CROP YIELD PREDICTION USING MACHINE LEARNING

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

By

Sreenidhi Chappidi(38110549)
Sowmya U (38110548)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119

MARCH – 2022

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Sreenidhi Chappidi(38110549)**and **Sowmya U (38110548)** who carried out the project entitled "**Prediction of Crop Yield Using Machine Learning "** under my supervision from December 2021 to March 2022.

Internal Guide

Dr. B. ANKAYARKANNI M.E., Ph.D.

Head of the Department

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

---

Submitted for Viva voce Examination held on __

Internal Examiner **External Examiner**

DECLARATION

I **Sreenidhi Chappidi (Reg No:38110549) and Sowmya U (Reg No: 38110548)** hereby declare that the Project Report entitled "**Prediction of Crop Yield Using Machine Learning** " done by us under the guidance of **Dr. B. Ankayarkanni M.E., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in 2021-2022.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and a deep sense of gratitude to my Project Guide **Dr. B. Ankayarkanni M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Among worldwide, agriculture has the major responsibility for improving the economic contribution of the nation. However, still the most agricultural fields are under developed due to the lack of deployment of ecosystem control technologies. Due to these problems, the crop production is not improved which affects the agriculture economy. Hence a development of agricultural productivity is enhanced based on the plant yield prediction. To prevent this problem, Agricultural sectors have to predict the crop from given dataset using machine learning techniques. The examination of dataset by coordinated ML techniques. A comparative study between machine learning algorithms had been carried out in order to determine which algorithm is the most accurate in predicting the best crop .In this we are going to predict the yield if a specific crop is selected else we will predict the yield of all the crops using the parameters District name, season and year.

Keywords: *dataset, Machine Learning-Regression methods, mean absolute error, R2-score*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligent
DA	Data Analytic
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In developing countries, farming is considered as the major source of revenue for many people. In modern years, the agricultural growth is engaged by several innovations, environments, techniques and civilizations. In addition, the utilization of information technology may change the condition of decision making and thus farmers may yield the best way. For decision making process, data mining techniques related to the agriculture are used. Data mining is a process of extracting the most significant and useful information from the huge number of datasets. Nowadays, we used machine learning approach with developed in crop or plant yield prediction since agriculture has different data like soil data, crop data, and weather data. Plant growth prediction is proposed for monitoring the plant yield effectively through the machine learning techniques.

It is also applicable for the automated process of farming is the beginning of a new era in Bangladesh that will be suitable for the farmers who seek experts to take suggestion about the appropriate crop on specific location of their land and don't want to forget any step of the cultivation throughout the process. Although, the opinion from experts is the most convenient way, this application is designed to give accurate solution in fastest manner possible. This research's main objective is to bring farming process a step closer to the digital platform.

1.1 DOMAIN OVERVIEW:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning

algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be

bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Fig 1.1 Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class

classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

Agriculture is one of the most important occupations practiced in our

country. It is the broadest economic sector and plays an important role in overall development of the country. About 60 % of the land in the country is used for agriculture in order to suffice the needs of 1.2 billion people. Thus, modernization of agriculture is very important and thus will lead the farmers of our country towards profit. Data analytic (DA) is the process of examining data sets in order to draw conclusions about the information they contain, increasingly with the aid of specialized systems and software. Earlier yield prediction was performed by considering the farmer's experience on a particular field and crop. However, as the conditions change day by day very rapidly, farmers are forced to cultivate more and more crops. Being this as the current situation, many of them don’t have enough knowledge about the new crops and are not completely aware of the benefits they get while farming them. Also, the farm productivity can be increased by understanding and forecasting crop performance in a variety of environmental conditions. Thus, the proposed system takes the location of the user as an input. From the location, the nutrients of the soil such as Nitrogen, Phosphorous, Potassium is obtained. This static data is the crop production and data related to demands of various crops obtained from various websites. It applies machine learning and prediction algorithm to identify the pattern among data and then process it as per input conditions.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

2.2 REVIEW OF LITERATURE SURVEY:

Title: Estimation of Organic Matter Content in Coastal Soil Using Reflectance Spectroscopy Research

Author: ZHENG Guanghui1, Dongryeol RYU2, ∗, JIAO Caixia1 and HONG Changqiao1 **Year**: 2015

Description:

Rapid determination of soil organic matter (SOM) using regression models based on soil reflectance spectral data serves an important function in precision agriculture. “Deviation of arch” (DOA)-based regression and partial least squares regression (PLSR) are two modelling approaches to predict SOM. However, few studies have explored the accuracy of the DOA

based regression and PLSR models. Therefore, the DOA-based regression and PLSR were applied to the visible near-infrared (VNIR) spectra to estimate SOM content in the case of various dataset divisions. A two-fold cross-validation scheme was adopted and repeated 10000 times for rigorous evaluation of the DOA-based models in comparison with the widely used PLSR model. Soil samples were collected for SOM analysis in the coastal area of northern Jiangsu Province, China. The results indicated that both modelling methods provided reasonable estimation of SOM, with PLSR outperforming DOA-based regression in general. However, the performance

of PLSR for the validation dataset decreased more noticeably. Among the four DOA-based regression models, a linear model provided the best estimation of SOM and a cut-off of SOM content (19.76 g kg−1), and the performance for calibration and validation datasets was consistent. As the SOM content exceeded 19.76 g kg−1, SOM became more effective in masking the spectral features of other soil properties to a certain extent. This work confirmed that reflectance spectroscopy combined with PLSR could serve as a non-destructive and cost-efficient way for rapid determination of SOM when

hyper spectral data were available. The DOA-based model, which requires only 3 bands in the visible spectra, also provided SOM estimation with acceptable accuracy.

Title: Preliminary Study of Soil Available Nutrient Simulation Using a Modified WOFOST Model and Time-Series Remote Sensing Observations

Author: Zhiqiang Cheng 1,2 ID, Jihua Meng 1,*, Yanyou Qiao 1, Yiming Wang 1,2, Wenquan Dong 1 and Yanxin Han 1,2

Year: 2017

Description:

The approach of using multispectral remote sensing (RS) to estimate soil available nutrients (SANs) has been recently developed and shows promising results. This method overcomes the limitations of commonly used methods by building a statistical model that connects RS-based crop growth and nutrient content. However, the stability and accuracy of this model require improvement. In this article, we replaced the statistical model by integrating the World Food Studies (WOFOST) model and time series of remote sensing (T

RS) observations to ensure stability and accuracy. Time series of HJ-1 A/B data was assimilated into the WOFOST model to extrapolate crop growth simulations from a single point to a large area using a specific assimilation method. Because nutrient-limited growth within the growing season is required and the SAN parameters can only be used at the end of the growing season in the original model, the WOFOST model was modified. Notably, the calculation order was changed, and new soil nutrient uptake algorithms were implemented in the model for nutrient-limited growth estimation. Finally, experiments were conducted in the spring maize plots of Hongxing Farm to analyze the effects of nutrient stress on crop growth and the SAN simulation accuracy. The results confirm the differences in crop growth status caused by a lack of soil nutrients. The new approach can take advantage of these differences to provide better SAN estimates. In general, the new approach can overcome the limitations of existing methods and simulate the SAN status with reliable accuracy.

Title: Distinguishing Heavy-Metal Stress Levels in Rice Using Synthetic Spectral Index Responses to Physiological Function Variations

Author: Ming Jin, Xiangnan Liu, Ling Wu, and Meiling Liu

Year: 2016

Description:

Accurately assessing the heavy-metal contamination in crops is crucial to food security. This study provides a method to distinguish heavy-metal stress levels in rice using the variations of two physiological functions as discrimination indices, which are obtained by assimilation of remotely sensed data with a crop growth model. Two stress indices, which correspond to dailytotalCO2 assimilationanddry-matterconversioncoefficient were incorporated into the World Food Study (WOFOST) crop growth model and calculated by assimilating the model with leaf area index (LAI), which was derived from time-series HJ1- CCD data. The stress levels are not constant with rice growth; thus, to improve the reliability, the two stress indices were obtained at both the first and the latter half periods of rice growth.

To com- pare the stress indices of different stress levels, a synthetic stress index was established by combining the two indices; then, three types of stress index discriminant spaces based on the synthetic index of different growth periods were constructed, in which the two-dimensional discriminant space based on two growth periods showed the highest accuracy, with a misjudgement rate of 4.5%. When the discrimination rules were applied at a regional scale, the average correct discrimination rate was 95.0%.

Title: Design and Characterization of a Fringing Field Capacitive Soil Moisture Sensor **Author:** Manash Protim Goswami, Babak Montazer, and Utpal Sarma, Member, IEEE **Year**: 2018

Description:

The optimization and implementation of a fringing field capacitive soil moisture sensor using the printed circuit board technology. It includes the analysis of a novel configuration of an interdigital sensor for measuring soil moisture with two existing configurations. The optimized designs were simulated by using a 3-D finite-element method and fabricated by using a copper clad board. The performance of the fabricated sensors was evaluated using four soil samples collected from different locations. The observations were compared with the standard gravimetric method to evaluate the soil water content of the samples. The characterization method and the results of the whole sensing system are discussed in terms of calibration, dynamic test, and repeatability

Title: Improving Spring Maize Yield Estimation at Field Scale by Assimilating

Time-Series HJ-1 CCD Data into the WOFOST Model Using a New Method with Fast Algorithms **Author:** Zhiqiang Cheng, Jihua Meng * and Yiming Wang

Year: 2016

Description:

Field crop yield prediction is crucial to grain storage, agricultural field management, and national agricultural decision-making. Currently, crop models are widely used for crop yield prediction. However, they are hampered by the uncertainty or similarity of input parameters when extrapolated to field scale. Data assimilation methods that combine crop models and remote sensing are the most effective methods for field yield estimation. In this study, the World Food Studies (WOFOST) model is used to simulate the growing process of spring maize. Common assimilation methods face some difficulties due to the scarce, constant, or similar nature of the input parameters. For example, yield spatial heterogeneity simulation, coexistence of common assimilation methods and the nutrient module, and time cost are relatively important limiting factors. To address the yield simulation problems at field scale, a simple yet effective method with fast algorithms is presented for assimilating the time-series HJ-1 A/B data into the WOFOST model in order to improve the spring maize yield simulation. First, the WOFOST model is calibrated and validated to obtain the precise mean yield. Second, the time-series leaf area index (LAI) is calculated from the HJ data using an empirical regression model. Third, some fast algorithms are developed to complete assimilation. Finally, several experiments are conducted in a large farmland (Hongxing) to

evaluate the yield simulation results. In general, the results indicate that the proposed method reliably improves spring maize yield estimation in terms of spatial heterogeneity simulation ability and prediction accuracy without affecting the simulation efficiency.

