PREDICTION OF CROP YIELD AND COST BY FINDING BEST LEARNING USING MACHINE LEARNING APPROCH

A PROJECT REPORT

Submitted by

MUKIL CHOKALINGAM M.[REGISTER NO:211417104154]

NAVEEN PR. [REGISTER NO: 211417104162]

NAVEEN NARAYAN M. [REGISTER NO: 211417104164]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2020

BONAFIDE CERTIFICATE

Certified that this project report **“PREDICTION OF CROP YIELD AND COST BY FINDING BEST LEARNING USING MACHINE LEARNING APPROCH”**

is the bonafide work of “ MUKIL CHOKALINGAM M. (211417104154), NAVEEN NARAYAN M. (211417104164), NAVEEN P R. (211417104162) ” who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. MURUGAVALLI,M.E.,Ph.D.,	**Mr. K.KAJENDRAN, M.C.A., M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	**ASSOCIATE PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mr. K. KAJENDRAN, M.C.A.,M.E.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

MUKIL CHOKALINGAM M.
NAVEEN NARAYAN M.
NAVEEN PR.

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 analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments etc. 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. The results show that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with entropy calculation, precision, Recall, F1 Score, Sensitivity, Specificity.

Keywords: dataset, Machine learning-Classification method

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

S.NO	**ABBREVIATION**	**EXPANSION**
1**.**	ML	Machine Learnning
2.	KNN	K-Nearest Neighbour
3.	SVM	Support Vector Machine
4.	GUI	Graphical User Interface
5.	UI	User Interface
6.	FP	False Positives
7.	FN	False Negatives

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

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 amount 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.

CHAPTER 2

LITERATURE SURVEY

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

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 modeling 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 cutoff 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 nondestructive 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

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

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 vari- ations 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 misjudgment 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

DESCRIPTION: The optimization and imple- mentation 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

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

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

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 favorable in the OC group, which indicates the optimal combination of month can be significant in statistical modeling 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 modeling. 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.

TITLE: Remote Sensing and Geospatial Technological Applications for Site-specific

Management of Fruit and Nut Crops: A Review

DESCRIPTION: Growth observation, impact assessment, and timely strategic response to small variations in crop production is known as precision agriculture. It has been used in a wide range of agricultural activities including field crop production, dairy farming, horticulture, and forest management. Site-specific crop management (SSCM) is one facet of precision agriculture which involves spatial referencing, crop and climate monitoring, attribute mapping, decision support systems, and differential action. SSCM is carried out with a greater degree of precision through the use of geospatial technologies. Geospatial technology is a combination of four essential tools: remote sensing, geographic information systems (GIS), global positioning systems (GPS), and information technology or data management. SSCM has become very common in management of field and row crops during recent years. Yet, its application for non-traditional horticultural crops has not been very widespread. Companies, such as Lanworth (http://lanworth.com/), which gather intelligence on natural resources, use geospatial technologies to map, analyze, and forecast agriculture and forest output. This is based on crop and forest growth parameters which is a large amount of data, terabytes in size. It helps in the SSCM of agricultural and forest products and increases the yield in a sustainable manner. The application of aerial or satellite imaging, along with GPS and GIS, is the first step towards the successful application of SSCM for fruit and nut crops. The use of SSCM for horticultural crops such as fruit and nut crops has potential for increasing net returns and optimizing resource. The delineation of orchards and spatial analysis using geospatial technology can provide additional information for management decision making, such as the determination of fruit yield, the quantification and scheduling of precise and proper fertilizer, irrigation needs, and the application of pesticides for pest and disease management. Ultimately, it will improve profits for producers. The use of remote sensing has become common for the general detection of the growth and health of orchards on a larger scale. However, digital imaging technology is increasingly being used for intensive site-specific management of orchards as well, for instance: estimating the amount of fruits on individual trees, fruit quality, and also leaf area index or crown cover. The overall goal to provide a review of studies that use geospatial technology and especially remote sensing for conducting SSCM in fruit and nut crops, including oranges, peaches, pecans, apples, grapes, blueberries, as well as other crops around the world.

It provides a comprehensive review on the development and application of SSCM for fruit and nut crops. The review also confirmed that not many studies (compared to row crop agriculture) have been conducted so far on fruit and nut crops' SSCM and one-stop of information on available technology for the fruit and nut crop SSCM. The study provides a detailed application of geospatial and information technology in the blueberry crop management with a case study in Georgia, U.S., and in a few locations in southeast Georgia. They can be replicated as described or modified based on local requirements and needs. It provides a detailed analysis of several characteristics responsible for blueberry production and describes, through references, how to develop geospatial models using those characteristics to plan for its management. Similar models can be developed using related crop characteristics to manage the crop.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

It present a crop/weeds classification approach based on a three-steps procedure. The first step is a robust pixel-wise segmentation (i.e., soil/plant) and image patches containing plants are extracted in the second step. The third step, a deep CNN for crop/weed classification is used. The extracted blobs in the masked image containing plants information are fed to a CNN classifier based on a fine-tuned model of VGG-16 exploiting the ability of deep CNN in object classification and to reduce the limitations of CNNs in generalizing when a limited amount of data is available. The classification step can then be specialized to the types of plants needed by the application scenario. It evaluated the complete pipeline, including the first background removal phase and the subsequent classification stage. Experimental results demonstrate that can achieve good classification results on challenging data.

Precision agriculture is gaining increasing attention because of the possible reduction of agricultural inputs (e.g., fertilizers and pesticides) that can be obtained by using high-tech equipment, including robots. To focus on an agricultural robotics system that addresses the weeding problem by means of selective spraying or mechanical removal of the detected weeds. To describe a deep learning based method to allow a robot to perform an accurate weed/crop classification using a sequence of two Convolutional Neural Networks (CNNs) applied to RGB images. The first network, based on encoder-decoder segmentation architecture, performs a pixel wise, plant-type agnostic, segmentation between vegetation and soil that enables to extract a set of connected blobs representing plant instances.

3.2 PROPOSED SYSTEM

In this section of the report, you will load in the data, check for cleanliness, and then trim and clean your dataset for analysis. Make sure that you document your steps carefully and justify your cleaning decisions.

3.3 FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

3.3.1 TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

3.3.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of drastic increase in the expenses. Python is open source and ready available for all users. Since the project is runned in python and jupyter notebook hence is cost efficient.

3.4 SYSTEM CONFIGURATION

3.4.1 HARDWARE CONFIGURATION

Processor	: Pentium IV/III
Hard disk	: minimum 300 GB
RAM	: minimum 4 GB

3.4.2 SOFTWARE CONFIGURATION

Operating System	: Windows
Tool	: Anaconda with Jupyter Notebook

3.5 SOFTWARE SPECIFICATION

3.5.1 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.Here in this thesis, we are providing basic info of the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning.

3.6 SUPERVISED LEARNING

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data.Let's explore these methods in more detail.The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output 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 linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics. Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have 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 regression and categorical for classification.

3.6.1CLASSIFICATION

As the name suggests, Classification is the task of “classifying things” into sub-categories. But, by a machine. If that doesn’t sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato.Between an A grade and a F grade.In Machine Learning and Statistics, Classification is the problem of identifying to which of a set of categories (sub populations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

TYPES OF CLASSIFICATION

Classification is of two types:

- **Binary Classification:** When we have to categorize given data into 2 distinct classes. Example – On the basis of given health conditions of a person, we have to determine whether the person has a certain disease or not.

- **Multiclass Classification:** The number of classes is more than 2. For Example

On the basis of data about different species of flowers, we have to determine which specie our observation belongs to

Fig 2: Binary and Multiclass Classification. Here x1 and x2 are our variables upon which the class is predicted. Suppose we have to predict whether a given patient has a certain disease or not, on the basis of 3 variables, called features. Which means there are two possible outcomes:

1. The patient has the said disease. Basically a result labeled “Yes” or “True”.

2. The patient is disease free. A result labeled “No” or “False”.

This is a binary classification problem. We have a set of observations called training data set, which comprises of sample data with actual classification results. We train a model, called Classifier on this data set, and use that model to predict whether a certain patient will have the

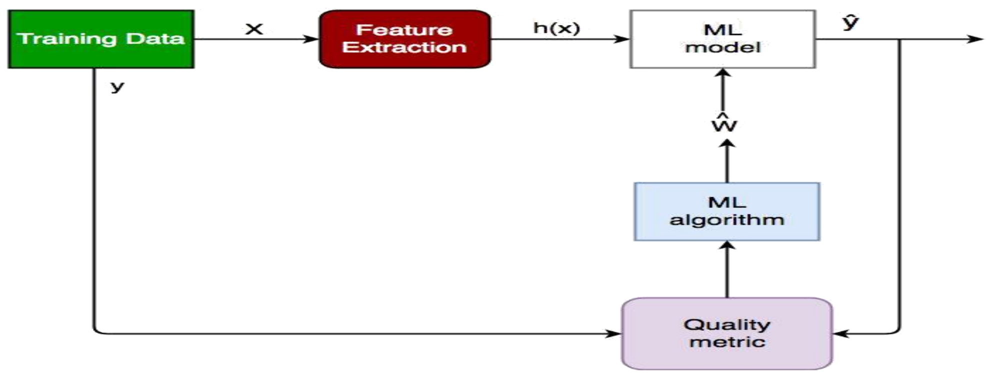


Fig 3.6.1.1 Generalized Classification Block Diagram.

1. X: pre-classified data, in the form of a N*M matrix. N is the no. of observations and M is the number of features

2. Y: An N-d vector corresponding to predicted classes for each of the N observations.

3. Feature Extraction: Extracting valuable information from input X using a series of transforms.

4. ML Model: The “Classifier” we’ll train.

5. y’ : Labels predicted by the Classifier.

6. Quality Metric: Metric used for measuring the performance of the model.

7. ML Algorithm: The algorithm that is used to update weights w’, which update the model and “learns” iteratively.

Types of Classifiers (Algorithms)

There are various types of classifiers. Some of them are :

- Linear Classifiers: Logistic Regression

- Tree Based Classifiers: Decision Tree Classifier

- Support Vector Machines
- Artificial Neural Networks
- Bayesian Regression
- Gaussian Naive Bayes Classifiers
- Stochastic Gradient Descent (SGD) Classifier
- Ensemble Methods: Random Forests, AdaBoost, Bagging Classifier, Voting Classifier, Extra Trees Classifier

Practical Applications of Classification

- Google's self driving car uses deep learning enabled classification techniques which enables it to detect and classify obstacles.
- Spam E-mail filtering is one of the most widespread and well recognized uses of Classification techniques.
- Detecting Health Problems, Facial Recognition, Speech Recognition, Object Detection, Sentiment Analysis all use Classification at their core.

3.7 REGRESSION

A **regression** problem is when the output variable is a real or continuous value, such as "salary" or "weight". A **classification** problem is when the output variable is a category like filtering emails "spam" or "not spam"

3.8 UNSUPERVISED LEARNING

Unsupervised learning is the algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. In our dataset we have the outcome variable or Dependent variable i.e Y having only two set of values, either M (Malign) or B(Benign). So we will use Classification algorithm of supervised learning.

3.8.1 CLUSTERING

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples. Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.For example, the data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers.Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

3.8.2CLUSTERING METHODS:

1. **Density-Based Methods:** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Example DBSCAN

(Density-Based Spatial Clustering of Applications with Noise) , OPTICS (Ordering Points to Identify Clustering Structure) etc.

2. **Hierarchical Based Methods:** The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two categories

• Agglomerative (bottom up approach)• Divisive (top down approach) .

3. **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon randomized Search) etc.

4. **Grid-based Methods:** In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example

STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest) etc.

CLUSTERING ALGORITHMS:

- K-Means Clustering.
- Mean-Shift Clustering for a single sliding window.
- The entire process of Mean-Shift Clustering.
- DBSCAN Smiley Face Clustering.
- EM Clustering using GMMs.
- Agglomerative Hierarchical Clustering.

