CROP YIELD PREDECTION BASED ON RNN ALGORITHM

ABSTRACT

With global warming and increasing of extreme climate evens, climate change may impose positive or negative effects on crop growth and yield. The traditional crop productivity simulations based on crop models are normally site-specific. The simulation accuracy can be improved by using the detailed field management information, such as temperature, climate, soil, location and water PH value. Crop yield estimation is of great importance to food security. Normalized Difference Vegetation Index (NDVI), as an effective crop monitoring tool, is extensively used in crop yield estimation. However, there are few studies focusing on the aspect of mixed crops grown together. The most fit regression models with extracted NDVI in the corresponding crop planting areas are determined. They work reasonably well in small regions, especially in the areas where crop types are unknown exactly. Further improvements to the regression models are possible by incorporating other physical factors such as soil types and geographical information.

SCOPE OF THE PROJECT

To maximize the crop yield, selection of the appropriate crop that will be sown plays a vital role. It depends on various factors like the type of soil and its composition, climate, geography of the region, crop yield, market prices etc. Techniques like Recurrent neural networks, K-nearest neighbors and Decision Trees have carved a niche for themselves in the context of crop selection which is based on various factors. Crop selection based on the effect of natural calamities like famines has been done based on machine learning. The use of artificial neural networks to choose the crops based on soil and climate has been shown by researchers. A plant nutrient management system has been proposed based on machine learning methods to meet the needs of soil, maintain its fertility levels, and hence improve the crop yield. A crop selection method called CSM has been proposed which helps in crop selection based on its yield prediction and other factors.

EXISTING SYSTEM

Ancient people cultivate the crops in their own land and so they have been accommodated to their needs. Therefore, the natural crops are cultivated and have been used by many creatures such as human beings, animals and birds. The greenish goods produced in the land which have been taken by the creature leads to a healthy and welfare life. Since the invention of new innovative technologies and techniques the agriculture field is slowly degrading. Due to these, abundant invention people are been concentrated on cultivating artificial products that is hybrid products where there leads to an unhealthy life. Nowadays, modern people don’t have awareness about the cultivation of the crops in a right time and at a right place. Because of these cultivating techniques the seasonal climatic conditions are also being changed against the fundamental assets like soil, water and air which lead to insecurity of food. Crop production is a complex phenomenon that is influenced by agro climatic input parameters. Agriculture input parameters varies from field to field and farmer to farmer. Collecting such information on a larger area is a daunting task. However, the climatic information collected in India at every square meter area in different parts of the area tabulated by Indian Meteorological Department. Also, the yield of every crop in each state is collected and published by the department of agriculture and cooperation every year.

DISADVANTAGES

1. No proper estimation of crop yield.
2. Low cultivation.
3. Demand for vegetables.

PROPOSED SYSTEM

The prediction of crop yield has direct impact on national and international economies and play important role in the food management and food security. Deep learning gains importance on crop monitoring, crop type classification and crop yield estimation applications with the recent advances in image classification using deep Recurrent Neural Networks. Traditional crop yield prediction approaches based on remote sensing consist of classical Machine Learning methods.Crop yield estimation is a difficult task since it is affected by various factors such as genetic potential of crop cultivar, soil, weather, cultivation practices and biotic stress.

ADVANTAGES

1. Improves crop yield productivity.
2. Easy to analysis the crop to cultivate.
3. Quick analysis.

SYSTEM SPECIFICATION

Software Requirements:

• Operating system	-	Windows/ Ubuntu
• Coding Language	-	Python
• Back-End	-	Anaconda 3.6.

Hardware Requirements:

• Processor	-	Intel Pentium Dual Core
• Speed	-	2.2GHz
• RAM	-	4 GB
• Hard Disk	-	200 GB

LITERATURE SURVEY:

TITTLE: Crop yield variation trend and distribution pattern in recent ten years
AUTHOR: ShanningBao ; Chunxiang Cao

DESCRIPTION:

In recent ten years, a perception exists that the agricultural management and crop cultivars have been improved obviously. But the crop yield variation trend due to above reason remain unknown yet. To evaluate the main food crop (maize, soybean and rice) yield trend from 2007 to 2016, the MODIS product (MCD12Q2) was used to extract the mature date of different crops. A two-band variant of the enhanced vegetation index at mature date was applied to establish empirical yield estimation model, coupling with statistical crop yield data. The validation show the estimated yield had accuracy of 90.9%, 91.7% and 83.3%, respectively. The average maize and soybean yield in study area presented increasing trend, but rice yield presented declining. However, maize yield in 22 cities and soybean yield in 19 cities show decreasing trend actually. Through statistical analysis, the crop yield distribution pattern was proved to be almost fixed. Most cities occupies approximate position on the ranking of relevant crop yield. It was demonstrated that some cities, for example Chifeng city, was suitable to develop specific agriculture economy. This paper can be used to give suggestion for agriculture planning and management.

TITTLE:L-Band Vegetation Optical Depth for Crop Phenology Monitoring and Crop Yield Assessment

AUTHOR:D. Chaparro ; M. Piles

DESCRIPTION:

Vegetation Optical Depth (VOD) at L-band is highly sensitive to the water content and above-ground biomass of vegetation. Hence, it has great potential for monitoring crop phenology and for providing crop yield forecasts. Recently, the Multi-Temporal Dual Channel Algorithm (MT -DCA) has been proposed to retrieve L-band VOD from Soil Moisture Active Passive (SMAP) measurements. In previous research, SMAP VOD has been compared to crop phenology and has been used to derive crop yield estimates. Here, we review and expand these initial research studies. In particular, we quantify the capability of VOD to detect different crop stages, and test different VOD metrics (i.e., maximum, range and integrals of VOD) to provide crop yield estimates in the United States Corn Belt. Results show that VOD captures 50% to 70% of crop changes during growing and maturing phases, and that it explains between 44% (in heterogeneous crop regions) and 74% (in homogenous croplands) of final crop yields.

TITTLE: Simulating Regional Crop Growth and Yield by Using GIS-based EPIC Model

AUTHOR: ZhaYan ; Yang Peng

DESCRIPTION:

With global warming and increasing of extreme climate evens, climate change may impose positive or negative effects on crop growth and yield. The traditional crop productivity simulations based on crop models are normally site-specific. In this study, the spatial crop model is developed by integrating Geographical Information System (GIS) with Erosion Productivity Impact Calculator (EPIC) model to simulate regional crop growth and yield. Data are exchanged using ASCII or binary data format between GIS and EPIC model without a common user interface. The GIS-based EPIC model is applied to simulate the average corn and wheat yield of 1980s in North China. Compared with the statistical yields, the crops yield of Shandong and Beijing is underestimated by the GIS-based EPIC model, especially for Beijing. However, the errors are mostly less than 10%, except that in Beijing and Shandong. Thus, the simulation accuracy of the GIS-based EPIC model is acceptable. The simulation accuracy can be improved by using the detailed field management information, such as irrigation, fertilization and tillage.

TITTLE: Crop Lodging Analysis from UasOrthophoto Mosaic, Sentinel-2 Image and Crop Yield Monitor Data

AUTHOR: TeemuKumpumäki ; Petri Linna

DESCRIPTION:

Crop lodging is surveyed from different image sources and from the crop yield map. Crop lodging has effect on remotely sensed and field level measurements when indirectly measuring, for example, soil variation, nutrients, or crop condition. The interpretation of results may be incorrect especially in the case when it is not possible to detect lodging from the analyzed dataset. Such situation may arise, for example, when analyzing Sentinel-2 (S2) images or crop yield monitor data. In this study crop lodging is inspected from UAS-based or-thophoto mosaic taken 50 meters above the ground level, S2 image and crop yield monitor connected to a combine harvester.

TITTLE: Use Of Deep Neural Networks For Crop Yield Prediction: A Case Study Of Soybean Yield in Lauderdale County, Alabama, USA
AUTHOR: Anıl SuatTerliksiz ; D. TurgayAltýlar

DESCRIPTION:

World population is constantly increasing and it is necessary to have sufficient crop production. Monitoring crop growth and yield estimation are very important for the economic development of a nation. The prediction of crop yield has direct impact on national and international economies and play important role in the food management and food security. Deep learning gains importance on crop monitoring, crop type classification and crop yield estimation applications with the recent advances in image classification using deep Convolutional Neural Networks. Traditional crop yield prediction approaches based on remote sensing consist of classical Machine Learning methods such as Support Vector Machines and Decision Trees. Convolutional Neural Network (CNN] and Long-Short Term Memory Network (LSTM] are deep neural network models that are proposed for crop yield prediction recently. This study focused on soybean yield prediction of Lauderdale County, Alabama, USA using 3D CNN model that leverages the spatiotemporal features. The yield is provided from USDA NASS Quick Stat tool for years 2003-2016. The satellite data used is collected from NASA's MODIS land products surface reflectance, land surface temperature and land surface temperature via Google Earth Engine. The root mean squared error (RMSE] is used as the evaluation metric in order to be able to compare the results with other methods that generally uses RMSE as the evaluation metric.

