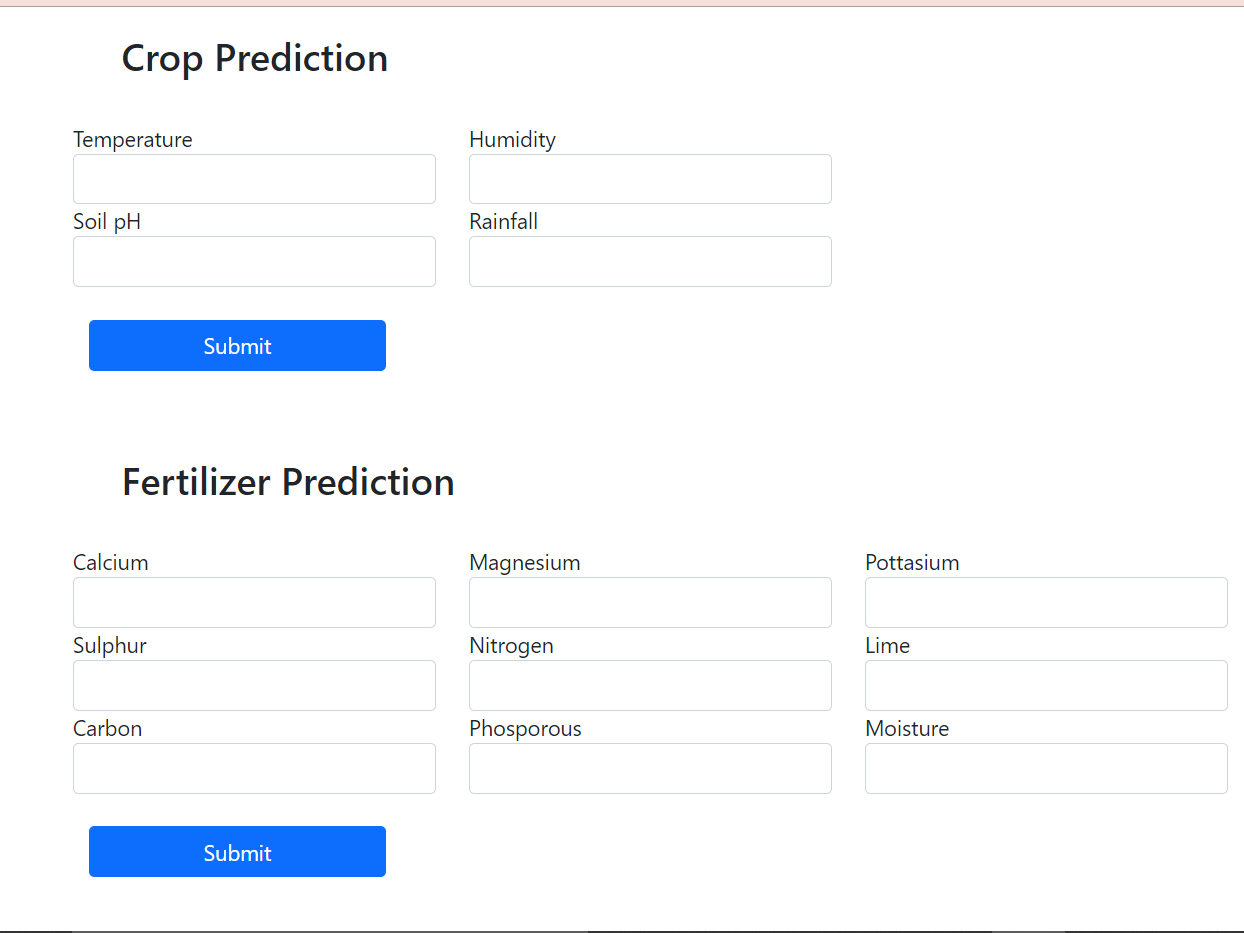
Random forest is the suitable model for most beneficial fertilizer recommendation.

**Flask Web App**

**User Interface and Output:**

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.



index.html

Output:

**Code:**

**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 class="row m-4">

<h3>Fertilizer Prediction</h3>

</div>

<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> </body>

**Conclusion and Future Scope**

**Conclusion:**

Presently our farmers are not effectively using technology and analysis, so there may be a chance of wrong selection of crop for cultivation that will reduce their income. To reduce those type of loses we have developed a farmer friendly system with UI, that will predict which would be the best suitable crop based on the given weather and land data and this system will also provide recommendation about the most beneficial fertilizer with fertilizer features. So, this makes the farmers to take right decision in selecting the crop for cultivation such that agricultural sector will be developed by innovative idea.

**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.

1. 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.
2. S. Veenadhari, Dr. Bharat Misra, Dr. CD singh (2014). Machine Learning Approach for Forecasting Crop Yield based on Climatic Parameters. ICCCI-2014

1. 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).
2. Arun K, Navin K, and Vishal V (2018). Efficient Crop Yield Prediction using Machine Learning, (pp.3151-3159), IRJET.
3. 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.

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

1. 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.

1. 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.

1. 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.
2. S.Bhanumathi, M.Vineeth and N.Rohit (2019). Crop Yield Prediction and Efficient use of Fertilizers, (pg.0769- 0773), ICCSP.

1. 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.