CHAPTER 4

ARCHITECTURE

4.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

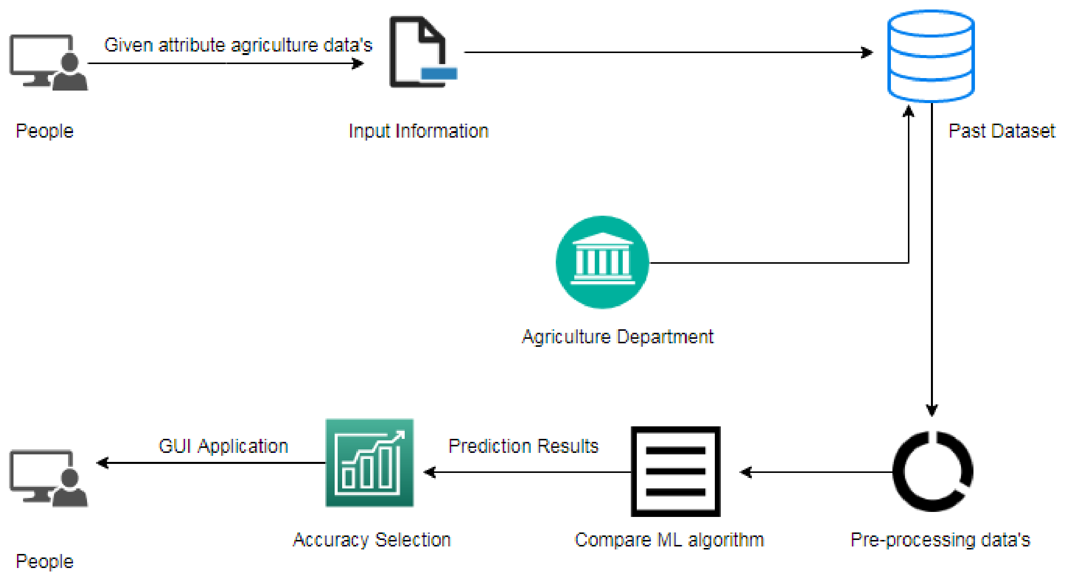


Fig 4.1.1 Architecture Diagram

4.2 UML DIAGRAMS

4.2.1 USE CASE DIAGRAM

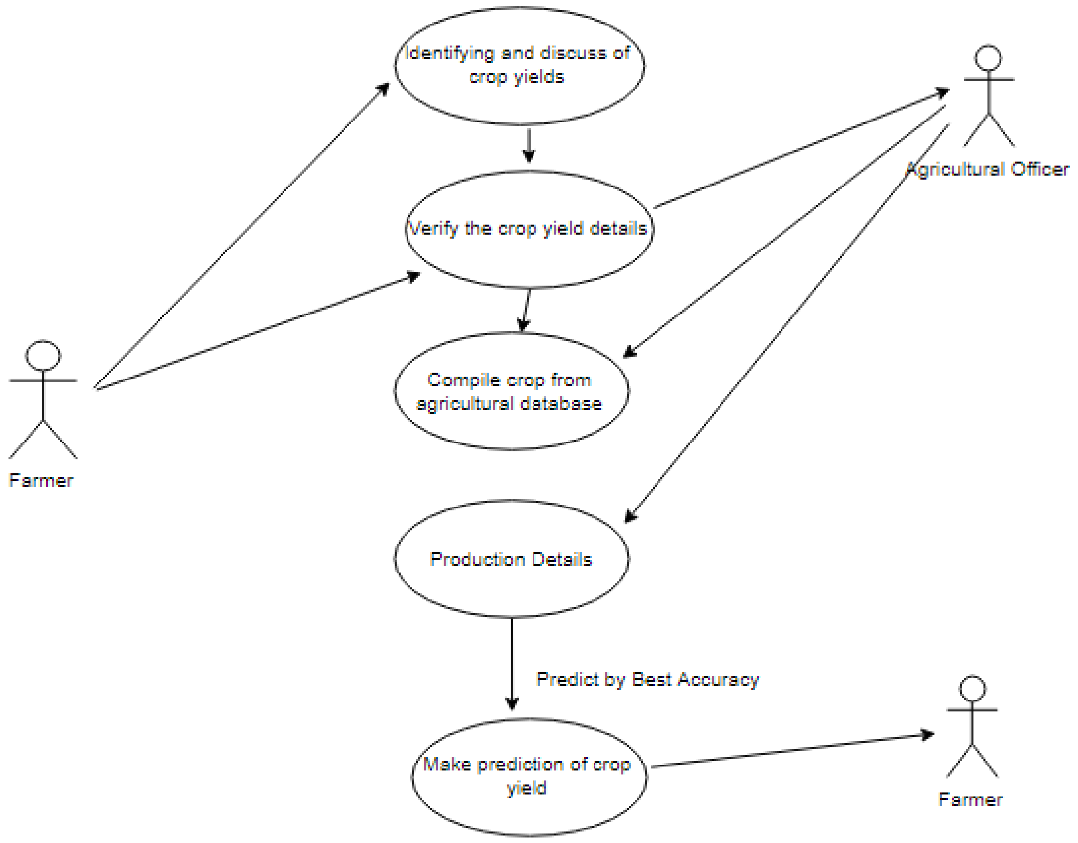
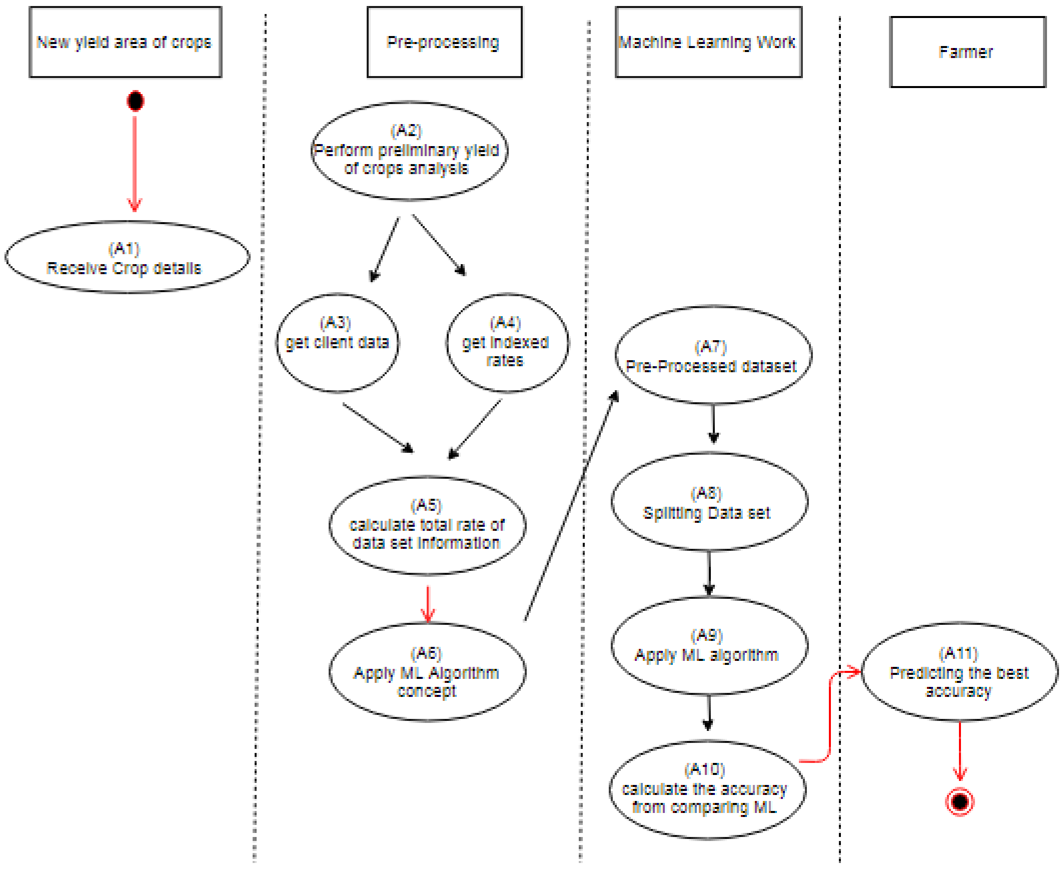

Identifying and discuss of crop yields
Verify the crop yield details
Compile crop from agricultural database
Production Details
Predict by Best Accuracy
Make prediction of crop yield
Agricultural Officer
Farmer
Farmer

Fig 4.2.1 Use Case Diagram

4.2.2 ACTIVITY DIAGRAM



New yield area of crops
Pre-processing
Machine Learning Work
Farmer
(A2) Perform preliminary yield of crops analysis
(A1) Receive Crop details
(A3) get client data
(A4) get indexed rates
(A7) Pre-Processed dataset
(A5) calculate total rate of data set information
(A8) Splitting Data set
(A6) Apply ML Algorithm concept
(A9) Apply ML algorithm
(A11) Predicting the best accuracy
(A10) calculate the accuracy from comparing ML

Fig 4.2.2 Activity Diagram

4.2.3 SEQUENCE DIAGRAM

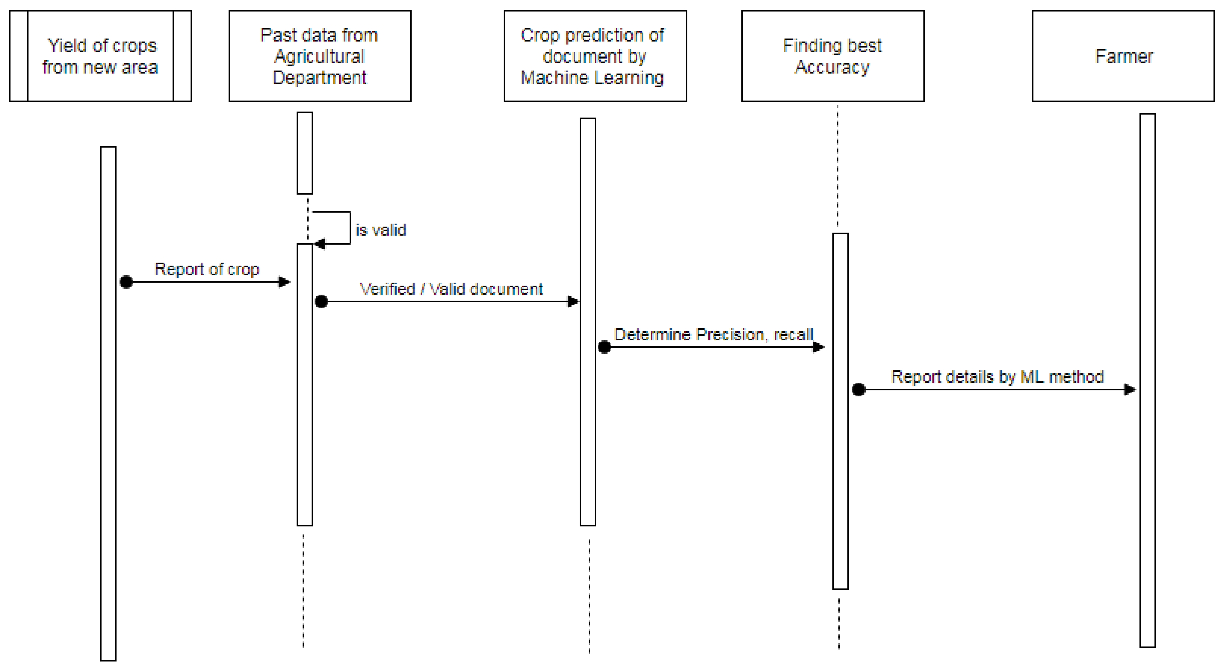

Yield of crops from new area
Past data from Agricultural Department
Crop prediction of document by Machine Learning
Finding best Accuracy
Farmer
is valid
Report of crop
Verified / Valid document
Determine Precision, recall
Report details by ML method

Fig 4.2.3 Sequence Diagram

CHAPTER 5

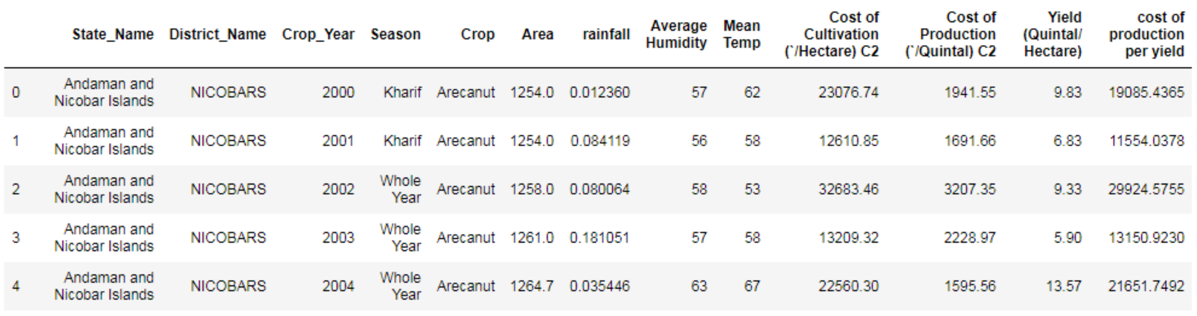
SYSTEM MODULE

5.1 MODULE

- Data validation and pre-processing technique (Module-01)
- Exploration data analysis of visualization and training a model by given attributes (Module-02)
- Performance measurements of logistic regression and decision tree algorithms (Module-03)
- Performance measurements of Support vector classifier and Random forest (Module-04)
- Performance measurements of KNN and Naive Bayes (Module-05)
- GUI based prediction of crop yield and yield cost (Module-06)

5.2 MODULE DESCRIPTION

5.2.1 **VARIBLE IDENTIFICATION PROCESS /DATA VALIDATION PROCESS:** 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 analyzed by regression algorithms; classification algorithms can be used to analyze discrete data. (For example to show the data type format of given dataset)

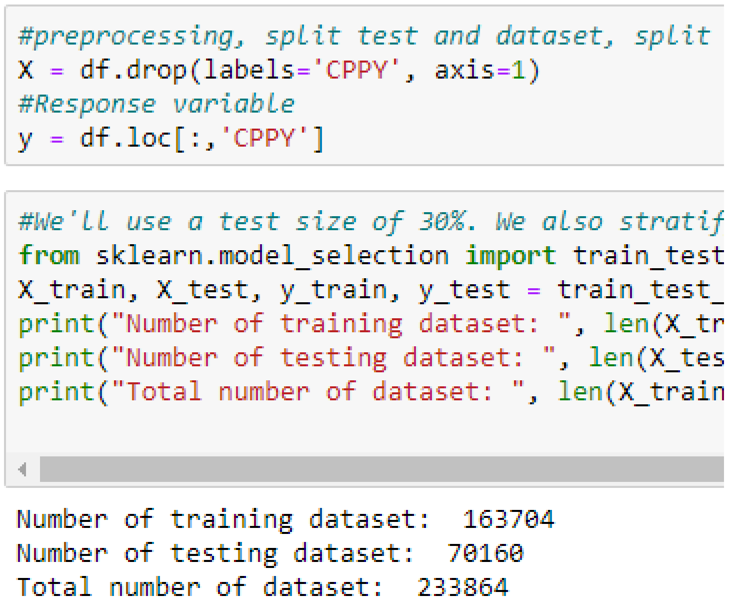


	State_Name	District_Name	Crop_Year	Season	Crop	Area	rainfall	Average Humidity	Mean Temp	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)	cost of production per yield
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	0.012360	57	62	23076.74	1941.55	9.83	19085.4365
1	Andaman and Nicobar Islands	NICOBARS	2001	Kharif	Arecanut	1254.0	0.084119	56	58	12610.85	1691.66	6.83	11554.0378
2	Andaman and Nicobar Islands	NICOBARS	2002	Whole Year	Arecanut	1258.0	0.080064	58	53	32683.46	3207.35	9.33	29924.5755
3	Andaman and Nicobar Islands	NICOBARS	2003	Whole Year	Arecanut	1261.0	0.181051	57	58	13209.32	2228.97	5.90	13150.9230
4	Andaman and Nicobar Islands	NICOBARS	2004	Whole Year	Arecanut	1264.7	0.035446	63	67	22560.30	1595.56	13.57	21651.7492

Fig: Given data frame

DATA VALIDATION/ CLEANING/PREPARING PROCESS:

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.



```
#preprocessing, split test and dataset, split
X = df.drop(labels='CPPY', axis=1)
#Response variable
y = df.loc[:,'CPPY']
```

```
#We'll use a test size of 30%. We also stratif
from sklearn.model_selection import train_test
X_train, X_test, y_train, y_test = train_test_
print("Number of training dataset: ", len(X_tr
print("Number of testing dataset: ", len(X_tes
print("Total number of dataset: ", len(X_train
```

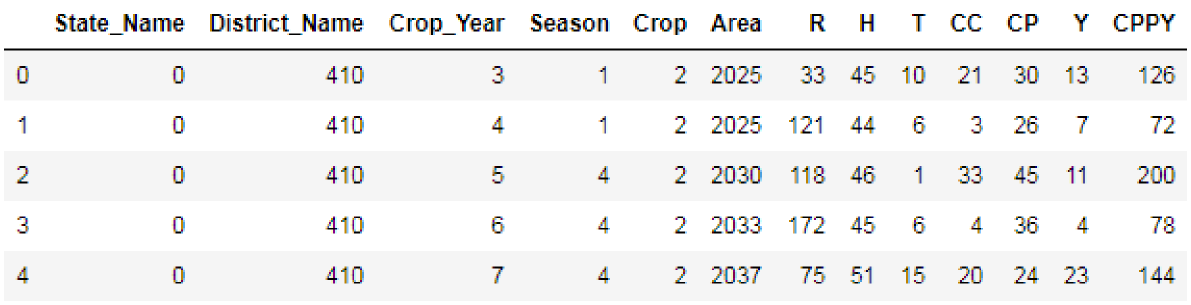
```
Number of training dataset:  163704
Number of testing dataset:  70160
Total number of dataset:  233864
```

Fig: Spliting the given dataset

DATA PREPROCESSING:

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format; for example, Random

Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set.