TITLE: Fuzzy Logic based Crop Yield Prediction using Temperature and Rainfall parameters predicted through ARMA, SARIMA, and ARMAX models

AUTHOR: Shivam Bang ; Rajat Bishnoi ; Ankit Singh Chauhan ; Akshay Kumar Dixit ; Indu Chawla

DESCRIPTION:

Agriculture plays a significant role in the economy of India. This makes crop yield prediction an important task to help boost India's growth. Crops are sensitive to various weather phenomena such as temperature and rainfall. Therefore, it becomes crucial to include these features when predicting the yield of a crop. Weather forecasting is a complicated process. In this work, three methods are used to forecast- ARMA (Auto Regressive Moving Average), SARIMA (Seasonal Auto Regressive Integrated Moving Average) and ARMAX (ARMA with exogenous variables). The performance of the three is compared and the best model is used to predict rainfall and temperature which are in turn used to predict the crop yield based on a fuzzy logic model.

TITLE: Design and Implementation of Mobile Application for Crop Yield Prediction using Machine Learning

AUTHOR: Meeradevi ; Hrishikesh Salpekar

DESCRIPTION:

India is primarily a country based on agriculture and depends largely on agriculture for its economy. If farmer knew which crop would yield higher, he would be able to grow that crop.The project is based on designing an app that will allow farmers to predict the region's production of specific crops depending on physical parameters such as rainfall and temperature. Using crop dataset of various crops from various regions of India, rainfall and temperature dataset for same regions , the proposed model is used to predict crop yield. The aim of proposed model is to develop a tool which is meant to deliver prediction based on the crop of the individual. The product version being developed focuses on the prediction, but eventually it can be added as more data is available and included as features in the data model. The proposed model provides farmers with the detailed recommendation set to optimize their crop selection based on individual factors such as location, farm size, temperature, rainfall, and various crop dataset.All data used in this study are publicly available.

TITLE: Prediction of major crop yields of Tamilnadu using K-means and Modified KNN

AUTHOR: A Suresh ; P. Ganesh Kumar ; M. Ramalatha

DESCRIPTION:

Agriculture is the principal source of livelihood for more than 40 percent of the population of this state. According to Food and Agricultural Organization (FAO) researchers, between 2010 and 2050 the world population will increase by one third. The demand for crop production will increase by 60percent higher than the current production. Hence prediction plays a major role to find out the demand of crop production for maximizing the yield. For that in this paper we propose a prediction method for the major crops of Tamilnadu using K-means and Modified K Nearest Neighbor (KNN). Matlab and WEKA are used as the tool for clustering and classification respectively. The number result shows that our method is better than traditional data mining approach.

TITLE: A paradigm for rice yield prediction in Tamilnadu

AUTHOR: A Suresh ; P. Ganesh Kumar ; M. Ramalatha

DESCRIPTION:

Rice is the staple food of Tamil Nadu from time immemorial. Tamil Nadu produces the most varieties of Rice. Water for the agriculture purpose of Tamil Nadu is dependent on the river basins and the seasonal rainfall for its accomplishment. The major sources of irrigation are through the 17 major river basins spread throughout Tamil Nadu. The river basin data abets in agricultural rotation planning and crop decision. The yield outcome of the crop is dependent on various factors such as soil, climate, fertiliser and irrigation that are essential to agriculture. The rice production and its yield in Tamil Nadu is studied and a paradigm that pilots the prediction of rice yield based on the parameters that affects the rice yield is framed. This paradigm can enable the policy makers to make fact based decisions.

TOPIC: Agriculture Environment Monitoring System using Android Wi-Fi

AUTHOR: Mr. Kalpataru B. Patil Mr. Mahesh B. Girase Mr. Vijay A. Mali Mr. Sachin S. Koli

YEAR: 2018

DESCRIPTION: Agriculture has been the main occupation in our country for centuries. But now, due to the migration of people from rural to urban areas, there is an obstacle in agriculture. So, to overcome this problem, we look for intelligent farming techniques that use IoT. This project includes several functions, such as GPS-based remote monitoring, humidity and temperature detection, intruder intrusion, security and adequate irrigation facilities. It uses wireless sensor networks to continuously observe soil properties and environmental factors. Several sensor nodes are implemented in different locations in the farm. The control of these parameters is carried out via any remote device or Internet service and operations are performed via interface sensors, GSM, voice kit, etc. with a microcontroller. This concept is created as a product and is given to the well-being of the farmer. Advanced development in wireless sensor networks can be used to monitor various parameters in agriculture. Due to the uneven natural distribution of rainwater, it is very difficult for farmers to monitor and control the distribution of water in the farm field across the farm or according to crop needs. There is no ideal irrigation method for all climatic conditions, soil structure and crop variety. Farmers suffer big financial losses due to bad weather forecasts and wrong irrigation methods in this context, with the evolution of miniaturized sensors and wireless technologies, it is possible to remotely control parameters such as temperature, humidity and moisture. The main objective of this document is to develop an intelligent wireless sensor network (WSN) for an agricultural environment. Monitoring the agricultural environment by various factors such as

temperature and humidity along with other factors can be significant. A traditional approach to measuring these factors in an agricultural environment hasled people to measure manually and verify them at different times. This document investigates a remote monitoring system that uses the wireless network. These nodes send data wirelessly to a central server, which collects data, stores it and allows them to be analyzed, viewed as necessary and also sent to the customer's mobile phone. Key words: Smart Agriculture, Android Wi-Fi, Soil Moisture Sensor, Humidity & Temperature Sensor, Fire Sensor

TOPIC: Monitoring and Control Systems in Agriculture Using Intelligent Sensor Techniques: A Review of the Aeroponic System

AUTHOR: Imran Ali Lakhiar, Gao Jianmin, Tabinda Naz Syed, Farman Ali Chandio, Noman Ali Buttar, and Waqar Ahmed Qureshi

YEAR: 2018

DESCRIPTION: In recent years, intelligent sensor techniques have achieved significant attention in agriculture. It is applied in agriculture to plan the several activities and missions properly by utilising limited resources with minor human interference. Currently, plant cultivation using new agriculture methods is very popular among the growers. However, the aeroponics is one of the methods of modern agriculture, which is commonly practiced around the world. In the system, plant cultivates under complete control conditions in the growth chamber by providing a small mist of the nutrient solution in replacement of the soil. The nutrient mist is ejected through atomization nozzles on a periodical basis. During the plant cultivation, several steps including temperature, humidity, light intensity, water nutrient solution level, pH and EC value, CO2 concentration, atomization time, and atomization interval time require proper attention for flourishing plant growth. Therefore, the object of this review study was to provide significant knowledge about early fault detection and diagnosis in aeroponics using intelligent techniques (wireless sensors). So, the farmer could monitor several paraments without using laboratory instruments, and the farmer could control the entire system remotely. Moreover, the technique also provides a wide range of information which could be essential for plant researchers and provides a greater understanding of how the key parameters of aeroponics correlate with plant growth in the system. It offers full control of the system, not by constant manual attention from the operator but to a large extent by wireless sensors. Furthermore, the adoption of the intelligent techniques in the aeroponic system could reduce the concept of the usefulness of the system due to complicated manually monitoring and controlling process.

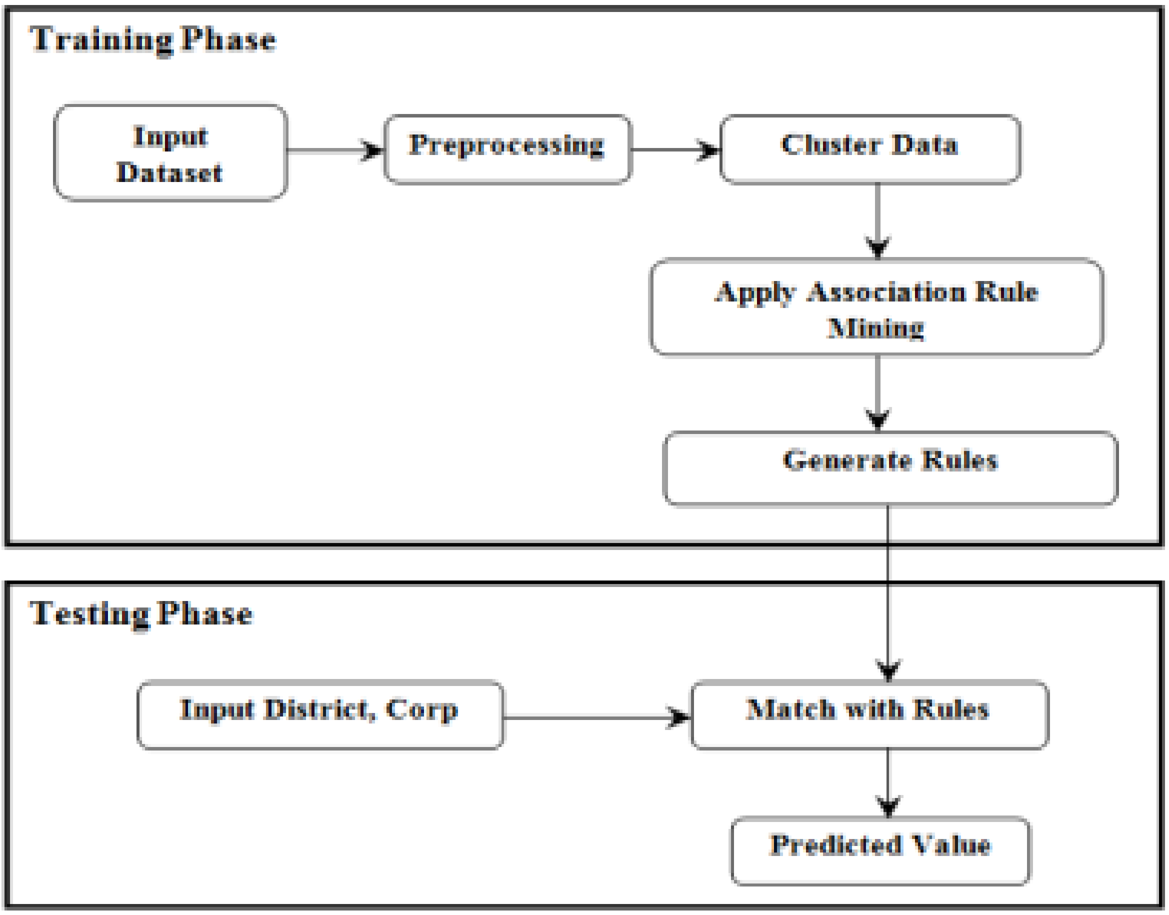
TOPIC: Wireless Monitoring of Soil Moisture, Temperature & Humidity Using Zigbee in Agriculture

