AGRO ANALYTICS

Submitted in partial fulfilment of the requirements

of the Degree of

B. E. Computer Engineering

By

Mohit Shah	**60004130094**
Mihin Sumaria	**60004130110**
Manan Shah	**60004148019**

Guide:

Mrs. Lynette D'mello

Assistant Professor

Department of Computer Engineering

D. J. Sanghvi College of Engineering

Mumbai – 400056

University of Mumbai

2016-17

CERTIFICATE

This is to certify that the project entitled **“Agro Analytics”** is a bonafide work of **Mohit Shah (60004130094), Mihin Sumaria (60004130110) and Manan Shah (60004148019)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the Degree of B. E. Computer Engineering.

Prof. Lynette D’Mello

Internal Guide

Dr. (Prof.) Narendra Shekokar

Head of Department

Dr. Hari Vasudevan

Principal

Project Report Approval for B. E.

This Project Report entitle ***Agro Analytics*** by ***Mohit Shah, Mihin Sumaria and Manan Shah*** is approved for the Degree of **B. E. Computer Engineering**.

Examiners:

1. ____________________

2. ____________________

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)	(Signature)	(Signature)
Mohit Shah	Mihin Sumaria	Manan Shah
60004130094	60004130110	60004148019

Date:

Abstract

Agricultural statistics and forecast is an important resource that the government has not explored commensurate to its impact. The aim of our project is to make this process computerized by implementing principles of data mining and analytics. More specifically, our project aims at targeting the social issue of drought, analysing data based on crop produce, amount of rainfall, agricultural inputs, irrigation, and similar factors for every crop in the state of Maharashtra.

Based on the extensive research carried out through this project, effective countermeasures and suggestions will be given, which if implemented expeditiously, can help tackling the problem of drought in our state.

Data can be mined and analysed to find various trends and relations, such as – contrast between total irrigation area and type of crop; total principal and non-principal crop amount versus district-wise rainfall etc.

The end result of the project will be research based reports specifying these trends, studied and analysed from data taken over the past few years. Actions to minimize the damage of drought will also be suggested.

Contents

List of Figures

Chapter 1

Introduction

1.1 Description

In the current scenario, the government is collecting data only in its raw form, and this data is of no use to the end user, that is the farmers. Collecting this raw data, standardizing it, analysing it, and feeding it to a system that will provide relational trends is the aim of our project.

These relational trends will act as solutions for farmers, especially in drought afflicted areas. For example, the cultivation of Kharif Rice is resource intensive and should be only done in a rainfall rich period, otherwise it could use up the natural reserves of ground water, leading to deficiency in the water table, which consequently leads to drought like conditions. This example, as naïve as it may sound, represents a broad class of trends which when extrapolated efficiently using the existing data mining algorithms, can produce a plethora of solutions, which when adhered to, will help alleviate the aforementioned drought like conditions.

Another example could be of the crop Rabi Maize. Rabi crops or Rabi harvest are agricultural crops sown in winter and harvested in the spring. The Rabi crops are sown around mid-November, after the monsoon rains are over, and harvesting begins in the months of April or May. The crops are grown either with rainwater that has percolated into the ground, or with irrigation. A good rain in winter spoils the Rabi crops but is good for kharif crops. Our project aims at identifying specific areas that are suitable for specific crops, so as to avoid deficit of crops.

If you look at these examples, you will see that the amount of rainfall that occurs in an area, the crop produced in that area, the season and other such parameters, are all correlated and can form trends that can give us solutions for suitable farming practices that can help farmers successfully avoid drought-like situations.

The ultimate objective of this project is to stand as a system that when fed with data regarding various parameters, successfully produces trends and correlations that can help the user of the system develop solutions to tackle or minimize the damage of drought. Computerization of this process will drastically reduce the time required to study patterns and carry out extensive research to generate reports, and will give a close estimation of the required outcome.

1.2 Problem Formulation

The practice of farming is one of the major occupations in our country, and a major produce of a variety of crops come from the state of Maharashtra. With an enormous agricultural sector in the state, the climatic conditions prevailing here are somewhat contradictory to what is required.

Maharashtra is a drought ridden state, with one of the most devastating droughts occurring only in the recent past. Drought is a prolonged period of abnormally low rainfall leading to a scarcity of water for human use and farming. Of late, and rightly so, the rapid degeneration of Maharashtra in the wake of drought, increasing farmer suicides and the water crisis in the state is being splashed across every newspaper, in India and abroad. It’s disheartening, to say the least! It’s being labelled as a ‘man-made’ drought. But this phenomenon isn’t a recent one. It finds its beginnings in years prior to 2012 that led to a disastrous drought that year. From then on, it’s been five long, consecutive years of depleting water, indebtedness and farmers killing themselves, with no end in sight.

Data Mining is an emerging research field in agricultural crop yield analysis. In this project, our focus is on the applications of Data Mining techniques in agricultural field. Different Data Mining techniques are in use, such as K-Means, K-Nearest Neighbour (KNN) and Support Vector Machines (SVM) for very recent applications of Data Mining techniques in agricultural field. In this project, consider the problem of predicting yield production. Yield prediction is a very important agricultural problem that remains to be solved based on the available data. The problem of yield prediction can be solved by employing Data Mining techniques. This work aims at finding suitable data models that achieve a high accuracy and a high generality in terms of yield prediction capabilities. For this purpose, different types of Data Mining techniques were evaluated on different data sets.

Currently, the government of Maharashtra and the Central Government of India are collecting data only in its raw form, and this data is of no use to the end user, that is the farmers. Collecting this raw data, standardizing it, analysing it, and feeding it to a system that will provide relational trends is the aim of our project. These relational trends will act as solutions for farmers, especially in drought afflicted areas. Using various parameters such as amount of rainfall, production of crops, area-wise statistics, agricultural inputs etc., agricultural tendencies will be formulated and a report-based system will provide solutions to farmers for eschewing droughts and sustaining development.

1.3 Motivation

Maharashtra is divided into five geographical regions, comprising six administrative divisions—Konkan, Pune, Nashik, Marathwada (Aurangabad) and Vidarbha (Amravati and Nagpur). The Vidarbha region reported the most farmer suicides, 1,541, in 2015. Nagpur (362) and Amravati (1,179) witnessed the maximum farmer suicides in this region. Vidarbha was followed by Aurangabad (1,130) that forms the Marathwada region. As many as 3,228 farmers committed suicide in Maharashtra in 2015, the highest since 2001, according to data tabled in the Rajya Sabha on March 4, 2016–that is almost nine farmers every day. Vidarbha and Marathwada, with 5.7 million farmers, accounted for 83% of all farmer suicides in Maharashtra in 2015.

The farmer suicides, which have remained unstoppable for past few years in eight districts of Marathwada, have crossed the staggering 400-mark in just over a four-month period in 2016. Compared to 2015, as many as 92 more farmers have embraced deaths in the first four and half months of 2016, highlighting the failure of the government schemes launched in August to curb the spate of suicides.

In past few months, 1,548 distressed farmers have been reported dead in the Marathwada region which is witnessing fourth successive years of drought with wells, rivers and dams having gone dry. Jayakwadi dam in Aurangabad district in Marathwada, which is witnessing the worst drought in a century, has only 1% water left of its 2.17 billion cubic metre capacity. As many as 246 districts in ten states across the country have already been declared drought-affected according to this Lok Sabha reply on March 10, 2016. Of these, 21 districts in Maharashtra, or 15,747 villages, are drought-affected. The drought in Vidarbha is more of an agriculture drought and not hydrological.

Looking at these staggering numbers, one can easily figure out that there is an undying need to formulate solutions for farmers, techniques that can be used by the farmers to produce crops and forecasts that can help farmers be better prepared for droughts. The farming sector has been hit hard by the recent situation of rainfall scarcity, and this is affecting the farmers in almost all regions of the state of Maharashtra. This truly, is a motivation behind developing a system that can learn about various attributes and provide answers instead of only storing raw data. Merging the current information system with a result-oriented application using Data Mining techniques to alleviate the effects of drought is the motivation of this project.

1.4 Proposed Solution

The system will analyze data in two phase, using a classifier for each phase. Data will be used for both, training and testing purposes. The extracted data includes records for rainfall, temperature and pressure for eight districts of Maharashtra on an average-per-month basis. 80 percent of the data will be used for training the system, and the remaining 20 percent of the data will be used for testing the accuracy of the system.

A. Phase One

As shown in figure 1.4.1, phase one of the system comprises of extraction of data related to parameters required for drought classification, which are, rainfall, temperature and pressure. This data will then be standardized to have a consistent format and loaded into the database.

The analysis of this data will produce trends and correlations with other parameters, which can be used to calculate the probability of drought and classifying it as low, medium or high.

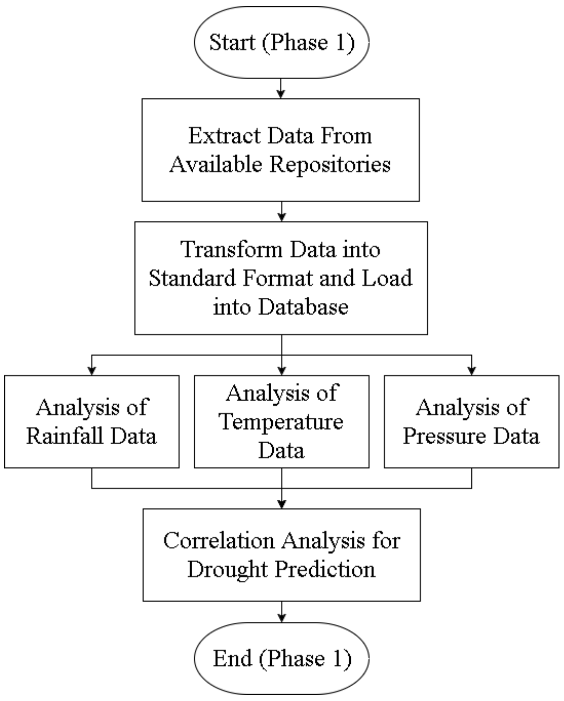

Start (Phase 1)
Extract Data From Available Repositories
Transform Data into Standard Format and Load into Database
Analysis of Rainfall Data
Analysis of Temperature Data
Analysis of Pressure Data
Correlation Analysis for Drought Prediction
End (Phase 1)

Figure 1.4.1: Expected Outcome for Phase One of Agro Analytics

This phase will be implemented using a classifier with the following attributes:

1. District (8 districts)

2. Year and Month (2011-2016, 12 months)

3. Recorded Rainfall

4. Average Rainfall

5. Recorded Temperature

6. Average Temperature

7. Recorded Pressure

8. Average Pressure

The result of this classifier will be probability of drought as low, medium or high.

B. Phase Two

As shown in figure 1.4.2, phase two of the system will obtain the result and attributes from phase one and feed it into the next classifier. Along with that, data regarding other parameters like soil type, fertilizer input etc. will also be correlated to find the Crop Growth Probability Index which will be mentioned in the form of a report.

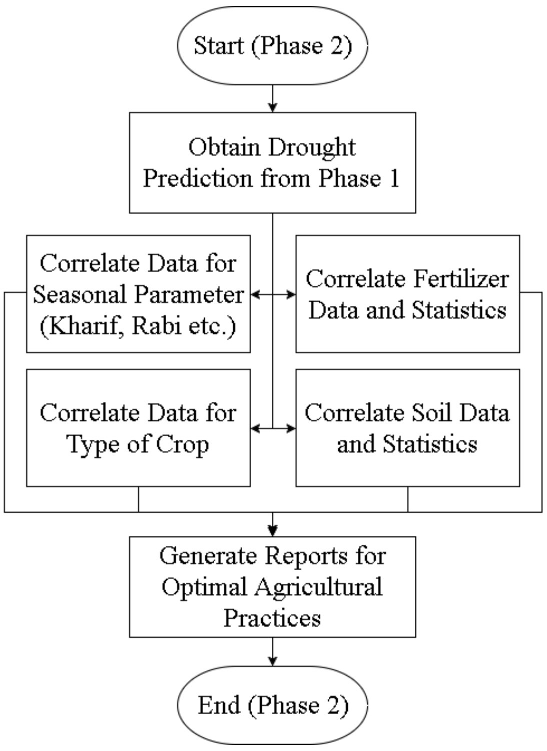

Start (Phase 2)
Obtain Drought Prediction from Phase 1
Correlate Data for Seasonal Parameter (Kharif, Rabi etc.)
Correlate Fertilizer Data and Statistics
Correlate Data for Type of Crop
Correlate Soil Data and Statistics
Generate Reports for Optimal Agricultural Practices
End (Phase 2)

Figure 1.4.2: Expected Outcome for Phase Two of Agro Analytics

This phase will be implemented using a classifier with the following attributes:

1. Attributes and Results from Classifier for Phase One
2. Crop Type (5 crops)
3. Available Soil Type and Nutrients
4. Required Soil Type and Nutrients
5. Required Rainfall
6. Required Temperature

1.5 Scope of the Project

As mentioned earlier, currently, the government is collecting data only in its raw form, and this data is of no use to the end user, that is the farmers. Collecting this raw data, standardizing it, analysing it, and feeding it to a system that will provide relational trends is the scope of our project.

This project is the demonstration of a promising system that is only aimed at helping farmers in drought afflicted areas. The present research has mainly revolved around collection and cataloguing of data in its raw form. This data can first be standardized and the analyzed to find various trends that can help build solutions for farmers.

This project is suitable to be used in areas that have similar situations. In India, states like Maharashtra, Madhya Pradesh, Gujarat etc. have been facing similar situations in terms of droughts and agricultural produce. Extracting accurate data and implementing correct algorithms to that data via this system can help formulate solutions to get rid or lessen the effects of droughts in these states.

Droughts affect our farmers the most, as is made evident by the staggering number of farmers' suicide cases in the state of Maharashtra, due to droughts. Any individual in a constituency must be looked after by its representative. Thus, this issue must be utmost importance to the government. To help reduce the sufferings of farmers, governments of various states and local bodies of a number of districts can make use of this system to generate reports and find solutions for their farmers.

Chapter 2

Review of Literature

In our review of relevant literature, for use of classification algorithms in agriculture, following are the papers that we have come across. A brief overview for each paper is given in the following section:

These papers make use of large datasets of soil and weather for agricultural decision-making by making use of classifiers like Naïve Bayes and Random Forests. The data for soil classification was acquired from the FAO/UNESCO and the ORSTROM/INRA systems, which are based on the USDA soil taxonomy. The data is available in a ready-format for a comprehensive soil classification system, which was developed with international cooperation. Statistical analysis and data mining was performed on a relevant subset of the soil database, collected from the various regions of Kanchipuram, Chandragiri, Mandalm Chittoor districts, taking 1500 samples from a total of 2045 samples. The data was stored in an excel sheet, which was then formatted to replace any null or missing values. This file was now encoded to a comma delimited (CSV) file format for further processing. The paper concludes by saying that the classification algorithms, among which Naïve Bayes gave the best results, can be used for soil profile classification, verification of valid patterns among soil data. [1]

This paper makes use of data of rural labour, arable land, and crop output for building a decision tree, and a set of spatial classification rules. It goes onto first illustrate how classification rule mining is done for a given dataset, using a decision tree. The decision tree algorithm that is made use of is the Iterative Dichotomiser 3 (ID3). The experimental data is a total output value of the aforementioned parameters, for 30 provinces and cities in China. Cluster analysis, using association method, on this data, with Euclidean distance as a parameter, is used, to label the data. This labelled data is used with the ID3 algorithm, and an inverted tree, which gives specific rules for predicting crop productivity, is generated. [3]

This paper makes use of spatial data mining based on k-means clustering to analyse rainfall and temperature data, for improvement of crop outputs. The paper first illustrates the basic steps of data mining, which are: data cleaning, data integration, data selection, data transformation, data mining and knowledge representation. It further goes on to explain how spatial data mining is potentially useful for extracting patterns from large datasets, and also explains the steps involved in k-means clustering. The paper concludes by providing visualizations of the cluster analysis for temperature and rainfall, and that the results can be improved by considering more parameters. [4]

This paper proposes an architecture for prediction of crop production output, based upon crop, soil and soil inputs, weather, etc. The proposed system suggests the use of C4.5, Naïve Bayes, ANNs and Decision trees for production of rules and classification of test data. The system will help in improving agriculture by making use of predictions based on historical data of the aforementioned parameters, depending on the season a crop is being grown in, and the weather at particular period of time. The goal of this research paper is to maximize the gains of the farmers by optimizing crop yield. [6]

This paper makes use of density based clustering and multiple linear regression for estimation of crop production in the East Godavari District, using rainfall, land, fertilizers and crop output data. The data that is made use of in this paper, are from the years 1955 to 2009 for the previously stated district of Andhra Pradesh. The input variables are as follows - 'Year', 'Rainfall', 'Area of Sowing', 'Yield', 'Fertilizers' (Nitrogen, Phosphorous and Potassium) and 'Production’. Multiple Linear Regression finds the linear relationship between dependent and independent variables, and is based on the least squares method. Density-based clustering is a type of clustering that works with a given threshold, i.e. the density of the neighbourhood does not exceed given threshold, for any given cluster. These two algorithms are made use of to perform region specific crop yield analysis, and the crop yield is estimated for 40 years’ intervals of sample data as input. The resulting output gives us an error rate ranging from -14% to +13% for the given interval. [7]

This paper is a brief survey of data mining techniques that can be used, and have been used in the past for agricultural datasets. The paper further provides an overview of the following data mining techniques – Classification, Clustering, Association Rule Mining, and Regression. The paper goes onto illustrate several examples for these algorithms, one of them being as follows. Support Vector Machines can be used in case of crop classification, in case of changing weather scenarios. [9]

After surveying the above papers, classification algorithms seem the most appropriate type of algorithms to implement our proposed solution. The algorithms that we will be making use of are – Decision Tree – Iterative Dichotomiser 3 (ID3), Random Forest Classifier, and Support

Vector Machines – Radial Basis Function, due to their ability to work with agricultural data, as seen in the aforementioned papers, in case of changing weather scenarios.