	State_Name	District_Name	Crop_Year	Season	Crop	Area	R	H	T	CC	CP	Y	CPPY
0	0	410	3	1	2	2025	33	45	10	21	30	13	126
1	0	410	4	1	2	2025	121	44	6	3	26	7	72
2	0	410	5	4	2	2030	118	46	1	33	45	11	200
3	0	410	6	4	2	2033	172	45	6	4	36	4	78
4	0	410	7	4	2	2037	75	51	15	20	24	23	144

Fig: After preprocessing given data frame

5.2.2EXPLORATION 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 in applied 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.

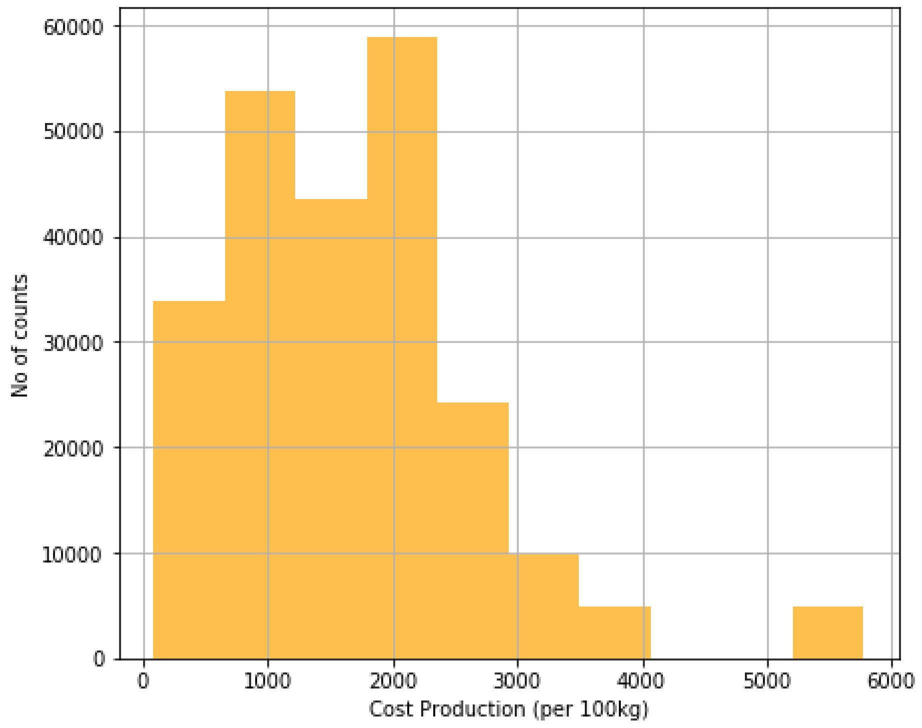


Fig: Cost production per 100kg by counts

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 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 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.

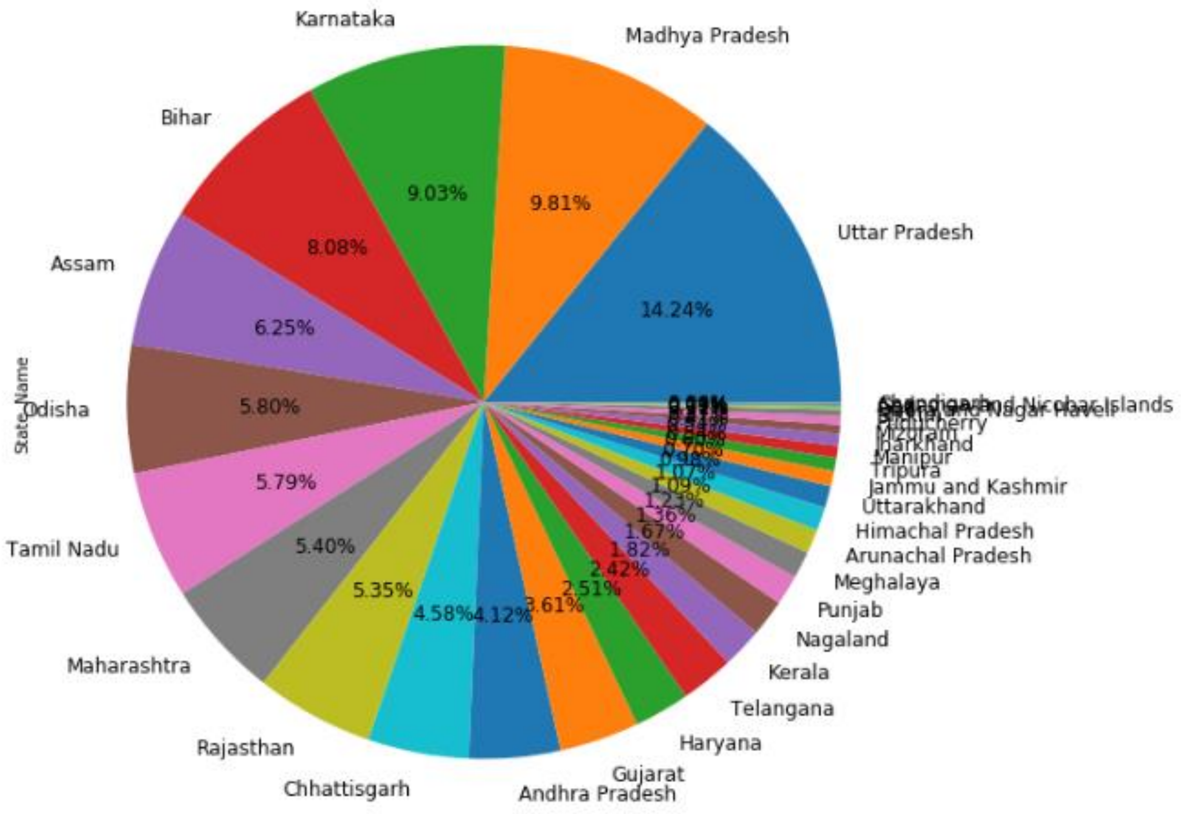


Fig: Percentage level of crop yield production by state

5.2.3LOGISTIC REGRESSION:

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts P(Y=1) as a function of

X.LOGISTIC REGRESSION ASUMPTIOMS:

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little.
- The independent variables are linearly related to the log odds.
- Logistic regression requires quite large sample sizes.

DECISION TREE:

It is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It 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.

5.2.4 SUPPORT VECTOR MACHINES(SVM):

A classifier that categorizes the data set by setting an optimal hyper plane between data. I chose this classifier as it is incredibly versatile in the number of different kernelling functions that can be applied and this model can yield a high predictability rate. Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning.

- How to disentangle the many names used to refer to support vector machines.
- The representation used by SVM when the model is actually stored on disk.
- How a learned SVM model representation can be used to make predictions for new data.
- How to learn an SVM model from training data.
- How to best prepare your data for the SVM algorithm.
- Where you might look to get more information on SVM.

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

5.5.5K-NEAREST NEGIBOUR (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.

NAIVE BAYES ALGORITHM:

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically. Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method. The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

- Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

5.5.6 GUI BASED PREDICTION OF CROP YIELD AND YIELD COST

Tkinter is a python library for developing GUI (Graphical User Interfaces). We use the tkinter library for creating an application of UI (User Interface), to create windows and all other graphical user interface and Tkinter will come with Python as a standard package, it can be used for security purpose of each users or accountants. There will be two kinds of pages like registration user purpose and login entry purpose of users.

CHAPTER 6

SYSTEM DESIGN

6.1 MODULE:1 DATA VALIDATION PROCESS AND PREPROCESSING TECHNIQUE

```
#import libraries for access and functional purpose

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as s

df = p.read_csv("df4.csv")

df.head(10)

listcrops = p.Categorical(df['Crop'])

listcrops

df['Crop'].value_counts()

df['Crop'].nunique()

df['Crop'].unique()

df.shape

df.describe()

df.info()

df[df.dtypes[df.dtypes == 'float64'].index].describe()
```

```
p.Categorical(df['State_Name']).describe()

p.Categorical(df['Mean Temp']).describe()

p.Categorical(df['Season']).describe()

p.Categorical(df['Average Humidity']).describe()

p.Categorical(df['rainfall']).describe()

p.Categorical(df['Crop']).describe()

df.duplicated()

sum(df.duplicated())

df.nunique()

df.corr()

print("Minimum yield of crops is  (100kg/2.47 acre):", df["Yield (Quintal/
Hectare) "].min())

print("Maximun yield of crops is  (100kg/2.47 acre):", df["Yield (Quintal/ Hectare)
"].max())

print("Minimum cost production for c2 scheme(per 2.47 acre):", df["Cost of
Production (`/Quintal) C2"].min())

print("Maximun cost production for c2 scheme(per 2.47 acre):", df["Cost of
Production (`/Quintal) C2"].max())

df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)
df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

#show the dataframe

df.head()

p.crosstab(df.State_Name,df.Crop)

df.rename(columns={'Mean Temp':'T'}, inplace=True)

df.rename(columns={'Average Humidity':'H'}, inplace=True)

df.rename(columns={'rainfall':'R'}, inplace=True)

df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)

df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

df['Y'].unique()

df.columns

from sklearn.preprocessing import LabelEncoder

var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',

    'R', 'H', 'T', 'CC', 'CP', 'Y', 'CPPY']

le = LabelEncoder()

for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)

df.head()
```

MODULE 2: EXPLORATION DATA ANALYSIS OF TRAIN A MODEL BY GIVEN ATTRIBUTES

```
#import libraries for access and functional purpose

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as s

#read the given dataset

df = p.read_csv("df.csv")

df.rename(columns={'Mean Temp':'T'}, inplace=True)

df.rename(columns={'Average Humidity':'H'}, inplace=True)

df.rename(columns={'rainfall':'R'}, inplace=True)

#Rename the data

df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)

df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

df.dropna()
```

```
from sklearn.preprocessing import LabelEncoder

var_mod = ['Y']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['YPr']= df.Y.map({'13':0, '7':0, '11':0, '4':0, '23':0, '39':1, '10':0, '18':0, '36':1,
'47':1, '8':0, '3':0,

       '38':1, '46':1, '5':0, '9':0, '2':0, '44':1, '17':0, '41':1, '6':0, '16':0, '35':1, '19':0,

       '0':0, '43':1, '12':0, '45':1, '25':0, '33':1, '29':0, '37':1, '32':1, '21':0, '42':1,

       '48':1, '30':1, '34':1, '26':0, '20':0, '31':1, '24':0, '27':0, '22':0, '40':1, '28':0,

       '1':0, '14':0, '15':0})

df['CPPY'].unique()

from sklearn.preprocessing import LabelEncoder

var_mod = ['CPPY']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['CPPY'].unique()

df['CPPYPr']= df.CPPY.map({'126':0, '72':0, '200':0, '78':0, '144':0, '212':0, '149':0,
'151':0, '178':0, '312':0,
```

'187':0, '69':0, '205':0, '276':1, '76':0, '134':0, '66':0, '254':1, '150':0, '291':1,

'155':0, '106':0, '87':0, '119':0, '183':0, '170':0, '28':0, '265':1, '180':0, '36':0,

'293':1, '326':1, '148':0, '234':1, '3':0, '124':0, '111':0, '182':0, '166':0, '169':0,

'270':1, '243':1, '98':0, '230':0, '282':1, '334':1, '188':0, '65':0, '34':0, '328':1,

'160':0, '115':0, '89':0, '152':0, '240':1, '141':0, '233':1, '32':0, '281':1, '335':1,

'109':0, '47':0, '136':0, '112':0, '261':1, '229':0, '42':0, '325':1, '213':0, '175':0,

'125':0, '196':0, '292':1, '82':0, '235':1, '79':0, '105':0, '123':0, '210':0, '39':0,

'146':0, '185':0, '201':0, '41':0, '164':0, '184':0, '222':0, '250':0, '301':1,

'218':0, '217':0, '168':0, '264':1, '46':0, '103':0, '284':1, '198':0, '56':0, '171':0,

'331':1, '256':1, '99':0, '58':0, '247':1, '286':1, '67':0, '133':0, '143':0, '204':0,

'227':0, '194':0, '6':0, '280':1, '225':0, '48':0, '258':1, '94':0, '244':1, '294':1,

'215':0, '132':0, '277':1, '71':0, '219':0, '315':1, '97':0, '91':0, '156':0, '289':1,

'100':0, '295':1, '147':0, '214':0, '19':0, '223':0, '90':0, '269':1, '300':1, '114':0,

'135':0, '173':0, '55':0, '113':0, '127':0, '73':0, '177':0, '274':1, '118':0, '191':0,

'145':0, '307':1, '162':0, '228':0, '50':0, '5':0, '248':1, '203':0, '61':0, '129':0,

'192':0, '83':0, '158':0, '206':0, '242':1, '267':1, '137':0, '92':0, '176':0, '298':1,

'25':0, '296':1, '186':0, '239':1, '193':0, '165':0, '310':1, '20':0, '287':1, '75':0,

'107':0, '271':1, '138':0, '121':0, '241':1, '74':0, '43':0, '232':0, '154':0, '181':0,

'195':0, '102':0, '237':1, '110':0, '268':1, '49':0, '104':0, '64':0, '318':1, '13':0,

```
    '128':0, '153':0, '31':0, '93':0, '309':1, '257':0, '52':0, '262':1, '202':0, '174':0,
    '101':0, '224':0, '86':0, '45':0, '263':1, '167':0, '17':0, '29':0, '30':0, '54':0,
    '303':1, '163':0, '251':1, '142':0, '273':1, '96':0, '35':0, '327':1, '53':0, '15':0,
    '190':0, '308':1, '226':0, '139':0, '62':0, '2':0, '1':0, '27':0, '84':0, '285':1,
    '246':1, '23':0, '189':0, '299':1, '231':0, '24':0, '16':0, '333':1, '306':1, '18':0,
    '288':1, '199':0, '260':1, '323':1, '37':0, '278':1, '324':1, '140':0, '4':0, '245':1,
    '322':1, '329':1, '159':0, '80':0, '197':0, '275':1, '11':0, '40':0, '33':0, '0':0,
    '320':1, '290':1, '68':0, '220':0, '21':0, '330':1, '12':0, '157':0, '302':1, '44':0,
    '221':0, '216':0, '279':1, '63':0, '313':1, '311':1, '332':1, '266':1, '238':1,
    '211':0, '172':0, '14':0, '117':0, '161':0, '122':0, '60':0, '321':1, '95':0, '208':0,
    '272':1, '131':0, '207':0, '81':0, '314':1, '51':0, '283':1, '120':0, '38':0, '9':0,
    '130':0, '70':0, '179':0, '7':0, '10':0, '108':0, '255':1, '77':0, '88':0, '22':0,
    '57':0, '85':0, '304':1, '253':1, '249':0, '297':1, '116':0, '305':1, '317':1, '252':1,
    '236':1, '209':0, '259':1, '26':0, '319':1, '316':1, '59':0, '336':1, '8':0})
df.head()
df['YPr'].unique()
```

