DAYANANDA SAGAR UNIVERSITY

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SCHOOL OF ENGINEERING**

**DAYANANDA SAGAR UNIVERSITY KUDLU GATE**

**BANGALORE - 560068**



## MINI PROJECT REPORT

***ON***

# “{OptimizingAgriculture Production}”

**SUBMITTED TO THE 6th SEMESTER DATA SCIENCE (19CS3613) COURSE**

**BACHELOR OF TECHNOLOGY**

***IN***

### COMPUTER SCIENCE & ENGINEERING

***Submitted by***

Kaushal Yadav (ENG19CS0139)

Krishna Kant Bishnoi (ENG19CS0147)

Lakshman Parihar M (ENG19CS0152)

***Under the supervision of***

### {Dr. Revathi V}

**DAYANANDA SAGAR UNIVERSITY**

### School of Engineering, Kudlu Gate, Bangalore-560068



**CERTIFICATE**

#### This is to certify that, Kaushal Yadav, Krishna Kant Bishnoi, Lakshman Parihar M bearing USN ENG19CS0139, ENG19CS0147, ENG19CS0152 has satisfactorily completed there’s Mini Project as prescribed by the University for the 6th semester B.Tech. programme in Computer Science & Engineering for Data Science (19CS3613) course during the year 2022 at the School of Engineering, Dayananda Sagar University., Bangalore.

Date: 12-may-2022

Signature of the faculty in-charge

|  |  |
| --- | --- |
| Max Marks | Marks Obtained |
|  |  |

Signature of Chairman Department of Computer Science & Engineering

**ACKNOWLEDGEMENT**

From the very core of our heart, we would like to express our sincere gratitude to **{Dr. Revathi V}** for his invaluable guidance, support, motivation and patience during the course of this mini- project work. We are always indebted to him for his kind support and constant encouragement.

We extend our sincere thanks to our **Chairman Dr. A Srinivas** who continuously helped throughout the project and without his guidance, this project would have been an uphill task.

It requires lots of efforts in terms of cooperation and support to fulfil various tasks involved during the project. We are always grateful to our peers and friends who have always encouraged us and guided us whenever we needed assistance.

Kaushal Yadav (ENG19CS0139)

Krishna Kant Bishnoi (ENG19CS0147)

Lakshman Parihar M (ENG19CS0152)

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE. NO** |
| 1. | Cover Page | 1 |
| 2. | Certificate | 2 |
| 3. | Acknowledgement | 3 |
| 4. | Content | 4 |
| 5. | Abstract | 5 |
| 6. | Introduction | 6 |
| 7. | Problem Statement | 6 |
| 8. | Objective | 6 |
| 9. | S/W & H/W Requirements | 7 |
| 10. | Proposed methodology | 7 |
| 11. | Factors helping to grow crops | 8 |
| 12. | Code | 9 |
| 13. | Result | 18 |
| 14 | Some information | 19 |
| 15 | Conclusion | 20 |
| 16 | References | 20 |

**ABSTRACT**

As we all know that agriculture depends largely on the nature of soil and the climatic conditions and many a times, we face unpredictable changes in climate like, non-seasonal rainfall or heat waves or fluctuations in humidity levels, etc. and all such events cause a great loss to our farmers and farming, because of which they are not able to utilize their agricultural land to its fullest. So, to solve all such problems, we have built a Machine Learning Model by the virtue of which we can help farmers, optimize the agricultural production, because this predictive model will help them understand that for a particular soil & given climatic condition, which crop will be best suitable for the harvest.

There are 7 key factors that we’ve taken into account which will help us in determining, exactly which crop should be grown and at what period of time, viz. Amount of Nitrogen, Phosphorus and Potassium in soil, Temperature in degree Celsius, Humidity, pH and Rainfall in mm.

Tools used: Python & Jupyter Notebook Libraries used: Numpy, Pandas, Seaborn, Matplotlib, ipywidgets and sklearn.

Machine Learning Algorithms used: Clustering Analysis and Logistic Regression.

Keywords: Temperature, Rainfall, Crops, Clustering Analysis.