AUTHOR: Prof C. H. Chavan, Mr.P. V.Karande

YEAR: 2014

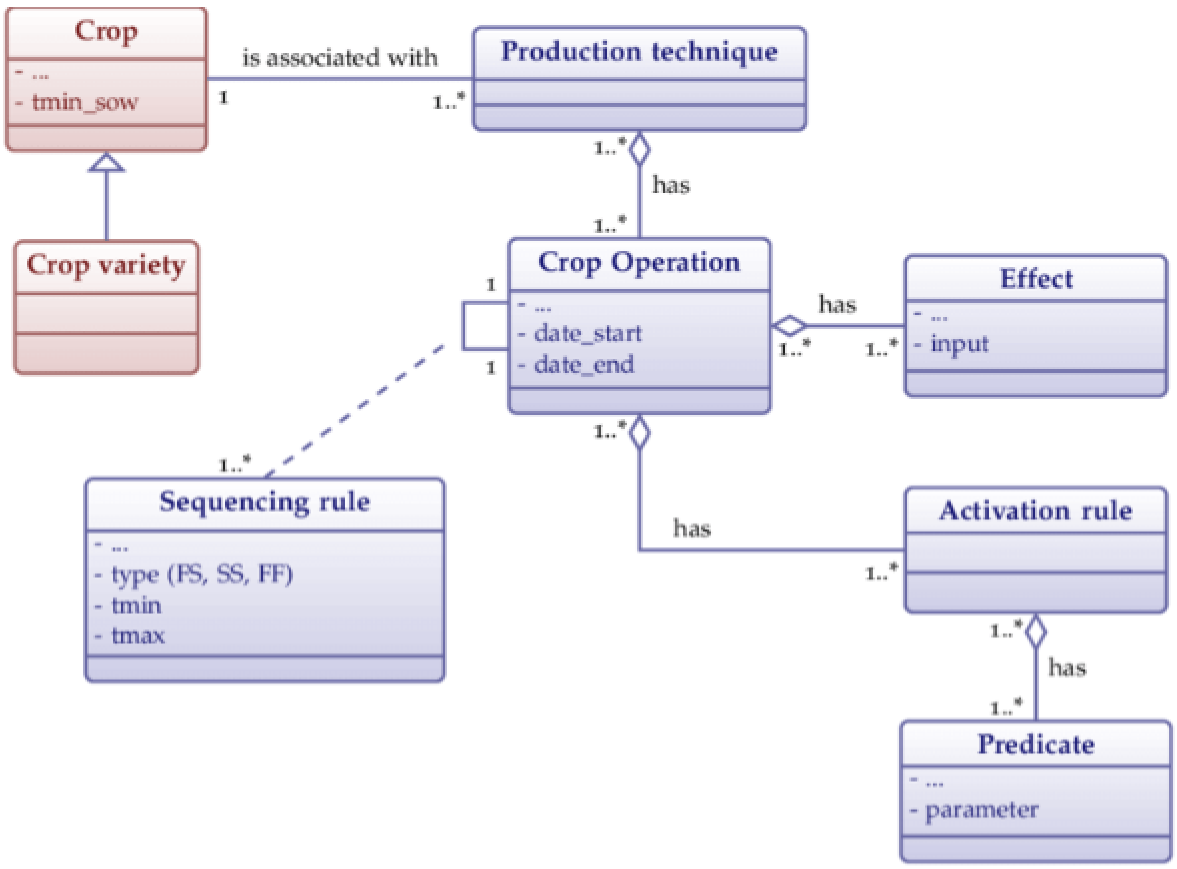
DESCRIPTION: The main objective of the present paper is to develop a smart wireless sensor network (WSN) for an agricultural environment. Monitoring agricultural environment for various factors such as soil moisture, temperature and humidity along with other factors can be of significance. A traditional approach to measure these factors in an agricultural environment meant individuals manually taking measurements and checking them at various times. This paper investigates a remote monitoring system using Zigbee. These nodes send data wirelessly to a central server, which collects the data, stores it and will allow it to be analyzed then displayed as needed and can also be sent to the client mobile.

ARCHITECTURE DIAGRAM:

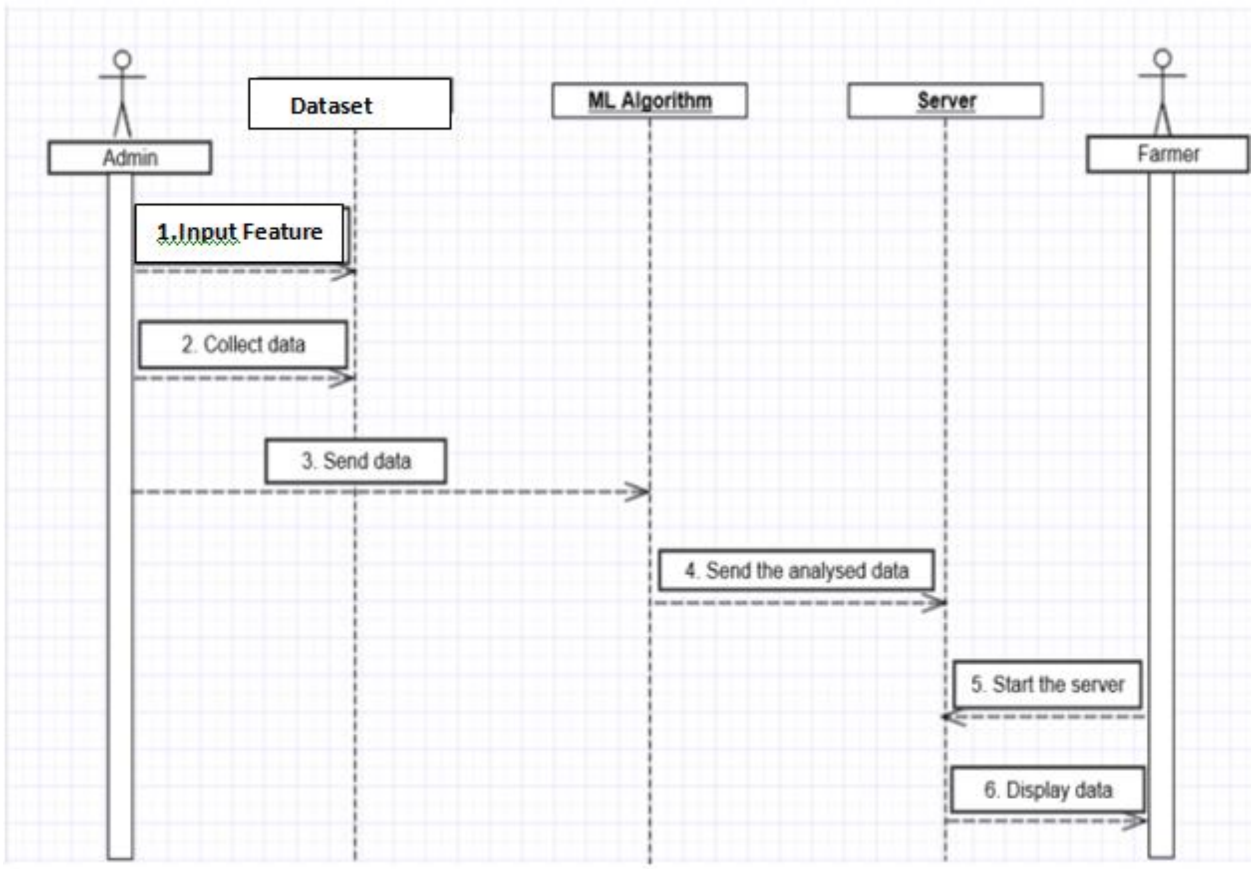


UML DIAGRAMS

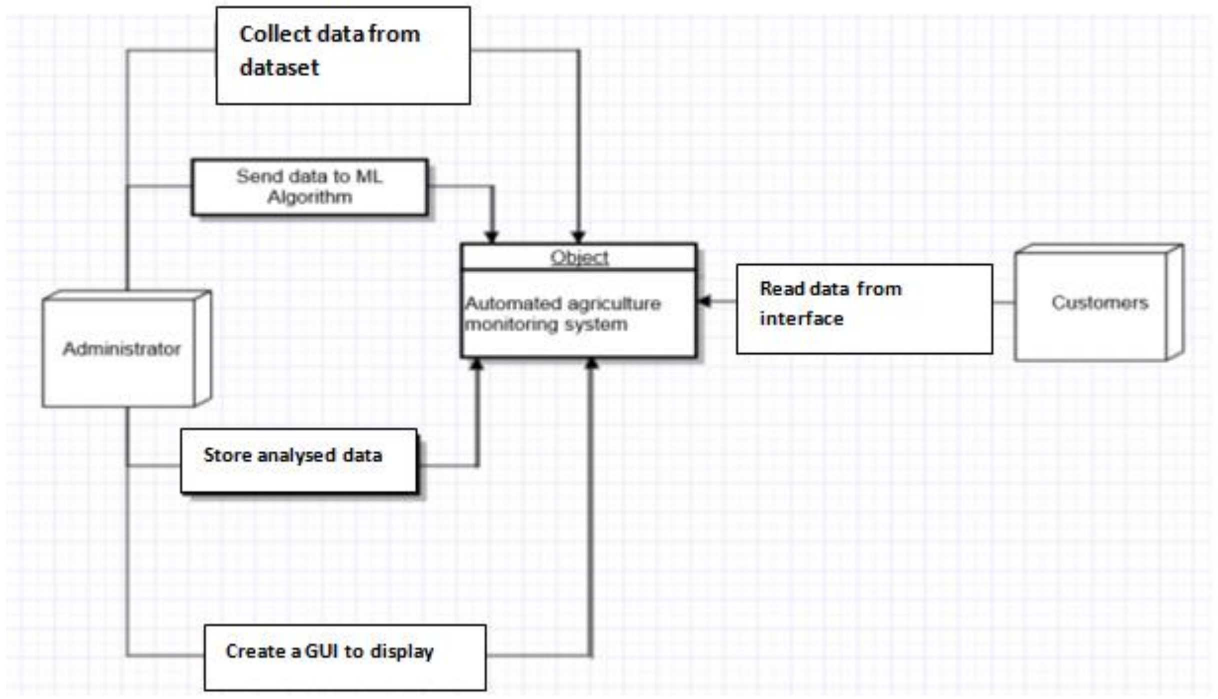
CLASS DIAGRAM:



SEQUENCE DIAGRAM



CONTEXT DIAGRAM



CHAPTER 4 FUNTIONAL AND NON-FUNTIONA; REQUIREMENTS

4. PROPOSED SYSTEM ANALYSIS AND DESIGN

4.1 INTORDUCTION

In product development, it is important to understand the difference between the baseline functionality necessary for any system to compete in that product domain, and features that make the system different from their competitor's products. Some strategies have important implications for software architecture. Specifically, it is not just the Software requirements specifications of the initial release that must be supported in the architecture. The Software requirements specifications of initial products need to be explicitly taken into consideration.

4.2 REQUIREMENT ANALYSIS

4.2.1 FUNCTIONAL REQUIREMENTS

In Software engineering and systems engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. ... This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability.

4.2.1.1 PRODUCT PERSPECTIVE

The product is supposed to be an open source, under the GNU general Public License. It is a web based system implementing client-server model. The portal System provides simple mechanism for users to share and acquire knowledge.

4.2.1.2 PRODUCT FEATURES

The following are the main features

Cross platform support:

Offers operating support for most of the known and commercial operating systems. User account: The system allows the user to create their accounts in the system and provide features of updating and viewing profiles.

Number of users being supported by the system:

Though the number is precisely not mentioned but the system is able to support a large number of online users at a time. Search: search is simply local search engine based on key words.

Discussion Forum:

Provides users with a platform to discuss and help each other with their problems

Ticketing system: Allows user to submit his issue to the admin in case his problems are not solved by FAQs and discussion forums.

FAQs section: Frequently asked section contains answer of problem which tablet user frequently faced.

4.2.1.3 USER CHARACTERISTICS

It is considered that the user do have the basic knowledge of operating the internet and to have access to it. The administrator is expected to be familiar with the interface of the tech support system.

4.2.1.4 ASSUMPTION AND DEPENDENCIES

This software highly depends on type and version of browser being installed in the system i.e. browser version should be used which have HTML5 support.

4.2.1.5 DOMAIN REQUIREMENT

Domain requirement is the Requirement that comes from the application domain of the system that reflects the characteristics of that domain. Therefore, as our System is an inventory System, the domain requirement of this system should concern about the requirements that reflect characteristic of Inventory System.

4.2.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and **requirements** engineering, a **non-functional requirement** (NFR) is a **requirement** that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with **functional requirements** that **define** specific behavior or functions.

4.2.2.1 PRODUCT REQUIREMENTS

4.2.2.1.1 EFFICIENCY

SPACE EFFICIENCY

Storage **efficiency** is the ability to store and manage data that consumes the least amount of **space** with little to no impact on performance; resulting in a lower total operational cost. **Efficiency** addresses the real-world demands of managing costs, reducing complexity and limiting risk.

TIME EFFICIENCY

The state or quality of being **efficient**, or able to accomplish something with the least waste of **time** and effort is Time efficiency; competency in performance. And accomplishment of or ability to accomplish a job with a minimum expenditure of **time** and effort.

4.2.2.1.2 RELIABILITY

Reliability Posted by: Margaret Rouse WhatIs.com Contributor(s): KajBackholm Word of the Day 5G 5G is the coming fifth-generation wireless broadband technology based on the IEEE 802.11ac standard. An important goal of 5G is to erase the differences between wireline and wireless networking to accommodate the growing mobility of network users. Subscribe to the Word of the Day Word of the Day Archive 20 Newest and Updated Terms competitive advantage mobile application management (MAM) Avro (Apache Avro) quality assurance (QA) gross revenue voice recognition (speaker recognition) Amazon Pinpoint employee engagement software Microsoft Project Honolulu project scope unstructured data hands-off infrastructure management Microsoft Windows Insider Program for Business risk map (risk heat map) VMware vCenter Server (formerly VMware VirtualCenter) Advanced Message Queuing Protocol (AMQP) network engineer cloud storage service Ansible cloud backup (online backup) Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs according to its specifications. It has long been considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability - RAS, for short - are considered to be important aspects to design into any system. In theory, a reliable product is totally free of technical errors; in practice, however, vendors frequently express a product's reliability quotient as a percentage.

4.2.2.1.3 PORTABILITY

Portability is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment.

4.2.2.1.4 USABILITY

Usability is the ease of use and learnability of a human-made object such as a tool or device. In **software engineering**, **usability** is the degree to which a **software**can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

4.2.2.2 ORGANISATIONAL REQUIREMENTS

4.2.2.2.1 IMPLEMENTATION REQUIREMENTS

LANGUAGE SPECIFICATRION

PYTHON

Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python graphical user interfaces (GUIs)

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

There are many other interfaces available, which you can find them on the net.

PYTHON TKINTER GUI

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.

- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python

importtkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No.	Operator & Description
1	**Button** The Button widget is used to display buttons in your application.
2	**Canvas** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	**Checkbutton**

	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	**Entry** The Entry widget is used to display a single-line text field for accepting values from a user.
5	**Frame** The Frame widget is used as a container widget to organize other widgets.
6	**Label** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	**Listbox** The Listbox widget is used to provide a list of options to a user.
8	**Menubutton** The Menubutton widget is used to display menus in your application.
9	**Menu** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	**Message** The Message widget is used to display multiline text fields for accepting values from a user.

11	**Radiobutton** The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	**Scale** The Scale widget is used to provide a slider widget.
13	**Scrollbar** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	**Text** The Text widget is used to display text in multiple lines.
15	**Toplevel** The Toplevel widget is used to provide a separate window container.
16	**Spinbox** The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	**PanedWindow** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	**LabelFrame** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

19	**tkMessageBox** This module is used to display message boxes in your applications.

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

4.2.2.2.2 ENGINEERING STANDARD REQUIREMENTS

This standard replaces IEEE 830-1998, IEEE 1233-1998, IEEE 1362-1998. ISO/IEC/IEEE 29148:2011 contains provisions for the processes and products related to the engineering of requirements for systems and software products and services throughout the life cycle. It defines the construct of a good requirement, provides attributes and characteristics of requirements, and discusses the iterative and recursive application of requirements processes throughout the life cycle. ISO/IEC/IEEE 29148:2011 provides additional guidance in the application of requirements engineering and management processes for requirements-related activities in ISO/IEC 12207 and ISO/IEC 15288. Information items applicable to the engineering of requirements and their content are defined. The content of ISO/IEC/IEEE 29148:2011 can be added to the existing set of requirements-related life cycle processes defined by ISO/IEC 12207 or ISO/IEC 15288, or can be used independently**.**

4.2.2.3. OPERATIONAL REQUIREMENTS

Operational requirements are those statements that "identify the essential capabilities, associated requirements, performance measures, and the process or series of actions to be taken in effecting the results that are desired in order to address mission area deficiencies, evolving applications or threats, emerging technologies, or system cost improvements. The operational requirements assessment starts with the Concept of Operations (CONOPS) and goes to a greater level of detail in identifying mission performance assumptions and constraints and current deficiencies of or enhancements needed for operations and mission success. Operational requirements are the basis for system requirements.