SPLITING TRAIN/TEST:

```
#preprocessing, split test and dataset, split response variable

X = df.drop(labels='CPPY', axis=1)

#Response variable

y = df.loc[:,'CPPY']

#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

print("Number of training dataset: ", len(X_train))

print("Number of testing dataset: ", len(X_test))

print("Total number of dataset: ", len(X_train)+len(X_test))

count_classes = p.value_counts(df['Crop'], sort = True).sort_index()

count_classes.plot(kind = 'bar', figsize=(20,15))

plt.title("Crop details")

plt.xlabel("Catogeries")

plt.ylabel("Strength values")

no=sum(df['CPPYPr']==0)

yes=sum(df['CPPYPr']==1)
```

```
colors=['orange','black']

locations=[1,2]

heights=[no,yes]

labels=['Unexpected Cost Production','Expected Cost Production']

plt.bar(locations,heights,color=colors,tick_label=labels,alpha=0.7)

plt.xlabel('Yield of Crost Production')

plt.ylabel('No. of each crop')

plt.title('Prediction results expecting from farmer by yield of crost production
amount')

no=sum(df['YPr']==0)

yes=sum(df['YPr']==1)

colors=['orange','black']

locations=[1,2]

heights=[no,yes]

labels=['No Yield','Yield']

plt.bar(locations,heights,color=colors,tick_label=labels,alpha=0.7)

plt.xlabel('Yield of Crop')

plt.ylabel('No. of each crop')

plt.title('Prediction results expecting from farmer by yield of crop')
```

```
df['CP'].hist(figsize=(7,6), color='orange', alpha=0.7)

plt.xlabel('Cost Production (per 100kg)')

plt.ylabel('No of counts')

plt.title('Cost Production per 100kg by counts')

df['CPPY'].hist(figsize=(7,6), color='black', alpha=0.7)

plt.xlabel('Cost Production of crop')

plt.ylabel('No of counts')

plt.title('Cost Production of crop by counts')

# Heatmap plot diagram

fig, ax = plt.subplots(figsize=(15,10))

s.heatmap(df.corr(), ax=ax, annot=True)

df.boxplot(column="CP", by="Season", figsize=(15,10))

#Propagation by variable

def PropByVar(df, variable):

    dataframe_pie = df[variable].value_counts()

    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)

    ax.set_title(variable + ' (in Percentage)', fontsize = 15)

    return n.round(dataframe_pie/df.shape[0]*100,2)
```

```
PropByVar(df, 'Crop')

#Propagation by variable

def PropByVar(df, variable):

    dataframe_pie = df[variable].value_counts()

    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize = 12)

    ax.set_title(variable + ' (in Percentage)', fontsize = 15)

    return n.round(dataframe_pie/df.shape[0]*100,2)

PropByVar(df, 'State_Name')

count_classes = p.value_counts(df['State_Name'], sort = True).sort_index()

count_classes.plot(kind = 'bar', figsize=(20,15))

plt.title("Dataset by each states")

plt.xlabel("Number of States")

plt.ylabel("Given data counts")

#Density Plots

plt = df.plot(kind= 'density', subplots=True, layout=(4,3), sharex=False,

                     sharey=False,fontsize=12, figsize=(15,10))

def y_No_y_bar_plot(df, bygroup):

    dataframe_by_Group = p.crosstab(df[bygroup], columns=df["YPr"], normalize
= 'index')
```

```
    dataframe_by_Group = n.round((dataframe_by_Group * 100), decimals=2)

    ax = dataframe_by_Group.plot.bar(figsize=(10,5));

    vals = ax.get_yticks()

    ax.set_yticklabels(['{:3.0f}%'.format(x) for x in vals]);

    ax.set_xticklabels(dataframe_by_Group.index,rotation = 0, fontsize = 15);

    ax.set_title('Crop Yield Prediction Vs No Crop Yield Prediction (%) (by ' +
dataframe_by_Group.index.name + ')\n', fontsize = 15)

    ax.set_xlabel(dataframe_by_Group.index.name, fontsize = 12)

    ax.set_ylabel('(%)', fontsize = 12)

    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)

    rects = ax.patches

    # Add Data Labels

    for rect in rects:

        height = rect.get_height()

        ax.text(rect.get_x() + rect.get_width()/2,

                height + 2,

                str(height)+'%',

                ha='center',

                va='bottom',
                fontsize = 12)

    return dataframe_by_Group

y_No_y_bar_plot(df, 'Season')

def c_No_y_bar_plot(df, bygroup):

    dataframe_by_Group = p.crosstab(df[bygroup], columns=df["CPPYPr"],
normalize = 'index')

    dataframe_by_Group = n.round((dataframe_by_Group * 100), decimals=2)

    ax = dataframe_by_Group.plot.bar(figsize=(10,5));

    vals = ax.get_yticks()

    ax.set_yticklabels(['{:3.0f}%'.format(x) for x in vals]);

    ax.set_xticklabels(dataframe_by_Group.index,rotation = 0, fontsize = 15);

    ax.set_title('Crop Yield Production cost Vs No Crop Yield Production cost (%)
(by ' + dataframe_by_Group.index.name + ')\n', fontsize = 15)

    ax.set_xlabel(dataframe_by_Group.index.name, fontsize = 12)

    ax.set_ylabel('(%)', fontsize = 12)

    ax.legend(loc = 'upper left',bbox_to_anchor=(1.0,1.0), fontsize= 12)

    rects = ax.patches

    # Add Data Labels

    for rect in rects:

        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2,

                height + 2,

                str(height)+'%',

                ha='center',

                va='bottom',

                fontsize = 12)

    return dataframe_by_Group

c_No_y_bar_plot(df, 'Season')
```

MODULE 3 : PERFORMANCE MEASUREMENTS OF LR AND DT

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n

#read the given dataset

df = p.read_csv("df.csv")

df.rename(columns={'Mean Temp':'T'}, inplace=True)

df.rename(columns={'Average Humidity':'H'}, inplace=True)

df.rename(columns={'rainfall':'R'}, inplace=True)
df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)

df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

from sklearn.preprocessing import LabelEncoder

var_mod = ['Y']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['YPr']= df.Y.map({'13':0, '7':0, '11':0, '4':0, '23':0, '39':1, '10':0, '18':0, '36':1,
'47':1, '8':0, '3':0,

       '38':1, '46':1, '5':0, '9':0, '2':0, '44':1, '17':0, '41':1, '6':0, '16':0, '35':1, '19':0,

       '0':0, '43':1, '12':0, '45':1, '25':0, '33':1, '29':0, '37':1, '32':1, '21':0, '42':1,

       '48':1, '30':1, '34':1, '26':0, '20':0, '31':1, '24':0, '27':0, '22':0, '40':1, '28':0,

       '1':0, '14':0, '15':0})

df.columns

from sklearn.preprocessing import LabelEncoder

var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',

       'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']
```

```
le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df.head()

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score,
roc_auc_score

X = df.drop(labels='YPr', axis=1)

#Response variable

y = df.loc[:,'YPr']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y

from sklearn.linear_model import LogisticRegression

logR= LogisticRegression()



logR.fit(X_train,y_train)



predictR = logR.predict(X_test)
```

```
print("")

print('Classification report of Logistic Regression Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Logistic Regression is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Logistic Regression is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]

FN = cm1[1][0]

TP = cm1[1][1]
```

```
FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)
```

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

dtree.fit(X_train, y_train)

predictDT = dtree.predict(X_test)

print("")

print('Classification report of Decision Tree Classifier Results:')

print("")

print(classification_report(y_test,predictDT))

x = (accuracy_score(y_test,predictDT)*100)

print('Accuracy result of Decision Tree Classifier is', x)

print("")

cm2=confusion_matrix(y_test,predictDT)

print('Confusion Matrix result of Decision Tree Classifier is:\n',
confusion_matrix(y_test,predictDT))

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
```

```
print('Specificity : ', specificity1)

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")
```

```
PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.preprocessing import LabelEncoder

var_mod = ['CPPY']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['CPPYPr']= df.CPPY.map({'126':0, '72':0, '200':0, '78':0, '144':0, '212':0, '149':0,
'151':0, '178':0, '312':0,

       '187':0, '69':0, '205':0, '276':1, '76':0, '134':0, '66':0, '254':1, '150':0, '291':1,

       '155':0, '106':0, '87':0, '119':0, '183':0, '170':0, '28':0, '265':1, '180':0, '36':0,

       '293':1, '326':1, '148':0, '234':1, '3':0, '124':0, '111':0, '182':0, '166':0, '169':0,

       '270':1, '243':1, '98':0, '230':0, '282':1, '334':1, '188':0, '65':0, '34':0, '328':1,

       '160':0, '115':0, '89':0, '152':0, '240':1, '141':0, '233':1, '32':0, '281':1, '335':1,

       '109':0, '47':0, '136':0, '112':0, '261':1, '229':0, '42':0, '325':1, '213':0, '175':0,

       '125':0, '196':0, '292':1, '82':0, '235':1, '79':0, '105':0, '123':0, '210':0, '39':0,

       '146':0, '185':0, '201':0, '41':0, '164':0, '184':0, '222':0, '250':0, '301':1,
```

'218':0, '217':0, '168':0, '264':1, '46':0, '103':0, '284':1, '198':0, '56':0, '171':0,
'331':1, '256':1, '99':0, '58':0, '247':1, '286':1, '67':0, '133':0, '143':0, '204':0,
'227':0, '194':0, '6':0, '280':1, '225':0, '48':0, '258':1, '94':0, '244':1, '294':1,
'215':0, '132':0, '277':1, '71':0, '219':0, '315':1, '97':0, '91':0, '156':0, '289':1,
'100':0, '295':1, '147':0, '214':0, '19':0, '223':0, '90':0, '269':1, '300':1, '114':0,
'135':0, '173':0, '55':0, '113':0, '127':0, '73':0, '177':0, '274':1, '118':0, '191':0,
'145':0, '307':1, '162':0, '228':0, '50':0, '5':0, '248':1, '203':0, '61':0, '129':0,
'192':0, '83':0, '158':0, '206':0, '242':1, '267':1, '137':0, '92':0, '176':0, '298':1,
'25':0, '296':1, '186':0, '239':1, '193':0, '165':0, '310':1, '20':0, '287':1, '75':0,
'107':0, '271':1, '138':0, '121':0, '241':1, '74':0, '43':0, '232':0, '154':0, '181':0,
'195':0, '102':0, '237':1, '110':0, '268':1, '49':0, '104':0, '64':0, '318':1, '13':0,
'128':0, '153':0, '31':0, '93':0, '309':1, '257':0, '52':0, '262':1, '202':0, '174':0,
'101':0, '224':0, '86':0, '45':0, '263':1, '167':0, '17':0, '29':0, '30':0, '54':0,
'303':1, '163':0, '251':1, '142':0, '273':1, '96':0, '35':0, '327':1, '53':0, '15':0,
'190':0, '308':1, '226':0, '139':0, '62':0, '2':0, '1':0, '27':0, '84':0, '285':1,
'246':1, '23':0, '189':0, '299':1, '231':0, '24':0, '16':0, '333':1, '306':1, '18':0,
'288':1, '199':0, '260':1, '323':1, '37':0, '278':1, '324':1, '140':0, '4':0, '245':1,
'322':1, '329':1, '159':0, '80':0, '197':0, '275':1, '11':0, '40':0, '33':0, '0':0,
'320':1, '290':1, '68':0, '220':0, '21':0, '330':1, '12':0, '157':0, '302':1, '44':0,

```
    '221':0, '216':0, '279':1, '63':0, '313':1, '311':1, '332':1, '266':1, '238':1,
    '211':0, '172':0, '14':0, '117':0, '161':0, '122':0, '60':0, '321':1, '95':0, '208':0,
    '272':1, '131':0, '207':0, '81':0, '314':1, '51':0, '283':1, '120':0, '38':0, '9':0,
    '130':0, '70':0, '179':0, '7':0, '10':0, '108':0, '255':1, '77':0, '88':0, '22':0,
    '57':0, '85':0, '304':1, '253':1, '249':0, '297':1, '116':0, '305':1, '317':1, '252':1,
    '236':1, '209':0, '259':1, '26':0, '319':1, '316':1, '59':0, '336':1, '8':0})


from sklearn.preprocessing import LabelEncoder
var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
    'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']
le = LabelEncoder()
for i in var_mod:
  df[i] = le.fit_transform(df[i]).astype(str)
df.head()
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score,
roc_auc_score
X = df.drop(labels='CPPYPr', axis=1)
#Response variable
y = df.loc[:,'CPPYPr']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

from sklearn.linear_model import LogisticRegression

logR= LogisticRegression()

logR.fit(X_train,y_train)

predictR = logR.predict(X_test)

print("")

print('Classification report of Logistic Regression Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Logistic Regression is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Logistic Regression is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )
print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]

FN = cm1[1][0]

TP = cm1[1][1]

FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

dtree.fit(X_train, y_train)

predictDT = dtree.predict(X_test)

print("")

print('Classification report of Decision Tree Classifier Results:')

print("")

print(classification_report(y_test,predictDT))

x = (accuracy_score(y_test,predictDT)*100)

print('Accuracy result of Decision Tree Classifier is', x)

print("")

cm2=confusion_matrix(y_test,predictDT)
```

```
print('Confusion Matrix result of Decision Tree Classifier is:\n',
confusion_matrix(y_test,predictDT))

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)
```

```
FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)
```

MODULE 4 : PERFORMANCE MEASUREMENTS OF RF ANDSVM:

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n

#read the given dataset

df = p.read_csv("df4.csv")
```

```
df.rename(columns={'Mean Temp':'T'}, inplace=True)

df.rename(columns={'Average Humidity':'H'}, inplace=True)

df.rename(columns={'rainfall':'R'}, inplace=True)

df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)

df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

from sklearn.preprocessing import LabelEncoder

var_mod = ['Y']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['YPr']= df.Y.map({'13':0, '7':0, '11':0, '4':0, '23':0, '39':1, '10':0, '18':0, '36':1,
'47':1, '8':0, '3':0,

       '38':1, '46':1, '5':0, '9':0, '2':0, '44':1, '17':0, '41':1, '6':0, '16':0, '35':1, '19':0,

       '0':0, '43':1, '12':0, '45':1, '25':0, '33':1, '29':0, '37':1, '32':1, '21':0, '42':1,

       '48':1, '30':1, '34':1, '26':0, '20':0, '31':1, '24':0, '27':0, '22':0, '40':1, '28':0,

       '1':0, '14':0, '15':0})

df.columns
```

```
from sklearn.preprocessing import LabelEncoder

var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',

       'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df.head()