## INTRODUCTION

* Agriculture is the science and art of cultivating plants and livestock.
* The history of agriculture began thousands of years ago.
* Machine learning is among the trending technologies; hence, there exist several technologies and systems that run on a machine learning framework.
* In recent times, several machine learning systems in agriculture have been tested and created.
* Research of several machine learning algorithms’ effectiveness in agriculture and other application domains has also been conducted and this is because machine learning is a very effective tool for efficient use of resources, prediction, and management, which are needed in agriculture.
* Machine learning is the ability of an electrical processing system to acquire knowledge and apply that knowledge.
* Future agriculture will use sophisticated technologies such as robots, temperature and moisture sensors, aerial images, and GPS technology.
* These advanced devices and precision agriculture and robotic systems will allow farms to be more profitable, efficient, safe, and environmentally friendly

## PROBLEM STATEMENT:

* Build a predictive model so as to suggest the most suitable crops to grow based on the available climate and soil conditions

## OBJECTIVE:

* Our objective is, building a predictive model so as to suggest the most suitable crops to grow, based on the nature of soil and the given climatic conditions.

## REQUIREMENTS:

**HARDWARE REQUIREMENTS:**

1. Any operating system (windows, MacOS etc)
2. minimum RAM --> 8GB
3. minimum CPU --> 4 cores-8 thread
4. SSD --> 512GB

**SOFTWARE REQUIREMENTS:**

1. programming languages --> Python
2. jupyter notebook

## PROPOSED METHODOLOGY

* Analysing and finding the optimal climatic condition and essential nutrients to plant and grow crops.
* Using Unsupervised learning technique, clustering the crops based on hidden pattern in it.
* Predicting the classifications of crop using Popular supervised learning method.
* Verifying the prediction result using metrics like Confusion matrix and classification report.
* Finally, Finding the optimal requirements for particular crops to check the model’s accuracy.

## FACTORS HELPING TO GROW CROPS:

* **N** - ratio of Nitrogen content in soil
* **P** - ratio of Phosphorous content in soil
* **K** - ration of Potassium content in soil
* **Temperature** - temperature in degree Celsius
* **Humidity** - relative humidity in %
* **ph.** - ph. value of the soil
* **Rainfall** - rainfall in mm