The following are the operation requirements.

- Economic
- Environmental
- Social
- Ethical
- Health and Safety
- Sustainability

ECONOMIC

Software engineering economics is about making decisions related to software engineering in a business context. ... Software engineering economics provides a way to study the attributes of software and software processes in a systematic way that relates them to economic measures.

ENVIRONMENTAL

System requirements are all of the requirements at the system level that describe the functions which the system as a whole should fulfill to satisfy the stakeholder needs and requirements. The system environment includes everything external to the software that might vary from one user to the next, or vary over time. If the system behaves differently depending on the amount of available memory, that's part of the system environment. If the system expects a message exchange over the network to finish within a certain amount of time, the network speed/reliability might be part of the system environment. If the system expects to use a certain amount of disk space, the disk capacity is part of the system environment. Those are simple examples, but I hope they get the point across.

SOCIAL

The consideration of social factors in software engineering activities, processes and CASE tools is deemed to be useful to improve the quality of both development process and produced software. Examples include the role of situational awareness and multi-cultural factors in collaborative software development. On the other hand, the dynamicity of the social contexts in which software could operate (e.g., in a cloud environment) calls for engineering social adaptability as a runtime iterative activity. Examples include approaches which enable software to gather users' quality feedback and use it to adapt autonomously or semi-autonomously.

ETHICAL REQUIREMENT

Principles that when followed, promote values such as trust, good behavior, fairness, and/or kindness. There is not one consistent set of standards that all companies follow, but each company has the right to develop the standards that are meaningful for their organization.

SAFETY REQUIREMENTS

A goal is a statement of the importance of achieving a desired target regarding some behavior, datum, characteristic, interface, or constraint. It is above the level of a policy and not sufficiently formalized to be verifiable.

A quality goal is a goal stating the importance of achieving a desired target regarding some quality factor or subfactor. A safety goal is a quality goal stating the importance of achieving a desired target regarding some safety factor (e.g., Health Safety) or safety subfactor (e.g., Hazard Protection"). Examples of such safety goals would be "The system must not harm its users" or "The petroleum refinery control system must eliminate the hazards involving chemical explosions".

A policy is any strategic decision that establishes a desired goal. A quality policy is a policy mandating a desired criterion (or type of criteria) of a quality factor or subfactor. A safety policy is a quality policy mandating a safety criterion (or type of safety criterion). An example of a safety policy would be "The petroleum refinery control system must keep the pressures within reactant tanks significantly below their maximum pressure ratings".

A requirement is any mandatory, externally observable, verifiable (e.g., testable), and validatable behavior, datum, characteristic, or interface. A quality requirement is any requirement that specifies a minimum amount of a mandatory quality subfactor in terms of a quality criterion and quality metric. A quality requirement is a kind of non-functional requirement like a data requirement, an interface requirement, or a constraint. A safety requirement is any quality requirement that specifies a minimum, mandatory amount of safety (subfactor) in terms of a system-specific quality criterion and a minimum level of an associated metric. An example of a safety requirement would be “The petroleum refinery control system shall keep the pressures within reactant tanks at least 30% below their maximum pressure ratings at all times”. Sometimes, safety requirements can be specified in an equivalent inverse of the normal way (e.g., in terms of the maximum acceptable amount of harm rather than the minimum acceptable amount of harm protection).

SUSTAINABILITY

Sustainable engineering is the process of designing or operating systems such that they use energy and resources sustainably, in other words, at a rate that does not compromise the natural environment, or the ability of future generations to meet their own needs.

CHAPTER 5 MODULES

MODULES

There are several modules in the field of agriculture. Some of them are listed below.

DATA COLLECTION

Historical weather data is taken from the gridded surface meteorological dataset Variables are observed daily and include minimum and maximum air temperature and relative humidity, precipitation, incoming shortwave radiation (sunlight), and average wind speed. An overview of the variables used. We note that all variables that are included parametrically are also included non-parametrically, with the exception of the quadratic terms in time and total precipitation. This allows for a parametric 'main effect,' while also allowing these variables to form nonlinear combinations with other input data, which could be useful if the effects of these variables depend partially on the levels of other variables. When training the model, we convert all nonparametric covariates into a matrix of their principal components, retaining those that comprise 95% of the variance of the data.

CROP SELECTION AND CROP YIELD PREDICTION

To maximize the crop yield, selection of the appropriate crop that will be sown plays a vital role. It depends on various factors like the type of soil and its composition, climate, geography of the region, crop yield, market prices etc. Techniques like Artificial neural networks, K-nearest neighbors and Decision Trees have carved a niche for themselves in the context of crop selection which is based on various factors. Crop selection based on the effect of natural calamities like famines has been done based on machine learning. The use of artificial neural networks to choose the crops based on soil and climate has been shown by researchers. A plant nutrient management system has been proposed based on machine learning methods to meet the needs of soil, maintain its fertility levels, and hence improve the crop yield. A crop selection method called CSM has been proposed which helps in crop selection based on its yield prediction and other factors.

Weather Processing

Indian agriculture mainly relies on seasonal rains for irrigation. Therefore, an accurate forecast of weather can reduce the enormous toil faced by farmers in India including crop selection, watering and harvesting. As the farmers have poor access to the Internet as a result of digital-divide, they have to rely on the little information available regarding weather reports. Up-to-date as well as accurate weather information is still not available as the weather changes dynamically over time. Researchers have been working on improving the accuracy of weather predictions by using a variety of algorithms. Artificial Neural networks have been adopted extensively for this purpose. Likewise, weather prediction based on machine learning technique. These algorithms have shown better results over the conventional algorithms.

CROP YIELD PREDICTION

Data Mining is widely applied to agricultural issues. Data Mining is used to analyze large data sets and establish useful classifications and patters in the data sets. The overall goal of the Data Mining process is to extract the information from a data set and transform it into understandable structure for further use. This paper analyzes the crop yield production based on available data. The Data mining technique was used to predict the crop yield for maximizing the crop productivity.

CHAPTER 6 ALGORITHM

RNN ALGORITHM

Recurrent Neural Networks (RNN)

Recurrent Neural Networks or RNN as they are called in short, are a very important variant of neural networks heavily used in Natural Language Processing. In a general neural network, an input is processed through a number of layers and an output is produced, with an assumption that two successive inputs are independent of each other.

This assumption is however not true in a number of real-life scenarios. For instance, if one wants to predict the price of a stock at a given time or wants to predict the next word in a sequence it is imperative that dependence on previous observations is considered.

Recurrent neural networks, of which LSTMs ("long short-term memory" units) are the most powerful and well known subset, are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies (but also including text, genomes, handwriting and the spoken word). What differentiates RNNs and LSTMs from other neural networks is that they take time and sequence into account, they have a temporal dimension.

The purpose of this post is to give students of neural networks an intuition about the functioning of recurrent neural networks and purpose and structure of LSTMs.

Research shows them to be one of the most powerful and useful types of neural network, although recently they have been surpassed in language tasks by the attention mechanism, transformers and memory networks. RNNs are applicable even to images, which can be decomposed into a series of patches and treated as a sequence.

Since recurrent networks possess a certain type of memory, and memory is also part of the human condition, we'll make repeated analogies to memory in the brain.

Recurrent networks are distinguished from feed forward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose: There is information in the sequence itself, and recurrent nets use it to perform tasks that feed forward networks can't.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before. One way to think about RNNs is this: they are a way to share weights over time.

Just as human memory circulates invisibly within a body, affecting our behavior without revealing its full shape, information circulates in the hidden states of recurrent nets. The English language is full of words that describe the feedback loops of memory. When we say a person is haunted by their deeds, for example, we are simply talking about the consequences that past outputs wreak on present time. The French call this "*Le passé qui ne passe pas*," or "The past that does not pass away."

We’ll describe the process of carrying memory forward mathematically:

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

The hidden state at time step t is h_t. It is a function of the input at the same time step x_t, modified by a weight matrix W (like the one we used for feed forward nets) added to the hidden state of the previous time step h_t-1 multiplied by its own hidden-state-to-hidden-state matrix U, otherwise known as a transition matrix and similar to a Markov chain. The weight matrices are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via back propagation and be used to adjust their weights until error can’t go any lower.

The sum of the weight input and hidden state is squashed by the function φ – either a logistic sigmoid function or tan h, depending – which is a standard tool for condensing very large or very small values into a logistic space, as well as making gradients workable for back propagation.

Because this feedback loop occurs at every time step in the series, each hidden state contains traces not only of the previous hidden state, but also of all those that preceded h_t-1 for as long as memory can persist.