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score,
roc_auc_score
```

PREDICTION OF CROP BY YIELD :

```
X = df.drop(labels='YPr', axis=1)

#Response variable

y = df.loc[:,'YPr']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
```

```
rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")

print('Classification report of Random Forest Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Random Forest is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Random Forest is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]
```

```
FN = cm1[1][0]

TP = cm1[1][1]

FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)
```

```
print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.svm import SVC

s = SVC()

s.fit(X_train,y_train)

predicts = s.predict(X_test)

print("")

print('Classification report of Support Vector Machines Results:')

print("")

print(classification_report(y_test,predicts))

x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)

print("")

cm2=confusion_matrix(y_test,predicts)

print('Confusion Matrix result of Support Vector Machines is:\n',cm2)

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")
```

```
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.preprocessing import LabelEncoder

var_mod = ['CPPY']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['CPPYPr']= df.CPPY.map({'126':0, '72':0, '200':0, '78':0, '144':0, '212':0, '149':0,
'151':0, '178':0, '312':0,

      '187':0, '69':0, '205':0, '276':1, '76':0, '134':0, '66':0, '254':1, '150':0, '291':1,

      '155':0, '106':0, '87':0, '119':0, '183':0, '170':0, '28':0, '265':1, '180':0, '36':0,

      '293':1, '326':1, '148':0, '234':1, '3':0, '124':0, '111':0, '182':0, '166':0, '169':0,

      '270':1, '243':1, '98':0, '230':0, '282':1, '334':1, '188':0, '65':0, '34':0, '328':1,

      '160':0, '115':0, '89':0, '152':0, '240':1, '141':0, '233':1, '32':0, '281':1, '335':1,

      '109':0, '47':0, '136':0, '112':0, '261':1, '229':0, '42':0, '325':1, '213':0, '175':0,
```

'125':0, '196':0, '292':1, '82':0, '235':1, '79':0, '105':0, '123':0, '210':0, '39':0,

'146':0, '185':0, '201':0, '41':0, '164':0, '184':0, '222':0, '250':0, '301':1,

'218':0, '217':0, '168':0, '264':1, '46':0, '103':0, '284':1, '198':0, '56':0, '171':0,

'331':1, '256':1, '99':0, '58':0, '247':1, '286':1, '67':0, '133':0, '143':0, '204':0,

'227':0, '194':0, '6':0, '280':1, '225':0, '48':0, '258':1, '94':0, '244':1, '294':1,

'215':0, '132':0, '277':1, '71':0, '219':0, '315':1, '97':0, '91':0, '156':0, '289':1,

'100':0, '295':1, '147':0, '214':0, '19':0, '223':0, '90':0, '269':1, '300':1, '114':0,

'135':0, '173':0, '55':0, '113':0, '127':0, '73':0, '177':0, '274':1, '118':0, '191':0,

'145':0, '307':1, '162':0, '228':0, '50':0, '5':0, '248':1, '203':0, '61':0, '129':0,

'192':0, '83':0, '158':0, '206':0, '242':1, '267':1, '137':0, '92':0, '176':0, '298':1,

'25':0, '296':1, '186':0, '239':1, '193':0, '165':0, '310':1, '20':0, '287':1, '75':0,

'107':0, '271':1, '138':0, '121':0, '241':1, '74':0, '43':0, '232':0, '154':0, '181':0,

'195':0, '102':0, '237':1, '110':0, '268':1, '49':0, '104':0, '64':0, '318':1, '13':0,

'128':0, '153':0, '31':0, '93':0, '309':1, '257':0, '52':0, '262':1, '202':0, '174':0,

'101':0, '224':0, '86':0, '45':0, '263':1, '167':0, '17':0, '29':0, '30':0, '54':0,

'303':1, '163':0, '251':1, '142':0, '273':1, '96':0, '35':0, '327':1, '53':0, '15':0,

'190':0, '308':1, '226':0, '139':0, '62':0, '2':0, '1':0, '27':0, '84':0, '285':1,

'246':1, '23':0, '189':0, '299':1, '231':0, '24':0, '16':0, '333':1, '306':1, '18':0,

'288':1, '199':0, '260':1, '323':1, '37':0, '278':1, '324':1, '140':0, '4':0, '245':1,

```
        '322':1, '329':1, '159':0, '80':0, '197':0, '275':1, '11':0, '40':0, '33':0, '0':0,
        '320':1, '290':1, '68':0, '220':0, '21':0, '330':1, '12':0, '157':0, '302':1, '44':0,
        '221':0, '216':0, '279':1, '63':0, '313':1, '311':1, '332':1, '266':1, '238':1,
        '211':0, '172':0, '14':0, '117':0, '161':0, '122':0, '60':0, '321':1, '95':0, '208':0,
        '272':1, '131':0, '207':0, '81':0, '314':1, '51':0, '283':1, '120':0, '38':0, '9':0,
        '130':0, '70':0, '179':0, '7':0, '10':0, '108':0, '255':1, '77':0, '88':0, '22':0,
        '57':0, '85':0, '304':1, '253':1, '249':0, '297':1, '116':0, '305':1, '317':1, '252':1,
        '236':1, '209':0, '259':1, '26':0, '319':1, '316':1, '59':0, '336':1, '8':0})


from sklearn.preprocessing import LabelEncoder
var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
        'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
df.head()
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score,
roc_auc_score
X = df.drop(labels='CPPYPr', axis=1)
```

```
#Response variable

y = df.loc[:,'CPPYPr']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")

print('Classification report of Random Forest Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Random Forest is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Random Forest is:\n',cm1)

print("")
```

```
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]

FN = cm1[1][0]

TP = cm1[1][1]

FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)
```

```
print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.svm import SVC

s = SVC()

s.fit(X_train,y_train)

predicts = s.predict(X_test)

print("")

print('Classification report of Support Vector Machines Results:')

print("")

print(classification_report(y_test,predicts))

x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)
print("")

cm2=confusion_matrix(y_test,predicts)

print('Confusion Matrix result of Support Vector Machines is:\n',cm2)

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")
```

```
TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)
```

MODULE 5 : PERFORMANCE MEASUREMENTS OF NB AND KNN:

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n
```

```
#read the given dataset

df = p.read_csv("df4.csv")

df.rename(columns={'Mean Temp':'T'}, inplace=True)

df.rename(columns={'Average Humidity':'H'}, inplace=True)

df.rename(columns={'rainfall':'R'}, inplace=True)

df.rename(columns={'Cost of Cultivation (`/Hectare) C2':'CC'}, inplace=True)

df.rename(columns={'Cost of Production (`/Quintal) C2':'CP'}, inplace=True)

df.rename(columns={'Yield (Quintal/ Hectare) ':'Y'}, inplace=True)

df.rename(columns={'cost of production per yield':'CPPY'}, inplace=True)

from sklearn.preprocessing import LabelEncoder

var_mod = ['Y']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['YPr']= df.Y.map({'13':0, '7':0, '11':0, '4':0, '23':0, '39':1, '10':0, '18':0, '36':1,
'47':1, '8':0, '3':0,

       '38':1, '46':1, '5':0, '9':0, '2':0, '44':1, '17':0, '41':1, '6':0, '16':0, '35':1, '19':0,

       '0':0, '43':1, '12':0, '45':1, '25':0, '33':1, '29':0, '37':1, '32':1, '21':0, '42':1,

       '48':1, '30':1, '34':1, '26':0, '20':0, '31':1, '24':0, '27':0, '22':0, '40':1, '28':0,
       '1':0, '14':0, '15':0})
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
       'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
df.head()
X = df.drop(labels='YPr', axis=1)
#Response variable
y = df.loc[:,'YPr']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)
from sklearn.neighbors import KNeighborsClassifier
rfc = KNeighborsClassifier()


rfc.fit(X_train,y_train)
```

```
predictR = rfc.predict(X_test)

print("")

print('Classification report of KNeighborsClassifier Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of KNeighborsClassifier is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of KNeighborsClassifier is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]
```

```
FN = cm1[1][0]

TP = cm1[1][1]

FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)
```

```
print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.naive_bayes import GaussianNB

s = GaussianNB()

s.fit(X_train,y_train)

predicts = s.predict(X_test)

print("")

print('Classification report of Naive Bayes Results:')

print("")

print(classification_report(y_test,predicts))

x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Naive Bayes is:', x)

print("")

cm2=confusion_matrix(y_test,predicts)

print('Confusion Matrix result of Naive Bayes is:\n',cm2)

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")
```

```
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.preprocessing import LabelEncoder

var_mod = ['CPPY']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(str)

df['CPPYPr']= df.CPPY.map({'126':0, '72':0, '200':0, '78':0, '144':0, '212':0, '149':0,
'151':0, '178':0, '312':0,

       '187':0, '69':0, '205':0, '276':1, '76':0, '134':0, '66':0, '254':1, '150':0, '291':1,

       '155':0, '106':0, '87':0, '119':0, '183':0, '170':0, '28':0, '265':1, '180':0, '36':0,

       '293':1, '326':1, '148':0, '234':1, '3':0, '124':0, '111':0, '182':0, '166':0, '169':0,

       '270':1, '243':1, '98':0, '230':0, '282':1, '334':1, '188':0, '65':0, '34':0, '328':1,

       '160':0, '115':0, '89':0, '152':0, '240':1, '141':0, '233':1, '32':0, '281':1, '335':1,

       '109':0, '47':0, '136':0, '112':0, '261':1, '229':0, '42':0, '325':1, '213':0, '175':0,
```

'125':0, '196':0, '292':1, '82':0, '235':1, '79':0, '105':0, '123':0, '210':0, '39':0,
'146':0, '185':0, '201':0, '41':0, '164':0, '184':0, '222':0, '250':0, '301':1,
'218':0, '217':0, '168':0, '264':1, '46':0, '103':0, '284':1, '198':0, '56':0, '171':0,
'331':1, '256':1, '99':0, '58':0, '247':1, '286':1, '67':0, '133':0, '143':0, '204':0,
'227':0, '194':0, '6':0, '280':1, '225':0, '48':0, '258':1, '94':0, '244':1, '294':1,
'215':0, '132':0, '277':1, '71':0, '219':0, '315':1, '97':0, '91':0, '156':0, '289':1,
'100':0, '295':1, '147':0, '214':0, '19':0, '223':0, '90':0, '269':1, '300':1, '114':0,
'135':0, '173':0, '55':0, '113':0, '127':0, '73':0, '177':0, '274':1, '118':0, '191':0,
'145':0, '307':1, '162':0, '228':0, '50':0, '5':0, '248':1, '203':0, '61':0, '129':0,
'192':0, '83':0, '158':0, '206':0, '242':1, '267':1, '137':0, '92':0, '176':0, '298':1,
'25':0, '296':1, '186':0, '239':1, '193':0, '165':0, '310':1, '20':0, '287':1, '75':0,
'107':0, '271':1, '138':0, '121':0, '241':1, '74':0, '43':0, '232':0, '154':0, '181':0,
'195':0, '102':0, '237':1, '110':0, '268':1, '49':0, '104':0, '64':0, '318':1, '13':0,
'128':0, '153':0, '31':0, '93':0, '309':1, '257':0, '52':0, '262':1, '202':0, '174':0,
'101':0, '224':0, '86':0, '45':0, '263':1, '167':0, '17':0, '29':0, '30':0, '54':0,
'303':1, '163':0, '251':1, '142':0, '273':1, '96':0, '35':0, '327':1, '53':0, '15':0,
'190':0, '308':1, '226':0, '139':0, '62':0, '2':0, '1':0, '27':0, '84':0, '285':1,
'246':1, '23':0, '189':0, '299':1, '231':0, '24':0, '16':0, '333':1, '306':1, '18':0,
'288':1, '199':0, '260':1, '323':1, '37':0, '278':1, '324':1, '140':0, '4':0, '245':1,

```
        '322':1, '329':1, '159':0, '80':0, '197':0, '275':1, '11':0, '40':0, '33':0, '0':0,
        '320':1, '290':1, '68':0, '220':0, '21':0, '330':1, '12':0, '157':0, '302':1, '44':0,
        '221':0, '216':0, '279':1, '63':0, '313':1, '311':1, '332':1, '266':1, '238':1,
        '211':0, '172':0, '14':0, '117':0, '161':0, '122':0, '60':0, '321':1, '95':0, '208':0,
        '272':1, '131':0, '207':0, '81':0, '314':1, '51':0, '283':1, '120':0, '38':0, '9':0,
        '130':0, '70':0, '179':0, '7':0, '10':0, '108':0, '255':1, '77':0, '88':0, '22':0,
        '57':0, '85':0, '304':1, '253':1, '249':0, '297':1, '116':0, '305':1, '317':1, '252':1,
        '236':1, '209':0, '259':1, '26':0, '319':1, '316':1, '59':0, '336':1, '8':0})
from sklearn.preprocessing import LabelEncoder
var_mod = ['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
        'R', 'H', 'T', 'CC', 'CP', 'Y','CPPY']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(str)
df.head()
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score, average_precision_score,
roc_auc_score
X = df.drop(labels='CPPYPr', axis=1)
#Response variable
```

```
y = df.loc[:,'CPPYPr']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1, stratify=y)

from sklearn.neighbors import KNeighborsClassifier

rfc = KNeighborsClassifier()

rfc.fit(X_train,y_train)

predictR = rfc.predict(X_test)

print("")

print('Classification report of K-Neighbors Classifier Results:')

print("")

print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of K-Neighbors Classifier is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of K-Neighbors Classifier is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
```

```
print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm1[0][0]

FN = cm1[1][0]

TP = cm1[1][1]

FP = cm1[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

from sklearn.naive_bayes import GaussianNB

s = GaussianNB()

s.fit(X_train,y_train)

predicts = s.predict(X_test)

print("")

print('Classification report of Naive Bayes Results:')

print("")

print(classification_report(y_test,predicts))

x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Naive Bayes is:', x)

print("")
```

```
cm2=confusion_matrix(y_test,predicts)

print('Confusion Matrix result of Naive Bayes is:\n',cm2)

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity1)

print("")

TN = cm2[0][0]

FN = cm2[1][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
```

```
TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)
```