Chapter 3

System Analysis

3.1 Functional Requirements

The functional requirements for our project are as stated below:

1. Classification of Districts: The system will need a preliminary classification of districts based on the area of cultivation, average rainfall, crop produce and agricultural inputs, in the form of training labels. The system will then learn using these training labels, and should be then able to classify the given districts into separate classes for individual crops.

2. Crop suggestion: The system should be able to suggest new crops for different districts, so as to maximize crop output based on that district's parameters. It should also show the degree of crop favourability for a particular district based on the parameters discussed earlier, in terms of 'High', 'Medium', and 'Low'.

3. Season-wise predictions: The system should be able to identify suitable seasons for every crop, and the corresponding trend for that crop in a given district. For example, the system should be able to suggest Kharif crops such as Rice, Maize, Bajra if the rainfall in a given district is similar to the typical conditions observed in the Kharif Season.

4. Rainfall Prediction: The main aim of the project is to alert the end-users, the farmers, of any upcoming drought like situations so that they can prepare themselves. The system should be able to analyse the rainfall data, and predict any erratic changes in the rainfall, or any major deviations from the standard amount for that particular month, for that particular district.

3.2 Non-Functional Requirements

The non-functional requirements of the system are as follows:

1. Reliability: The system, in case of failure, should not experience a loss of data, or any of the acquired learning from past training sessions.

2. Performance: The system should not take up a lot of resources for the training or testing sessions, to generate the output in form of prediction, and should run smoothly on the platform that it is installed on. Also, it should give accurate results.

3. Availability: The system should be in a specified state before the start of an operation at any random time.

4. Scalability: The system is going to be used for the state of Maharashtra first, but should be scalable to accommodate other states facing similar drought conditions in later stages, and should also account for the change in the agricultural practices of the state.

5. Maintainability: On the occurrence of a failure in the system, the time required to restore it to its former state should be less.

3.3 Specific Requirements

Hardware Requirements:

The project is a software solution and does not require any specific hardware unit, apart from the following:

1. Standard computer set that can perform statistical analysis, with:
   - Keyboard
   - Mouse
2. Processor: Intel Core II Duo or higher.
3. Required RAM: 32 MB min.
4. Required Memory: 16 GB min.

Software Requirements:

The following software needs to be installed on the host computer to run the system.

1. The system will require Python and its various libraries such as, NumPy, MatPlotLib, Seaborn, Pandas, Scikit-learn, BeautifulSoup. Pandas will be used to convert the standardized data in the .csv format to arrays that can be operated upon, to find correlation, variance, etc. for the given data. MatPlotLib will be used to visualize the trends that will be discovered as a result of the analysis, and Seaborn will give aesthetic value to these trends.

2. Microsoft Excel will be used to store the agricultural data in a standardized format, for each district and crop.

3. Scikit-learn will be used to implement learning algorithms for our project, namely Random Forest, Support Vector Machine and Iterative Dichotomiser 3.

3.4 Use Case Diagram and Description

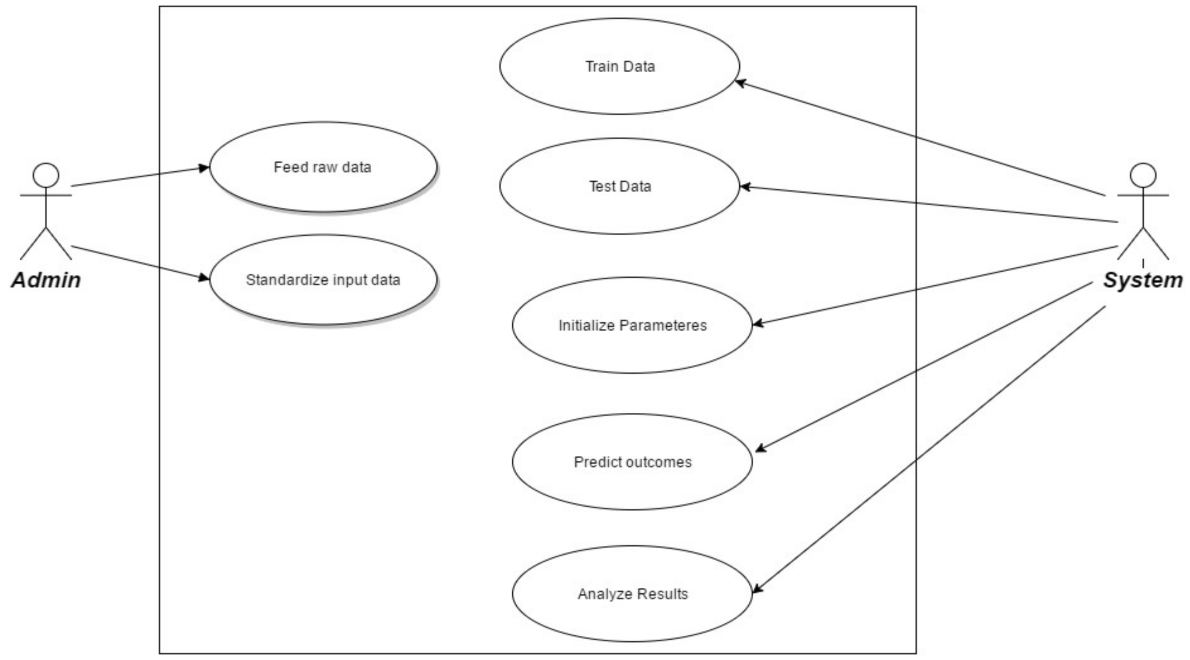


Figure 3.4.1: Use Case Diagram

The use case diagram shows the main functions performed by the admin, as well as the system. Each use case represents a sub-task that is performed to compute the results. The description of the diagram is as follows:

• The admin standardizes the data available from the surveys, to a machine-readable format, so that it can be operated upon using python and its libraries. The data will then be fed to the system for further processing. The system will now divide this data into two data sets – training and testing data.

• Training labels, provided by the user as a part of the standardization, will be used to form a model which will be used for the testing data.

• The system will be initializing parameters for the model based on the training data.

• The testing data will be classified according to the training results and the parameters initialized in the previous step, by the system.

• The system will now classify outcomes, and analyse results to provide viable solutions to the end-users.

Chapter 5

Design

5.1 Architectural Design

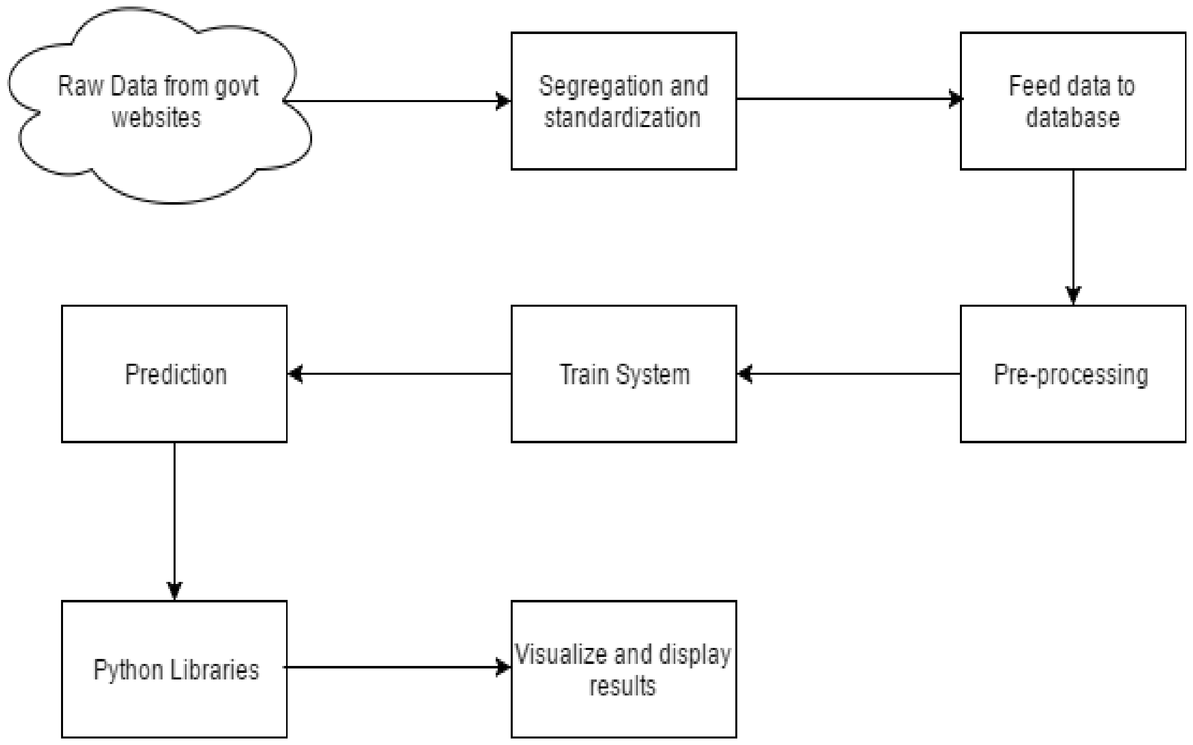

Raw Data from govt websites
Segregation and standardization
Feed data to database
Pre-processing
Train System
Prediction
Python Libraries
Visualize and display results

Figure 5.1.1: Architectural Design

The design of the flow of the system is as follows:

• Raw data is collected.

• Data is segregated and standardized.

• It is fed to the database; we will be using Excel as ours.

• Its sent for pre-processing where the data is fed next to train the system.

• The algorithms are used to predict results.

• Python scripts and libraries will be used to then Visualize the obtained results for a better understanding of trends.

5.2 User Interface Design

As such, Agro Analytics is a backend application and has little significance of User Interface Design. The data that we have used and the results that we obtained have been incorporated in a website accessible at http://www.agroanalytics.info.

The Home page displays all options for data and results and also provides navigation to references used in creating this system.

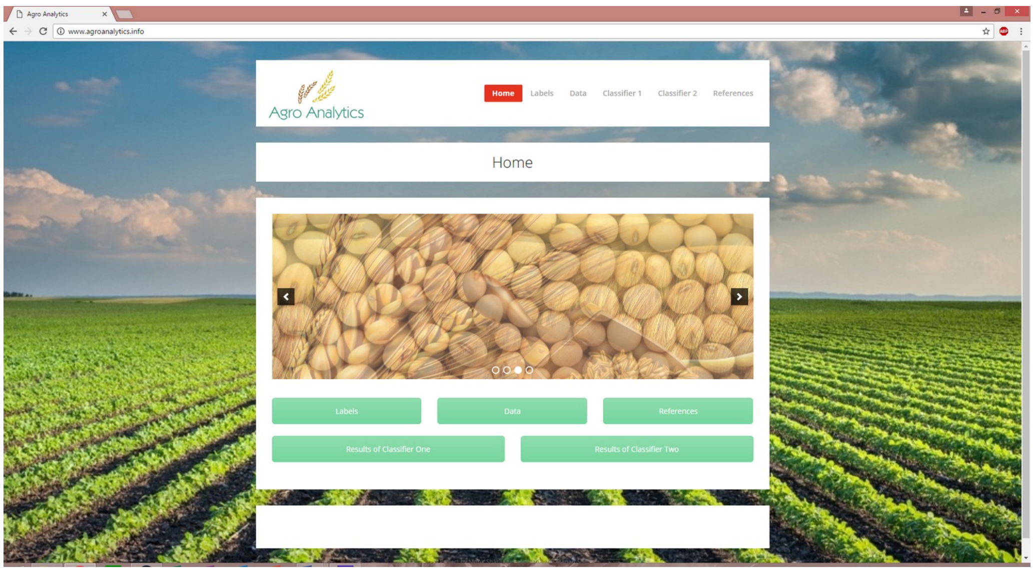
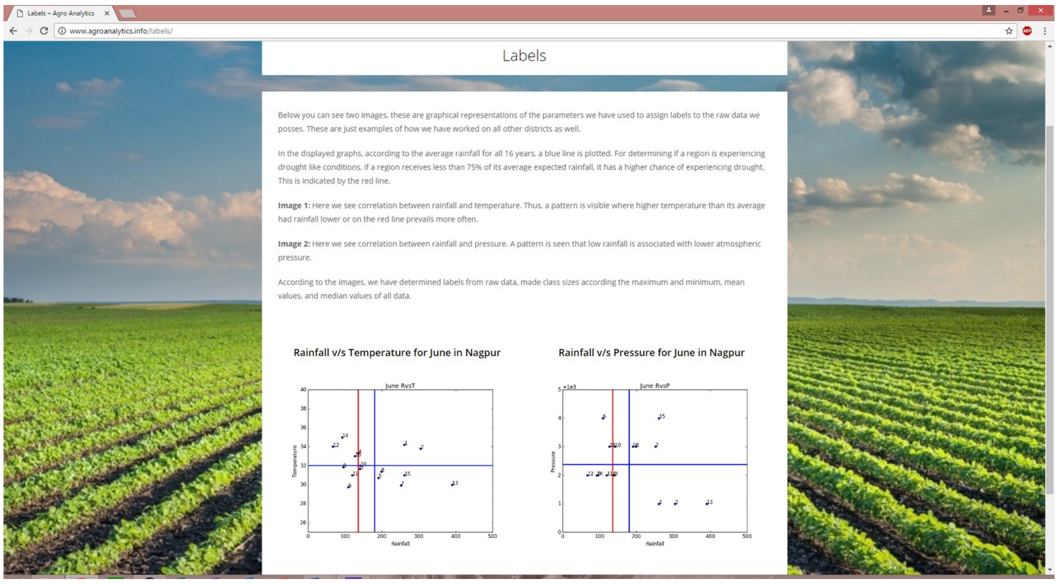


Figure 5.2.1 Home Page

According to the images, we have determined labels from raw data, made class sizes according the maximum and minimum, mean values, and median values of all data.



Labels

Below you can see two images, these are graphical representations of the parameters we have used to assign labels to the raw data we posses. These are just examples of how we have worked on all other districts as well.

In the displayed graphs, according to the average rainfall for all 16 years, a blue line is plotted. For determining if a region is experiencing drought like conditions, if a region receives less than 75% of its average expected rainfall, it has a higher chance of experiencing drought. This is indicated by the red line.

Image 1: Here we see correlation between rainfall and temperature. Thus, a pattern is visible where higher temperature than its average had rainfall lower or on the red line prevails more often.

Image 2: Here we see correlation between rainfall and pressure. A pattern is seen that low rainfall is associated with lower atmospheric pressure.

According to the images, we have determined labels from raw data, made class sizes according the maximum and minimum, mean values, and median values of all data.