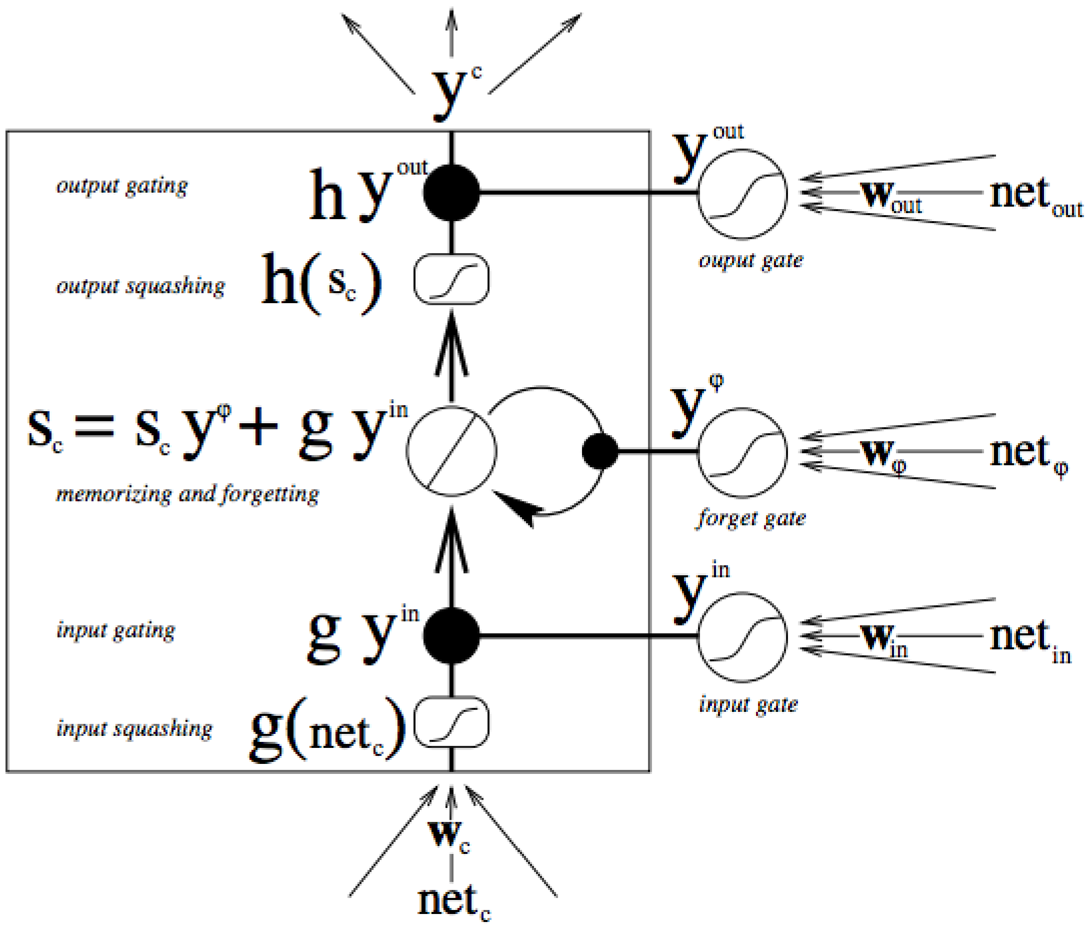
Given a series of letters, a recurrent network *will* use the first character to help determine its perception of the second character, such that an initial q might lead it to infer that the next letter will be u, while an initial t might lead it to infer that the next letter will be h.

LSTMs help preserve the error that can be backpropagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely. This is one of the central challenges to machine learning and AI, since algorithms are frequently confronted by environments where reward signals are sparse and delayed, such as life itself. (Religious thinkers have tackled this same problem with ideas of karma or divine reward, theorizing invisible and distant consequences to our actions.)

LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computer's memory. The cell makes decisions about what to store, and when to allow reads, writes and erasures, via gates that open and close. Unlike the digital storage on computers, however, these gates are analog, implemented with element-wise multiplication by sigmoids, which are all in the range of 0-1. Analog has the advantage over digital of being differentiable, and therefore suitable for back propagation.

Those gates act on the signals they receive, and similar to the neural network's nodes, they block or pass on information based on its strength and import, which they filter with their own sets of weights. Those weights, like the weights that modulate input and hidden states, are adjusted via the recurrent networks learning process. That is, the cells learn when to allow data to enter, leave or be deleted through the iterative process of making guesses, backpropagating error, and adjusting weights via gradient descent.

The diagram below illustrates how data flows through a memory cell and is controlled by its gates.



There are a lot of moving parts here, so if you are new to LSTMs, don't rush this diagram – contemplate it. After a few minutes, it will begin to reveal its secrets.

Starting from the bottom, the triple arrows show where information flows into the cell at multiple points. That combination of present input and past cell state is fed not only to the cell itself, but also to each of its three gates, which will decide how the input will be handled.

The black dots are the gates themselves, which determine respectively whether to let new input in, erase the present cell state, and/or let that state impact the network's output at the present time step. S_c is the current state of the memory cell, and g_y_in is the current input to it. Remember that each gate can be open or shut, and they will recombine their open and shut states at each step. The cell can forget its state, or not; be written to, or not; and be read from, or not, at each time step, and those flows are represented here.

The large bold letters give us the result of each operation.

Here’s another diagram for good measure, comparing a simple recurrent network (left) to an LSTM cell (right). The blue lines can be ignored; the legend is helpful.

CHAPTER 7 SYSTEM TESTING AND MAINTENANCE:

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently.

In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

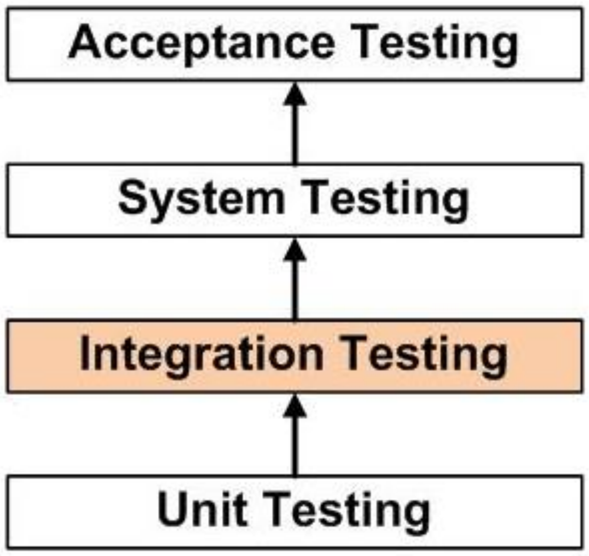
Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non-testing leads to errors that may appear few months later.

This will create two problems, Time delay between the cause and appearance of the problem. The effect of the system errors on files and records within the system. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. This is the final step in system life cycle. Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

Integration Testing is a level of software testing where individual units are combined and tested as a group.



The purpose of this level is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration testing.

METHOD

Any of Black Box Testing, White Box Testing, and Gray Box Testing methods can be used. Normally, the method depends on your definition of ‘unit’.

TASKS

- Integration Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test Cases/Scripts
  - Prepare

  - Review
  - Rework
  - Baseline
- Integration Test
  - Perform

UNIT TESTING:

Unit testing verification efforts on the smallest unit of software design, module. This is known as “Module Testing”. The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

BLACK BOX TESTING

Black box testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

WHITE-BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software thattests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

GREY BOX TESTING

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. To test the Web Services application usually the Grey box testing is used. Grey box testing is performed by end-users and also by testers and developers.

INTEGRATION TESTING:

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

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.

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.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need

- The Status of Nodes information is done automatically in the Cache Updating process