Title: A generalized regression-based model for forecasting winter wheat yields in Kansas and Ukraine using MODIS data

Author: Becker-Reshef, E. Vermote, M. Lindeman

Year: 2010

Description:

As new remote sensing instruments and data become available their utility for improving established terrestrial monitoring tasks need to be evaluated. An empirical, generalized remotely sensed based yield model was developed and successfully applied at the state level in Kansas using daily, high quality 0.05° NDVI time series data to drive the regression model, a percent crop mask as a filter to identify the purest winter wheat pixels, and

USDA NASS county crop statistics for model calibration. The model predictions of production in Kansas closely matched the USDA/NASS reported numbers with a 7% error. This empirical regression model that was developed in Kansas was successfully applied directly in Ukraine. The model forecast winter wheat production in Ukraine six weeks prior to harvest with a 10% error of the official production numbers. In 2009 the model was run in real-time in Ukraine and forecast production within 7% of the official statistics which were released after the harvest. Wheat is one of the key cereal crops grown worldwide, providing the primary caloric and nutritional source for millions of people around the world. In order to ensure food security and sound, actionable mitigation strategies and policies for management of food shortages, timely and accurate estimates of global crop production are essential. It combines a new BRDF-corrected, daily surface reflectance dataset developed from NASA's Moderate resolution Imaging Spectro-radiometer (MODIS) with detailed official crop statistics to develop an empirical, generalized approach to forecast wheat yields. The first step of this study was to develop and evaluate a regression-based model for forecasting winter wheat production in Kansas. This regression-based model was then directly applied to forecast winter wheat production in Ukraine. The forecasts of production in Kansas closely matched the USDA/NASS reported numbers with a 7% error. The same regression model forecast winter wheat production in Ukraine within 10% of the official reported production numbers six weeks prior to harvest. Using new data from MODIS, this method is simple, has limited data requirements, and can provide an indication of winter wheat production shortfalls and surplus prior to harvest in regions where minimal ground data is available.

Title: Machine Learning Approaches to Corn Yield Estimation Using Satellite Images and Climate Data: A Case of Iowa State

Author: Kim, NariLee, Yang-Won

Year: 2016

Description:

Machine learning, which is an efficient empirical method for classification and prediction, is another approach to crop yield estimation. It described the corn yield estimation

in Iowa State using four machine learning approaches such as RF (Random Forest), ERT (Extremely Randomized Trees) and DL (Deep Learning). Also, comparisons of the validation statistics among them were presented. To examine the seasonal sensitivities of the corn yields, three period groups were set up: (1) MJJAS (May to September), (2) JA (July and August) and (3) OC (optimal combination of month). In overall, the DL method showed the highest accuracies in terms of the correlation coefficient for the three period groups. The accuracies were relatively favourable in the OC group, which

indicates the optimal combination of month can be significant in statistical modelling of crop yields. The differences between our predictions and USDA (United States Department of Agriculture) statistics were about 6-8 %, which shows the machine learning approaches can be a viable option for crop yield modelling. Monitoring crop yield is important for many agronomy issues such as farming management, food security and international crop trade. Because South Korea highly depends on imports of most major grains except for rice, reasonable estimations of crop yields are more required under recent conditions of climate changes and various disasters. Remote sensing data has been widely used in the estimation of crop yields by employing statistical methods such as regression model. It conducted multivariate regression analyses to estimate corn and soybean yields in Iowa using MODIS (Moderate Resolution Imaging Spector radiometer) NDVI (Normalized Difference Vegetation Index), climate factors and soil moisture and presented regression models for the estimation of winter wheat yields using MODIS NDVI and weather data in Shandong, China. It estimated corn and soybean yields using several MODIS products and climatic variables for Midwestern United States (US) and represented prediction errors of about 10 %. It built multiple regression models using MODIS NDVI and weather data to estimate rice yields in North Korea and showed the RMSE of 0.27 ton/ha. Most of the previous studies are based on the multivariate regression analysis using the relationship between crop yields and agro-environmental factors such as vegetation index, climate variables and soil properties.

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

- In our research, which we found in the previous research papers is that everyone uses climatic factors like rainfall, sunlight and agricultural factors like soil type, nutrients possessed by the soil (Nitrogen, Potassium, etc.)
- But the problem is we need to gather the data and then a third party does this prediction and then it is explained to the farmer and this takes a lot of effort for the farmer and he doesn't understand the science behind these factors.

3.2 PROPOSED SYSTEM

To make it simple and which can be directly used by the farmer this paper uses simple factors like which state and district is the farmer from, which crop, crop year and in what season (as in Kharif, Rabi, etc.).

The system prepared predict crops yield of almost all kinds of crops that are planted in India. This script makes novel by the usage of simple parameters like State, district, season, area and the user can predict the yield of the crop in which year he or she wants to.

The client has to enter the district, season and year. After submitting the inputs, it will output the best crop to be planted

STEPS

- Define a problem
- Preparing data
- Evaluating algorithms
- Improving results
- Predicting results

3.3 OBJECTIVE

- Data validation
- Data Cleaning/ Preparing
- Data Visualization
- Using algorithms (Like random forest, Decision tree Logistic regression algorithm)

3.4 SCOPE

The scope of this project is to investigate a dataset of crop records for agricultural sector using machine learning technique. To identifying crop predicting by farmer is more difficult. We try to reduce this risk factor behind selection of the crop.

3.5 HARDWARE AND SOFTWARE REQUIREMENTS

3.5.1 Software Requirements

Operating System: Windows

Tool: Anaconda with Jupyter Notebook

3.5.2 Hardware requirements

Processor: Pentium IV/III

Hard disk: minimum 80 GB

RAM: minimum 2 GB

3.5.3 PROJECT REQUIREMENTS

3.5.3.1 Functional requirements

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

3.5.3.2 Non-Functional Requirements

Process of functional steps,
Problem define
Preparing data
Evaluating algorithms
Improving results
Prediction the result

3.6 SOFTWARE DESCRIPTION

Anaconda is a free and open distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook

- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

3.6.1Conda

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

3.6.2 The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3.6.3 Notebook Document

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

3.6.4 Jupyter Notebook App

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

3.6.5 Kernel

A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or

with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down.

3.6.6 Notebook Dashboard

The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

3.7 System Architecture

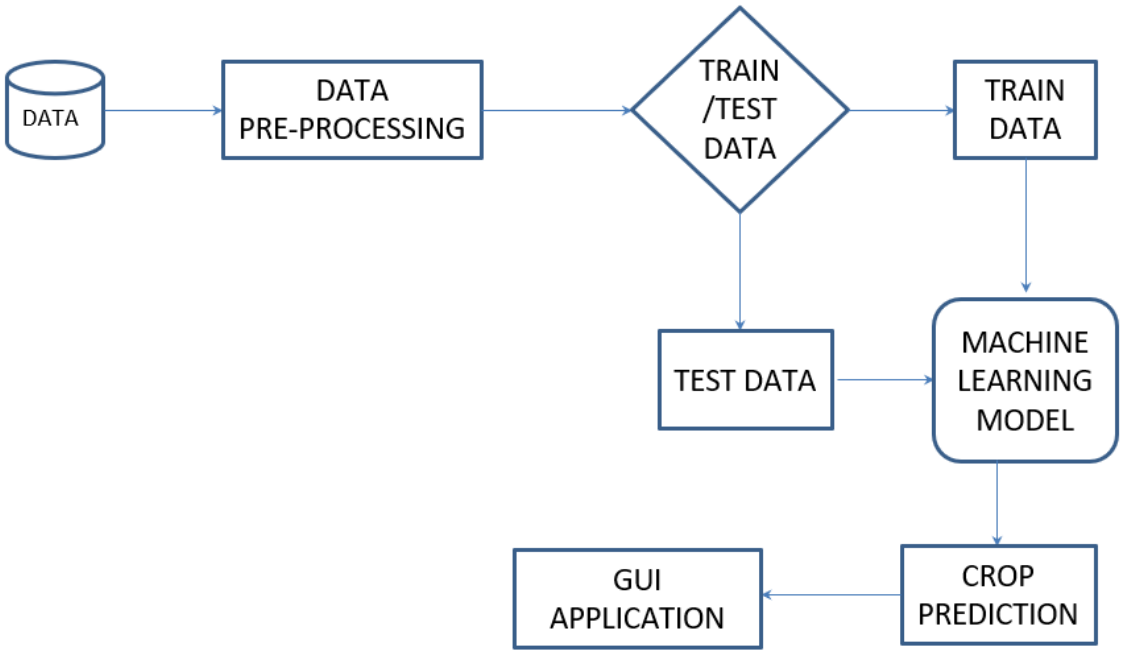


Fig 3.1 system architecture

3.8 Modules

- Data validation and Data cleaning (Module 1)
- Exploration data analysis of visualization and normalizing the data (Module 2)
- Training a model by given attributes and using Random Forest algorithm (Module 3)
- Performance measurements of KNN and Decision Tree (Module-05)
- Performance measurements of Linear and Lasso Regression (Module-05)
- GUI based prediction of crop yield (Module-06**)**