MODULE 6:

```
from tkinter import *

import numpy as np

import pandas as pd

df = pd.read_csv("re.csv")

df

df.shape
df.columns

l1=['Above_1_ton_per_hec', 'Below_1_ton_per_hec', 'Above_3_tons_per_hec',

      'Above_100_tons_per_hec', 'Above_70_tons_per_hec',

      'Above_40_tons_per_hec', 'Above_6_tons_per_hec',

      'Above_50_tons_per_hec', 'Above_10_tons_per_hec']

l2=['Below_20000_hec',

      'Above_20000_hec', 'Above_40000_hec', 'Above_80000_hec']

l3=[ 'Andaman and Nicobar Islands', 'Andhra Pradesh', 'Arunachal Pradesh',

      'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh',

      'Dadra and Nagar Haveli', 'Goa', 'Gujarat', 'Haryana',

      'Himachal Pradesh', 'Jammu and Kashmir', 'Jharkhand', 'Karnataka',

      'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya',

      'Mizoram', 'Nagaland', 'Odisha', 'Puducherry', 'Punjab', 'Rajasthan',

      'Sikkim', 'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh',

      'Uttarakhand']

l4=['Kharif', 'Whole year', 'Rabi', 'Autumn', 'Summer',

      'Winter']

l5=['TA50', 'TA60', 'TA80']

l6=['HB25', 'HA25', 'HA60', 'HA80']
```

```
df['Crop'].unique()

cropyc=['Sugercane', 'Rice', 'Tobacco', 'Wheat', 'Coconut', 'Yam',

       'Sweetpotato', 'Turmeric', 'WaterMelon', 'Tapioca', 'Urad',

       'Varagu']

l7=['Above_1_ton_per_hec', 'Below_1_ton_per_hec', 'Above_3_tons_per_hec',

       'Above_100_tons_per_hec', 'Above_70_tons_per_hec',

       'Above_40_tons_per_hec', 'Above_6_tons_per_hec',

       'Above_50_tons_per_hec', 'Above_10_tons_per_hec', 'Below_20000_hec',

       'Above_20000_hec', 'Above_40000_hec', 'Above_80000_hec',

       'Andaman and Nicobar Islands', 'Andhra Pradesh', 'Arunachal Pradesh',

       'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh',

       'Dadra and Nagar Haveli', 'Goa', 'Gujarat', 'Haryana',

       'Himachal Pradesh', 'Jammu and Kashmir', 'Jharkhand', 'Karnataka',

       'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya',

       'Mizoram', 'Nagaland', 'Odisha', 'Puducherry', 'Punjab', 'Rajasthan',

       'Sikkim', 'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh',

       'Uttarakhand', 'Kharif', 'Whole year', 'Rabi', 'Autumn', 'Summer',

       'Winter', 'TA50', 'TA60', 'TA80', 'HB25', 'HA25', 'HA60', 'HA80']

l8=[]
```

```
for x in range(0,len(l7)):
    l8.append(0)
df.replace({'Crop':{'Sugercane':0, 'Rice':1, 'Tobacco':2, 'Wheat':3, 'Coconut':4,
'Yam':5,
       'Sweetpotato':6, 'Turmeric':7, 'WaterMelon':8, 'Tapioca':9, 'Urad':10,
       'Varagu':11}},inplace=True)
X= df[l7]


y = df[["Crop"]]
np.ravel(y)
tr=pd.read_csv("tere.csv")
X_test= tr[l7]
y_test = tr[["Crop"]]
np.ravel(y_test)
def randomforest():
    from sklearn.ensemble import RandomForestClassifier
    clf4 = RandomForestClassifier()
    clf4 = clf4.fit(X,np.ravel(y))
```

```
# calculating accuracy---------------------------------------------------------------

from sklearn.metrics import accuracy_score

y_pred=clf4.predict(X_test)

print(accuracy_score(y_test, y_pred))

print(accuracy_score(y_test, y_pred,normalize=False))

# -----------------------------------------------------


terms = [State.get(),Season.get(),Tl.get(),Hl.get(),Yl.get()]


for k in range(0,len(l7)):

    for z in terms:

        if(z==l7[k]):

            l8[k]=1


inputtest = [l8]

predict = clf4.predict(inputtest)

predicted=predict[0]
```

```
    h='no'
    for a in range(0,len(cropyc)):
        if(predicted == a):
            h='yes'
            break
    if (h=='yes'):
        t2.delete("1.0", END)
        t2.insert(END, cropyc[a])
    else:
        t2.delete("1.0", END)
        t2.insert(END, "Not Found")
root = Tk()
root.title("Indian Crop Prediction")
#root.configure(background='black')
canvas = Canvas(root,width=1880,height=1253)
canvas.pack()
photo = PhotoImage(file ='im3.png')
canvas.create_image(0,0,image=photo,anchor=NW)
State = StringVar()
```

```
State.set(None)

Season = StringVar()

Season.set(None)

Tl = StringVar()

Tl.set(None)

Hl = StringVar()

Hl.set(None)

Yl = StringVar()

Yl.set(None)

Yc = StringVar()

Yc.set(None)

fn= StringVar()

vn= StringVar()

dn=StringVar()

rl= IntVar()

w2 = Label(canvas, justify=LEFT, text="Crop Prediction using Machine
Learning", fg="white", bg="red")

w2.config(font=("Elephant", 20))

w2.grid(row=1, column=0, columnspan=2, padx=100)
```

```
w2 = Label(canvas, justify=LEFT, text="Indian Agricultural Department",
fg="green")

w2.config(font=("Times Roman", 15))

w2.grid(row=2, column=0, columnspan=2, padx=100)

FNLb = Label(canvas, text="Farmer Name",fg="yellow", bg="black")

FNLb.grid(row=5, column=0, pady=15, sticky=W)

VNLb = Label(canvas, text="Village Name",fg="yellow", bg="black")

VNLb.grid(row=6, column=0, pady=15, sticky=W)

DNLb = Label(canvas, text="District Name",fg="yellow", bg="black")

DNLb.grid(row=7, column=0, pady=15, sticky=W)

stLb = Label(canvas, text="State Name",fg="yellow", bg="black")

stLb.grid(row=8, column=0, pady=15, sticky=W)

seLb = Label(canvas, text="Season Details",fg="yellow", bg="black")

seLb.grid(row=9, column=0, pady=15, sticky=W)

tlLb = Label(canvas, text="Temperature Level",fg="yellow", bg="black")

tlLb.grid(row=5, column=1, pady=15, sticky=W)

hlLb = Label(canvas, text="Humidity Level",fg="yellow", bg="black")

hlLb.grid(row=6, column=1, pady=15, sticky=W)

ylLb = Label(canvas, text="Yield level (per Hectare)",fg="yellow", bg="black")
```

```
ylLb.grid(row=7, column=1, pady=15, sticky=W)

ycLb = Label(canvas, text="Yield cost amount  (per Hectare)",fg="yellow",
bg="black")

ycLb.grid(row=8, column=1, pady=15, sticky=W)

ranfLb = Label(canvas, text="RF", fg="white", bg="red")

ranfLb.grid(row=15, column=0, pady=10, sticky=W)

OPTIONS1 = sorted(l1)

OPTIONS2 = sorted(l2)

OPTIONS3 = sorted(l3)

OPTIONS4 = sorted(l4)

OPTIONS5 = sorted(l5)

OPTIONS6 = sorted(l6)

FNEn = Entry(canvas, textvariable=fn)

FNEn.grid(row=5, column=0)

VNEn = Entry(canvas, textvariable=vn)

VNEn.grid(row=6, column=0)

DNEn = Entry(canvas, textvariable=dn)

DNEn.grid(row=7, column=0)

stEn = OptionMenu(canvas, State,*OPTIONS3)
```

```
stEn.grid(row=8, column=0)


seEn = OptionMenu(canvas, Season,*OPTIONS4)

seEn.grid(row=9, column=0)

tlEn = OptionMenu(canvas, Tl,*OPTIONS5)

tlEn.grid(row=5, column=1)

hlEn = OptionMenu(canvas, Hl,*OPTIONS6)

hlEn.grid(row=6, column=1)

ylEn = OptionMenu(canvas, Yl,*OPTIONS1)

ylEn.grid(row=7, column=1)

ycEn = OptionMenu(canvas, Yc,*OPTIONS2)

ycEn.grid(row=8, column=1)

rnf = Button(canvas, text="Check the result",
command=randomforest,bg="cyan",fg="green")

rnf.grid(row=15, column=1,padx=10)

t2 = Text(canvas, height=1, width=40,bg="orange",fg="black")

t2.grid(row=15, column=0 , padx=10)

root.mainloop()
```

MODULE 7 CROP YIELD AND CROP PREDICTION:

```
from tkinter import *

import numpy as np

import pandas as pd

df = pd.read_csv("re1.csv")

df

df.shape

df.columns

l1=['Sugercane', 'Rice', 'Tobacco', 'Wheat', 'Coconut', 'Yam',

       'Sweetpotato', 'Tapioca', 'Turmeric', 'WaterMelon', 'Urad', 'Varagu',

       'Banana']

l2=['DINDIGUL', 'THE NILGIRIS', 'KARUR', 'KRISHNAGIRI', 'MADURAI',

       'NAGAPATTINAM', 'NAMAKKAL', 'PERAMBALUR', 'PUDUKKOTTAI',
'SALEM',

       'THANJAVUR', 'THENI', 'THIRUVARUR', 'TIRUCHIRAPPALLI',
'TIRUNELVELI',

       'TIRUPPUR', 'TIRUVANNAMALAI', 'VELLORE', 'VILLUPURAM',
'VIRUDHUNAGAR',

       'COIMBATORE', 'CUDDALORE', 'DHARMAPURI', 'ERODE',
'KANNIYAKUMARI',
    'KANCHIPURAM', 'ARIYALUR', 'RAMANATHAPURAM',
'SIVAGANGA', 'THIRUVALLUR',

    'TUTICORIN']

l4=[ 'Summer', 'Winter', 'Kharif', 'Rabi', 'Whole Year',

    'Autumn']

l5=['TA50', 'TA60', 'TA80']

l6=['HB25', 'HA25', 'HA60', 'HA80']

df['Class'].unique()

cropyc=['Cost Yield is ten tons and Cost Production is ten lakshs',

    'Cost Yield is one ton and Cost Production is twenty thousand',

    'Cost Yield is twenty tons and Cost Production is fourty lakshs',

    'Cost Yield is two tons and Cost Production is fourty thousand',

    'Cost Yield is three tons and Cost Production is sixty thousand',

    'Cost Yield is six tons and Cost Production is six lakshs ',

    'Cost Yield is two tons and Cost Production is fourty  thousand',

    'Cost Yield is five tons and Cost Production is one lakshs',

    'Cost Yield is ten tons and Cost Production is two lakshs']

l7=['Sugercane', 'Rice', 'Tobacco', 'Wheat', 'Coconut', 'Yam',

    'Sweetpotato', 'Tapioca', 'Turmeric', 'WaterMelon', 'Urad', 'Varagu',
        'Banana','DINDIGUL', 'THE NILGIRIS', 'KARUR', 'KRISHNAGIRI', 
'MADURAI',
        'NAGAPATTINAM', 'NAMAKKAL', 'PERAMBALUR', 'PUDUKKOTTAI', 
'SALEM',
        'THANJAVUR', 'THENI', 'THIRUVARUR', 'TIRUCHIRAPPALLI', 
'TIRUNELVELI',
        'TIRUPPUR', 'TIRUVANNAMALAI', 'VELLORE', 'VILLUPURAM', 
'VIRUDHUNAGAR',
        'COIMBATORE', 'CUDDALORE', 'DHARMAPURI', 'ERODE', 
'KANNIYAKUMARI',
        'KANCHIPURAM', 'ARIYALUR', 'RAMANATHAPURAM', 
'SIVAGANGA', 'THIRUVALLUR',
        'TUTICORIN','Summer', 'Winter', 'Kharif', 'Rabi', 'Whole Year',
        'Autumn','TA50', 'TA60', 'TA80','HB25', 'HA25', 'HA60', 'HA80']
l8=[]
for x in range(0,len(l7)):
    l8.append(0)
df.replace({'Class':{'Cost Yield is ten tons and Cost Production is ten lakshs':0,
        'Cost Yield is one ton and Cost Production is twenty thousand':1,
        'Cost Yield is twenty tons and Cost Production is fourty lakshs':2,
        'Cost Yield is two tons and Cost Production is fourty thousand':3,
      'Cost Yield is three tons and Cost Production is sixty thousand':4,

      'Cost Yield is six tons and Cost Production is six lakshs ':5,

      'Cost Yield is two tons and Cost Production is fourty  thousand':6,

      'Cost Yield is five tons and Cost Production is one lakshs':7,

      'Cost Yield is ten tons and Cost Production is two lakshs':8}},inplace=True)

X= df[l7]

y = df[["Class"]]

np.ravel(y)

tr=pd.read_csv("tere1.csv")

X_test= tr[l7]

y_test = tr[["Class"]]

np.ravel(y_test)

def randomforest():

    from sklearn.ensemble import RandomForestClassifier

    clf4 = RandomForestClassifier()

    clf4 = clf4.fit(X,np.ravel(y))


    # calculating accuracy-----------------------------------------------------------------

    from sklearn.metrics import accuracy_score
```

```
y_pred=clf4.predict(X_test)

print(accuracy_score(y_test, y_pred))

print(accuracy_score(y_test, y_pred,normalize=False))

# -----------------------------------------------------

terms = [Dist.get(),Season.get(),Tl.get(),Hl.get(),Crop.get()]

for k in range(0,len(l7)):

    for z in terms:

        if(z==l7[k]):

            l8[k]=1

inputtest = [l8]

predict = clf4.predict(inputtest)

predicted=predict[0]

h='no'

for a in range(0,len(cropyc)):

    if(predicted == a):

        h='yes'

        break

if (h=='yes'):

    t2.delete("1.0", END)
```

```
        t2.insert(END, cropyc[a])

    else:

        t2.delete("1.0", END)

        t2.insert(END, "Not Found")

root = Tk()

root.configure(background='black')

Dist = StringVar()

Dist.set(None)

Season = StringVar()

Season.set(None)

Crop =StringVar()

Crop.set(None)

Tl = StringVar()

Tl.set(None)

Hl = StringVar()

Hl.set(None)

Yl = StringVar()

Yl.set(None)

fn= StringVar()
```

```
vn= StringVar()

rl= IntVar()

w2 = Label(root, justify=LEFT, text="Prediction of Yield and Yield cost by Crop
using Machine Learning", fg="white", bg="red")

w2.config(font=("Elephant", 20))

w2.grid(row=1, column=0, columnspan=2, padx=100)

w2 = Label(root, justify=LEFT, text="Tamilnadu Agricultural Department",
fg="green")

w2.config(font=("Times Roman", 15))

w2.grid(row=2, column=0, columnspan=2, padx=100)

FNLb = Label(root, text="Farmer Name",fg="yellow", bg="black")

FNLb.grid(row=5, column=0, pady=15, sticky=W)

VNLb = Label(root, text="Village Name",fg="yellow", bg="black")

VNLb.grid(row=6, column=0, pady=15, sticky=W)

RFLb = Label(root, text="Rainfall Value",fg="yellow", bg="black")

RFLb.grid(row=8, column=1, pady=15, sticky=W)


dtLb = Label(root, text="District Name",fg="yellow", bg="black")

dtLb.grid(row=8, column=0, pady=15, sticky=W)
```

```
seLb = Label(root, text="Season Details",fg="yellow", bg="black")

seLb.grid(row=9, column=0, pady=15, sticky=W)

tlLb = Label(root, text="Temperature Level",fg="yellow", bg="black")

tlLb.grid(row=5, column=1, pady=15, sticky=W)

hlLb = Label(root, text="Humidity Level",fg="yellow", bg="black")

hlLb.grid(row=6, column=1, pady=15, sticky=W)

crLb = Label(root, text="Crop Details",fg="yellow", bg="black")

crLb.grid(row=7, column=1, pady=15, sticky=W)

ranfLb = Label(root, text="RandomForest", fg="white", bg="red")

ranfLb.grid(row=15, column=0, pady=10, sticky=W)

OPTIONS1 = sorted(l1)

OPTIONS2 = sorted(l2)

OPTIONS4 = sorted(l4)

OPTIONS5 = sorted(l5)

OPTIONS6 = sorted(l6)

FNEn = Entry(root, textvariable=fn)

FNEn.grid(row=5, column=0)

VNEn = Entry(root, textvariable=vn)

VNEn.grid(row=6, column=0)
```

```
RFEn = Entry(root, textvariable=rl)

RFEn.grid(row=8, column=1)

crEn = OptionMenu(root, Crop,*OPTIONS1)

crEn.grid(row=7, column=1)

dtEn = OptionMenu(root, Dist,*OPTIONS2)

dtEn.grid(row=8, column=0)

seEn = OptionMenu(root, Season,*OPTIONS4)

seEn.grid(row=9, column=0)

tlEn = OptionMenu(root, Tl,*OPTIONS5)

tlEn.grid(row=5, column=1)

hlEn = OptionMenu(root, Hl,*OPTIONS6)

hlEn.grid(row=6, column=1)

rnf = Button(root, text="Check the result",
command=randomforest,bg="cyan",fg="green")

rnf.grid(row=15, column=1,padx=10)

t2 = Text(root, height=1, width=70,bg="orange",fg="black")

t2.grid(row=15, column=0 , padx=10)

root.mainloop()
```