## CODE:

## #For data manipulations

## import numpy as np

## import pandas as pd

## #For data visualizations

## import matplotlib. pyplot as plt

## import seaborn as sns

## #For interactive analysis

## from ipywidgets import interact

## #Reading the dataset

## data = pd. read\_csv('data.csv')

## #Shape of the dataset

## print ("Shape of the dataset:", data.shape)

## #Checking the head of the dataset

## data.head()

## #Checking the missing values

## data.isnull().sum()

## #Checking the crops present

## data['label'].value\_counts()

## #Checking average climatic and soil requirements

## print(" Average Ratio of Nitrogen in the Soil : {0:.2f}".format(data['N'].mean()))

## print(" Average Ratio of Phosphorous in the Soil : {0:.2f}".format(data['P'].mean()))

## print(" Average Ratio of Potassium in the Soil : {0:.2f}".format(data['K'].mean()))

## print(" Average Temperature in Celsius : {0:.2f}".format(data['temperature'].mean()))

## print(" Average Relative Humidity in % : {0:.2f}".format(data['humidity'].mean()))

## print(" Average PH Value of the Soil : {0:.2f}".format(data['ph'].mean()))

## print(" Average Rainfall in mm : {0:.2f}".format(data['rainfall'].mean()))

## #Checking the detailed statistics for each crop

## @interact

## def summary(crops = list(data['label'].value\_counts().index)):

## x = data[data['label'] == crops]

## print("...........................................")

## print("Statistics for Nitrogen")

## print("Minimum Nitrogen required:", x['N'].min())

## print("Average Nitrogen required:", x['N'].mean())

## print("Maximum Nitrogen required:", x['N'].max())

## print("...........................................")

## print("Statistics for Phosphorous")

## print("Minimum Phosphorous required:", x['P'].min())

## print("Average Phosphorous required:", x['P'].mean())

## print("Maximum Phosphorous required:", x['P'].max())

## print("...........................................")

## print("Statistics for for Pottasium")

## print("Minimum Pottasium required:", x['K'].min())

## print("Average Pottasium required:", x['K'].mean())

## print("Maximum Pottasium required:", x['K'].max())

## print("...........................................")

## print("Statistics for Temperature")

## print("Minimum Temperature required: {0:.2f}".format(x['temperature'].min()))

## print("Average Temperature required: {0:.2f}".format(x['temperature'].mean()))

## print("Maximum Temperature required: {0:.2f}".format(x['temperature'].max()))

## print("...........................................")

## print("Statistics for Humidity")

## print("Minimum Humidity required: {0:.2f}".format(x['humidity'].min()))

## print("Average Humidity required: {0:.2f}".format(x['humidity'].mean()))

## print("Maximum Humidity required: {0:.2f}".format(x['humidity'].max()))

## print("...........................................")

## print("Statistics for PH")

## print("Minimum PH required: {0:.2f}".format(x['ph'].min()))

## print("Average PH required: {0:.2f}".format(x['ph'].mean()))

## print("Maximum PH required: {0:.2f}".format(x['ph'].max()))

## print("...........................................")

## print("Statistics for Rainfall")

## print("Minimum Rainfall required: {0:.2f}".format(x['rainfall'].min()))

## print("Average Rainfall required: {0:.2f}".format(x['rainfall'].mean()))

## print("Maximum Rainfall required: {0:.2f}".format(x['rainfall'].max()))

## #Comparing Average requirement and conditions for each crop

## @interact

## def compare(conditions = ['N', 'P', 'K', 'temperature', 'ph', 'humidity', 'rainfall']):

## print("Average Value for", conditions, "is {0:.2f}".format(data[conditions].mean()))

## print("...........................................")

## print("Rice : {0:.2f}".format(data[(data['label'] == 'rice')][conditions].mean()))

## print("Black grams : {0:.2f}".format(data[(data['label'] == 'blackgram')][conditions].mean()))

## print("Banana : {0:.2f}".format(data[(data['label'] == 'banana')][conditions].mean()))

## print("Jute : {0:.2f}".format(data[(data['label'] == 'jute')][conditions].mean()))

## print("Coconut : {0:.2f}".format(data[(data['label'] == 'coconut')][conditions].mean()))

## print("Apple : {0:.2f}".format(data[(data['label'] == 'apple')][conditions].mean()))

## print("Papaya : {0:.2f}".format(data[(data['label'] == 'papaya')][conditions].mean()))

## print("Muskmelon : {0:.2f}".format(data[(data['label'] == 'muskmelon')][conditions].mean()))

## print("Grapes : {0:.2f}".format(data[(data['label'] == 'grapes')][conditions].mean()))

## print("Watermelon : {0:.2f}".format(data[(data['label'] == 'watermelon')][conditions].mean()))

## print("Kidney Beans : {0:.2f}".format(data[(data['label'] == 'kidneybeans')][conditions].mean()))

## print("Mung Beans : {0:.2f}".format(data[(data['label'] == 'mungbean')][conditions].mean()))

## print("Oranges : {0:.2f}".format(data[(data['label'] == 