3.8.1 Module-01: Variable Identification Process / data validation process and Data cleaning:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the

training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers uses this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model. For example, time series data can be analysed by regression algorithms; classification algorithms can be used to analyse discrete data. (For example, to show the data type format of given dataset)

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0

Fig 3.2 Given Data Frame

3.8.1.1. Data cleaning:

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary

goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

```
df_input = df_input[df_input['Area'] > 0 ]
df_input = df_input[df_input['Production'] > 0 ]
df_input = df_input.replace(r'^\s*$', np.NaN, regex=True)
df_input = df_input.dropna()
df_input["ProductionPerArea"] = ((df_input["Production"])/(df_input["Area"]))
df_input = df_input.drop(columns=['State_Name','Area','Production'])
```

Fig 3.3: Cleaning the Dataset

3.8.2 Module-02: Exploration data analysis of visualization and normalization:

3.8.2.1 Exploration data analysis of visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and inapplied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data. How to chart time series data with line plots and categorical quantities with bar charts. How to summarize data distributions with histograms and box plots. How to summarize the relationship between variables with scatter plots.

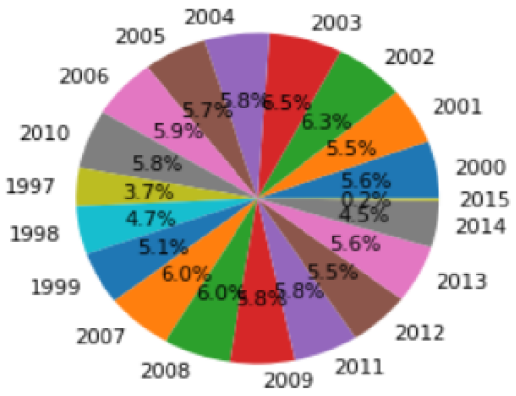


Fig 3.4 Percentage Level Of Data In Different Years

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results. Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary 22 distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data. Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

It couldn't fit the model on the training data and can't say that the say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

```
p1 = np.percentile(np.array(df_input['ProductionPerArea']), 25)
p2 = np.percentile(np.array(df_input['ProductionPerArea']), 99)
df_input = df_input[df_input['ProductionPerArea'] > p1]
df_input = df_input[df_input['ProductionPerArea'] < p2]
```

Fig 3.5 Removing outliers

3.8.2.2 Data Normalization

To train data with string datatype we use label encoder to convert them to numeric data. Label encoding simply assigns each data point a number starting from .the attributes we are label encoding is district name, season and crop. Then to scale each data attribute we use two types of scaling techniques which are MinMax

transform and normalization. The attributes district name, season, crop is scaled by MinMax transform and production per area is scaled by normalization. MinMax transform is done by subtracting each datapoint from minimum among datapoints then dividing by maximum- minimum. normalization is done by dividing datapoint with standard deviation. We choose normalization to productionperarea as many data points are low and min max transform is yielding good results.

3.8.3 Module 3:Training a model by given attributes and using Random Forest algorithm

At First, we import all the required modules. Then we use appropriate cleaning methods to clean our data. Then we split our data according to the required output. We split our dataset into train and test data using train_test_split technique. The X prefix indicates the element esteems and y prefix indicates target esteems. This splits the dataset into train and test information in the proportion which we mentioned. Here we are using a proportion of 80:20. At this point we epitomize any calculation. Here we fit our preparation information into this calculation so our system can get prepared utilizing this information. Here the training part is finished

```
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small
x_train, x_test, y_train, y_test =
 train_test_split(df.iloc[:,:-1], df.iloc[:,-1], test_size=0.2, random_state=2)
```

Fig 3.6 Splitting A Data Set in To Training And Training Data

3.8.3.1 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. The following are the basic steps involved in performing the random forest algorithm

pick N random records from the dataset. Build a decision tree based on these N records. Choose the number of trees you want in your algorithm and repeat steps 1 and 2. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the

average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

3.8.4 Module 4 :Performance measurements of kNN and Decision Tree

3.8.4.1 KNN

K-Nearest Neighbor (KNN) K-Nearest Neighbor is a supervised machine learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors. The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the “k” is the number of neighbors it checks). Makes predictions about the validation set using the entire training set. KNN makes a prediction about a new instance by searching through the entire set to find the k “closest” instances. “Closeness” is determined using a proximity measurement (Euclidean) across all features.

3.8.4.2 Decision Tree

It is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It 24 works for both continuous as well as categorical output variables.

Assumptions of Decision tree:

- At the beginning, we consider the whole training set as the root.
- Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision

node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers

3.8.5 Module 5:Performance measurements of Lasso and Linear Regression

3.8.5.1 Linear Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression*. If there is a single input variable (x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called multiple linear regression*. The linear regression model gives a sloped straight line describing the relationship within the variables**.**

The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

3.8.5.2 Lasso Regression

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Lasso Regression uses L1 regularization technique (will be discussed later in this article). It is used when we have a greater number of features because it automatically performs feature selection

3.8.6 Module 6: GUI based prediction of crop yield

- we created a webpage and deployed it in Heroku. we used HTML, CSS and js for the frond end.
- In the backend we used python language and flask framework to implement

3.8.7 Parameter calculations

Regression refers to predictive modelling problems that involve predicting a numeric value.

It is different from classification that involves predicting a class label. Unlike classification, you cannot use classification accuracy to evaluate the predictions made by a regression model.

Instead, you must use error metrics specifically designed for evaluating predictions made on regression problems.

There are four error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)
- R2_score

3.8.7.1 Mean Squared Error

Mean Squared Error, or MSE for short, is a popular error metric for regression problems. It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here "*least squares*" refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

- MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

Where *y_i* is the i'th expected value in the dataset and *yhat_i* is the i'th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

The squaring also has the effect of inflating or magnifying large errors. That is, the larger the difference between the predicted and expected values, the larger the resulting squared positive error. This has the effect of "*punishing*" models more for larger errors when MSE is used as a loss function. It also has the effect of "*punishing*" models by inflating the average error score when used as a metric.

3.8.7.2 Root Mean Squared Error:

RMSE, is an extension of the mean squared error**.** Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units "*dollars*," then the RMSE error score will also have the unit "*dollars*" and not "*squared dollars*" like the MSE.
As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.
The RMSE can be calculated as follows:

- **RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)**

Where *y_i* is the i'th expected value in the dataset, *yhat_i* is the i'th predicted value, and *sqrt()* is the square root function**.**

We can restate the RMSE in terms of the MSE as:

- RMSE = sqrt(MSE)

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an example of jensens inequality

3.8.7.3 Mean Absolute Error

MAE, is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.
That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or *abs()* is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted

value may be positive or negative and is forced to be positive when calculating the MAE.
The MAE can be calculated as follows:

- MAE = 1 / N * sum for i to N abs (y_i – yhat_i)

Where *y_i* is the i'th expected value in the dataset, *yhat_i* is the i'th predicted value and *abs ()* is the absolute function

3.8.7.4 R2_score:

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable

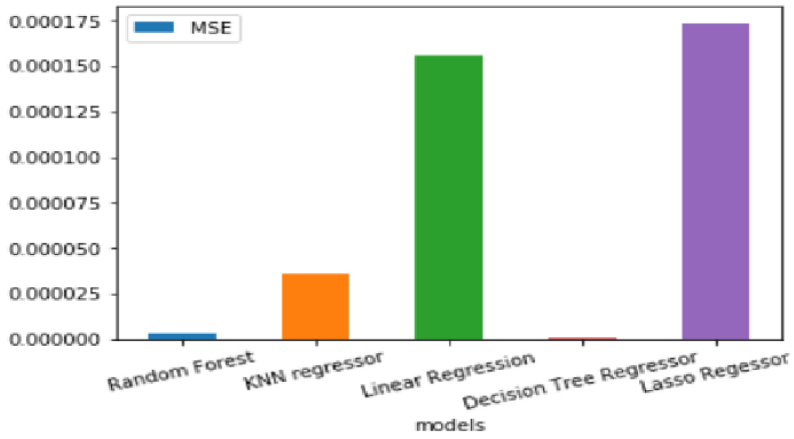
$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}$$

CHAPTER 4

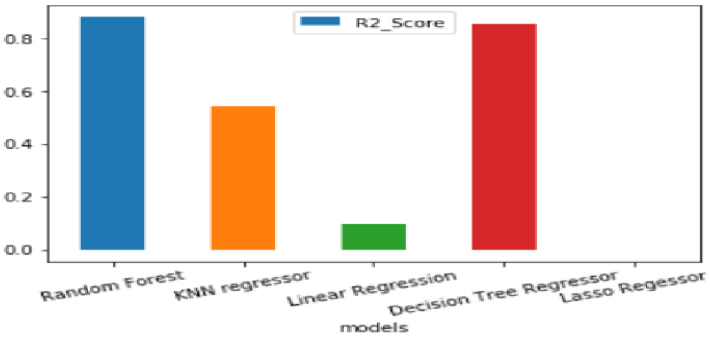
RESULTS AND DISCUSSION

	MSE	MAE	RMSE	R2_Score	models
0	3.426823e-06	0.001505	0.004558	0.880158	Random Forest
1	3.595377e-05	0.003637	0.008913	0.541729	KNN regressor
2	1.557740e-04	0.006636	0.012498	0.098844	Linear Regression
3	1.761657e-07	0.001490	0.005049	0.852930	Decision Tree Regressor
4	1.737817e-04	0.007141	0.013166	-0.000002	Lasso Regessor