Rainfall v/s Temperature for June in Nagpur

Rainfall v/s Pressure for June in Nagpur

Figure 5.2.2 Labels Page

Data that we have used was obtained in its raw format and was then pre-processed for labeling all, rainfall, temperature and pressure data.

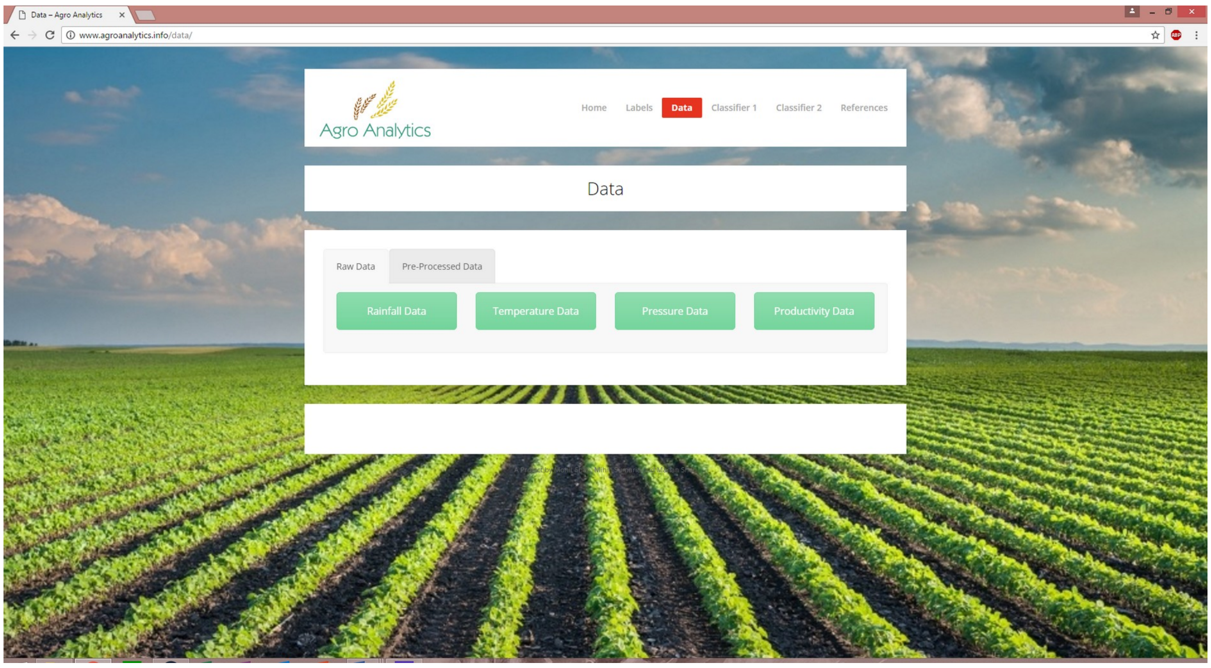
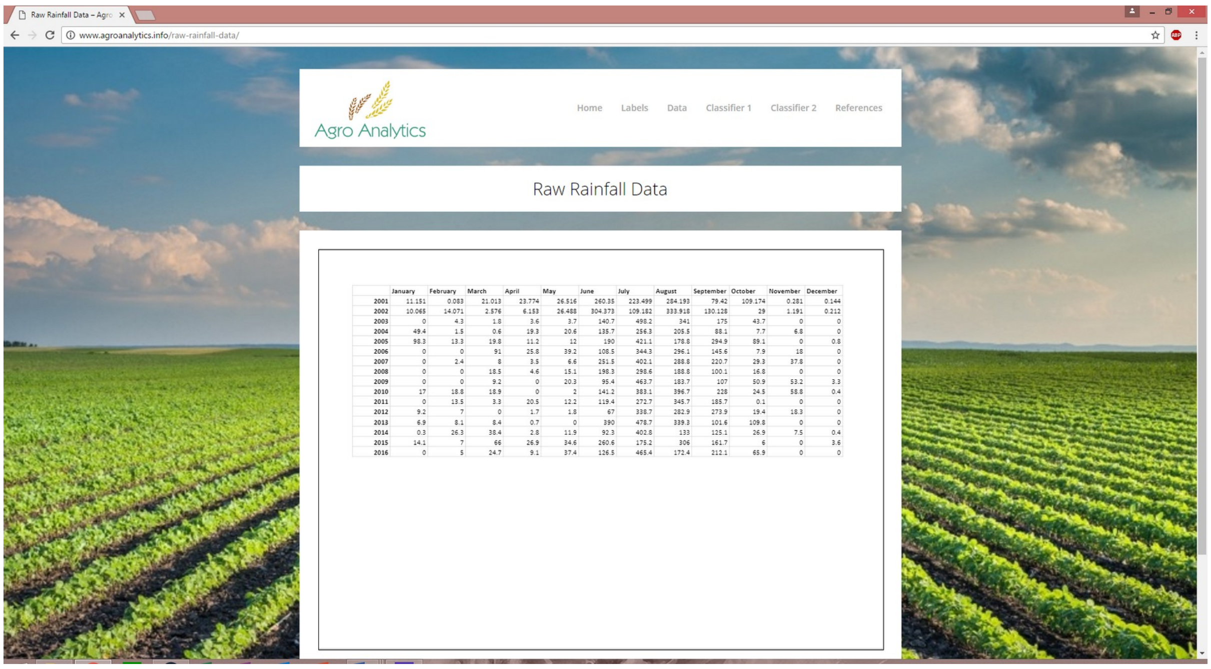


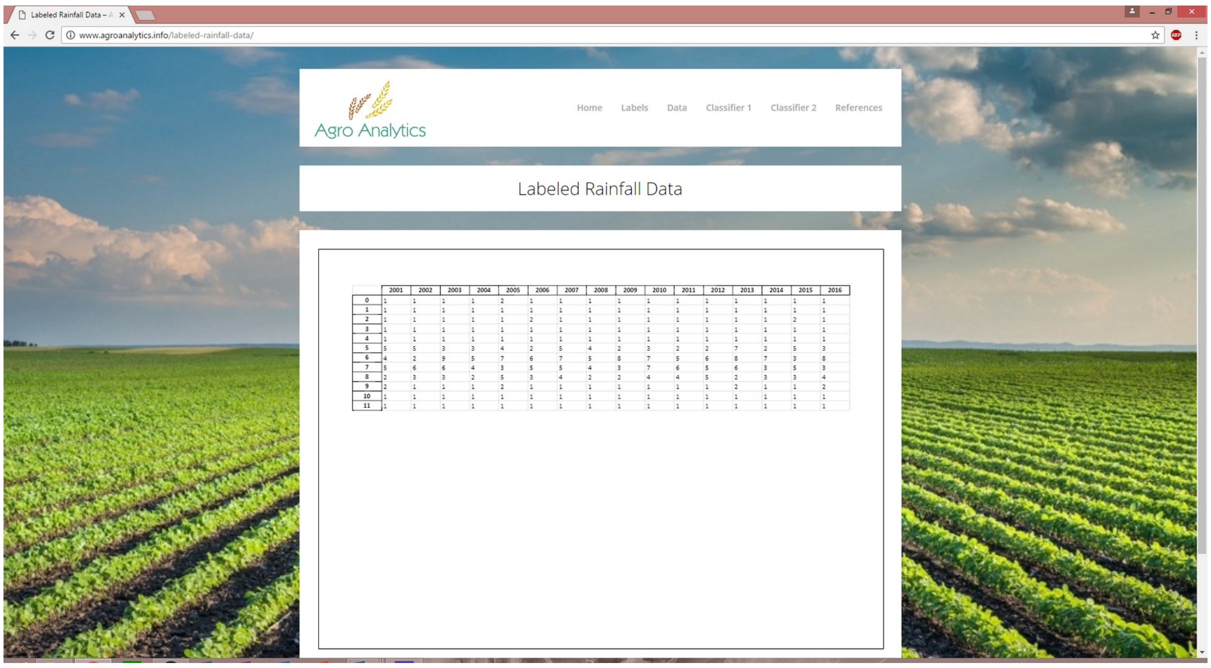
Figure 5.2.3 Raw and Pre-processed Data

An example of raw rainfall and pre-processed rainfall data is shown in Figure 5.2.4 and Figure 5.2.5. This data is displayed as PDFs, and can be easily downloaded or converted to database format.



	January	February	March	April	May	June	July	August	September	October	November	December
2001	11.151	0.083	21.013	23.774	26.516	260.35	223.499	284.193	79.42	109.174	0.281	0.144
2002	10.065	14.071	2.576	6.153	26.488	304.373	109.182	333.918	130.128	29	1.191	0.212
2003	0	4.3	1.8	3.6	3.7	140.7	498.2	341	175	43.7	0	0
2004	49.4	1.5	0.6	19.3	20.6	135.7	256.3	205.5	88.1	7.7	6.8	0
2005	98.3	13.3	19.8	11.2	12	190	421.1	178.8	294.9	89.1	0	0.8
2006	0	0	91	25.8	39.2	108.5	344.3	296.1	145.6	7.9	18	0
2007	0	2.4	8	3.5	6.6	251.5	402.1	288.8	220.7	29.3	37.8	0
2008	0	0	18.5	4.6	15.1	198.3	298.6	188.8	100.1	16.8	0	0
2009	0	0	9.2	0	20.3	95.4	463.7	183.7	107	50.9	53.2	3.3
2010	17	18.8	18.9	0	2	141.2	383.1	396.7	228	24.5	58.8	0.4
2011	0	13.5	3.3	20.5	12.2	119.4	272.7	345.7	185.7	0.1	0	0
2012	9.2	7	0	1.7	1.8	67	338.7	282.9	273.9	19.4	18.3	0
2013	6.9	8.1	8.4	0.7	0	390	478.7	339.3	101.6	109.8	0	0
2014	0.3	26.3	38.4	2.8	11.9	92.3	402.8	133	125.1	26.9	7.5	0.4
2015	14.1	7	66	26.9	34.6	260.6	175.2	306	161.7	6	0	3.6
2016	0	5	24.7	9.1	37.4	126.5	465.4	172.4	212.1	65.9	0	0

Figure 5.2.4 Raw Rainfall Data


Labeled Rainfall Data – A
www.agroanalytics.info/labeled-rainfall-data/
Agro Analytics
Home
Labels
Data
Classifier 1
Classifier 2
References

Labeled Rainfall Data

	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
0	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	2	1	1	1	1	1	1	1	1	2	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	3	3	4	2	5	4	2	3	2	2	7	2	5	3
6	4	2	9	5	7	6	7	5	8	7	5	6	8	7	3	8
7	5	6	6	4	3	5	5	4	3	7	6	5	6	3	5	3
8	2	3	3	2	5	3	4	2	2	4	4	5	2	3	3	4
9	2	1	1	1	2	1	1	1	1	1	1	1	2	1	1	2
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 5.2.5 Pre-processed Rainfall Data

An example of the results obtained and the accuracy achieved has been displayed below. The website shows results for both classifiers using various algorithms.

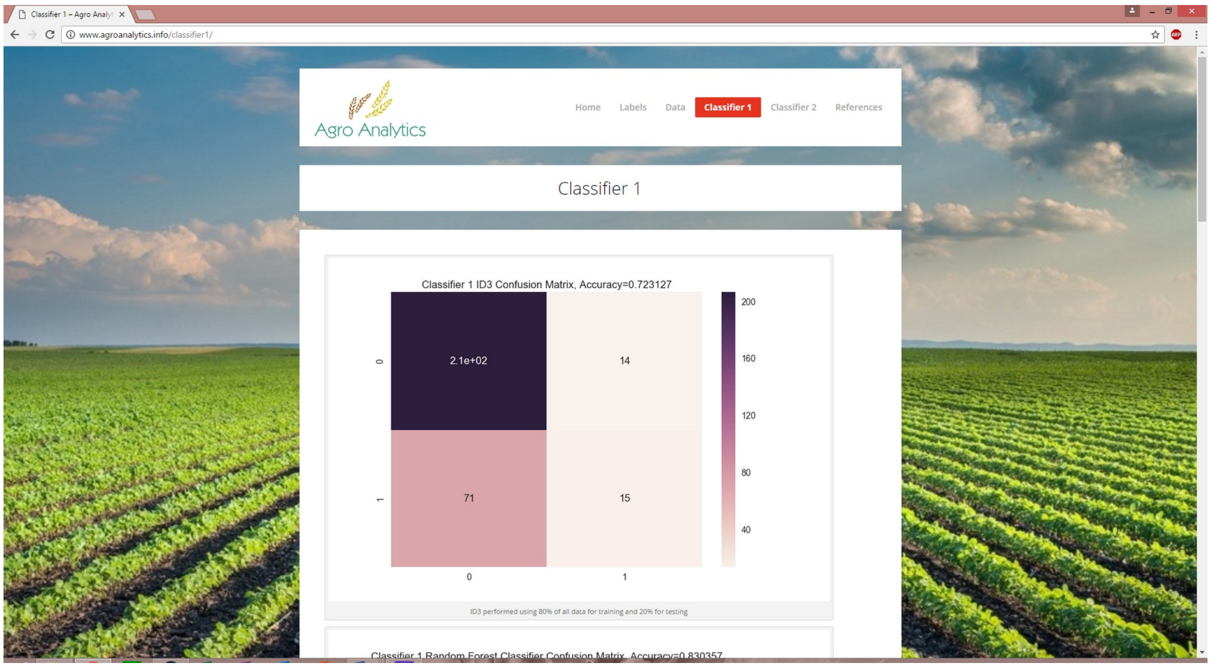

Classifier 1 – Agro Analy
www.agroanalytics.info/classifier1/
Agro Analytics
Home
Labels
Data
Classifier 1
Classifier 2
References
Classifier 1
Classifier 1 ID3 Confusion Matrix, Accuracy=0.723127
2.1e+02
14
71
15
0
1
0
1
200
160
120
80
40
ID3 performed using 80% of all data for training and 20% for testing
Classifier 1 Random Forest Classifier Confusion Matrix, Accuracy=0.830357

Figure 5.2.6 Results of Classifier 1

Chapter 6

Implementation

6.1 Algorithms/Methods Used

Algorithms Used:

1. SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. [10]

The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data. [10]

The above example describes a linear classifier, but SVM can be used to model a non-linear classifier, too. This is done by mapping the data onto a higher dimensional feature space, and finding a corresponding maximum marginal hyperplane. This hyperplane maps onto a non-linear plane in the lower dimensional feature space. This is done using the kernel trick, where the resulting algorithm is formally similar, except that the dot product is replaced by a non-linear kernel function. [11]

In our project, we make use of the Gaussian Radial Basis Function kernel, a popular kernel function used in various kernelized algorithms. [12]

2. Random Forest

The random forest is an ensemble approach that can also be thought of as a form of nearest neighbor predictor. It is a 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 overfitting to their training set. Random forests does not overfit. You can run as many trees as you want. It is fast. Running on a data set with 50,000 cases and 100 variables, it produced 100 trees in 11 minutes on an 800 MHz machine. For large data sets the major memory requirement is the storage of the data itself, and three integer arrays with the same dimensions as the data. If proximities are calculated, storage requirements grow as the number of cases times the number of trees. When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance. After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data. [12, 13, 14] Random Forest is one of the algorithms used to classify the training instances at the both the aforementioned phases.

3. Iterative Dichotomiser 3

In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains. Following are the steps to be followed to induct a decision tree using ID3:

1. Calculate the entropy of every attribute using the data set S

2. Split the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum)

3. Make a decision tree node containing that attribute

4. Recurse on subsets using remaining attributes.

ID3 does not guarantee an optimal solution; it can get stuck in local optima. It uses a greedy approach by selecting the best attribute to split the dataset on each iteration. One improvement that can be made on the algorithm can be to use backtracking during the search for the optimal decision tree. ID3 can overfit to the training data. To avoid overfitting, smaller decision trees should be preferred over larger ones. This algorithm usually produces small trees, but it does not always produce the smallest possible tree. ID3 is harder to use on continuous data. If the values of any given attribute is continuous, then there are many more places to split the data on this attribute, and searching for the best value to split by can be time consuming.

The ID3 algorithm is used by training on a dataset S to produce a decision tree which is stored in memory. At runtime, this decision tree is used to classify new unseen test cases by working down the decision tree using the values of this test case to arrive at a terminal node that tells you what class this test case belongs to.