BUILD THE TEST PLAN

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CODE

PROJECT CODE:

```
from tkinter import *
import csv
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import tkinter.messagebox



root = Tk()
root.geometry('850x800')
root.title("crop selector")

pH = DoubleVar()
temp = IntVar()
var = IntVar()
c = StringVar()
c1 = StringVar()
var1 = IntVar()
value1=0
name1="noname"



def doNothing() :


    canvas= Canvas(root,width = 2000,height = 1000)
    canvas.pack()
    label_0 = Label(root, text="CROP SELECTOR", width=20, font=("bold", 20))
    label_0.place(x=310, y=53)

    csv.register_dialect('myDialect',delimiter=',',skipinitialspace=True)
    count=0
    posi=0
    t=0
    maxx=0
    with open('crops.csv', 'r') as csvFile:
        reader = csv.reader(csvFile, dialect='myDialect')
        for row in reader:
            t=0
            if((row[1]<entry_2.get()) and (row[2]>entry_2.get())):
                t=t+1
            if((var1.get()==1 and row[5]=='food')or(var2.get()==1 and
row[5]=='cash')or(var3.get()==1 and row[5]=='plantation')):
                t=t+1
            if((var.get()==1 and row[6]=='summer')or(var.get()==2 and
row[6]=='rainy')or(var.get()==3 and row[6]=='winter')):
                t=t+1
            if((row[3]<entry_1.get()) and (row[4]>entry_1.get())):
                t=t+1
            if(row[8]==droplist1):
                t=t+1
            if((row[7]<entry_9.get()) and (row[8]>entry_9.get())):
                t=t+1
            if(t>maxx):
                if((var1.get()==1 and row[5]=='food')or(var2.get()==1 and
row[5]=='cash')or(var3.get()==1 and row[5]=='plantation')):
                    maxx=t
                    posi=count
            print(t)
            count=count+1
    k=0
    with open('crops.csv', 'r') as csvFile:
        reader = csv.reader(csvFile, dialect='myDialect')
        for row in reader:
            if(k==posi):
                name1=row[0]
                value1=maxx*100/6
            k=k+1
    csvFile.close()
    canvas.destroy()
    root.destroy()
    import extra
    extra.fun(value1,name1)
    return
def Check():
    if(var.get()==0 or ((var1.get()==var2.get()==var3.get()) and (not(var1.get()==1
or var2.get()==1 or var3.get()==1)))):
        import popup
        popup.popUp()
    else:
        doNothing()
    return

label_0 = Label(root, text="CROP SELECTOR", width=20, font=("bold", 20))
label_0.place(x=310, y=53)

label_1 = Label(root, text="pH", width=20, font=("bold", 10))
label_1.place(x=250, y=180)

entry_1 = Entry(root)
entry_1.place(x=440, y=180)

label_9 = Label(root, text="rainfall", width=20, font=("bold", 10))
label_9.place(x=250, y=440)

entry_9 = Entry(root)
entry_9.place(x=440, y=440)

label_2 = Label(root, text="temperature", width=20, font=("bold", 10))
label_2.place(x=250, y=130)

entry_2 = Entry(root)
entry_2.place(x=440, y=130)

label_3 = Label(root, text="weather", width=20, font=("bold", 10))
label_3.place(x=250, y=230)
```

```
radioButt_1 = Radiobutton(root, text="summer", padx=5, variable=var, value=1)
radioButt_1.place(x=430, y=230)
radioButt_2 = Radiobutton(root, text="rainy", padx=20, variable=var, value=2)
radioButt_2.place(x=505, y=230)
radioButt_3 = Radiobutton(root, text="winter", padx=35, variable=var, value=3)
radioButt_3.place(x=580, y=230)

label_4 = Label(root, text="state", width=20, font=("bold", 10))
label_4.place(x=250, y=280)

list1 = ['Andhra Pradesh','Arunachal
Pradesh','Assam','Bihar','Chattisgarh','Goa','Gujarat','Haryana','Himachal Pradesh'
    ,'Jammu and Kashmir','Jharkhand','Karnataka','Kerala','Madhya
Pradesh','Maharashtra','Manipur','Meghalaya','Mizoram',
    'Nagaland','Odisha','Punjab','Rajasthan','Sikkim','Tamil
Nadu','Telengana','Tripura','Uttar Pradesh','Uttarakhand','West Bengal']

droplist = OptionMenu(root, c, *list1)
droplist.config(width=15)
c.set('select your state')
droplist.place(x=440, y=280)

label_8 = Label(root, text="type of soil", width=20, font=("bold", 10))
label_8.place(x=250, y=330)

list2 = ['Alluvial','Black','Loamy','Sandy']
```

```
droplist1 = OptionMenu(root, c1, *list2)
droplist1.config(width=15)
c1.set('select your soil')
droplist1.place(x=440, y=330)



var2 = IntVar()
var3 = IntVar()

label_5 = Label(root, text="type of crop", width=20, font=("bold", 10))
label_5.place(x=250, y=380)

check_1 = Checkbutton(root, text="food", variable=var1)
check_1.place(x=430, y=380)

check_2 = Checkbutton(root, text="cash", variable=var2)
check_2.place(x=505, y=380)

check_3 = Checkbutton(root, text="plantation", variable=var3)
check_3.place(x=580, y=380)

button1=Button(root, text='Submit', width=20, bg='brown',
fg='white',command=Check)
button1.place(x=650, y=500)
root.mainloop()

from tkinter import*
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
def goBack():
    import ndsemi
def fun(value1,name1):
    root = Tk()

    canvas1 = Canvas(root, width=1000, height=50)
    canvas1.pack()
    label_0 = Label(root, text=name1, width=20, font=("chiller", 20))
    label_0.place(x=350, y=00)

    def insert_number():
        x1 = value1
        x2 = 100-x1
        x3 = 0

        figure1 = Figure(figsize=(5, 4), dpi=100)
        subplot1 = figure1.add_subplot(111)
        xAxis = [float(x1), float(x2), float(x3)]
        yAxis = [float(x1), float(x2), float(x3)]
        subplot1.bar(xAxis, yAxis, color='g')
        bar1 = FigureCanvasTkAgg(figure1, root)
        bar1.get_tk_widget().pack(side=LEFT, fill=BOTH, expand=0)

        figure2 = Figure(figsize=(5, 4), dpi=100)
        subplot2 = figure2.add_subplot(111)
        labels2 = 'Matched', 'NotMatched', ' '
        pieSizes = [float(x1), float(x2), float(x3)]
        explode2 = (0, 0.1, 0)
        subplot2.pie(pieSizes, explode=explode2, labels=labels2, autopct='%1.1f%%',
shadow=True, startangle=90)
        subplot2.axis('equal')
        pie2 = FigureCanvasTkAgg(figure2, root)
        pie2.get_tk_widget().pack()

    insert_number()

    button1 = Button(root, text='Submit', width=20, bg='brown', fg='white',
command=goBack)
    button1.place(x=650, y=500)
    root.mainloop()

import os
import argparse
import numpy as np
import tensorflow as tf
from collections import namedtuple

from utils import next_experiment_path
from batch_generator import BatchGenerator


# TODO: add help info
```

```
parser = argparse.ArgumentParser()
parser.add_argument('--seq_len', dest='seq_len', default=256, type=int)
parser.add_argument('--batch_size', dest='batch_size', default=32, type=int)
parser.add_argument('--epochs', dest='epochs', default=30, type=int)
parser.add_argument('--window_mixtures', dest='window_mixtures', default=10,
type=int)
parser.add_argument('--output_mixtures', dest='output_mixtures', default=20,
type=int)
parser.add_argument('--lstm_layers', dest='lstm_layers', default=3, type=int)
parser.add_argument('--units_per_layer', dest='units', default=400, type=int)
parser.add_argument('--restore', dest='restore', default=None, type=str)
args = parser.parse_args()

epsilon = 1e-8



class WindowLayer(object):
    def __init__(self, num_mixtures, sequence, num_letters):
        self.sequence = sequence  # one-hot encoded sequence of characters --
[batch_size, length, num_letters]
        self.seq_len = tf.shape(sequence)[1]
        self.num_mixtures = num_mixtures
        self.num_letters = num_letters
        self.u_range = -tf.expand_dims(tf.expand_dims(tf.range(0.,
tf.cast(self.seq_len, dtype=tf.float32)), axis=0),
                                        axis=0)
    def __call__(self, inputs, k, reuse=None):
        with tf.variable_scope('window', reuse=reuse):
            alpha = tf.layers.dense(inputs, self.num_mixtures, activation=tf.exp,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='alpha')
            beta = tf.layers.dense(inputs, self.num_mixtures, activation=tf.exp,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='beta')
            kappa = tf.layers.dense(inputs, self.num_mixtures, activation=tf.exp,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='kappa')

            a = tf.expand_dims(alpha, axis=2)
            b = tf.expand_dims(beta, axis=2)
            k = tf.expand_dims(k + kappa, axis=2)

            phi = tf.exp(-np.square(self.u_range + k) * b) * a  # [batch_size, mixtures,
length]
            phi = tf.reduce_sum(phi, axis=1, keep_dims=True)  # [batch_size, 1,
length]

            # whether or not network finished generating sequence
            finish = tf.cast(phi[:, 0, -1] > tf.reduce_max(phi[:, 0, :-1], axis=1),
tf.float32)

            return tf.squeeze(tf.matmul(phi, self.sequence), axis=1), \
                tf.squeeze(k, axis=2), \
                tf.squeeze(phi, axis=1), \
                tf.expand_dims(finish, axis=1)

    @property
    def output_size(self):
        return [self.num_letters, self.num_mixtures, 1]



class MixtureLayer(object):
    def __init__(self, input_size, output_size, num_mixtures):
        self.input_size = input_size
        self.output_size = output_size
        self.num_mixtures = num_mixtures

    def __call__(self, inputs, bias=0., reuse=None):
        with tf.variable_scope('mixture_output', reuse=reuse):
            e = tf.layers.dense(inputs, 1,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='e')
            pi = tf.layers.dense(inputs, self.num_mixtures,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='pi')
            mu1 = tf.layers.dense(inputs, self.num_mixtures,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='mu1')
            mu2 = tf.layers.dense(inputs, self.num_mixtures,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='mu2')
            std1 = tf.layers.dense(inputs, self.num_mixtures,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='std1')
            std2 = tf.layers.dense(inputs, self.num_mixtures,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='std2')
            rho = tf.layers.dense(inputs, self.num_mixtures,

kernel_initializer=tf.truncated_normal_initializer(stddev=0.075), name='rho')

            return tf.nn.sigmoid(e), \
                   tf.nn.softmax(pi * (1. + bias)), \
                   mu1, mu2, \