4.1. Graph comparing MSE values in various algorithms

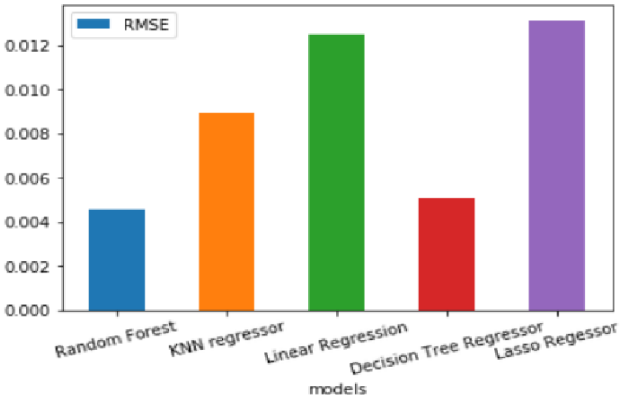


4.2. Graph comparing R2_Score values in various algorithms



4.3. Graph comparing MAE values in various algorithms

4.4 Graph comparing RMSE values in various algorithms



- The above graphs describe the comparison of various error values of crop yield prediction.
- To find out the best algorithm which gives less error we have compared the five machine algorithms
- From the analysis after the comparison of machine learning algorithms we got decision Tree and Random Forest gave less error and high R2_score compared to other. We use Random Forest algorithm for prediction as it is collection of decision trees.

CHAPTER 5

CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. Finally we predict the crop using machine learning algorithm with different results. This brings some of the following insights about crop prediction. As maximum types of crops will be covered under this system, farmer may get to know about the crop which may never have been cultivated and lists out all possible crops, it helps the farmer in decision making of which crop to cultivate. Also, this system takes into consideration the past production of data which will help the farmer get insight into the demand and the cost of various crops in market.

5.1 Future Work:

- ➢ using neural networks so that they may give better results
- ➢ Taking an extra parameter cost so that predicting of crop becomes easier.
- ➢ To optimize the work to implement in Artificial Intelligence environment.

REFERENCES

[1] Bendre, M. R., Thool, R.C., Thool, V. R., “Big Data in Precision Agriculture : Weather Forecasting for Future Agriculture”, 1 st International Conference on Next Generation Computing Technologies, pp.744-750, 2015
.
[2] Grajales, D.F.P., Mosquera, G.J.A, Mejia, F., Piedrahita, L.C., Basurto, C., “Crop-Planning, Making Smarter Agriculture With Climate Data”,Fourth International Conference on Agro-GeoInformatics, pp.240-244, 2015.

[3] Hemageetha, N., “A survey on application of data mining techniques to analyze the soil for agricultural purpose”, 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp.3112-3117, 2016.

[4] Mayank Champaneri, Chaitanya Chandvidkar , Darpan Chachpara, Mansing Rathod, “Crop yield prediction using machine learning” International Journal of Science and Research ,April 2020.

[5] Pavan Patil, Virendra Panpatil, Prof. Shrikant Kokate, “Crop Prediction System using Machine Learning Algorithms”, International Research Journal of Engineering and Technology, Feb 2020.

[6] Raval Agrawal, H., Agrawal, P., “Review on Data Mining Tools”, International Journal of Innovative Science, Engineering & Technology, Vol. 1, Issue 2, pp.52-56, 2014.

[7] Sabri Arik, Tingwen Huang, Weng Kin Lai, Qingshan Liu , “Soil Property Prediction: An Extreme Learning Machine Approach” Springer, vol. 3, Issue 4,666-680,2015.

[8] Shivnath Ghosh,Santanu Koley, “Machine Learning for Soil Fertility and Plant Nutrient Management using Back Propagation Neural Networks” IJRITCC, vol. 2, Issue 2,292-297,2014.

[9] Tan, L., “Cloud-based Decision Support and Automation for Precision Agriculture in Orchards”, ScienceDirect, pp. 330-335, IFAC–Papers OnLine 49-16, pp.330-335, 2016
.
[10] Zhihao Hong,Z. Kalbarczyk,R. K. Iyer, “A Data-Driven Approach to Soil Moisture Collection and Prediction” IEEE Xplore,vol. 2, Issue 2,292-297,2016.

APPENDICES

A) SOURCE CODE

```
#import libraries for access and functional purpose
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as s
df_input=pd.read_csv("C:/Users/ADITYA/Desktop/dataworld_set.csv")
df_input.shape
df_input.head()
df_input['Crop'].nunique()
df_input['Crop'].value_counts()
df_input['District_Name'].nunique()
df_input['District_Name'].value_counts()
df_input['Season'].nunique()
df_input['Season'].value_counts()
df_input['Crop_Year'].nunique()
df_input['Crop_Year'].value_counts()
df_input.info()
df_input.isnull().sum()
df_input.duplicated().sum()
df_input = df_input[df_input['Area'] > 0 ]
df_input = df_input[df_input['Production'] > 0 ]
df_input = df_input.replace(r'^\s*$', np.NaN, regex=True)
df_input = df_input.dropna()
df_input["ProductionPerArea"] = ((df_input["Production"])/(df_input["Area"]))
df_input = df_input.drop(columns=['State_Name','Area','Production'])
Crops = df_input['Crop']
CropsCount = {}
for crop in Crops:
    CropsCount[crop] = CropsCount.get(crop, 0)+1
#extract values and keys of dict:CropsCount
```

```
labels = list(CropsCount.keys())
values = list(CropsCount.values())
plt.pie(values, labels=labels,autopct='%1.1f%%')
plt.show()
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
categorical_columns = ['District_Name', 'Crop' ,'Season']
#label encoder dict
labels_dict = {}
#scaling dict
scaling_dict = {}
for column in categorical_columns:
  le = LabelEncoder()
  le.fit(df_input[column])
  df_input[column] = le.transform(df_input[column])
  labels_dict[column] = le.classes_
   df_input.describe()
  x = df_input['ProductionPerArea']
print(x.describe())
import matplotlib.pyplot as plt
plt.style.use('ggplot')
fig, ax = plt.subplots()
ax.boxplot((x), vert=False, showmeans=True, meanline=True,
          labels=('x'), patch_artist=True,
 medianprops={'linewidth': 2, 'color': 'purple'},
          meanprops={'linewidth': 2, 'color': 'red'})
plt.show()
  def deviationTransform(arr):
  d = np.std(arr)
  return [0,d]
def minMaxTransform(arr):
  min = np.min(arr)
```

```
  max = np.max(arr)
  return [min,max-min]
for column in categorical_columns:
  scaling_params = minMaxTransform(np.array(df_input[column]))
  df_input[column] = (df_input[column] -
scaling_params[0])/scaling_params[1]
  scaling_dict[column] = scaling_params
scaling_params = minMaxTransform(np.array(df_input['Crop_Year']))
scaling_dict['Crop_Year'] = scaling_params
df_input['Crop_Year'] = (df_input['Crop_Year'] -
scaling_params[0])/scaling_params[1]
scaling_params =
deviationTransform(np.array(df_input['ProductionPerArea']))
df_input['ProductionPerArea'] = (df_input['ProductionPerArea'] -
scaling_params[0])/scaling_params[1]
scaling_dict['ProductionPerArea'] = scaling_params


p1 = np.percentile(np.array(df_input['ProductionPerArea']), 25)
p2 = np.percentile(np.array(df_input['ProductionPerArea']), 99)
df_input = df_input[df_input['ProductionPerArea'] > p1]
df_input = df_input[df_input['ProductionPerArea'] < p2]
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,:-1], df.iloc[:,-1],
test_size=0.2, random_state=2)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import r2_score
def trained(clf,x_train,x_test,y_train,y_test):
  clf.fit(x_train,y_train)
  y_train_pred = clf.predict(x_train)
```

```
  print("MSE " + str(mean_squared_error(y_train, y_train_pred)))
  print("MAE " + str(mean_absolute_error(y_train, y_train_pred)))
  print("RMSE " + str(np.sqrt(mean_squared_error(y_train, y_train_pred))))
  #print("RMSLE " + str(np.sqrt(mean_squared_log_error(y_test, y_pred))))
  R2_score = r2_score(y_train, y_train_pred)
  print("R2 Score " + str(R2_score))
  y_pred = np.array(y_train_pred)
  y_test = np.array(y_train)
  data={}
  data['x'] = y_test
  data['y'] = y_pred
  sns.regplot(x="x", y="y", data=data);
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import r2_score
def tested(clf,x_train,x_test,y_train,y_test):
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print("MSE " + str(mean_squared_error(y_test, y_pred)))
    print("MAE " + str(mean_absolute_error(y_test, y_pred)))
    print("RMSE " + str(np.sqrt(mean_squared_error(y_test, y_pred))))
    #print("RMSLE " + str(np.sqrt(mean_squared_log_error(y_test, y_pred))))
    R2_score = r2_score(y_test,y_pred)
    print("R2 Score " + str(R2_score))
    y_pred = np.array(y_pred)
    y_test = np.array(y_test)
    data={}
    data['x'] = y_test
    data['y'] = y_pred
    sns.regplot(x="x", y="y", data=data);
```