Methods Used:

1. Data Collection:

Data can originate from myriad sources, and needs to be accumulated before it can be put to use. This can be done by directly importing files that may already be available in .csv or .xlsx formats. This could also be done by manually entering data tuples into a data repository, while referring to different sites. This can be a very tedious and time consuming process. Another method is, scraping off data from the internet using an API or a HTML DOM parser. Our data was collected from various government, as well as private websites. All of them have been listed below in the table below.

Data Source	**Description**
http://www.indiawaterportal.org/articles/district-wise-monthly-rainfall-data-list-raingauge-stations-india-meteorological-department	For rainfall
http://www.timeanddate.com	For temperature and pressure
http://mahaagri.gov.in/cropwatch/asp/rpt1.asp	For crop statistics
http://farmer.gov.in/Drought/DroughtExport.aspx?5+z96qEraJI=	Government announced droughts through Maharashtra

Table 6.1.1 Sources of Data

In case of rainfall data, directly importable data was available on the above mentioned website. But for pressure and temperature, historical data was available, but it had not been compiled together. So as to compile this data, we made use of the BeautifulSoup package of Python. Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. [17]

2. Data Pre-processing:

Datasets in any data mining applications can have missing data values. These missing values can get propagated due to faulty sensors, or lack of communication among the components in a data collection system. These missing values can affect the performance of a data mining system, and need to be addressed accordingly. This can be done by representing these missing values by an approximated mean value. In this project, missing values for parameters like ‘Rainfall’, ‘Temperature’, ‘Pressure’, ‘Crop Productivity’, are replaced with their corresponding averaged representatives. For example, if the ‘Rainfall’ data is missing for the month of July, 2013, then we replace it with the average rainfall for the month of July from all the remaining years. This is done for all missing values. Erroneous values can be dealt with in a similar manner. This data then can be restructured so as to make it readily usable by algorithms. We restructured our data such that the data was stored year against month, as the two axes.

3. Data Transformation/Wrangling:

Discretization or labelling is a process where we convert a set of continuous numerical values to a group of discrete values. Several classification algorithms need discrete values as an input, and cannot work with numerical values. The algorithms being employed in this project need labelled/discrete data. Therefore, we assign labels to our numerical data tuples. This can either be done by dividing data into fixed intervals, where each interval is assigned a label. The step size of the interval can be intuitive, but in our project, we have decided our step size corresponding the requirements of crops. Another method that can be used is unsupervised clustering, and let data fall into different classes and have labels assigned automatically. An optimal number of clusters can be decided using various parameters like silhouette score. We make use of the former, where we divided ‘Rainfall’, ’Pressure’, and ‘Temperature’ into 9, 6, and 5 intervals, respectively.

Below you can see two images, these are graphical representations of the parameters we have used to assign labels to the raw data we possess. These are just examples of how we have worked on all other districts as well.

In the displayed graphs, according to the average rainfall for all 16 years, a blue line is plotted. For determining if a region is experiencing drought like conditions, if a region receives less than 75% of its average expected rainfall, it has a higher chance of experiencing drought. This is indicated by the red line.

In Figure 6.1.1, we see correlation between rainfall and temperature. Thus, a pattern is visible where higher temperature than its average had rainfall lower or on the red line prevails more often.

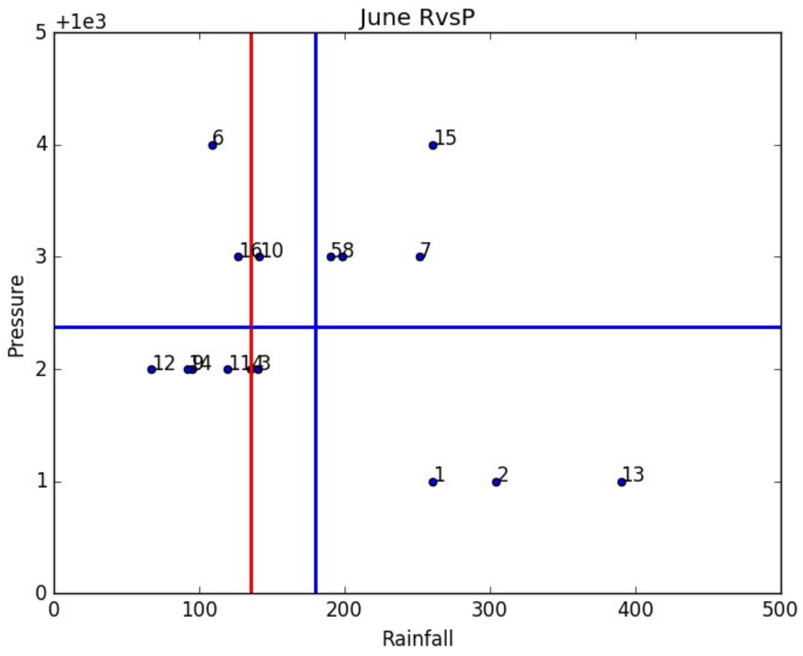


Figure 6.1.1 Rainfall v/s Pressure

In Figure 6.1.2, we see correlation between rainfall and pressure. A pattern is seen that low rainfall is associated with lower atmospheric pressure.

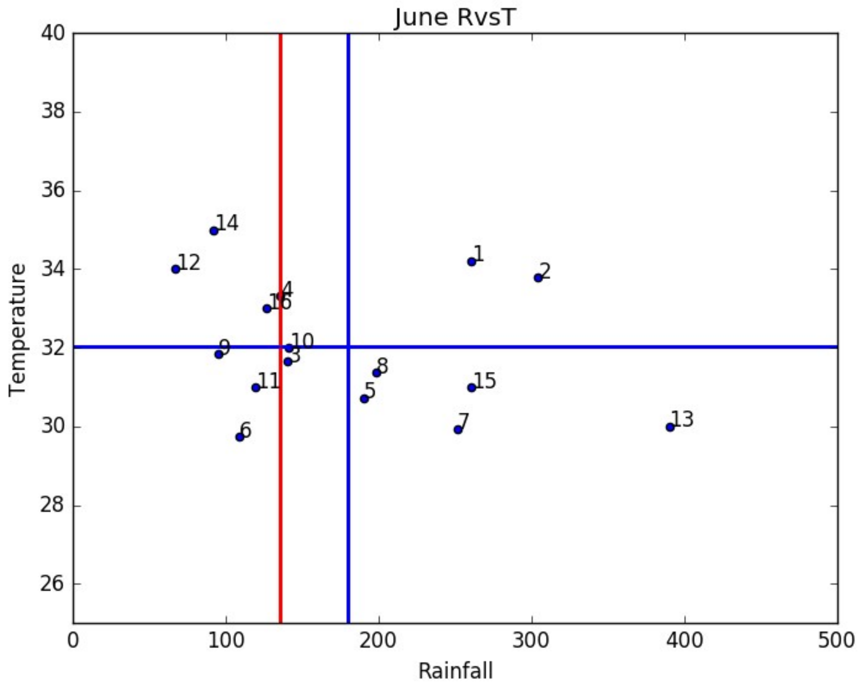


Figure 6.1.2 Rainfall v/s Pressure

4. Data Integration: Various parameters and their corresponding labelled tuples need to be integrated before they are fed into a classification algorithm. We have done by making a separate dataframe using pandas, which imported data from other data repositories.

5. Data Mining – Classification: This stage is where the actual learning algorithm runs, further building a model which will classify any unlabelled instances. For this, the integrated dataframe obtained from the previous step is divided into training and testing data. The training data is then fed into the algorithms, building a model. This model is used to predict the target class for our testing data. We make use of scikit-learn to implement these algorithms, which have inbuilt functions – fit(), to build a model upon training data, and predict(), to predict labels for unlabelled data using the model.

6. Visual Representation: The final results of any data mining application need to be visualized, so that it can be presented and comprehended easily. To evaluate the performance of classification algorithms, we make use of confusion matrices, and parameters like accuracy. It is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another). [18]

6.2 Working of the Project

Code for importing Data from Excel Sheets (.xlsx) – data.py

```
import pandas as pd
import numpy as np

def kharif(rf):
      khar=rf['June']+rf['July']+rf['August']+rf['September']
+rf['October']
      return khar
def wholeyear(rf):
      wy=rf.sum(axis=1)
      return wy
def tkharif(t):
      tk=(t['June']+t['July']+t['August']+t['September']
+t['October'])/5
      return tk
tempnagpur=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Nagpur')
temppune=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Pune')
```

```
tempnashik=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Nashik')
tempaurangabad=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Aurangabad')
tempamravati=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Amravati')
tempsolapur=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Solapur')
tempyavatmal=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Yavatmal')
templatur=pd.read_excel(io='Data\
Temperature_pandas.xlsx',sheetname='Latur')

pressnagpur=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Nagpur')
presspune=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Pune')
pressnashik=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Nashik')
pressaurangabad=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Aurangabad')
pressamravati=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Amravati')
presssolapur=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Solapur')
pressyavatmal=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Yavatmal')
presslatur=pd.read_excel(io='Data\
Pressure_pandas.xlsx',sheetname='Latur')

rfnagpur=pd.read_excel(io='Data\Rainfall.xlsx',sheetname='Nagpur')
rfpune=pd.read_excel(io='Data\Rainfall.xlsx',sheetname='Pune')
rfnashik=pd.read_excel(io='Data\Rainfall.xlsx',sheetname='Nashik')
rfaurangabad=pd.read_excel(io='Data\
Rainfall.xlsx',sheetname='Aurangabad')
rfamravati=pd.read_excel(io='Data\
Rainfall.xlsx',sheetname='Amravati')
rfsolapur=pd.read_excel(io='Data\Rainfall.xlsx',sheetname='Solapur')
rfyavatmal=pd.read_excel(io='Data\
Rainfall.xlsx',sheetname='Yavatmal')
rflatur=pd.read_excel(io='Data\Rainfall.xlsx',sheetname='Latur')

kharnagpur=kharif(rfnagpur)
kharpune=kharif(rfpune)
kharnashik=kharif(rfnashik)
kharaurangabad=kharif(rfaurangabad)
kharamravati=kharif(rfamravati)
kharsolapur=kharif(rfsolapur)
kharyavatmal=kharif(rfyavatmal)
kharlatur=kharif(rflatur)

wynagpur=wholeyear(rfnagpur)
wypune=wholeyear(rfpune)
wynashik=wholeyear(rfnashik)
wyaurangabad=wholeyear(rfaurangabad)
wyamravati=wholeyear(rfamravati)
```

```
wysolapur=wholeyear(rfsolapur)
wyyavatmal=wholeyear(rfyavatmal)
wylatur=wholeyear(rflatur)

tknagpur=tkharif(tempnagpur)
tkpune=tkharif(temppune)
tknashik=tkharif(tempnashik)
tkaurangabad=tkharif(tempaurangabad)
tkamravati=tkharif(tempamravati)
tksolapur=tkharif(tempsolapur)
tkyavatmal=tkharif(tempyavatmal)
tklatur=tkharif(templatur)
```

Code for Scraping Data from timeanddate.com - scrape.py

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
#places=['nagpur','pune','nashik','aurangabad','amravati']
#writer=pd.ExcelWriter('pressure.xlsx')
#for place in places:
i=2009
j=9
d={'2001':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.na
n,np.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6'
,'7','8','9','10','11','12']),
'2002':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2003':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2004':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2005':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2006':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2007':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2008':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2009':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2010':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2011':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2012':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2013':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2014':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2015':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12']),
'2016':pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,np.nan,n
p.nan,np.nan,np.nan,np.nan,np.nan],index=['1','2','3','4','5','6','7
','8','9','10','11','12'])}
df=pd.DataFrame(d)
while(i<=2016 and j<=12):
      print i,j

url="https://www.timeanddate.com/weather/india/amravati/historic?
month="+str(j)+"&year="+str(i)
      r=requests.get(url)
      soup=BeautifulSoup(r.text,"lxml")
      table=soup.find('table',{'class':'zebra tb-wt fw tb-hover'})
      td=int(table.find('tbody').find('tr',{'class':'sep-
t'}).find('th',text='Average').findNext('td').findNext('td').findNex
t('td').text.split()[0])
      print td
      df[str(i)][str(j)]=td
      j+=1
      if(j>12 and i<2016):
            j=1
            i+=1
df.to_csv('pressureamravat.csv')
```

Code for labelling of Rainfall Data – RainfallConversion.py

```
from data import *
import numpy as np
import pandas as pd
np.set_printoptions(threshold=np.inf)
pd.set_option('display.max_colwidth', -1)
labels=['1','2','3','4','5','6','7','8','9','10']
months=['2001','2002','2003','2004','2005','2006','2007','2008','200
9','2010','2011','2012','2013','2014','2015','2016']
#months=['January','February','March','April','May','June','July','A
ugust','September','October','November','December']
rf=np.ravel(np.array([rfnagpur.values,rfpune.values,rfnashik.values,
rfaurangabad.values,rfamravati.values,rfsolapur.values,rfyavatmal.va
lues,rflatur.values]))
rf=pd.cut(rf,10,right=False,labels=labels)
```

```
rf3=[]
j=0
for k in range(8):
      rf2=[]
      for i in range(16):
            rf2.append(rf[j:j+12])
            j+=12
      rf3.append(rf2)

rfnagpur=pd.DataFrame(rf3[0],columns=months)
rfpune=pd.DataFrame(rf3[10],columns=months)
rfnashik=pd.DataFrame(rf3[11],columns=months)
rfaurangabad=pd.DataFrame(rf3[12],columns=months)
rfamravati=pd.DataFrame(rf3[13],columns=months)
rfsolapur=pd.DataFrame(rf3[14],columns=months)
rfyavatmal=pd.DataFrame(rf3[15],columns=months)
rflatur=pd.DataFrame(rf3[16],columns=months)
writer = pd.ExcelWriter('Rainfall_pandas_labels.xlsx')
rfnagpur.to_excel(writer,'Nagpur')
rfpune.to_excel(writer,'Pune')
rfnashik.to_excel(writer,'Nashik')
rfaurangabad.to_excel(writer,'Aurangabad')
rfamravati.to_excel(writer,'Amravati')
rfsolapur.to_excel(writer,'Solapur')
rfyavatmal.to_excel(writer,'Yavatmal')
rflatur.to_excel(writer,'Latur')
```

Code for labelling of Temperature Data – TemperatureConversion.py

```
from data import *
np.set_printoptions(threshold=np.inf)
pd.set_option('display.max_colwidth', -1)
labels=['1','2','3','4','5']
months=['2001','2002','2003','2004','2005','2006','2007','2008','200
9','2010','2011','2012','2013','2014','2015','2016']
#months=['January','February','March','April','May','June','July','A
ugust','September','October','November','December']
temp=np.ravel(np.array([tempnagpur.values,temppune.values,tempnashik
.values,tempaurangabad.values,tempamravati.values,tempsolapur.values
,tempyavatmal.values,templatur.values]))
temp=pd.cut(temp,5,right=False,labels=labels)
temp3=[]
j=0
for k in range(8):
      temp2=[]
      for i in range(16):
            temp2.append(temp[j:j+12])
            j+=12
      temp3.append(temp2)

tempnagpur=pd.DataFrame(temp3[0],columns=months)
temppune=pd.DataFrame(temp3[10],columns=months)
```

```
tempnashik=pd.DataFrame(temp3[11],columns=months)
tempaurangabad=pd.DataFrame(temp3[12],columns=months)
tempamravati=pd.DataFrame(temp3[13],columns=months)
tempsolapur=pd.DataFrame(temp3[14],columns=months)
tempyavatmal=pd.DataFrame(temp3[15],columns=months)
templatur=pd.DataFrame(temp3[16],columns=months)
writer = pd.ExcelWriter('Temperature_pandas_labels.xlsx')
tempnagpur.to_excel(writer,'Nagpur')
temppune.to_excel(writer,'Pune')
tempnashik.to_excel(writer,'Nashik')
tempaurangabad.to_excel(writer,'Aurangabad')
tempamravati.to_excel(writer,'Amravati')
tempsolapur.to_excel(writer,'Solapur')
tempyavatmal.to_excel(writer,'Yavatmal')
templatur.to_excel(writer,'Latur')
"""
temp=np.sort(temp)
tempstd=np.std(temp)
tempmean=np.mean(temp)
tempmedian=np.median(temp)
print temp
print tempmean
print tempstd
print tempstd**2/len(temp)
print tempmedian