'orange')][conditions].mean()))

## print("Chick Peas : {0:.2f}".format(data[(data['label'] == 'chickpea')][conditions].mean()))

## print("Lentils : {0:.2f}".format(data[(data['label'] == 'lentil')][conditions].mean()))

## print("Cotton : {0:.2f}".format(data[(data['label'] == 'cotton')][conditions].mean()))

## print("Maize : {0:.2f}".format(data[(data['label'] == 'maize')][conditions].mean()))

## print("Moth Beans : {0:.2f}".format(data[(data['label'] == 'mothbeans')][conditions].mean()))

## print("Pigeon Peas : {0:.2f}".format(data[(data['label'] == 'pigeonpeas')][conditions].mean()))

## print("Mango : {0:.2f}".format(data[(data['label'] == 'mango')][conditions].mean()))

## print("Pomegranate : {0:.2f}".format(data[(data['label'] == 'pomegranate')][conditions].mean()))

## print("Coffee : {0:.2f}".format(data[(data['label'] == 'coffee')][conditions].mean()))

## #Checking the below and above Average Conditions

## @interact

## def compare(conditions = ['N', 'P', 'K', 'temperature', 'ph', 'humidity', 'rainfall']):

## print("Crops that require greater than average", conditions, '\n')

## print(data[data[conditions] > data[conditions].mean()]['label'].unique())

## print("...........................................")

## print("Crops that require less than average", conditions, '\n')

## print(data[data[conditions] <= data[conditions].mean()]['label'].unique())

## plt.subplot(3,4,1)

## sns.histplot(data['N'], color="yellow")

## plt.xlabel('Nitrogen', fontsize = 12)

## plt.grid()

## plt.subplot(3,4,2)

## sns.histplot(data['P'], color="orange")

## plt.xlabel('Phosphorous', fontsize = 12)

## plt.grid()

## plt.subplot(3,4,3)

## sns.histplot(data['K'], color="darkblue")

## plt.xlabel('Pottasium', fontsize = 12)

## plt.grid()

## plt.subplot(3,4,4)

## sns.histplot(data['temperature'], color="black")

## plt.xlabel('Temperature', fontsize = 12)

## plt.grid()

## plt.subplot(2,4,5)

## sns.histplot(data['rainfall'], color="grey")

## plt.xlabel('Rainfall', fontsize = 12)

## plt.grid()

## plt.subplot(2,4,6)

## sns.histplot(data['humidity'], color="lightgreen")

## plt.xlabel('Humidity', fontsize = 12)

## plt.grid()

## plt.subplot(2,4,7)

## sns.histplot(data['ph'], color="darkgreen")

## plt.xlabel('PH Le vel', fontsize = 12)

## plt.grid()

## plt.suptitle('Distribution for Agricultural Conditions', fontsize = 20)

## plt.show()

## 

## #Checking that crops those have unusual requirements

## print("Some Interesting Patterns")

## print("...........................................")

## print("Crops that require very High Ratio of Nitrogen Content in Soil:", data[data['N'] > 120]['label'].unique())

## print("Crops that require very High Ratio of Phosphorous Content in Soil:", data[data['P'] > 100]['label'].unique())

## print("Crops that require very High Ratio of Potassium Content in Soil:", data[data['K'] > 200]['label'].unique())

## print("Crops that require very High Rainfall:", data[data['rainfall'] > 200]['label'].unique())

## print("Crops that require very Low Temperature:", data[data['temperature'] < 10]['label'].unique())

## print("Crops that require very High Temperature:", data[data['temperature'] > 40]['label'].unique())

## print("Crops that require very Low Humidity:", data[data['humidity'] < 20]['label'].unique())

## print("Crops that require very Low pH:", data[data['ph'] < 4]['label'].unique())

## print("Crops that require very High pH:", data[data['ph'] > 9]['label'].unique())

## #Checking which crop to be grown according to the season

## print("Summer Crops")

## print(data[(data['temperature'] > 30) & (data['humidity'] > 50)]['label'].unique())

## print("...........................................")

## print("Winter Crops")

## print(data[(data['temperature'] < 20) & (data['humidity'] > 30)]['label'].unique())

## print("...........................................")

## print("Monsoon Crops")

## print(data[(data['rainfall'] > 200) & (data['humidity'] > 30)]['label'].unique())

## from sklearn.cluster import KMeans

## #removing the labels column

## x = data.drop(['label'], axis=1)

## #selecting all the values of data

## x = x.values

## #checking the shape

## print(x.shape)

## #Determining the optimum number of clusters within the Dataset

## plt.rcParams['figure.figsize'] = (10,4)

## wcss = [ ]

## for i in range(1,11):

## km = KMeans(n\_clusters = i, init = 'k-means++', max\_iter = 2000, n\_init = 10, random\_state = 0)

## km.fit(x)

## wcss.append(km.inertia\_)

## 

## #Plotting the results

## plt.plot(range(1,11), wcss)

## plt.title('Elbow Method', fontsize = 20)

## plt.xlabel('No of Clusters')

## plt.ylabel('wcss')

## plt.show

## 

## #Implementation of K Means algorithm to perform Clustering analysis

## km = KMeans(n\_clusters = 4, init = 'k-means++', max\_iter = 2000, n\_init = 10, random\_state = 0)

## y\_means = km.fit\_predict(x)

## #Finding the results

## a = data['label']

## y\_means = pd.DataFrame(y\_means)

## z = pd.concat([y\_means, a], axis = 1)

## z = z.rename(columns = {0: 'cluster'})

## #Checking the clusters for each crop

## print("Lets Check the results after applying K Means Clustering Analysis \n")

## print("Crops in First Cluster:", z[z['cluster'] == 0]['label'].unique())

## print("...........................................")