Random Forest :

```
  from sklearn.ensemble import RandomForestRegressor
  regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
  print("\t\t\t random-forest classifier")
  trained(regresser,x_train,x_test,y_train,y_test)

  from sklearn.ensemble import RandomForestRegressor
  regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
  print("\t\t\t random-forest classifier")
  tested(regresser,x_train,x_test,y_train,y_test)
from sklearn.ensemble import RandomForestRegressor
  regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
  regresser.fit(x_train,y_train)
  y_pred = regresser.predict(x_test)
  y_pred = np.array(y_pred)
  y_test = np.array(y_test)
  for i in range(y_test.size):
      if(y_test[i]>0 and y_test[i]<0.001):
          print(i,y_test[i],y_pred[i])
  from sklearn.metrics import mean_squared_error
  from sklearn.metrics import mean_absolute_error
  from sklearn.metrics import mean_squared_log_error
  from sklearn.metrics import r2_score
  import numpy as np
  import seaborn as sns
  import matplotlib.pyplot as plt
  def classify(clf,x_train,x_test,y_train,y_test):
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print(y_pred)
    y_train_pred = clf.predict(x_train)
    metricsList = []
    metricsList.append(mean_squared_error(y_train, y_train_pred))
```

```
    metricsList.append(mean_absolute_error(y_test, y_pred))
    metricsList.append(np.sqrt(mean_squared_error(y_test, y_pred)))
    metricsList.append(r2_score(y_test,y_pred))
    return metricsList
clfMetrics = []
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,:-1], df.iloc[:,-1],
test_size=0.2, random_state=2)
from sklearn.ensemble import RandomForestRegressor
regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
print("\t\t\t random-forest classifier")
clfMetrics.append((classify(regresser,x_train,x_test,y_train,y_test)))
print(clfMetrics)
print(clfMetrics)
```

Html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='css/main.css') }}"
    />
    <title>Crop Yield Prediction</title>
  </head>
  <body>
     <div class = output>
```

```
        <h1>Crop Yield Prediction</h1>
      <form>
        <br><label for="stateName">Choose a State:</label>
        <select name="stateName" id="stateName"
onChange="stateSelected()"></select></br>
        <br><label for="districtName">Choose a District:</label>
        <select name="districtName" id="districtName"></select></form></br>
        <br><label for="crop">Choose a Crop (optional):</label>
        <select name="crop" id="crop"></select></br>
        <br><label for="season">Choose a Season:</label>
        <select name="season" id="season"></select></br>
        <br><label for="year">Year:</label>
        <input type="text" name="year" id="year" value="2022"></br>
        <button onClick="predict()">Submit</button>
      </form>
      <div class= results>
      <p>Yield in tonnes/hectare ⬇️⬇️</p>
      <p id="results">Submit the Form to get Results 😎 </p>
      </div>
    </div>
  </body>
  <script src="{{ url_for('static', filename='js/main.js') }}"></script>
</html>
```

Css

```
html {
  overflow-y: auto;
}
body {
  background-image: url("crop_yield.jpeg");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
```

```
        color: yellow;
      }
.output {
        font-size: 24px;
        color: yellow;
        border-radius: 1em;
        padding: 1em;
        top: 50%;
        left: 50%;
        margin-right: -50%;
        transform: translate(-50%, -50%);
        position: absolute;
        bottom: 0;
      }
      table,
      tr,
      td {
        font-size: 20px;
        font-family: monospace;
        font-weight: bold;
      }
      h1 {
        font-size: 40px;
        font-family: monospace;
        text-align: center;
      }
      .results {
        color: rgb(0, 255, 0);
        overflow-y: auto;
        height: 600px;
        width: 500px;
      }
```

Javascript:

```
let stateDistrictMap;
        fetch(new URL(document.URL) + "/stateDistrictMap")
          .then((res) => res.json())
          .then((data) => {
            stateDistrictMap = data["result"];
            document.getElementById("stateName").innerHTML = List2PicklistValue(
              Object.keys(stateDistrictMap)
            );
          })
          .catch((error) => {
            console.log("error getting states and districts from server :(");
          });
        fetch(new URL(document.URL) + "options?param=Season")
          .then((res) => res.json())
          .then((data) => (document.getElementById("season").innerHTML =
        data.result))
          .catch((error) => {
            console.log("error getting seasons from server :(");
          });
        fetch(new URL(document.URL) + "options?param=Crop")
          .then((res) => res.json())
          .then((data) => (document.getElementById("crop").innerHTML =
        data.result))
          .catch((error) => {
            console.log("error getting crops from server :(");
          });
        function predict() {
          document.getElementById("results").innerHTML = "Fetching Data
        ⏳⏳⏳";
          let body = {
            district: document.getElementById("districtName").value,
            season: document.getElementById("season").value,
```

```
    year: document.getElementById("year").value,
    crop: document.getElementById("crop").value,
  };
  if (document.getElementById("stateName").value == "Select") {
    document.getElementById("results").innerHTML = "Select a State";
    return;
  }
  if (body.district == "Select") {
    document.getElementById("results").innerHTML = "Select a Ditrict";
    return;
  }
  if (body.season == "Select") {
    document.getElementById("results").innerHTML = "Select a Season ";
    return;
  }
  if (body.year == "") {
    document.getElementById("results").innerHTML = "Fill a Year ";
    return;
  }
  fetch(new URL(document.URL) + "predict", {
    method: "POST",
    mode: "same-origin",
    cache: "no-cache", // *default, no-cache, reload, force-cache,
only-if-cached
    credentials: "same-origin", // include, *same-origin, omit
    headers: {
      "Content-Type": "application/json",
    },
    redirect: "follow", // manual, *follow, error
    referrerPolicy: "no-referrer", // no-referrer, *no-referrer-when-downgrade,
origin, origin-when-cross-origin, same-origin, strict-origin,
strict-origin-when-cross-origin, unsafe-url
    body: JSON.stringify(body), // body data type must match "Content-Type"
```

```
header
  })
    .then((res) => {
      if (!res.ok) throw "response staus is not ok " + res.status;
      return res.json();
    })
    .then((data) => {
      let crops = data.result;
      let htmlString = "<table><tr><th>Crop</th><th>Yield</th></tr>";
      for (let i = 0; i < crops.length; i++) {
        htmlString +=
          "<tr><td>" +
          crops[i][0] +
          "</td><td>" +
          crops[i][1].toFixed(2) +
          "</td></tr>";
      }
      htmlString += "</table>";
      document.getElementById("results").innerHTML = htmlString;
      return;
    })
    .catch((error) => {
      console.log("error while prediction " + error.data);
      document.getElementById("results").innerHTML =
        "Something Went Wrong 😕😕😕";
    });
}
document.getElementById("districtName").innerHTML =
  "<option value='Select'>Select A State</option>";
function List2PicklistValue(list) {
  result = "<option value='Select'>Select</option>";
  for (idx in list) {
```

```
    result += "<option value='" + list[idx] + "'>" + list[idx] + "</option>\n";
  }
  return result;
}
 function stateSelected() {
  let state = document.getElementById("stateName").value;
  if (state == "Select") {
    document.getElementById("districtName").innerHTML =
      "<option value='Select'>Select A State</option>";
  } else {
    document.getElementById("districtName").innerHTML =
List2PicklistValue(
      stateDistrictMap[state]
    );
  }
}
```

Python code:

```
 import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import os
class model:
    def __init__(self):
        self.regressor = None
        self.params = {}
        self.labels_dict = {}
        self.scaling_dict = {}
        self.df = None
def deviationTransform(self, arr):
```

```
        d = int(np.std(arr))
        return [0, d]
    def minMaxTransform(self, arr):
        min = int(np.min(arr))
        max = int(np.max(arr))
        return [min, max-min]
    def getData(self, fileName):
        self.df = pd.read_csv(fileName)
      def preProcessData(self):
        if(os.getenv('PY_ENV') != None):
            self.df = self.df.sample(n=1000)
        self.df = self.df[self.df['Area'] > 0]
        self.df = self.df[self.df['Production'] > 0]
        self.df["ProductionPerArea"] = (
            (self.df["Production"])/(self.df["Area"]))
        # replace empty strings with nan
        self.df = self.df.replace(r'^\s*$', np.NaN, regex=True)
        # drop null values
        self.df = self.df.dropna()
        p1 = np.percentile(np.array(self.df['ProductionPerArea']), 25)
        p2 = np.percentile(np.array(self.df['ProductionPerArea']), 99)
        self.df = self.df[self.df['ProductionPerArea'] > p1]
        self.df = self.df[self.df['ProductionPerArea'] < p2]
        self.labels_dict['State_District_Map'] = {}
        for _, row in self.df.iterrows():
            key = row['State_Name']
            value = row['District_Name']
            if key in self.labels_dict['State_District_Map']:
                self.labels_dict['State_District_Map'][key].add(
                    row['District_Name'])
            else:
                self.labels_dict['State_District_Map'][key] = set(
```

```
                    [row['District_Name']])
            for key, value in self.labels_dict['State_District_Map'].items():
                self.labels_dict['State_District_Map'][key] = list(value)
            self.df = self.df.drop(columns=['State_Name', 'Area', 'Production'])
        def encodeAndNormalizeData(self):
            categorical_columns = ['District_Name', 'Crop', 'Season']
            for column in categorical_columns:
                le = LabelEncoder()
                le.fit(self.df[column])
                self.df[column] = le.transform(self.df[column])
                self.labels_dict[column] = list(le.classes_)
                scaling_params = self.minMaxTransform(np.array(self.df[column]))
                self.df[column] = (self.df[column] -
                                   scaling_params[0])/scaling_params[1]
                self.scaling_dict[column] = scaling_params
            scaling_params = self.minMaxTransform(np.array(self.df['Crop_Year']))
            self.scaling_dict['Crop_Year'] = scaling_params
            self.df['Crop_Year'] = (self.df['Crop_Year'] -
                                    scaling_params[0])/scaling_params[1]
            scaling_params = self.deviationTransform(
                np.array(self.df['ProductionPerArea']))
            self.df['ProductionPerArea'] = (
                self.df['ProductionPerArea'] - scaling_params[0])/scaling_params[1]
            self.scaling_dict['ProductionPerArea'] = scaling_params
        def classify(self, x_train, x_test, y_train, y_test):
            self.regressor.fit(x_train, y_train)
            y_pred = self.regressor.predict(x_test)
            y_train_pred = self.regressor.predict(x_train)
            print(r2_score(y_train, y_train_pred))
            print(r2_score(y_test, y_pred))
    def dumpData(self):
            json_params = {}
```