"""
```

Code for labelling Pressure Data – PressureConversion.py

```
from data import *
np.set_printoptions(threshold=np.inf)
pd.set_option('display.max_colwidth', -1)
labels=['1','2','3','4','5','6']
months=['2001','2002','2003','2004','2005','2006','2007','2008','200
9','2010','2011','2012','2013','2014','2015','2016']
#months=['January','February','March','April','May','June','July','A
ugust','September','October','November','December']
press=np.ravel(np.array([pressnagpur.values,presspune.values,pressna
shik.values,pressaurangabad.values,pressamravati.values,presssolapur
.values,pressyavatmal.values,presslatur.values]))
press=pd.cut(press,6,right=False,labels=labels)
press3=[]
j=0
for k in range(8):
      press2=[]
      for i in range(16):
            press2.append(press[j:j+12])
            j+=12
      press3.append(press2)

pressnagpur=pd.DataFrame(press3[0],columns=months)
```

```
presspune=pd.DataFrame(press3[10],columns=months)
pressnashik=pd.DataFrame(press3[11],columns=months)
pressaurangabad=pd.DataFrame(press3[12],columns=months)
pressamravati=pd.DataFrame(press3[13],columns=months)
presssolapur=pd.DataFrame(press3[14],columns=months)
pressyavatmal=pd.DataFrame(press3[15],columns=months)
presslatur=pd.DataFrame(press3[16],columns=months)
writer = pd.ExcelWriter('Pressure_pandas_labels.xlsx')
pressnagpur.to_excel(writer,'Nagpur')
presspune.to_excel(writer,'Pune')
pressnashik.to_excel(writer,'Nashik')
pressaurangabad.to_excel(writer,'Aurangabad')
pressamravati.to_excel(writer,'Amravati')
presssolapur.to_excel(writer,'Solapur')
pressyavatmal.to_excel(writer,'Yavatmal')
presslatur.to_excel(writer,'Latur')
```

Code for importing Labelled Data from Excel Sheets – datalabels.py

```
import pandas as pd

tempnagpurl=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Nagpur')
temppunel=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Pune')
tempnashikl=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Nashik')
tempaurangabadl=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Aurangabad')
tempamravatil=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Amravati')
tempsolapurl=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Solapur')
tempyavatmall=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Yavatmal')
templaturl=pd.read_excel(io='Data\
Temperature_pandas_labels.xlsx',sheetname='Latur')

pressnagpurl=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Nagpur')
presspunel=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Pune')
pressnashikl=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Nashik')
pressaurangabadl=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Aurangabad')
pressamravatil=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Amravati')
presssolapurl=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Solapur')
pressyavatmall=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Yavatmal')
```

```
presslaturl=pd.read_excel(io='Data\
Pressure_pandas_labels.xlsx',sheetname='Latur')

rfnagpurl=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Nagpur')
rfpunel=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Pune')
rfnashikl=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Nashik')
rfaurangabadl=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Aurangabad')
rfamravatil=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Amravati')
rfsolapurl=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Solapur')
rfyavatmall=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Yavatmal')
rflaturl=pd.read_excel(io='Data\
Rainfall_pandas_labels.xlsx',sheetname='Latur')

nagpurdroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Nagpur')
punedroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Pune')
nashikdroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Nashik')
aurangabaddroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Aurangabad')
amravatidroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Amravati')
solapurdroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Solapur')
yavatmaldroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Yavatmal')
laturdroughtyn=pd.read_excel(io='Data\
Drought10_pandas_labels.xlsx',sheetname='Latur')
```

Code for Integrating Data for Classifier 1 – IntegrationC1.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
years=['2001','2002','2003','2004','2005','2006','2007','2008','2009
','2010','2011','2012','2013','2014','2015','2016']
months=['January','February','March','April','May','June','July','Au
gust','September','October','November','December']
Classifier1Data=pd.DataFrame(np.zeros((1536,10)),columns=['District'
,'Year','Month','Rainfall','Average  Rainfall','Temperature','Average
Temperature','Pressure','Average                  Pressure','Drought
Classification'])
count=0
```

```
def avglabel(value,mini,maxi,classes):
      increment=(maxi-mini)/classes
      lb=mini
      label=1
      while(True):
            if(value>=lb and value<(lb+increment)):
                  break
            label+=1
            lb+=increment
      return label

def
integrate(district,rfdistrict,pressdistrict,tempdistrict,rfdistrictl
,pressdistrictl,tempdistrictl,districtdroughtyn):
      global count
      for year in range(16):
            for month in range(12):


rfaverage=avglabel(rfdistrict[rfdistrict.columns[month].encode('asci
i')].mean(),0,615.40,10)
                  rf=rfdistrictl.ix[month,years[year]]


pressaverage=avglabel(pressdistrict[pressdistrict.columns[month].enc
ode('ascii')].mean(),1000,1017,6)
                  press=pressdistrictl.ix[month,years[year]]


tempaverage=avglabel(tempdistrict[tempdistrict.columns[month].encode
('ascii')].mean(),18,39,5)
                  temp=tempdistrictl.ix[month,years[year]]
                  drought=districtdroughtyn.ix[month,years[year]]


Classifier1Data.ix[count]=[district,years[year],months[month],rf,rfa
verage,press,pressaverage,temp,tempaverage,drought]
                  count+=1

integrate('Nagpur',rfnagpur,pressnagpur,tempnagpur,rfnagpurl,pressna
gpurl,tempnagpurl,nagpurdroughtyn)
integrate('Pune',rfpune,presspune,temppune,rfpunel,presspunel,temppu
nel,punedroughtyn)
integrate('Nashik',rfnashik,pressnashik,tempnashik,rfnashikl,pressna
shikl,tempnashikl,nashikdroughtyn)
integrate('Aurangabad',rfaurangabad,pressaurangabad,tempaurangabad,r
faurangabadl,pressaurangabadl,tempaurangabadl,aurangabaddroughtyn)
integrate('Amravati',rfamravati,pressamravati,tempamravati,rfamravat
il,pressamravatil,tempamravatil,amravatidroughtyn)
```

```
integrate('Solapur',rfsolapur,presssolapur,tempsolapur,rfsolapurl,pr
esssolapurl,tempsolapurl,solapurdroughtyn)
integrate('Yavatmal',rfyavatmal,pressyavatmal,tempyavatmal,rfyavatma
ll,pressyavatmall,tempyavatmall,yavatmaldroughtyn)
integrate('Latur',rflatur,presslatur,templatur,rflaturl,presslaturl,
templaturl,laturdroughtyn)

writer=pd.ExcelWriter('Classifier1Data.xlsx')
Classifier1Data.to_excel(writer)
```

Code for Classifier 1 (ID3) – C1ID3.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from                    sklearn.metrics                          import
accuracy_score,confusion_matrix,precision_score,recall_score
import matplotlib.pyplot as plt
import seaborn as sns

C1Data=pd.read_excel(io='Data\Classifier1Data.xlsx')
#C1Data.drop(['District','Month','Year'],axis=1,inplace=True)
Districts=['Nagpur','Pune','Nashik','Aurangabad','Amravati','Solapur
','Yavatmal','Latur']
train=pd.DataFrame()
test=pd.DataFrame()
for district in Districts:
     train=train.append(C1Data[C1Data['District']==district][:150])
     test=test.append(C1Data[C1Data['District']==district][150:])
"""
train=C1Data.ix[:1229]
test=C1Data.ix[1229:]
"""
train.drop(['District','Month','Year'],axis=1,inplace=True)
test.drop(['District','Month','Year'],axis=1,inplace=True)
id3=DecisionTreeClassifier(random_state=0,criterion='entropy')
id3.fit(train.drop(['Drought Classification'],axis=1),train['Drought
Classification'])
C1Output=id3.predict(test.drop(['Drought Classification'],axis=1))
accuracy=accuracy_score(test['Drought Classification'],C1Output)
precision=precision_score(test['Drought
Classification'],C1Output,average='macro')
recall=recall_score(test['Drought
Classification'],C1Output,average='macro')
cm=confusion_matrix(test['Drought Classification'],C1Output)
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier   1   ID3   Confusion   Matrix,   Accuracy=%f,
Precision=%f, Recall=%f'%(accuracy,precision,recall))
```

```
plt.show()
```

Code for Classifier 1 (Support Vectort Classfication) – C1SVCRBF.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from                    sklearn.metrics                    import
accuracy_score,confusion_matrix,precision_score,recall_score
import matplotlib.pyplot as plt
import seaborn as sns

C1Data=pd.read_excel(io='Data\Classifier1Data.xlsx')
#C1Data.drop(['District','Month','Year'],axis=1,inplace=True)
Districts=['Nagpur','Pune','Nashik','Aurangabad','Amravati','Solapur
','Yavatmal','Latur']
train=pd.DataFrame()
test=pd.DataFrame()
for district in Districts:
     train=train.append(C1Data[C1Data['District']==district][:150])
     test=test.append(C1Data[C1Data['District']==district][150:])
"""
train=C1Data.ix[:1229]
test=C1Data.ix[1229:]
"""
train.drop(['District','Month','Year'],axis=1,inplace=True)
test.drop(['District','Month','Year'],axis=1,inplace=True)

clf=SVC(kernel='rbf')
clf.fit(train.drop(['Drought Classification'],axis=1),train['Drought
Classification'])
C1Output=clf.predict(test.drop(['Drought Classification'],axis=1))
accuracy=accuracy_score(test['Drought Classification'],C1Output)
precision=precision_score(test['Drought
Classification'],C1Output,average='macro')
recall=recall_score(test['Drought
Classification'],C1Output,average='macro')
cm=confusion_matrix(test['Drought Classification'],C1Output)
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier 1 SVC RBF Kernel Confusion Matrix, Accuracy=
%f, Precision=%f, Recall=%f'%(accuracy,precision,recall))
plt.show()
```

Code for Classifier 1 (Random Forest) – C1RFC.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from                      sklearn.metrics                      import
accuracy_score,confusion_matrix,precision_score,recall_score
import matplotlib.pyplot as plt
import seaborn as sns

C1Data=pd.read_excel(io='Data\Classifier1Data.xlsx')
#C1Data.drop(['District','Month','Year'],axis=1,inplace=True)
Districts=['Nagpur','Pune','Nashik','Aurangabad','Amravati','Solapur
','Yavatmal','Latur']
train=pd.DataFrame()
test=pd.DataFrame()
for district in Districts:
     train=train.append(C1Data[C1Data['District']==district][:150])
     test=test.append(C1Data[C1Data['District']==district][150:])
"""
train=C1Data.ix[:1229]
test=C1Data.ix[1229:]
"""
train.drop(['District','Month','Year'],axis=1,inplace=True)
test.drop(['District','Month','Year'],axis=1,inplace=True)

randomforest=RandomForestClassifier(n_estimators=100)
randomforest.fit(train.drop(['Drought
Classification'],axis=1),train['Drought Classification'])
C1Output=randomforest.predict(test.drop(['Drought
Classification'],axis=1))
accuracy=accuracy_score(test['Drought Classification'],C1Output)
precision=precision_score(test['Drought
Classification'],C1Output,average='macro')
recall=recall_score(test['Drought
Classification'],C1Output,average='macro')
cm=confusion_matrix(test['Drought Classification'],C1Output)
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier 1 Random Forest Classifier Confusion Matrix,
Accuracy=%f, Precision=%f, Recall=%f'%(accuracy,precision,recall))
plt.show()
```

Code for importing and labelling Crop Data – datacrops.py

```
# Bajra, Jowar, Sugarcane, (Maize), Soyabean

import pandas as pd
import numpy as np
from sklearn import preprocessing
"""datacrops=pd.read_csv('Data\CropMaharashtra.csv')
datacrops=datacrops[datacrops['Crop_Year']>=2001]
datacrops=datacrops[(datacrops['Crop']=='Jowar')                 |
(datacrops['Crop']=='Bajra')  |  (datacrops['Crop']=='Sugarcane')  |
(datacrops['Crop']=='Soyabean') | (datacrops['Crop']=='Cotton')]
datacrops['Productivity']=datacrops['Production
(tonnes)']/datacrops['Area (ha)']
datacrops=datacrops.drop(['State_Name'],axis=1)
datacrops=datacrops[(datacrops['District_Name']=='NAGPUR')        |
(datacrops['District_Name']=='PUNE')                               |
(datacrops['District_Name']=='NASHIK')                             |
(datacrops['District_Name']=='AURANGABAD')                         |
(datacrops['District_Name']=='AMRAVATI')                           |
(datacrops['District_Name']=='SOLAPUR')                            |
(datacrops['District_Name']=='YAVATMAL')                           |
(datacrops['District_Name']=='LATUR')]
print datacrops"""

datacrops=pd.read_csv('Data\CropProject.csv')
datacrops=datacrops[datacrops.Season.str.contains("Rabi")==False]
def labelassignment(cropname):
     m=preprocessing.MinMaxScaler(feature_range=(0,4))

dc=datacrops[(datacrops['Crop']==cropname)].dropna(how='any',axis=0,
subset=['Productivity'])
     sdc=m.fit_transform(dc['Productivity'])
     sdc=np.around(sdc)
     dc['CropLabel']=sdc
     return dc

def writing(dc,writer,sheetname):
     dc.to_excel(writer,sheetname)

dcbajra=labelassignment('Bajra')
dcjowar=labelassignment('Jowar')
dcsugarcane=labelassignment('Sugarcane')
dcsoya=labelassignment('Soyabean')

"""writer = pd.ExcelWriter('CropLabels.xlsx')
writing(dcbajra,writer,'Bajra')
writing(dcjowar,writer,'Jowar')
writing(dcsugarcane,writer,'Sugarcane')
writing(dcsoya,writer,'Soyabean')"""
```

Code for assigning Drought Labels – DroughtYN.py

```
from data import *
import numpy as np
import pandas as pd
# Nagpur '2002','2004','2011','2014'
# Pune '2001','2002','2003','2009','2011','2012','2014'
# Nashik '2001','2002','2003','2004','2009','2011','2012','2014'
# Aurangabad '2001','2002','2003','2009','2012','2014'
# Amravati '2002','2004','2009','2011','2014'
# Solapur '2001','2002','2003','2009','2011','2012'
# Yavatmal '2002','2004','2009','2014'
# Latur '2001','2002','2003','2004','2009','2011','2012','2014'
droughtdeclarations=[[1,3,10,13],
[0,1,2,8,10,11,13],
[0,1,2,3,8,10,11,13],
[0,1,2,8,11,13],
[1,3,8,10,13],
[0,1,2,8,10,11],
[2,3,8,13],
[0,1,2,3,8,10,11,13]
]

def droughtlabelassignment(years,rfdistrict):
	district=[]
	for year in years:
	#for year in range(16):
		dyear=[]
		for month in rfdistrict:
			label=0

	if(rfdistrict.ix[year,month]<0.8*rfdistrict[month].mean()):
				label=1
			dyear.append(label)
		district.append(dyear)
	return district

def structuring(district,years):
	districtdroughtyn=np.zeros((16,12))
	for i in district:
		for k in years:
			districtdroughtyn[k]=i
	return districtdroughtyn

def writing(districtdroughtyn,writer,sheetname):
```

```
months=['2001','2002','2003','2004','2005','2006','2007','2008','200
9','2010','2011','2012','2013','2014','2015','2016']

districtdroughtyn=pd.DataFrame(np.transpose(districtdroughtyn),colum
ns=months)
      districtdroughtyn.to_excel(writer,sheetname)

nagpur=droughtlabelassignment(droughtdeclarations[0],rfnagpur)
pune=droughtlabelassignment(droughtdeclarations[10],rfpune)
nashik=droughtlabelassignment(droughtdeclarations[11],rfnashik)
aurangabad=droughtlabelassignment(droughtdeclarations[12],rfaurangab
ad)
amravati=droughtlabelassignment(droughtdeclarations[13],rfamravati)
solapur=droughtlabelassignment(droughtdeclarations[14],rfsolapur)
yavatmal=droughtlabelassignment(droughtdeclarations[15],rfyavatmal)
latur=droughtlabelassignment(droughtdeclarations[16],rflatur)

nagpurdroughtyn=structuring(nagpur,droughtdeclarations[0])
punedroughtyn=structuring(pune,droughtdeclarations[10])
nashikdroughtyn=structuring(nashik,droughtdeclarations[11])
aurangabaddroughtyn=structuring(aurangabad,droughtdeclarations[12])
amravatidroughtyn=structuring(amravati,droughtdeclarations[13])
solapurdroughtyn=structuring(solapur,droughtdeclarations[14])
yavatmaldroughtyn=structuring(yavatmal,droughtdeclarations[15])
laturdroughtyn=structuring(latur,droughtdeclarations[16])

writer = pd.ExcelWriter('DroughtYN_pandas_labels.xlsx')
writing(nagpurdroughtyn,writer,'Nagpur')
writing(punedroughtyn,writer,'Pune')
writing(nashikdroughtyn,writer,'Nashik')
writing(aurangabaddroughtyn,writer,'Aurangabad')
writing(amravatidroughtyn,writer,'Amravati')
writing(solapurdroughtyn,writer,'Solapur')
writing(yavatmaldroughtyn,writer,'Yavatmal')
writing(laturdroughtyn,writer,'Latur')
```

Code for Integrating Data for Classifier 2 – IntegrationC2.py