CHAPTER 7

TESTING

7.1 INTRODUCTION

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing

7.2TYPES OF TESTS

7.2.1UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING:

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

BASIS PATH TESTING:

- Flow graph notation
- Kilometric complexity
- Deriving test cases
- Graph matrices Control

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box

.you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.2.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 8

CONCLUSION

8.1 APPLICATIONS:

- It is an integrated farm management application using mobile app.
- Agricultural sector to automate to identify the crop prediction process (real time world) and predicting by desktop application / web application.

8.2FUTURE ENHANCEMENT:

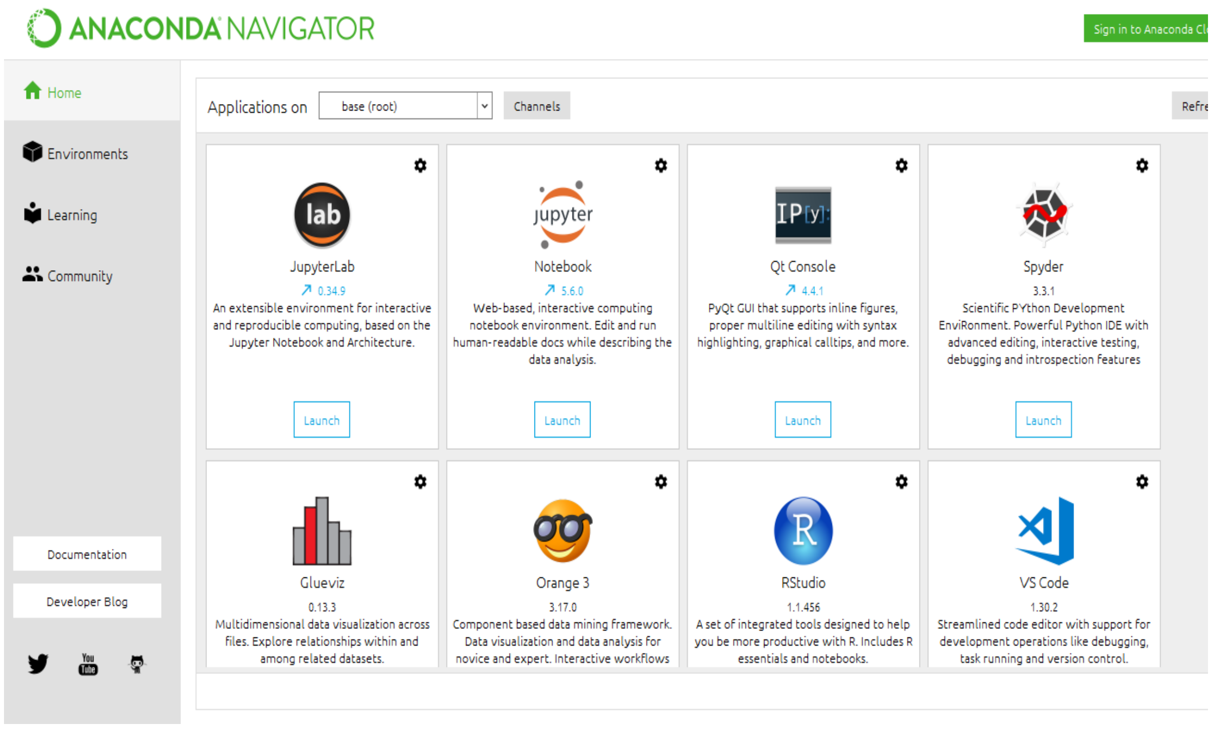
- Agricultural department wants to automate the detecting the yield crops from eligibility process (real time).
- To automate this process by show the prediction result in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.

8.3 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.

A.1 SAMPLE SCREENS

SOFTWARE INVOLMENT STEPS:



JupyterLab
↗ 0.34.9
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
↗ 5.6.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
↗ 4.4.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.3.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

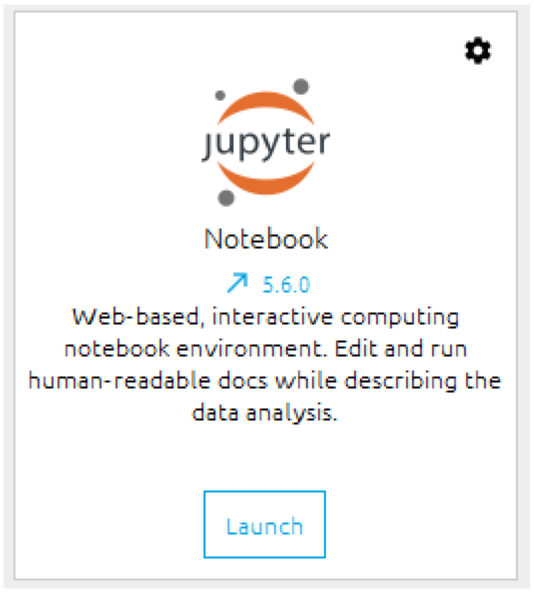
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.17.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
1.30.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Fig: Open the anaconda navigator

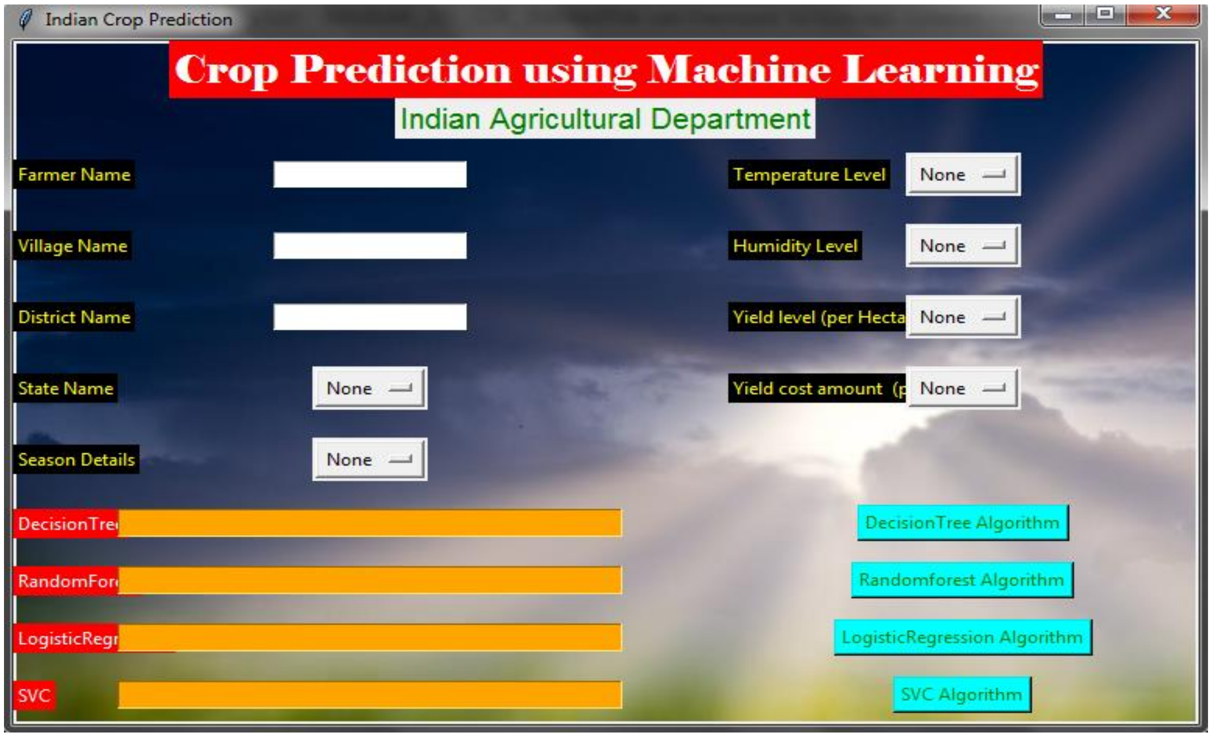


Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Fig: Launch the jupyter notebook platform

TESTING RESULTS:

INPUT:



Indian Crop Prediction

Crop Prediction using Machine Learning

Indian Agricultural Department

Farmer Name		Temperature Level	None
Village Name		Humidity Level	None
District Name		Yield level (per Hecta	None
State Name	None	Yield cost amount (p	None
Season Details	None		

DecisionTre

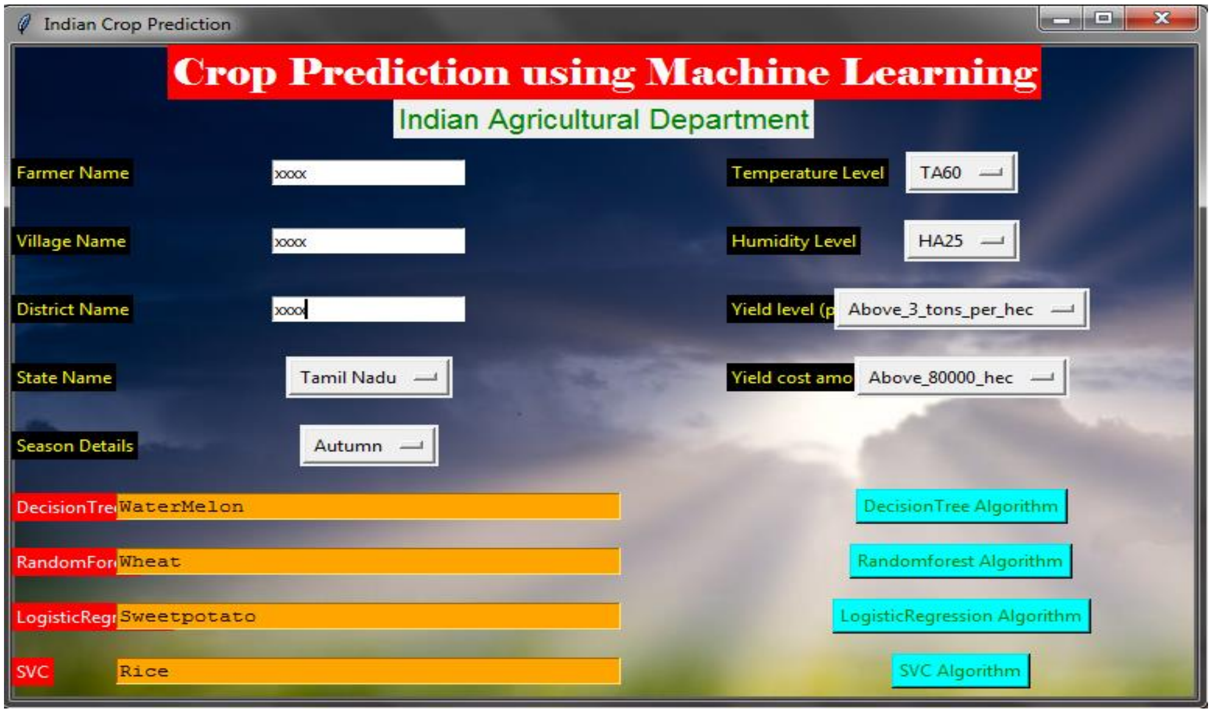
RandomFor

LogisticRegr

SVC

OUTPUT:

TEST 01:



Indian Crop Prediction

Crop Prediction using Machine Learning

Indian Agricultural Department

Farmer Name: xxxx

Village Name: xxxx

District Name: xxxx

State Name: Tamil Nadu

Season Details: Autumn

Temperature Level: TA60

Humidity Level: HA25

Yield level (p: Above_3_tons_per_hec

Yield cost amo: Above_80000_hec

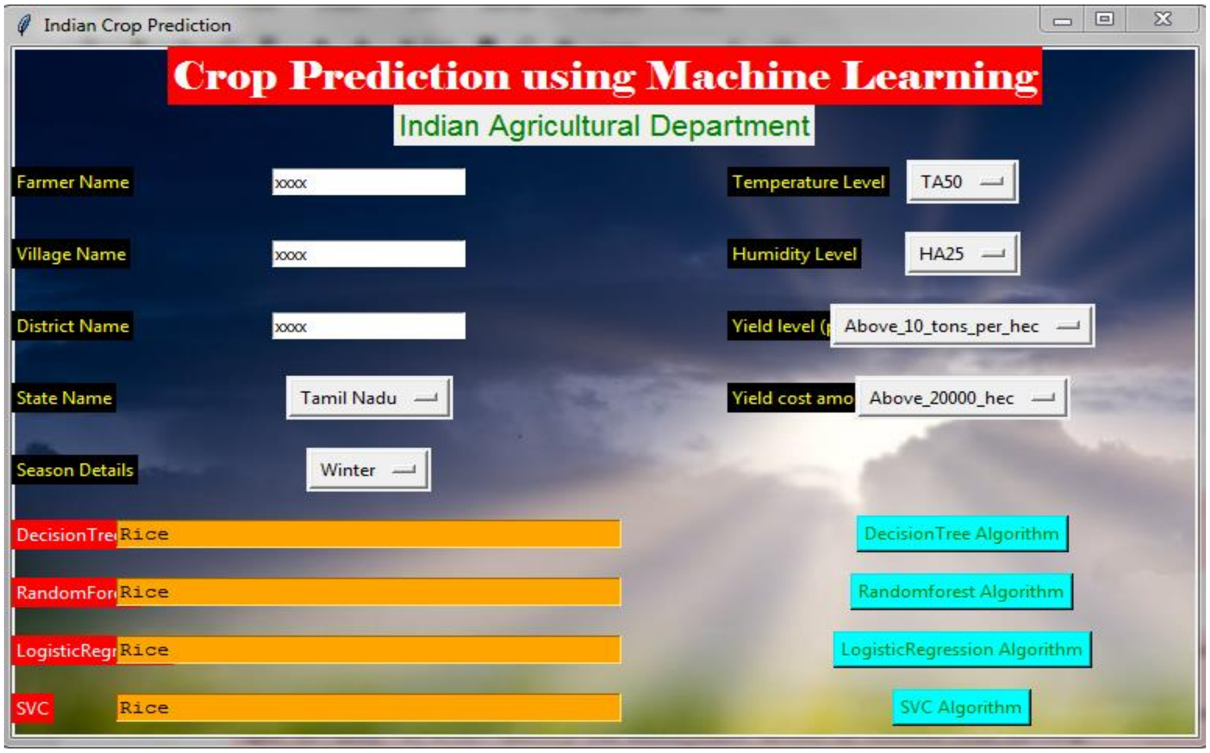
DecisionTre: WaterMelon

RandomFor: Wheat

LogisticRegr: Sweetpotato

SVC: Rice

TEST 02:



Indian Crop Prediction

Crop Prediction using Machine Learning

Indian Agricultural Department

Farmer Name: xxxx

Village Name: xxxx

District Name: xxxx

State Name: Tamil Nadu

Season Details: Winter

Temperature Level: TA50

Humidity Level: HA25

Yield level (: Above_10_tons_per_hec

Yield cost amo: Above_20000_hec

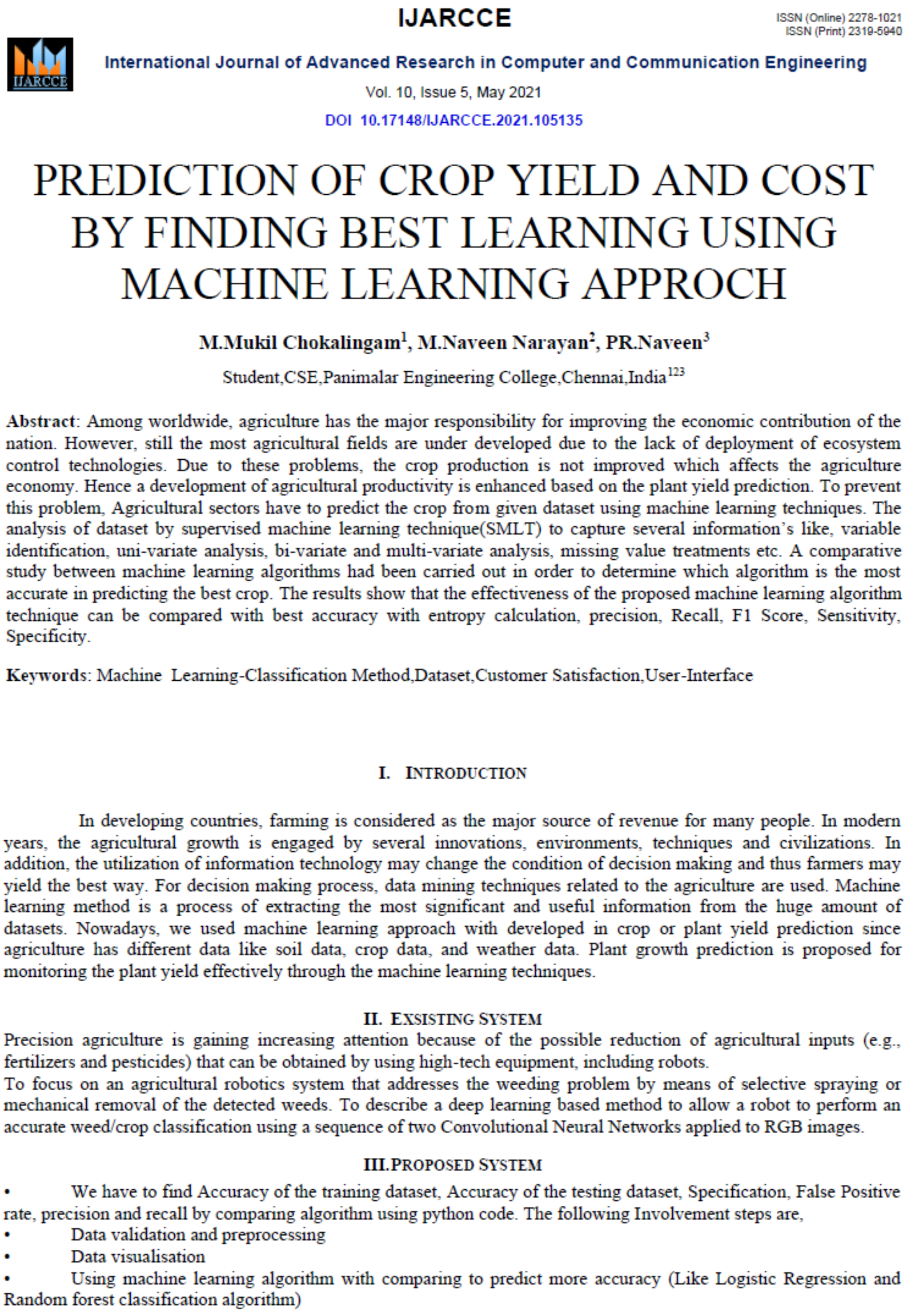
DecisionTre: Rice

RandomFor: Rice

LogisticRegr: Rice

SVC: Rice

A.2 PUBLICATIONS



IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.105135

PREDICTION OF CROP YIELD AND COST BY FINDING BEST LEARNING USING MACHINE LEARNING APPROCH

M.Mukil Chokalingam[1], M.Naveen Narayan[2], PR.Naveen[3]

Student,CSE,Panimalar Engineering College,Chennai,India[123]

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 analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments etc. 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. The results show that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with entropy calculation, precision, Recall, F1 Score, Sensitivity, Specificity.

Keywords: Machine Learning-Classification Method,Dataset,Customer Satisfaction,User-Interface

I. 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. Machine learning method is a process of extracting the most significant and useful information from the huge amount 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.

II. EXSISTING SYSTEM

Precision agriculture is gaining increasing attention because of the possible reduction of agricultural inputs (e.g., fertilizers and pesticides) that can be obtained by using high-tech equipment, including robots.
To focus on an agricultural robotics system that addresses the weeding problem by means of selective spraying or mechanical removal of the detected weeds. To describe a deep learning based method to allow a robot to perform an accurate weed/crop classification using a sequence of two Convolutional Neural Networks applied to RGB images.

III.PROPOSED SYSTEM

- We have to find Accuracy of the training dataset, Accuracy of the testing dataset, Specification, False Positive rate, precision and recall by comparing algorithm using python code. The following Involvement steps are,
- Data validation and preprocessing
- Data visualisation
- Using machine learning algorithm with comparing to predict more accuracy (Like Logistic Regression and Random forest classification algorithm)

PROBLEM DEFINITION :

• Agriculture is the most important sector that influences the economy of India. It contributes to 18% of India's Gross Domestic Product (GDP) and gives employment to 50% of the population of India.
• People of India are practicing Agriculture for years but the results are never satisfying due to various factors that affect the crop yield. To fulfill the needs of around 1.2 billion people, it is very important to have a good yield of

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 10, Issue 5, May 2021
DOI 10.17148/IJARCCE.2021.105135

crops. Due to factors like soil type, precipitation, seed quality, lack of technical facilities etc. the crop yield is directly influenced.
• To focuses on implementing crop yield prediction system by using Machine learning techniques by doing analysis on agriculture dataset. For evaluating performance Accuracy is used as one of the factors. The classifiers are further compared with the values of Precision, Recall and F1score. Lesser the value of error, more accurate the algorithm will work. The result is based on comparison among the classifiers.

Advantages:
● Our goal is push for assisting farmers, government using our predictions. All these publications state they have done better than their competitors but there is no article or public mention of their work being used practically to assist the farmers. If there are some genuine problems in rolling out that work to next stage, then identify those problems and try solving them.
● It is targeted to those farmers who wish to professionally manage their farm by planning, monitoring and analyzing all farming activities.

Algorithm :

- Data validation and pre-processing technique
- Exploration data analysis of visualization and training a model by given attributes
- Performance measurements of logistic regression and decision tree algorithms
- Performance measurements of Support vector classifier and Random forest
- Performance measurements of KNN and Naive Bayes
- GUI based prediction of crop yield and yield cost

Software Requirements :

Operating System	: Windows / Linux
Tool	: Anaconda with Jupyter Notebook

Hardware Requirements :

Processor	: Pentium IV/III
Hard disk	: minimum 80 GB
RAM	: minimum 2 GB

Techniques Used :

Python

IV. CONCLUSION

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

V. REFERENCES

[1]D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosyst. Eng.*, vol. 114, pp. 358–371, 2013.

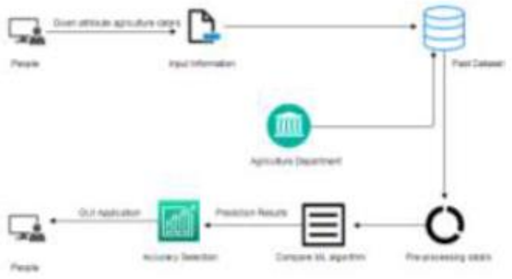
[2]S. K. Seelan, S. Laguette, G. M. Casady, and G. A. Seielstad, "Remote sensing applications for precision agriculture: A learning community approach," *Remote Sens. Environ.*, vol. 88, pp. 157–169, 2003.

[3]P. J. Pinter Jr. *et al.*, "Remote sensing for crop management," *Photogram- metric Eng. Remote Sens.*, vol. 69, pp. 647–664, 2003.

[4]L. Di, G. Y. Eugene, L. Kang, R. Shrestha, and Y.-Q. Bai, "RF-CLASS: A remote-sensing-based flood crop loss assessment cyber-service system for supporting crop statistics and insurance decision-making," *J. Integrative Agriculture*, vol. 16, pp. 408–423, 2017.

[5]M. A. Friedl *et al.*, "Global land cover mapping from MODIS: algorithms and early results," *Remote Sen. Environ.*, vol. 83, pp. 287–302, 2002

Architecture Diagram:



REFERENCES

- D. J. Mulla, “Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps,” *Biosyst. Eng.*, vol. 114, pp. 358–371, 2013.
- S. K. Seelan, S. Laguette, G. M. Casady, and G. A. Seielstad, “Remote sensing applications for precision agriculture: A learning community approach,” *Remote Sens. Environ.*, vol. 88, pp. 157–169, 2003.
- P. J. Pinter Jr. *et al.*, “Remote sensing for crop management,” *Photogram-metric Eng. Remote Sens.*, vol. 69, pp. 647–664, 2003.
- L. Di, G. Y. Eugene, L. Kang, R. Shrestha, and Y.-Q. Bai, “RF-CLASS: A remote-sensing-based flood crop loss assessment cyber-service system for supporting crop statistics and insurance decision-making,” *J. Integrative Agriculture*, vol. 16, pp. 408–423, 2017.
- M. A. Friedl *et al.*, “Global land cover mapping from MODIS: algorithms and early results,” *Remote Sen. Environ.*, vol. 83, pp. 287–302, 2002.
- X. Zhang *et al.*, “Monitoring vegetation phenology using MODIS,” *Remote Sen. Environ.*, vol. 84, pp. 471–475, 2003.
- F. Gao *et al.*, “Toward mapping crop progress at field scales through fusion of Landsat and MODIS imagery,” *Remote Sen. Environ.*, vol. 188, pp. 9–25, 2017.
- C. Boryan, Z. Yang, R. Mueller, and M. Craig, “Monitoring US agri- culture: the US department of agriculture, national agricultural statistics service, cropland data layer program,” *Geocarto Int.*, vol. 26, pp. 341–358, 2011.
- W. Han, Z. Yang, L. Di, and R. Mueller, “CropScape: A web service based application for exploring and disseminating US conterminous geospatial cropland data products for decision support,” *Comput. Electron. Agricul-*

ture, vol. 84, pp. 111–123, 2012.

- H. Xiang and L. Tian, "An automated stand-alone in-field remote sens- ing system (SIRSS) for in-season crop monitoring," *Comput. Electron. Agriculture*, vol. 78, pp. 1–8, 2011.
- Z. Sun, P. Yue, and L. Di, "GeoPWTManager: A task-oriented web geoprocessing system," *Comput. Geosci.*, vol. 47, pp. 34–45, 2012.
- Z. Sun *et al.*, "Automation of customized and near-real-time vegetation condition index generation through cyberinfrastructure-based geoprocess- ing workflows," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 11, pp. 4512–4522, Nov. 2014.
- Z. Sun, L. Di, A. Chen, P. Yue, and J. Gong, "The use of geospatial workflows to support automatic detection of complex geospatial features from high resolution images," in *Proc. 2nd Int. Conf. Agro-Geoinform. (Agro-Geoinform.)*, 2013, pp. 159–162.
- Z. Sun and P. Yue, "The use of Web 2.0 and geoprocessing services to support geoscientific workflows," in *Proc. 18th Int. Conf. Geoinform.*, 2010, pp. 1–5.
- P. Yue *et al.*, "GeoPW: Laying blocks for the geospatial processing web,"
- *Trans. GIS*, vol. 14, pp. 755–772, 2010.
- Z. Sun, H. Fang, L. Di, P. Yue, X. Tan, and Y. Bai, "Developing a web-based system for supervised classification of remote sensing images," *GeoInformatica*, vol. 20, pp. 1–21, 2016.
- Z. Sun, H. Fang, L. Di, and P. Yue, "Realizing parameterless automatic classification of remote sensing imagery using ontology engineering and cyberinfrastructure techniques," *Comput. Geosci.*, vol. 94, pp. 56–67, 2016.
- Z. Sun, H. Fang, M. Deng, A. Chen, P. Yue, and L. Di, "Regular shape similarity index: A novel index for accurate extraction of regular objects

from remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 7, pp. 3737–3748, Jul. 2015.

- M. C. You, Z. Sun, L. Di, and Z. Guo, "A web-based semi-automated method for semantic annotation of high schools in remote sensing images," in *Proc. 3rd Int. Conf. Agro-Geoinform.*, 2014, pp. 1–4.
- J. Surkan and L. Di, "Fast trainable pattern classification by a modifi- cation of Kanerva's SDM model," in *Proc. Int. Joint Conf. Neural Netw.*, 1989, pp. 347–349.