```
            json_params['labels'] = self.labels_dict
            json_params['scaling'] = self.scaling_dict
            self.params = json_params
    def predict(self, district, season, year, crop):
            if(not year or year < 2000 or year > 2099):
                return [['Year needs to between 2000 and 2099', '']]
            district_value = self.params['labels']['District_Name'].index(district)
            district_params = self.params['scaling']['District_Name']
            district_scaled = (
                district_value - district_params[0])/district_params[1]
            season_value = self.params['labels']['Season'].index(season)
            season_params = self.params['scaling']['Season']
            season_scaled = (season_value -
        season_params[0])/season_params[1]
            year_params = self.params['scaling']['Crop_Year']
            year_scaled = (year - year_params[0])/year_params[1]
            crop_params = self.params['scaling']['Crop']
            production_params = self.params['scaling']['ProductionPerArea']
            predictions = []
            crops_predicted = []
            if(crop == '' or crop == 'Select'):
                crops = self.params['labels']['Crop']
                for index, _ in enumerate(self.params['labels']['Crop']):
                    crop_scaled = (index - crop_params[0])/crop_params[1]
                    dfPred = pd.DataFrame(
                        [[district_scaled, year_scaled, season_scaled, crop_scaled]],
        columns=['District_Name', 'Crop_Year', 'Season', 'Crop'])
                    predictions.append(self.regressor.predict(
                        (dfPred)))
                    crops_predicted.append(
                        [crops[index],
        predictions[index][0]*production_params[1]+production_params[0]])
                return crops_predicted, 200
```

```
        else:
            crop_value = self.params['labels']['Crop'].index(crop)
            crop_scaled = (crop_value - crop_params[0])/crop_params[1]
            dfPred = pd.DataFrame(
                [[district_scaled, year_scaled, season_scaled, crop_scaled]],
columns=['District_Name', 'Crop_Year', 'Season', 'Crop'])
            production_pred = self.regressor.predict(dfPred)
            return [[crop,
production_pred[0]*production_params[1]+production_params[0]]], 200
    def main(self):
        try:
            self.getData('dataworld_set.csv')
            self.preProcessData()
            self.encodeAndNormalizeData()
            x_train, x_test, y_train, y_test = train_test_split(
                self.df.iloc[:, :-1], self.df.iloc[:, -1], test_size=0.33,
random_state=42)
            self.regressor = RandomForestRegressor(
                n_estimators=10, random_state=0)
            print("\t\t\t random-forest classifier")
            self.classify(x_train, x_test, y_train, y_test)
            self.dumpData()
        except Exception as err:
            print("error running model.py main method ", type(err), err)
    if __name__ == '__main__':
      mode = model()
      mode.main()
```

Flask:

```
import json
from flask import Flask, render_template, make_response, request
from model import model
import os
app = Flask(__name__)
```