```
from data import *
from datacrops import *
import numpy as np
import pandas as pd

def writing(dc,writer,sheetname):
      dc.to_excel(writer,sheetname)

def TRImport(cropname):
      Crop=pd.read_excel(io="Data\
CropLabels.xlsx",sheetname=cropname)
      k=[]
      t=[]
```

```
        for i in Crop.index:
            if(Crop.ix[i]['District']=='NAGPUR'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharnagpur[Crop.ix[i]['Year']-2001])
                else:
                    k.append(kharnagpur[Crop.ix[i]['Year']-2001])
                t.append(tknagpur[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='PUNE'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharpune[Crop.ix[i]['Year']-2001])
                else:
                    k.append(kharpune[Crop.ix[i]['Year']-2001])
                t.append(tkpune[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='NASHIK'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharnashik[Crop.ix[i]['Year']-2001])
                else:
                    k.append(kharnashik[Crop.ix[i]['Year']-2001])
                t.append(tknashik[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='AURANGABAD'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharaurangabad[Crop.ix[i]['Year']-
    2001])
                else:
                    k.append(kharaurangabad[Crop.ix[i]['Year']-
    2001])
                t.append(tkaurangabad[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='AMRAVATI'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharamravati[Crop.ix[i]['Year']-
    2001])
                else:
                    k.append(kharamravati[Crop.ix[i]['Year']-
    2001])
                t.append(tkamravati[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='SOLAPUR'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharsolapur[Crop.ix[i]['Year']-2001])
                else:
                    k.append(kharsolapur[Crop.ix[i]['Year']-2001])
                t.append(tksolapur[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='YAVATMAL'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharyavatmal[Crop.ix[i]['Year']-
    2001])
                else:
                    k.append(kharyavatmal[Crop.ix[i]['Year']-
    2001])
                t.append(tkyavatmal[Crop.ix[i]['Year']-2001])
            elif(Crop.ix[i]['District']=='LATUR'):
                if(Crop.ix[i]['Season']=='Whole Year'):
                    k.append(kharlatur[Crop.ix[i]['Year']-2001])
                else:
                    k.append(kharlatur[Crop.ix[i]['Year']-2001])
                t.append(tklatur[Crop.ix[i]['Year']-2001])
      m=preprocessing.MinMaxScaler(feature_range=(0,4))
      k=np.around(m.fit_transform(k))
      t=np.around(m.fit_transform(t))
      Crop['Rainfall']=k
      Crop['Temperature']=t
      return Crop

    Jowar=TRImport('Jowar')
    Bajra=TRImport('Bajra')
    Sugarcane=TRImport('Sugarcane')
    Soyabean=TRImport('Soyabean')

    writer=pd.ExcelWriter('Classifier2Data.xlsx')
    writing(Bajra,writer,'Bajra')
    writing(Jowar,writer,'Jowar')
    writing(Sugarcane,writer,'Sugarcane')
    writing(Soyabean,writer,'Soyabean')
```

Code for Classifier 2 (ID3) – C2ID3.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

C2Data=pd.read_excel(io='Data\
Classifier2Data.xlsx',sheetname='Soyabean')
C2Data.drop(['District','Year','Season','Crop','Area','Production','
Productivity'],axis=1,inplace=True)
train=C2Data.ix[:78]
test=C2Data.ix[78:]
id3=DecisionTreeClassifier(random_state=0,criterion='entropy')
id3.fit(train.drop(['CropLabel'],axis=1),train['CropLabel'])
C2Output=id3.predict(test.drop(['CropLabel'],axis=1))
accuracy=accuracy_score(test['CropLabel'],C2Output)
cm=confusion_matrix(test['CropLabel'],C2Output)
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier   2   ID3   Confusion   Matrix,   Accuracy=%f'%
(accuracy))
plt.show()
```

Code for Classifier 2 (Support Vector Classification) – C2SVC.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

C2Data=pd.read_excel(io='Data\
Classifier2Data.xlsx',sheetname='Soyabean')
C2Data.drop(['District','Year','Season','Crop','Area','Production','
Productivity'],axis=1,inplace=True)
train=C2Data.ix[:78]
test=C2Data.ix[78:]
clf=SVC(kernel='rbf')
clf.fit(train.drop(['CropLabel'],axis=1),train['CropLabel'])
C2Output=clf.predict(test.drop(['CropLabel'],axis=1))
accuracy=accuracy_score(test['CropLabel'],C2Output)
cm=confusion_matrix(test['CropLabel'],C2Output)
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier 2 SVC RBF Kernel Confusion Matrix, Accuracy=
%f'%(accuracy))
plt.show()
```

Code for Classifier 2 (Random Forest) – C2RFC.py

```
from data import *
from datalabels import *
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

C2Data=pd.read_excel(io='Data\
Classifier2Data.xlsx',sheetname='Soyabean')
C2Data.drop(['District','Year','Season','Crop','Area','Production','
Productivity'],axis=1,inplace=True)
train=C2Data.ix[:78]
test=C2Data.ix[78:]
randomforest=RandomForestClassifier(n_estimators=100)
randomforest.fit(train.drop(['CropLabel'],axis=1),train['CropLabel']
)
C2Output=randomforest.predict(test.drop(['CropLabel'],axis=1))
accuracy=accuracy_score(test['CropLabel'],C2Output)
cm=confusion_matrix(test['CropLabel'],C2Output)
```