## print("Crops in Second Cluster:", z[z['cluster'] == 1]['label'].unique())

## print("...........................................")

## print("Crops in Third Cluster:", z[z['cluster'] == 2] ['label'].unique())

## print("...........................................")

## print("Crops in Fourth Cluster:", z[z['cluster'] == 3]['label'].unique())

## #Splitting the Dataset for predictive modelling

## y = data['label']

## x = data.drop(['label'], axis=1)

## print("Shape of x:", x.shape)

## print("Shape of y:", y.shape)

## #Creating training and testing sets for results validation

## from sklearn.model\_selection import train\_test\_split

## x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 0)

## print("The Shape Of x train:", x\_train.shape)

## print("The Shape Of x test:", x\_test.shape)

## print("The Shape Of y train:", y\_train.shape)

## print("The Shape Of y test:", y\_test.shape)

## #Creating a Predictive Model

## from sklearn.linear\_model import LogisticRegression

## model = LogisticRegression()

## model.fit(x\_train, y\_train)

## y\_pred = model.predict(x\_test)

## #Evaluating the model performance

## from sklearn.metrics import confusion\_matrix

## #Printing the Confusing Matrix

## plt.rcParams['figure.figsize'] = (10,10)

## cm = confusion\_matrix(y\_test, y\_pred)

## sns.heatmap(cm, annot = True, cmap = 'Wistia')

## plt.title('Confusion Matrix For Logistic Regression', fontsize = 15)

## plt.show()

## 

## #Defining the classification Report

## from sklearn.metrics import classification\_report

## #Printing the Classification Report

## cr = classification\_report(y\_test, y\_pred)

## print(cr)

## 

## #head of dataset

## data.head()

## prediction = model.predict((np.array([[90, 40, 40, 20, 80, 7, 200]])))

## print("The Suggested Crop for given climatic condition is:",prediction)

## RESULT:

## The Suggested Crop for given climatic condition is: Rice

## CLUSTERING THE SIMILAR CROPS:

## Creating Clusters based on the input essential requirements using K-Means clustering algorithm.

## Finding optimal cluster value using Elbow method

## Determining the number of Clusters within the Dataset.

## Based on the given conditions implementing Hard Clustering.

## PREDICTIVE MODELLING BUILDING:

## Initially splitting the dataset into inputs (x) and targets (y).

## Splitting the dataset into Train and test set.

## Using Logistic Regression to do the predictions of the Crops.

## It employs the use of the sigmoid function that can take any real-valued number and map it into a probability value between 0 and 1 to predict the output class.

## There are two types of logistic regression: Binary and Multinomial.

## Binary Logistic Regression deals with two categories whereas multinomial deals with three or more categories.

## PREDICTION RESULT ANALYSIS:

## CONFUSION MATRIX:

## Through the Confusion Matrix we can see that the most of the predictions are correct.

## By this we came to know that the selected Logistic Regression for the Prediction performed really Good.

## CLASSIFICATION REPORT:

* From the classification Report we can see that the used Logistic regression performed really well and we got the accuracy as 97%.

## CONCLUSION:

Traditionally, judgment based on experience has been the basis for planning in agriculture, but in­creased specialization and the adoption of capital ­intensive production systems have stimulated the development of more formal planning methods based on the construction and analysis of a mathematical model. Improving agricultural productivity is important in order to improve farmer incomes, and it requires increases in yield, better productivity through the efficient utilization of resources, reduction in crop losses, and ensuring that farmers receive fair prices for output, thus our model helps farmers to choose the right crop for certain climatic conditions

## REFERENCES:

## Data set used: [https://drive.google.com/file/d/1bwv36vs6WKgrJeelGPXhNwI\_z120-tOi/view](#_Data_set_used:)

## 

* [Michael W. Berry](https://www.researchgate.net/scientific-contributions/Michael-W-Berry-38201876), [Azlinah Hj Mohamed](https://www.researchgate.net/profile/Azlinah-Mohamed), [Bee Wah Yap](https://www.researchgate.net/scientific-contributions/Bee-Wah-Yap-2033621324) Supervised and Unsupervised Learning for Data Science, research gate, January 2020.

# [Shweta Gupta](https://www.linkedin.com/in/shwetalakhimpur/),  “Data Science in the Indian Agriculture Industry”, guest blog, May 28, 2018

## YouTube