```
    classParams = {}
    port = int(os.environ.get('PORT', 5000))
@app.route('/', methods=['GET'])
    def index():
        return render_template('index.html')
@app.route('/stateDistrictMap', methods=['GET'])
    def getStateDistrictMap():
        result = clf.params['labels']['State_District_Map']
        return {'result': result}
@app.route('/options')
    def getOptions():
        result = '<option value=\'Select\'>Select</option>'
        for option in clf.params['labels'][request.args.get('param')]:
            result += '<option value=\'{option}\'>{option}</option>'.format(
                option=option)
        return {'result': result}
    @app.route('/predict', methods=['GET', 'POST'])
    def predict():
        body = json.loads(request.data)
        response = {}
        try:
            response = clf.predict(
                body['district'], body['season'], int(body['year']), body['crop'])
            return make_response({"result": response[0]}, response[1])
        except Exception as err:
            print("while predicting error has occured " + str(err))
            return make_response("error while prediction", 500)
    clf = model()
    clf.main()
    app.run(host='0.0.0.0', port=port, debug=False)
```

B) SCREEN SHOTS

Fig B.1 Output When Crop name is not given

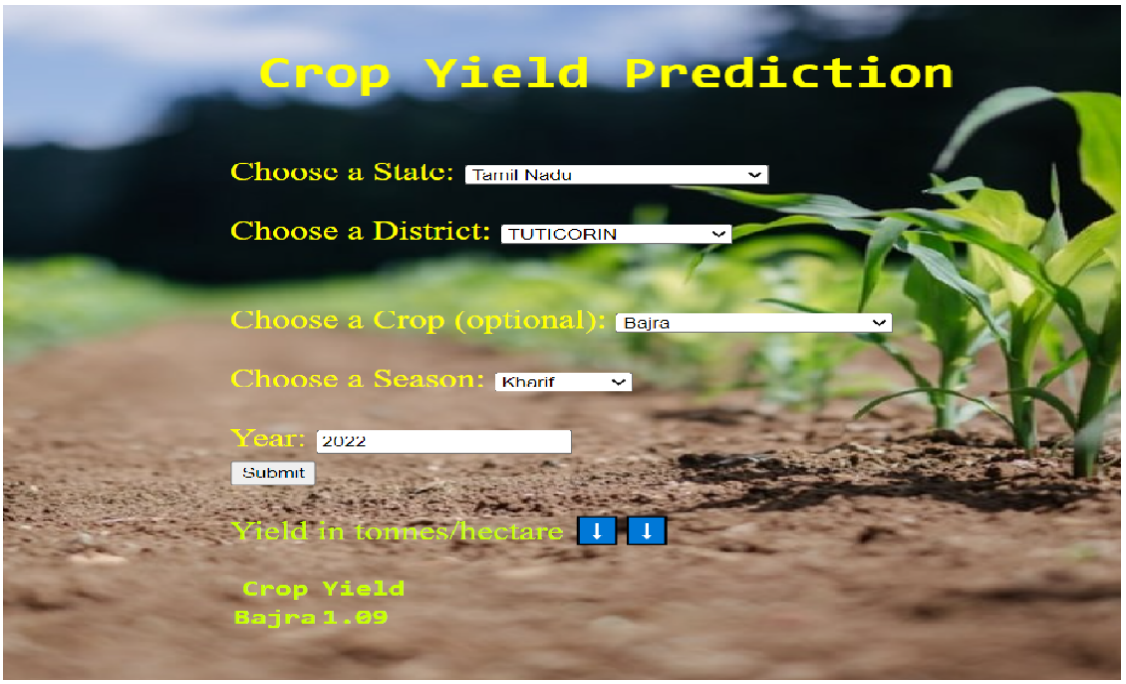


Fig B.2 Output When Crop Name is Given

C) PLAGARISM REPORT

PREDICTION OF CROP YIELD USING MACHINE LEARNING.docx

ORIGINALITY REPORT

9%	4%	5%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	towardsdatascience.com Internet Source	2%
2	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	2%
3	Potnuru Sai Nishant, Pinapa Sai Venkat, Bollu Lakshmi Avinash, B. Jabber. "Crop Yield Prediction based on Indian Agriculture using Machine Learning", 2020 International Conference for Emerging Technology (INCET), 2020 Publication	1%
4	ijsrcseit.com Internet Source	1%
5	Anand Kumar Pandey, B. Ankayarkanni. "Recommending E-Commerce Products on Cold Start and Long Tail Using Transaction Data", 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI) (48184), 2020 Publication	1%

D) PAPER

PREDICTION OF CROP YIELD USING MACHINE LEARNING APPROACH

Sreenidhi Ch, Sowmya U,
UG Student, Department of CSE,
Sathyabama Institute of Science and Technology, Chennai, India.
sreenidhi6677@gmail.com,
sowmyau0713@gmail.com

B.Ankayarkanni
Assistant professor, Department of CSE ,
Sathyabama Institute of Science and Technology, Chennai, India.

ankayarkanni.s@gmail.com

AB***STRACT: Among around the globe, agribusiness has the huge commitment in regards to improving the financial responsibility of the country. In any case, regardless the most cultivating fields are youthful in light of the shortfall of association of organic framework control developments. Due to these issues, the yield creation isn't improved which impacts the cultivating economy. From this time forward an improvement of cultivating benefit is redesigned reliant on the plant yield assumption. To thwart this issue, Agricultural zones need to anticipate the yield from given dataset using ML methodology. The examination of dataset by coordinated ML techniques. In this we are going to predict the yield if a specific crop is selected else, we will predict the yield of all the crops using the parameters District name, season and year.***

Keywords: dataset, Machine Learning-Regression methods, mean absolute error, R2-score

I INTRODUCTION

In our research, we found that most of previous papers used climatic factors like rainfall, sunlight. When focused based on soil they used parameters like soil type, soil PH, crop sensitivity etc., Regarding rice crop prediction they used parameters like soil, sunshine, fertilizer, temperature, paddy rainfall and pest. As these terms are bit difficult to explain to the farmers because he may not be able to understand these terms so to make this process easy, we are making an application which uses parameters like district name, season and year and it predicts the crop yield this make farmers easy to use the application as the terms are easy.

Normally Data Scientists utilize various types of Machine Learning calculations to the enormous informational collections. In India, we have more than hundreds of crops planted around the whole country. The data for this research has been taken from a site called data world. The data set contains parameters like state name, district name, crop, season, year, area and production

Agriculture is the foundation of each economy. Agricultural scientists in elective nations resulted those trails made for crop yield augmentation through favorable to pesticide state approaches resulted in unsafely high substance use. These examinations have announced a connection between substance use and crop yield

Agribusiness is partner exchange area that is profiting intensely from the occasion of finder innovation, information science, and AI (ML) methods inside the most recent years. These improvements get back to fulfill ecological and populace pressures round-looked by our general public, any place reports show a necessity for powerful worldwide agribusiness yield increment to create nourishment for a developing populace on a more sweltering planet. The vast majority of the work tired the area of yield predicting through cubic centimeter utilizes some sort of far-off detecting information over the homestead. Agribusiness looks to broaden and improve the crop yield and hence the nature of the yields to support human existence. Nonetheless, inside the current time, people will in general require a great deal of like a shot appreciated positions. There are less, and less people worried in crop development. moreover, the consistent increment of human populace makes the development of the yields at the legitimate time and opportune spot even a great deal of crucial, in light of the fact that the environment is dynamic and accordingly the movements from conventional climate design are a ton of continuous than before make. The data hole between antiquated ways that of developing and new agrarian advancements might be survived if the PC code might be intended to show the intelligent effect of environment factors, especially the effect of greatest occasions (for example warmth, rainfalls and overabundance water) happening at totally extraordinary developing periods of crops. The temperature change without a doubt influences the local and world food creation, along these lines arranging PC code to show crop forecasts needs new strategy for temperature change examines, circumstances for temperature change transformation, and policymakers which will restrict the overwhelming impacts of climate on food give. The dirt sort will adjustment after some time due to climate and vermin, consequently crop the executives should deal with an extravagant amount of data, straightforwardly or by implication related with each other. It will along these lines by thinking about a worked on the real world, to allow a brisk appraisal of the effect of temperature change in agribusiness.

Farming ought to adjust to those environment changes, and it will do accordingly by creating models which will in principle enhance the executives rehearses, augment the turns of the new yield to deal with the progressions of soil, novel rearing projects. By boosting the value of anticipating, the occasional environment changes might be discovered and recorded in an incredibly convenient way. Later on, by exploitation PC code upheld AI, one will conveniently evaluate the temperature change effect and check achievable circumstances that consolidate found out changes in climatic conditions and water dispersion. information {processing} is that the way toward dissecting the test information gathered over a sum and changed areas from totally various perspectives, separate patterns or examples {of information of information on info} and switch them into supportive data for clients. The examples, affiliations, or connections among this information will extra be reawakened into data that is offered to the client as recorded examples and future patterns. This data given by AI will encourage ranchers with crop development by foreseeing probabilities of crop misfortunes or stop misfortunes out and out.

Yield forecast benefits the ranchers in diminishing their misfortunes and to get best costs for their harvests. Agriculture is most significant occupation drilled in our country. It is the biggest financial area and assumes a significant part in by and large improvement of the country. Determination of harvest is

significant issue in horticultural arranging. Yield creation rate relies upon topography of an area, climate condition, soil type and gathering techniques. Various subsets of these impacting boundaries are utilized for various yields by various expectation models. AI strategy is utilized for expectation of harvest yield utilizing various calculations. AI strategies which are broadly utilized in expectation strategy are relapse tree, irregular timberland, convolution neural organization and K-closest calculation. There is consistently a huge danger factor for the ranchers to choose which specific yield ought to develop during season, on specific land asset. It relies upon various boundaries, for example, creation rate, cost and diverse government approaches. Independent of the capital put as far as soil, water and nature of seeds of the yield rate creation the harvest may bomb carrying shocking misfortunes to the rancher and his family

In this pre-owned AI approach with created in harvest or plant yield expectation since horticulture has diverse information like soil information, crop information, and climate information. Plant development forecast is proposed for checking the plant yield viably through the AI methods.

II RELATED WORK

Virendra Panpatil et al[1] had made huge work for Indian farms by building profitable yield recommendation schema.

The planned schema is used find the best season for planting, advancement of plant and Plant fulfillment. They used discrete classifier for generating better correctness for occurrence. The best agreeable area of framework that it can without much of a extent all-around all things be used to check on various yields.

Mayank et al[2] created improvised schema for crop yield using executed AI computations and with target to give effortless to use User Interface, rise the exactness of crop yield estimate, examining discrete climatic parameters.

Zhihao et al[3] two relapse administered AI strategies SVM, RVM to show viability in soil quality forecast. a shrewd remote gadget for detecting soil dampness and meteorological information.

Sabri Arik et al[4] includes a analysis regarding Soil Fertility , Plant Nutrient by utilizing back spread computation . The outcomes are exact and empowers advancement in soil properties.

It works best when contrasted with conventional techniques. Be that as it may, framework is moderate wasteful and not steady.

Shivnath et al[5] proposed about BPN to assess the test informational collection. BPN utilizes a concealed layer which supports in better execution in foreseeing soil properties. BPN present, is utilized to build up a self-prepared capacity to foresee soil properties with boundaries.

This results in more exactness and executes better compared to the customarily utilized strategies, in any case, at times the framework turns be moderate and irregularity is found in the yield.