```
dfcm=pd.DataFrame(cm)
plt.figure(1)
sns.heatmap(dfcm,annot=True)
plt.title('Classifier 2 Random Forest Classifier Confusion Matrix,
Accuracy=%f'%(accuracy))
plt.show()
```

Chapter 8

Results and Discussions

8.1 Results Obtained

The following two tables show an instance of the expected outcome for Agro Analytics in the system.

Table 8.1.1 and Table 8.2 below show the working of both the classifiers in the form of an example, depicting the working of both the phases of the system.

District	Nagpur
Year	2012
Month	June
Rec. Rainfall	67 mm
Avg. Rainfall	107 mm
Rec. Temp.	34 ºC
Avg. Temp.	32 ºC
Rec. Pressure	1002
Avg. Pressure	1002
Drought Classification	High

Table 8.1.1: Classifier for Phase One

Drought Classification	**Crop Type**	**Reqd. Rainfall**	**Reqd. Temperature**	**Crop Growth Probability Index**
High	Jowar	70-75 mm	25-28 ºC	Report

Table 8.1.2: Classifier for Phase Two

8.2 Results Visualized

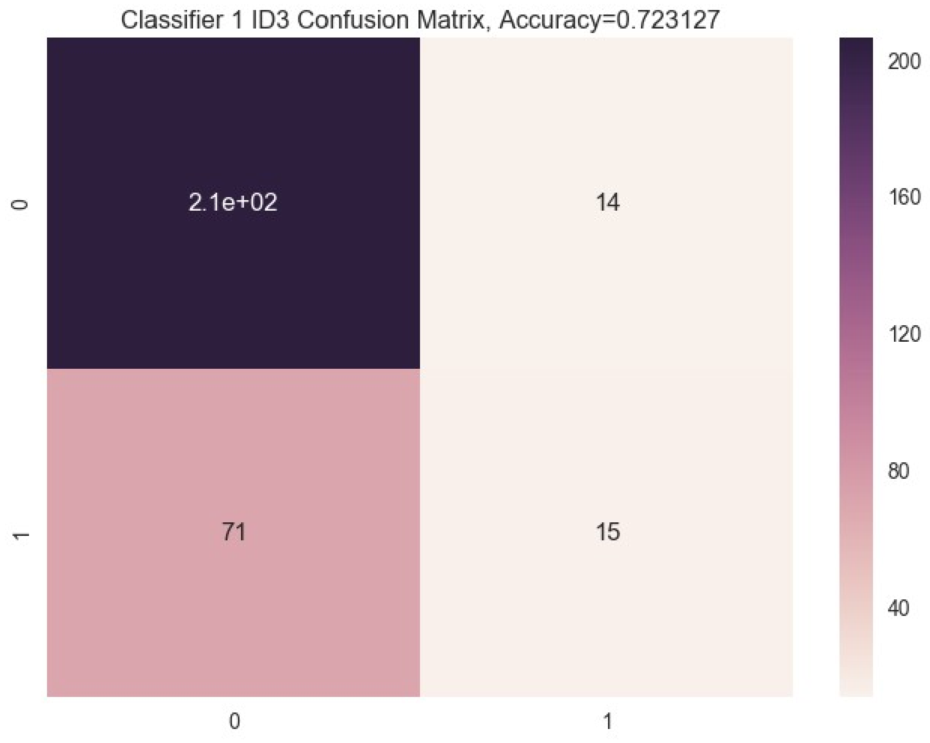
As discussed earlier classification algorithms are best evaluated using confusion matrices. Following are the results visualized for Classifier 1 and Classifier 2, respectively.

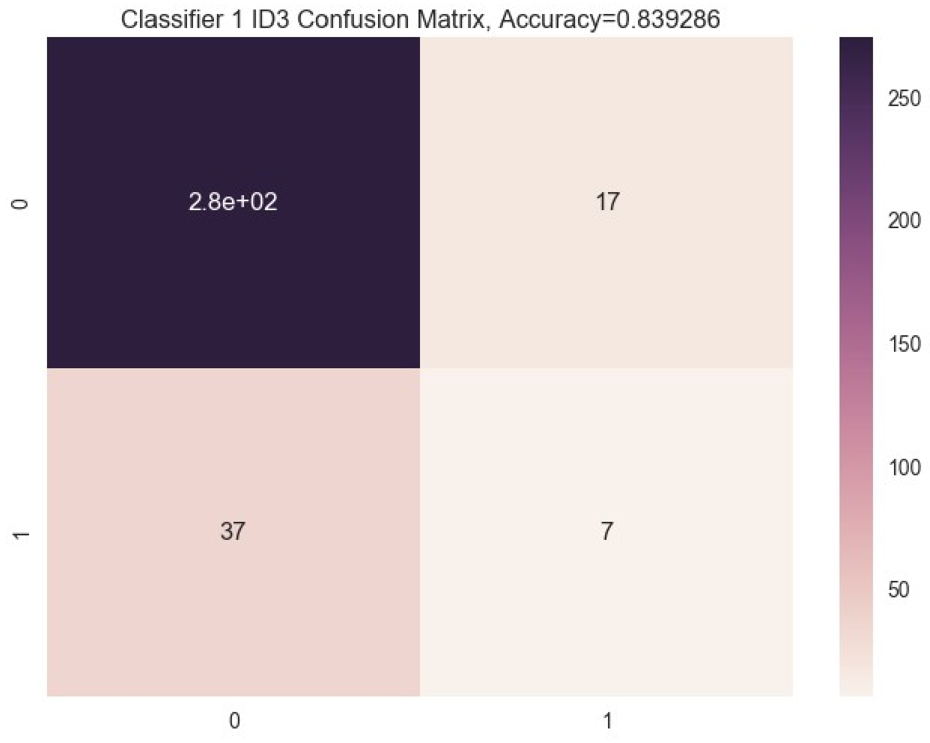
Classifier 1

The classifier was fed two test cases. In the first one, the entire data was split into 80% training, and 20% testing. As seen in earlier sections, this degraded the model's performance.

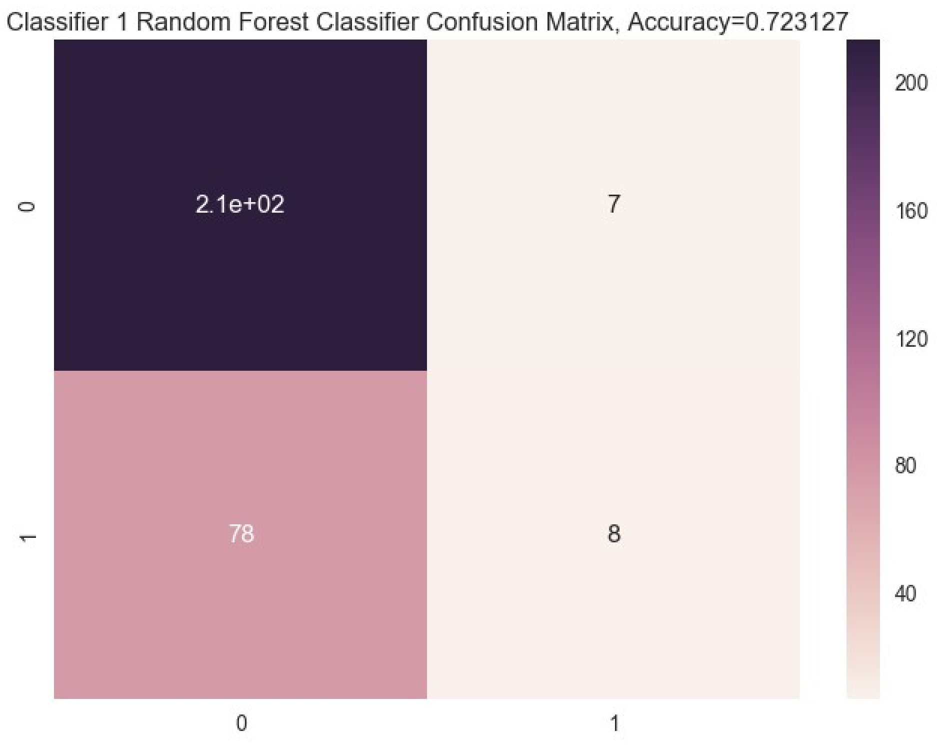
Hence, in the second test case we have divided the data into 80% training and 20% testing for each district, thus giving us better results.

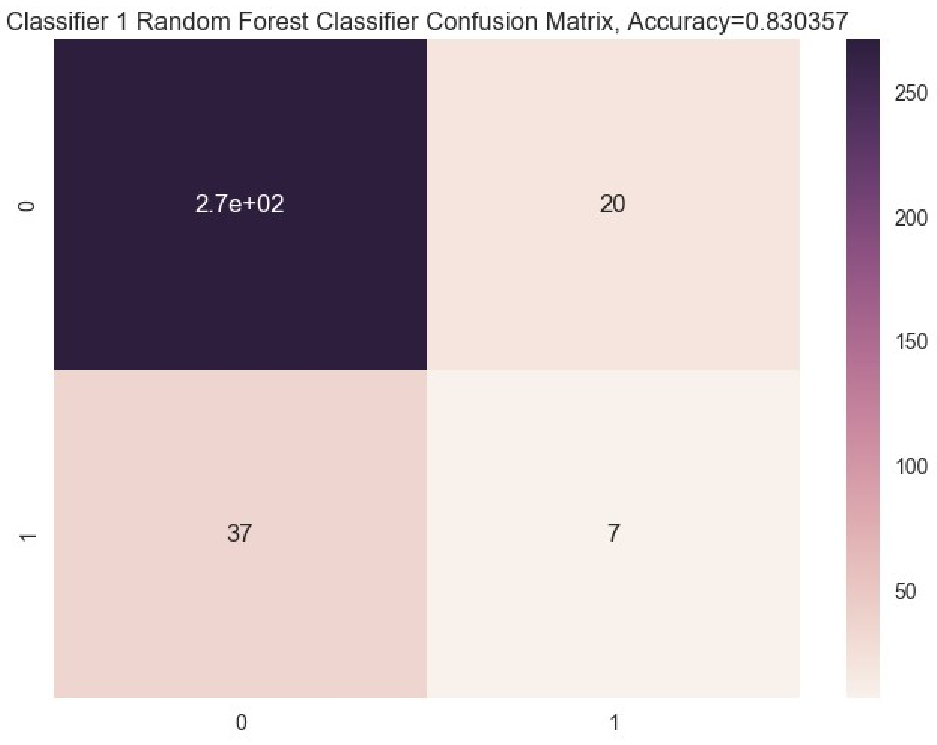
1. Iterative Dichotomiser 3





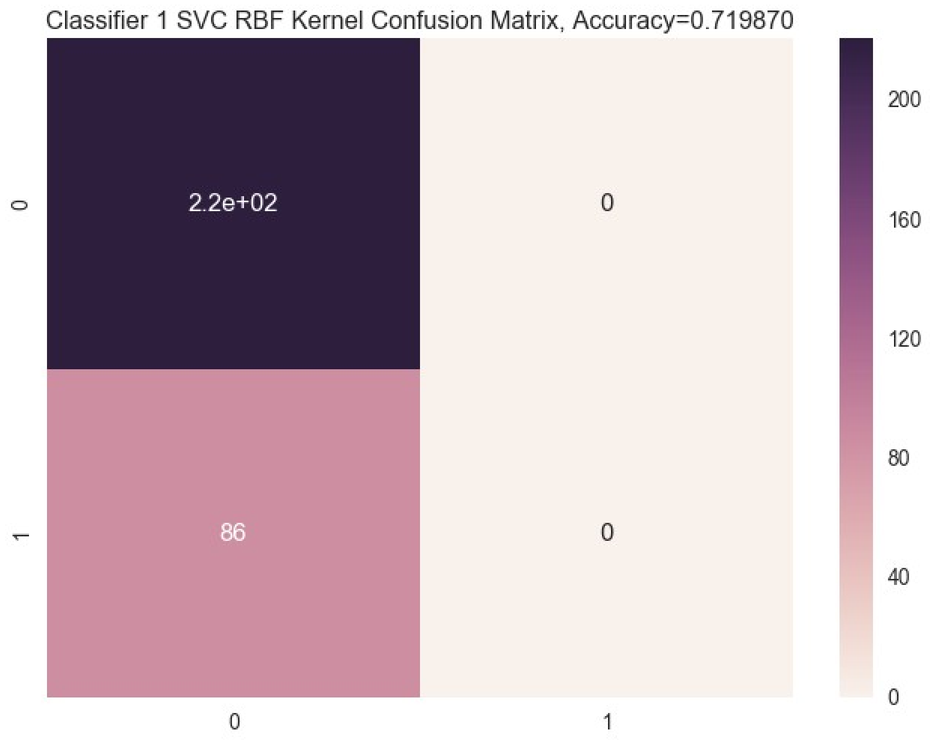
2. Random Forest