Raval et al[6] examine regarding the Knowledge Discovery Process and the rudiments of different Data Mining approaches, for example, Association rules, Classification etc.,

Agrawal et al examine regarding different Data Mining devices, for example, Dashboards, Text-Mining instruments. They give an outline about these apparatuses and the different situations where they can be conveyed.

Grajales et al[7] recommended a web app that uses open dataset like verifiable creation, land cover, nearby environment surroundings and coordinates them to give simple admittance to the ranchers. The suggested

engineering basically centers around open-source devices for the advancement of the application. The client can choose area for which the subtleties are accessible at a single tick.

Bendre et al[8] gathers information about GIS, GPS, VRT and RS are controlled utilizing Map Reduce calculation and direct relapse calculation to figure the climate information that can be utilized in accuracy agribusiness.

Verma, A. et al[9] used classification techniques for crop prediction like Naïve Bayes, KNN algorithm on soil datasets which consists of nutrients of soil like zinc, Phosphorous , copper , pH, iron, Sulphur, manganese, Organic Carbon, nitrogen, and potassium

Chakrabarty, A. et al[10] made crop prediction in Bangladesh. They considered parameters like soil composition, type of fertilizer, type of soil and its structure, soil consistency, reaction and texture

III PROPOSED SYSTEM

In this proposed work we need to load the information, check for null and duplicate values, and then we trim and clean our dataset for examination. We should make sure that we record our means carefully and legitimize your cleaning preferences. This is the most energizing stage in Applying Machine Learning to any Dataset. It is otherwise called Algorithm choice for Predicting the best outcomes. Usually, Data Scientists use various kinds of ML calculations to the huge data collections. Yet, at significant level every one of those various calculations can be of these two types of gatherings: regulated and solo learning. Managed learning: supervised learning is a schema where the necessary information and needed yield information is also given. Info and yield information are used for future data handling. Learning issues can be of two types Regression and Classification issues

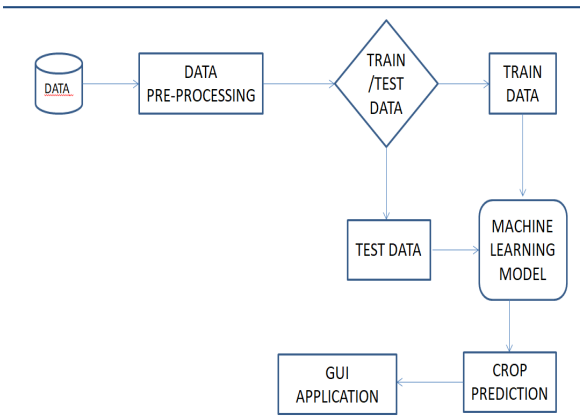


Fig. Architecture of proposed Model

3.1 TRAIN THE DATASET

At First, we import all the required modules. Then we use appropriate cleaning methods to clean our data. Then we split our data according to the required output. We split our dataset into train and test data using train_test_split technique. The X prefix indicates the element esteems and y prefix indicates target esteems. This splits the dataset into train and test information in the proportion which we mentioned. Here we are using a proportion of 80:20. At this point we epitomize any calculation. Here we fit our preparation information into this calculation so our system can get prepared utilizing this information. Here the training part is finished.

3.2 TEST THE DATASET

To get confidence in our trained model we use some data of the dataset as test data. Here we have divided the dataset into train and test data in 80:20 proportion. We predict the yield values with the trained model by sending the feature values from testing data. We now compare the predicted values with the actual values using metrics such as R2 score, mean squared error etc. these metrics tell us how close we are able to predict the actual data. Mean squared the low the better. R2 score the higher the better.r2 score calculates the

correlation between predicted and actual values.

IV MODULES DESCRIPTION

4.1 DATA VALIDATION AND CLEANING

4.1.1 DATA SET DESCRIPTION

We took our dataset from dataworld website. dataset has more than 250000 data points. As machine learning models work better when we have more data this dataset having large data is very useful. Dataset has the following attributes state, district, area, production, season, year.

4.1.2 DATA CLEANING

From the available attributes we drop the state attribute as we will use district. As state is dependent on the district, we don't need it for training. We are dropping rows which have negative values for area and production as these are supposed to positive and hence are inconsistent. We also drop any na,nan and empty values before training as these rows could disrupt training. For the target of our training model, we use a derived attribute production Per Area as we are interested in yield per area. Hence, we drop production and area as these are included in our derived target attribute Production Per Area.

To get confidence in our trained model we use some data of the dataset as test data. Here we have divided the dataset into train and test data in 67:33 proportion. We predict the yield values with the trained model by sending the feature values from testing data. We now compare the predicted values with the actual values using metrics such as R2 score, mean squared error etc. these metrics tell us how close we are able to predict the actual data. Mean squared the low the better. R2 score the higher the better.r2 score calculates the correlation between predicted and actual values.

4.2 DATA NORMALIZATION

4.2.1 PREPROCESSING AND SCALING

To train data with string datatype we use label encoder to convert them to numeric data. Label encoding simply assigns each data point a number starting from .the attributes we are label encoding is district name, season and crop. Then to scale each data attribute we use two types of scaling techniques which are MinMax transform and normalization. The attributes district name, season, crop is scaled by MinMax transform and production per area is scaled by normalization. MinMax transform is done by subtracting each datapoint from minimum among datapoints then dividing by maximum- minimum.

normalization is done by dividing datapoint with standard deviation. We choose normalization to productionperarea as many data points are low and min max transform is yielding good results.

4.3 RANDOM FOREST

It is a very famous ML algorithm that felicitates in cases of both classification and regression issues. This algorithm works on the notion of ensemble learning, which works on the principle of merging several classifiers to give the solution for any tough problems and increase the precision and performance of the applied model. Random Forest is a classifier or regressor that contains multiple decision trees on several subsets of a given dataset and considers the average to improve the dataset's high accuracy in case of classification problems in case of regression problems it helps us to lower the error value. In other words, rather than depending on a single decision tree, this algorithm considers predictions from every tree and predicts the final result based on the predictions. On observation we can say that:

$$No.\ of\ Trees\ in\ forest \propto Accuracy\ of\ the\ model$$

4.4 DECISION TREE

It is a very famous ML algorithm that is used in cases of both classification and regression issues. It consists of three nodes. The First node which is the initial node is the root node. The Interior nodes describes the features of the data set where as the branches represent the decision rules. Finally, the leaf node shows the result.

4.5 K-NEAREST NEIGHBOR (KNN)

It is a regulated AI figuring which keeps all events identify with planning data centers in n-dimensional spaces. It separates the closest k number of cases saved and gives the most broadly perceived class as the assumption and for authentic regarded data it reestablishes the mean of k nearest neighbors. In case of distance weighted nearest neighbor it stacks the responsibility of all of the k neighbors based on their distance using the going with request giving more critical burden to the closest neighbors. Makes assumptions regarding the endorsement set using the entire getting ready set. KNN makes an estimate about another case by means of glancing through the entire set to find the k "closest" events. "Closeness" is settled using a proximity assessment across all features.

4.6. LASSO REGRESSION

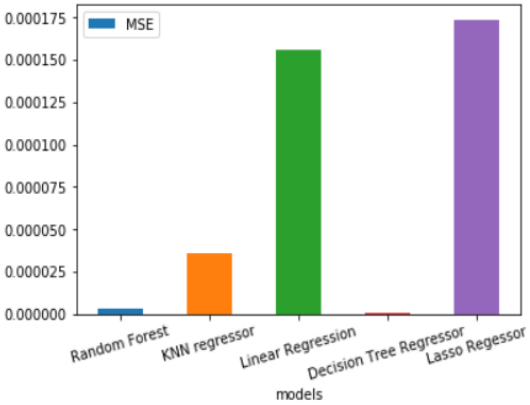
Lasso regression a kind of linear regression that uses depreciation, i.e., data points are diminished towards a mid-point, as done when finding the mean. This model is particularly useful and suits well for models that show high levels of multicollinearity as is the case in this particular dataset. It executes L1 regularization. Lasso solutions are quadratic programming problems, that use the following formula: $\Sigma(yi - \Sigma xij\beta j)2+ \lambda\Sigma|\beta j|pj=1\ jni=1$

The intensity of the L1 penalty is controlled by a tuning parameter. is essentially the amount of shrinking. If it is zero, we know that all features are taken into account, and it is equivalent to linear regression, in which just the residual sum of squares is used to form a predictive model. If it is infinity, it means that no features are taken into account. The bias increases while the variance decreases with an increase in λ, and vice-versa

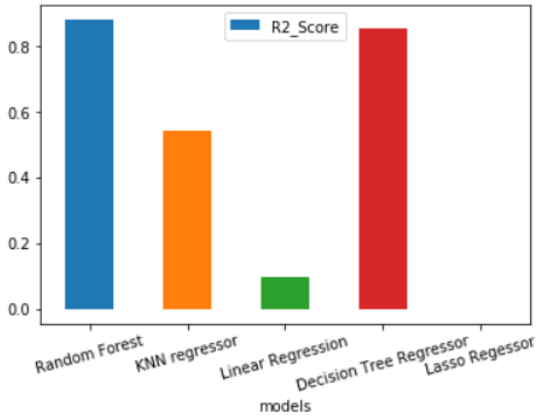
V Result and Discussion

	MSE	MAE	RMSE	R2_Score	models
0	3.426823e-06	0.001505	0.004558	0.880158	Random Forest
1	3.595377e-05	0.003637	0.008913	0.541729	KNN regressor
2	1.557740e-04	0.006636	0.012498	0.098844	Linear Regression
3	1.761657e-07	0.001490	0.005049	0.852930	Decision Tree Regressor
4	1.737817e-04	0.007141	0.013166	-0.000002	Lasso Regessor

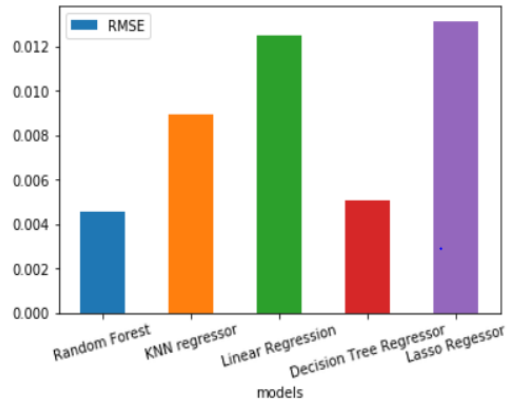
MSE PLOT



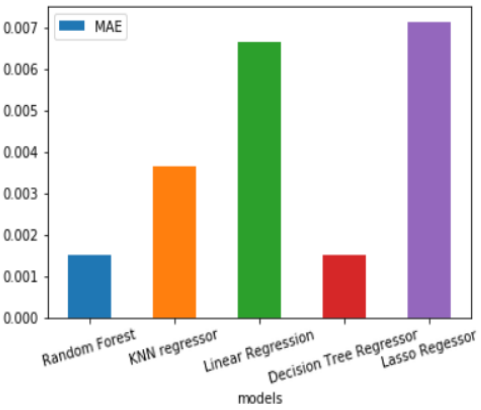
R2_Score plot

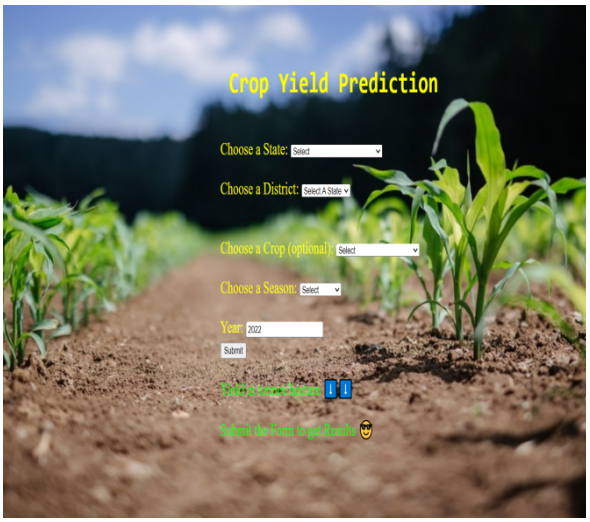


RMSE PLOT



MAE PLOT





VI CONCLUSION

The logical connection started from data cleaning, visualizing the data, preprocessing then training and testing the dataset examination finally model design and evaluation. Finally, we anticipate the yield using ML algorithms with different results. This brings a bit of the going with encounters about yield gauge. As most prominent sorts of yields will be covered under this structure, farmer may turn out to be more familiar with about the collect which may never have been created and penetrates down each possible yield, it helps the farmer in unique of which respect create.

It gives the yield if a specific crop is given else it will give the yield of all crops.

REFERENCES

[1] Crop Data (1997 – 2010) dataworld.com

[2] ML algorithms: Analyticsvidhya.com

[3] javascript codewithhugo.com

[4] Deployment heroku

[5] PavanPalle,Praful Nikam, Abhijeet Pandhe, Vijay Pagare,

Prof. DilipDalgade Crop Yield Prediction based Climatic boundaries 2019.

[6] Vishal Vats, Naveen Kumar, Arun Kumar, crop yield prediction using machine learning algorithms,2018.

[7] Alaslani, Maram Elrefaei, Lamiaa. (2019). Learning

with CNN for IRIS Recognition. International

Journal of Artificial Intelligence Applications. 10. 49-66. 10.5121/ijaia.2019.10505.

[8]Yuan, Jun Ni, Bingbing Kassim, Ashraf. (2014). Half-CNN: A General schema for complete-Image Regression.

[9] Training Recurrent Neural Networks, Ilya Sutskever, PhD Thesis, 2012.

[10] Thomas van Klompenburg,Ayalew Kassahun and Cagatay Catal, “Crop yield prediction using machine learning: A systematic literature review”, Computers and Electronics in Agriculture, Volume 177.

[11] Saeed Khaki,Lizhi Wang and Sotirios V. Archontoulis, CNN-RNN schema for Crop Yield Prediction , Frontiers in Plant Science, v. 10, 2019