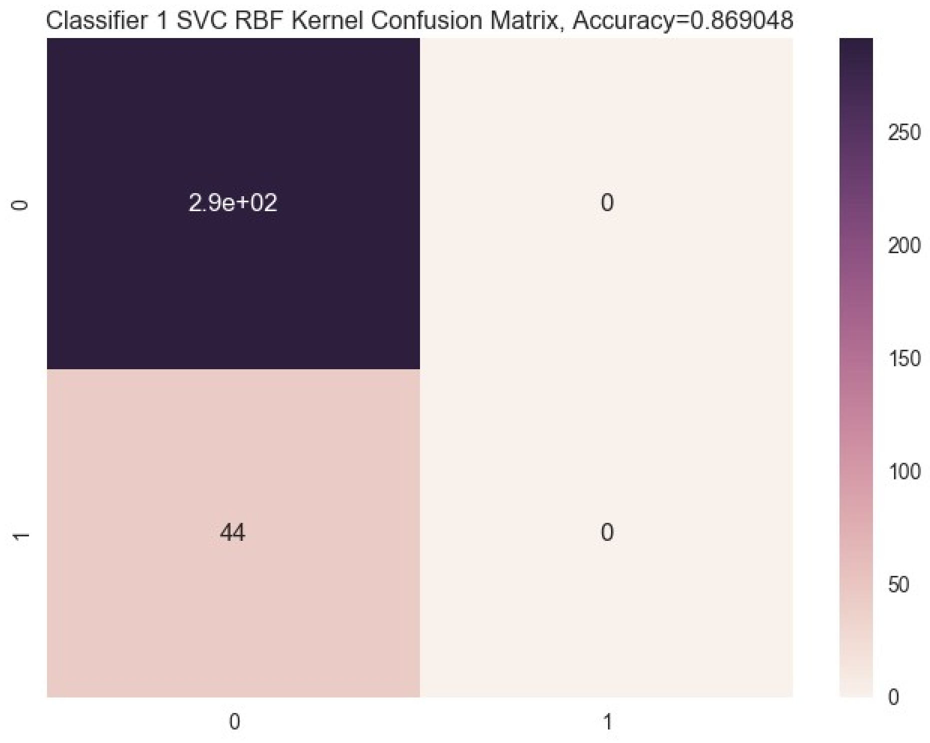




3. Support Vector Classification – Radial Basis Function

Among the above mentioned algorithms, ID3 consistently gave better results, for both the test cases.



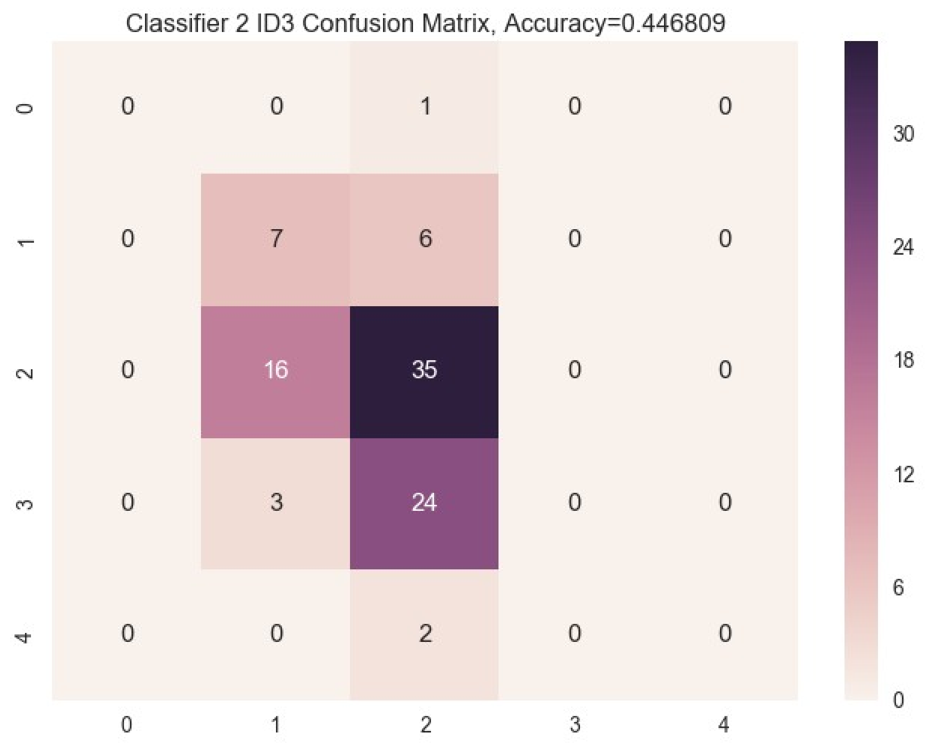


Among the above mentioned algorithms, ID3 consistently gave better results, for both the test cases.

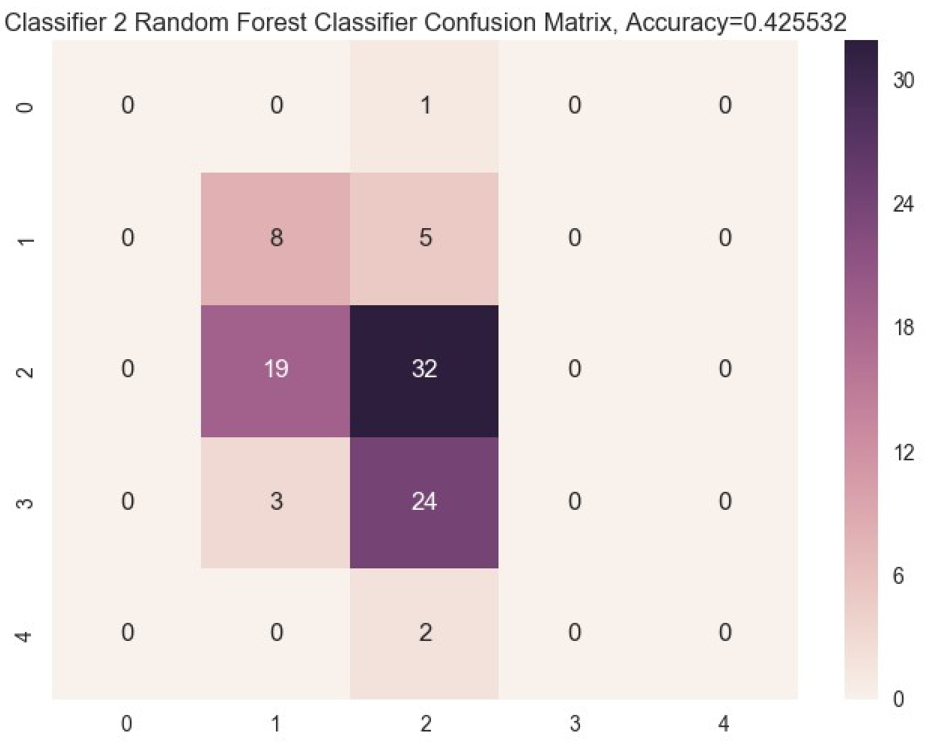
Classifier 2

1. Sugarcane

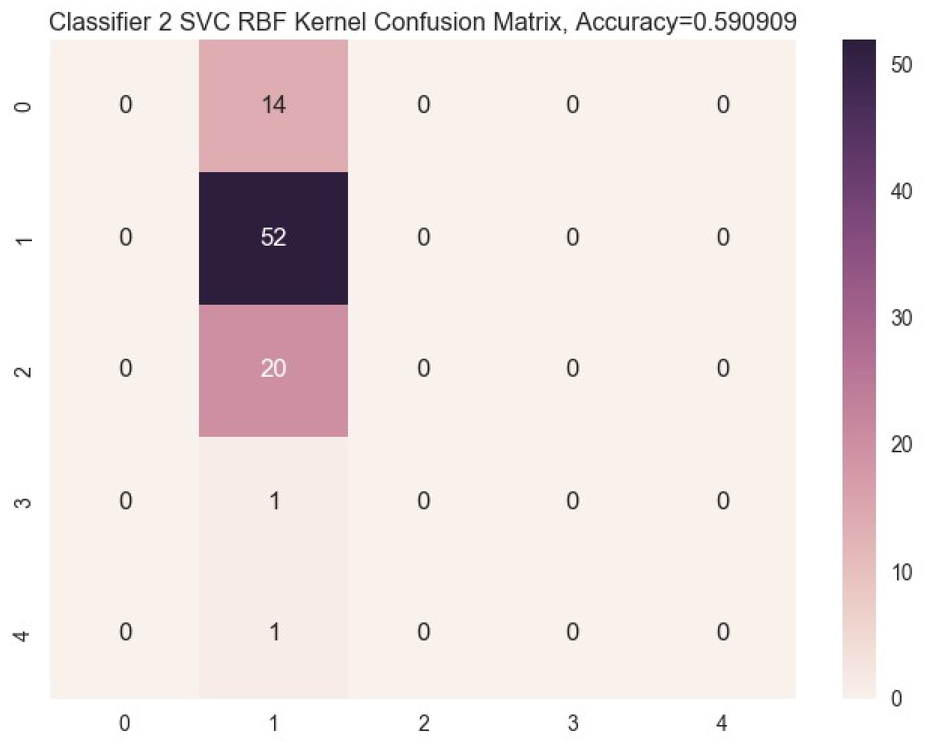
a. Iterative Dichotomiser 3



b. Random Forest

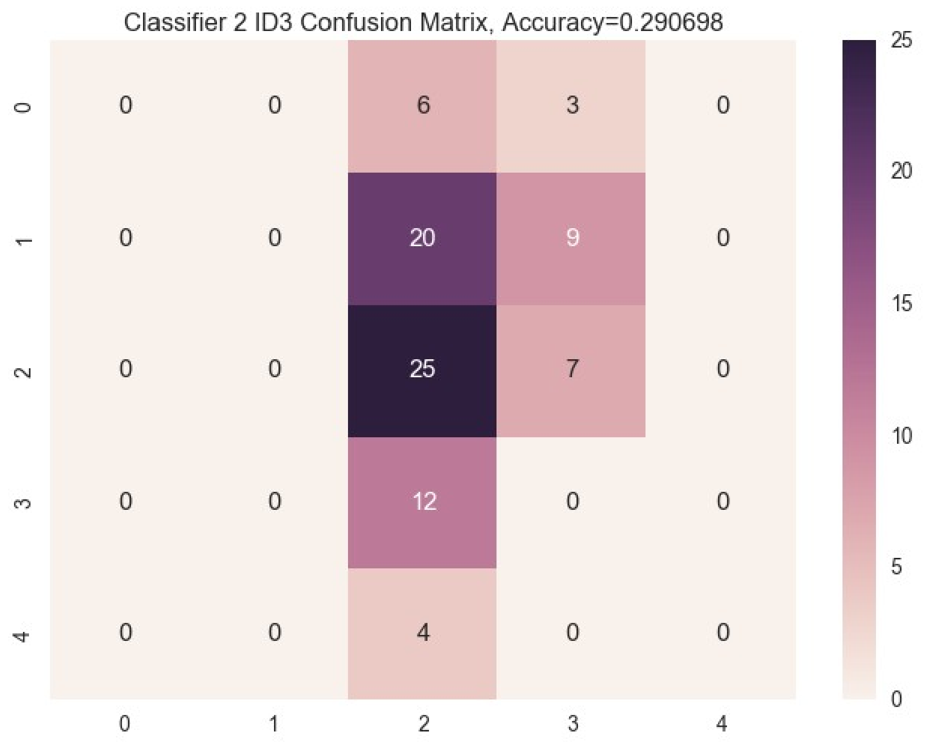


c. Support Vector Classification – Radial Basis Function

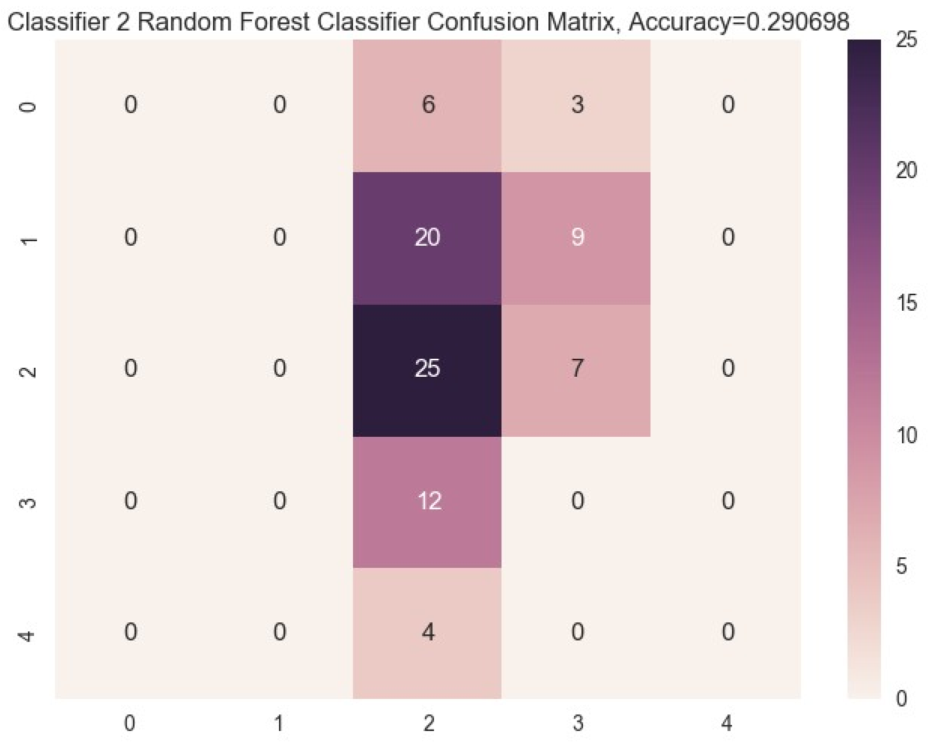


2. Jowar

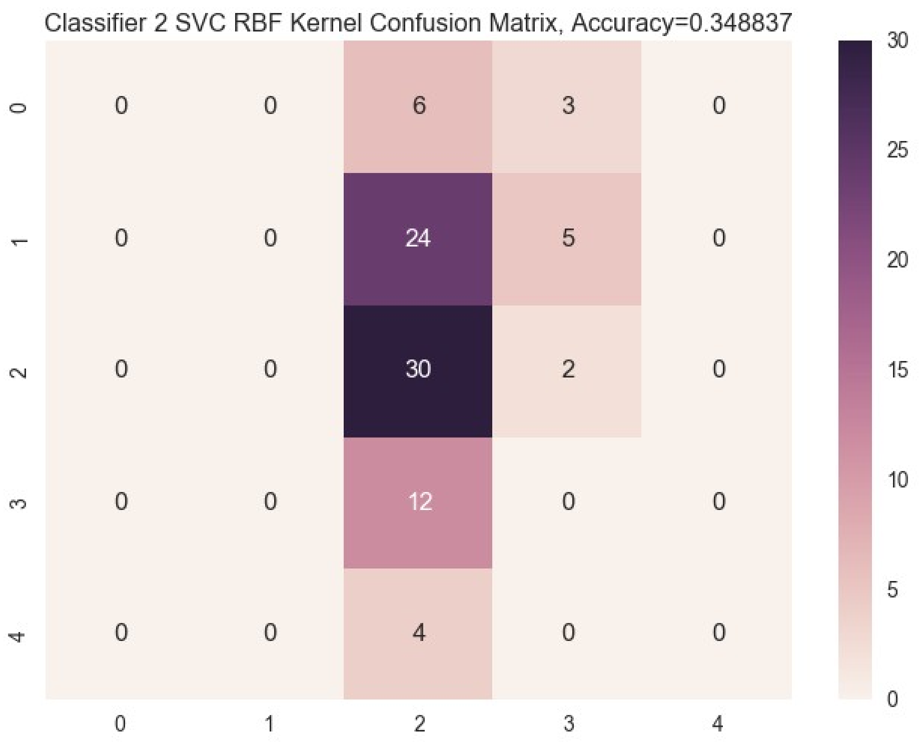
a. Iterative Dichotomiser 3



b. Random Forest

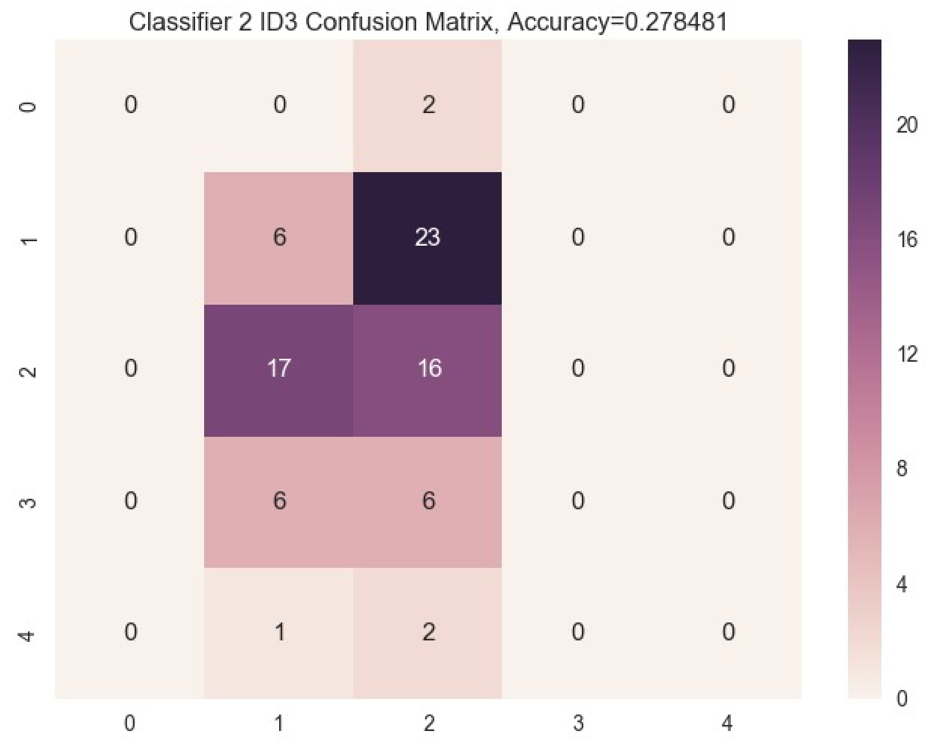


c. Support Vector Classification – Radial Basis Function

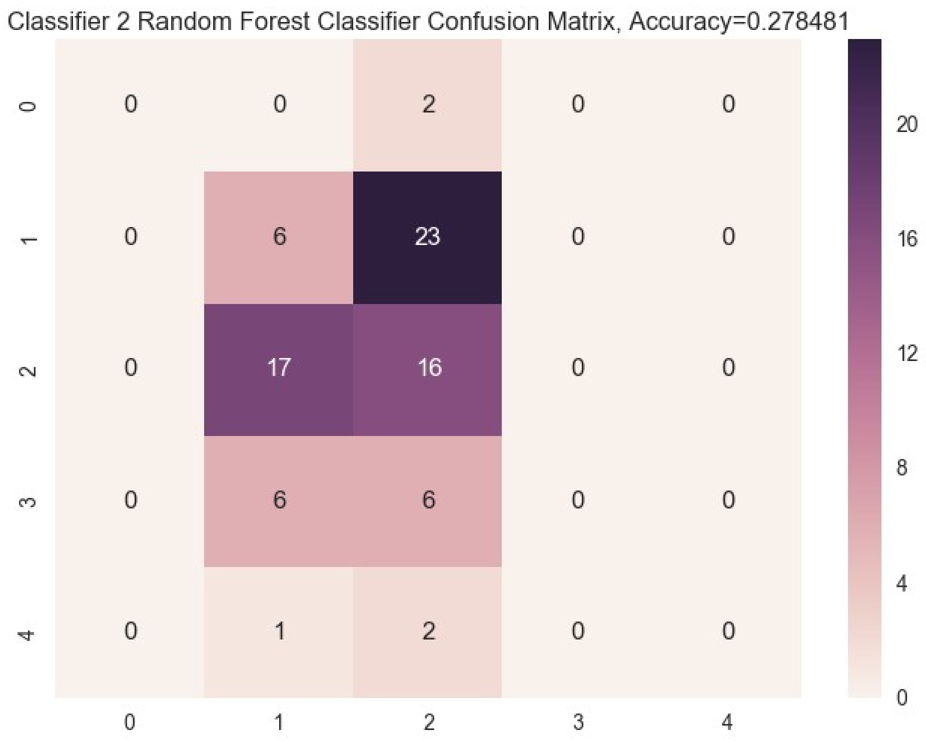


3. Bajra

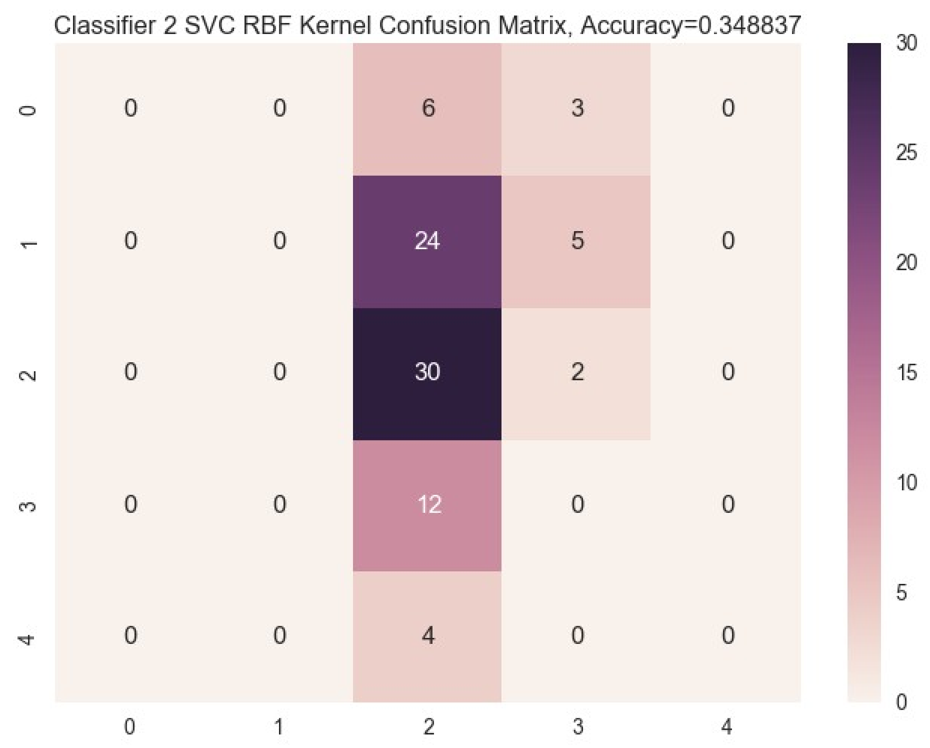
a. Iterative Dichotomiser 3



b. Random Forest

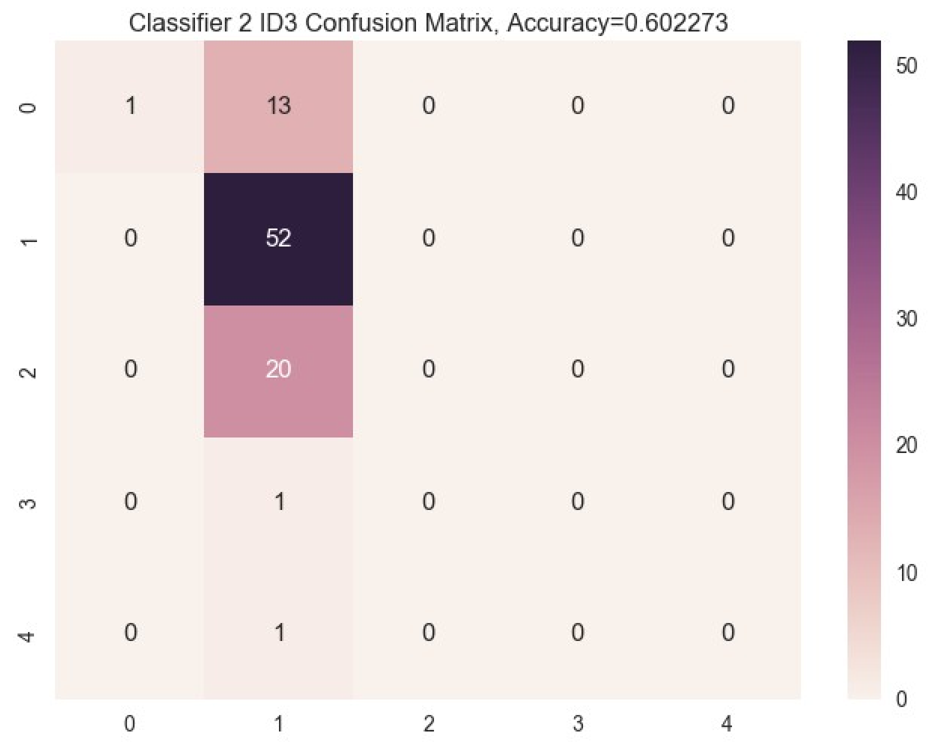


c. Support Vector Classification – Radial Basis Function

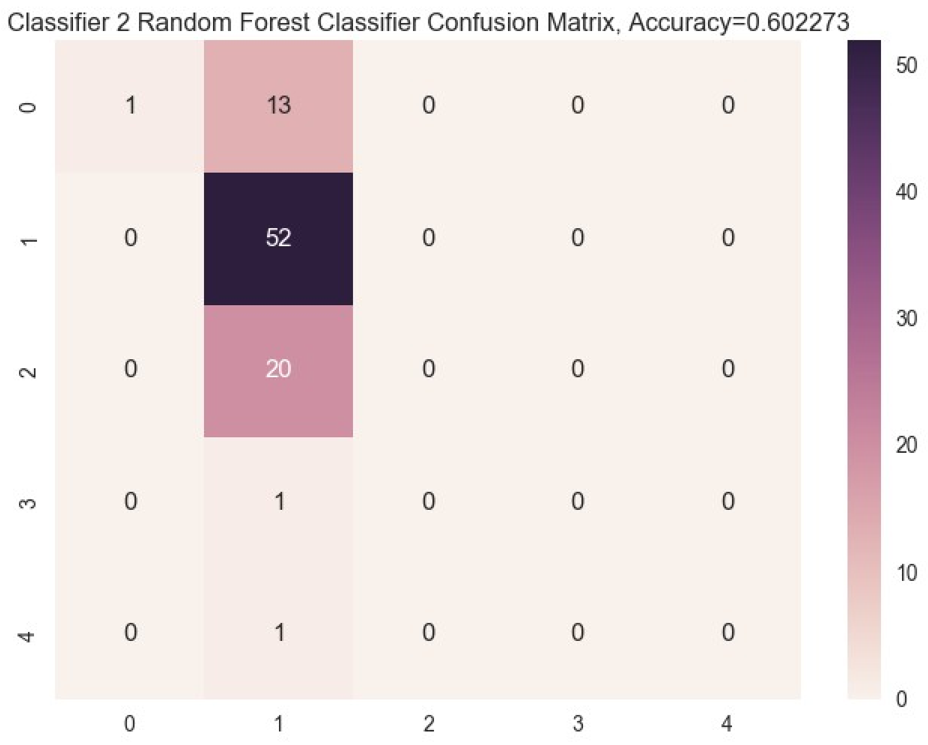


4. Soyabean

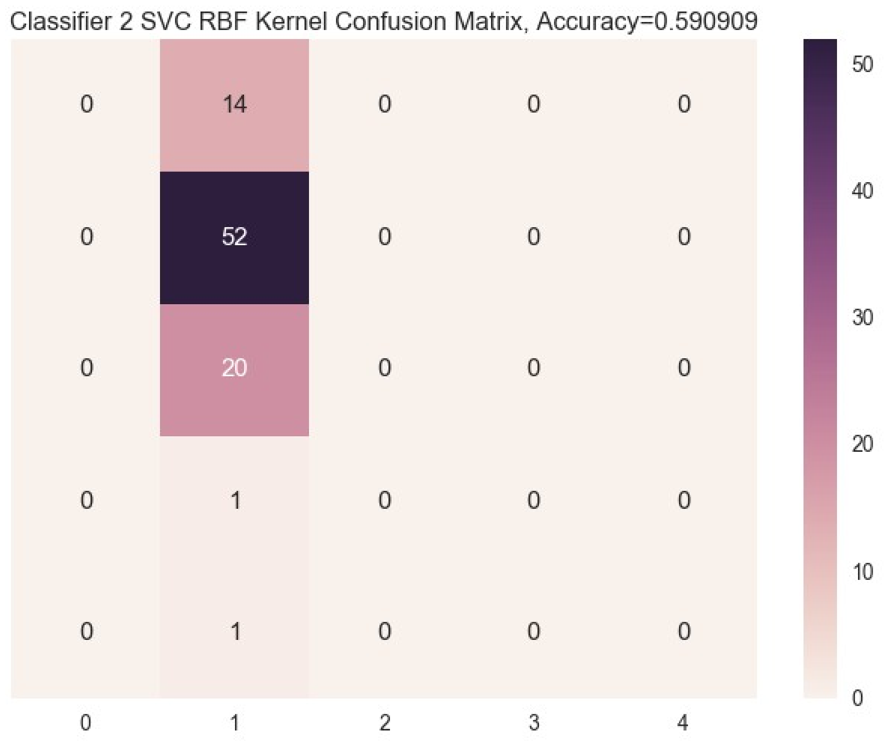
a. Iterative Dichotomiser 3



b. Random Forest



c. Support Vector Classification – Radial Basis Function



Here there is a considerable degradation in the accuracy of the algorithms, due to error magnification from the previous classifier's output. Here the performance for algorithms vary from crop to crop. These anomalies can be explained by the fact that irrigation and agricultural inputs, which are very crucial in today's agricultural practices, are not being considered as an input to the classifier.