                   tf.exp(std1 - bias), tf.exp(std2 - bias), \
                   tf.nn.tanh(rho)



class RNNModel(tf.nn.rnn_cell.RNNCell):
    def __init__(self, layers, num_units, input_size, num_letters, batch_size,
window_layer):
        super(RNNModel, self).__init__()
        self.layers = layers
        self.num_units = num_units
        self.input_size = input_size
        self.num_letters = num_letters
        self.window_layer = window_layer
        self.last_phi = None

        with tf.variable_scope('rnn', reuse=None):
            self.lstms = [tf.nn.rnn_cell.LSTMCell(num_units)
                          for _ in range(layers)]
            self.states = [tf.Variable(tf.zeros([batch_size, s]), trainable=False)
                           for s in self.state_size]

            self.zero_states = tf.group(*[sp.assign(sc)
                                          for sp, sc in zip(self.states,
                                                            self.zero_state(batch_size,
dtype=tf.float32))])

    @property
    def state_size(self):
        return [self.num_units] * self.layers * 2 + self.window_layer.output_size

    @property
    def output_size(self):
        return [self.num_units]

    def call(self, inputs, state, **kwargs):
        # state[-3] --> window
        # state[-2] --> k
        # state[-1] --> finish
        # state[2n] --> h
        # state[2n+1] --> c
        window, k, finish = state[-3:]
        output_state = []
        prev_output = []

        for layer in range(self.layers):
            x = tf.concat([inputs, window] + prev_output, axis=1)
            with tf.variable_scope('lstm_{}'.format(layer)):
                output, s = self.lstms[layer](x, tf.nn.rnn_cell.LSTMStateTuple(state[2 *
layer],
                                                                                state[2 * layer + 1]))
                prev_output = [output]
            output_state += [*s]

            if layer == 0:
                window, k, self.last_phi, finish = self.window_layer(output, k)

        return output, output_state + [window, k, finish]


def create_graph(num_letters, batch_size,
                 num_units=400, lstm_layers=3,
                 window_mixtures=10, output_mixtures=20):
    graph = tf.Graph()
    with graph.as_default():
        coordinates = tf.placeholder(tf.float32, shape=[None, None, 3])
        sequence = tf.placeholder(tf.float32, shape=[None, None, num_letters])
```

```
reset = tf.placeholder(tf.float32, shape=[None, 1])
bias = tf.placeholder_with_default(tf.zeros(shape=[]), shape=[])

def create_model(generate=None):
    in_coords = coordinates[:, :-1, :]
    out_coords = coordinates[:, 1:, :]

    _batch_size = 1 if generate else batch_size
    if generate:
        in_coords = coordinates

    with tf.variable_scope('model', reuse=generate):
        window = WindowLayer(num_mixtures=window_mixtures,
sequence=sequence, num_letters=num_letters)

        rnn_model = RNNModel(layers=lstm_layers, num_units=num_units,
                             input_size=3, num_letters=num_letters,
                             window_layer=window, batch_size=_batch_size)

        reset_states = tf.group(*[state.assign(state * reset)
                                  for state in rnn_model.states])

        outs, states = tf.nn.dynamic_rnn(rnn_model, in_coords,
                                         initial_state=rnn_model.states)

        output_layer = MixtureLayer(input_size=num_units, output_size=2,
                                    num_mixtures=output_mixtures)
        with tf.control_dependencies([sp.assign(sc) for sp, sc in
zip(rnn_model.states, states)]):
            with tf.name_scope('prediction'):
                outs = tf.reshape(outs, [-1, num_units])
                e, pi, mu1, mu2, std1, std2, rho = output_layer(outs, bias)

            with tf.name_scope('loss'):
                coords = tf.reshape(out_coords, [-1, 3])
                xs, ys, es = tf.unstack(tf.expand_dims(coords, axis=2), axis=1)

                mrho = 1 - tf.square(rho)
                xms = (xs - mu1) / std1
                yms = (ys - mu2) / std2
                z = tf.square(xms) + tf.square(yms) - 2. * rho * xms * yms
                n = 1. / (2. * np.pi * std1 * std2 * tf.sqrt(mrho)) * tf.exp(-z / (2. *
mrho))
                ep = es * e + (1. - es) * (1. - e)
                rp = tf.reduce_sum(pi * n, axis=1)

                loss = tf.reduce_mean(-tf.log(rp + epsilon) - tf.log(ep + epsilon))

            if generate:
                # save params for easier model loading and prediction
                for param in [('coordinates', coordinates),
                              ('sequence', sequence),
                              ('bias', bias),
                              ('e', e), ('pi', pi),
                              ('mu1', mu1), ('mu2', mu2),
                              ('std1', std1), ('std2', std2),
                              ('rho', rho),
                              ('phi', rnn_model.last_phi),
                              ('window', rnn_model.states[-3]),
                              ('kappa', rnn_model.states[-2]),
                              ('finish', rnn_model.states[-1]),
                              ('zero_states', rnn_model.zero_states)]:
                          tf.add_to_collection(*param)

        with tf.name_scope('training'):
            steps = tf.Variable(0.)
            learning_rate = tf.train.exponential_decay(0.001, steps,
staircase=True,
                                                   decay_steps=10000, decay_rate=0.5)

            optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
            grad, var = zip(*optimizer.compute_gradients(loss))
            grad, _ = tf.clip_by_global_norm(grad, 3.)
            train_step = optimizer.apply_gradients(zip(grad, var),
global_step=steps)

        with tf.name_scope('summary'):
            # TODO: add more summaries
            summary = tf.summary.merge([
                tf.summary.scalar('loss', loss)
```

```
                    ])

                return namedtuple('Model', ['coordinates', 'sequence', 'reset_states', 'reset',
'loss', 'train_step',
                                            'learning_rate', 'summary'])(
                            coordinates, sequence, reset_states, reset, loss, train_step,
learning_rate, summary
                      )
        train_model = create_model(generate=None)
        _ = create_model(generate=True)  # just to create ops for generation

    return graph, train_model



def main():
    restore_model = args.restore
    seq_len = args.seq_len
    batch_size = args.batch_size
    num_epoch = args.epochs
    batches_per_epoch = 100

    batch_generator = BatchGenerator(batch_size, seq_len)
    g, vs = create_graph(batch_generator.num_letters, batch_size,
                         num_units=args.units, lstm_layers=args.lstm_layers,
                         window_mixtures=args.window_mixtures,
                         output_mixtures=args.output_mixtures)
    config = tf.ConfigProto(allow_soft_placement=True)
    config.gpu_options.allocator_type = 'BFC'
    config.gpu_options.per_process_gpu_memory_fraction = 0.90

    with tf.Session(graph=g) as sess:
        model_saver = tf.train.Saver(max_to_keep=2)
        if restore_model:
            model_file = tf.train.latest_checkpoint(os.path.join(restore_model,
'models'))
            experiment_path = restore_model
            epoch = int(model_file.split('-')[-1]) + 1
            model_saver.restore(sess, model_file)
        else:
            sess.run(tf.global_variables_initializer())
            experiment_path = next_experiment_path()
            epoch = 0

        summary_writer = tf.summary.FileWriter(experiment_path, graph=g,
flush_secs=10)

summary_writer.add_session_log(tf.SessionLog(status=tf.SessionLog.START),
                                       global_step=epoch * batches_per_epoch)

        for e in range(epoch, num_epoch):
            print('\nEpoch {}'.format(e))
            for b in range(1, batches_per_epoch + 1):
                coords, seq, reset, needed = batch_generator.next_batch()
                if needed:
                    sess.run(vs.reset_states, feed_dict={vs.reset: reset})
                l, s, _ = sess.run([vs.loss, vs.summary, vs.train_step],
                                   feed_dict={vs.coordinates: coords,
                                              vs.sequence: seq})
                summary_writer.add_summary(s, global_step=e * batches_per_epoch +
b)
                print('\r[{:5d}/{:5d}] loss = {}'.format(b, batches_per_epoch, l), end='')

            model_saver.save(sess, os.path.join(experiment_path, 'models', 'model'),
                             global_step=e)


if __name__ == '__main__':
    main()
```

CONCLUSION

Crop yield prediction is still remaining as a challenging issue for farmers. The aim of this research is to propose and implement a rule based system to predict the crop yield production from the collection of past data.

REFERENCES

1. Chaochong, J. M. (2008). Forecasting Agricultural Production via Generalized Regression Neural Network.IEEE .

2. Gupta, A. S. (2016). Need Of Smart Water Systems In India. International Journal of Applied Engineering Research, 2216-2223.

3. Hong-Ying L, Y.-L. H.-J.-M. (2012).Z.Crop yield forecasted model based on time series techniques. Journal of Northeast Agricultural University (English Edition).

4. J. Khazaei, M. R. (2008). Yield estimation and clustering of chickpea genotypes using soft computing techniques. Agron J 2008.

5. Ji, B. S.,& Wan, J. (2007). Artificial neutral networks for rice yield prediction in mountainous regions. Joural of Agricultural science.

6. Kumar, R. M. (2009). Crop Selection Method to Maximize Crop Yield Rate Using Machine Learning Technique.International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, 1 (1).

7. M.Shashi, Y. a. (2009). Atmospheric Temperature Prediction using Support Vector Machines. International Journal of Computer Theory and Engineering, 1 (1).

8. Miss.Snehal, S. D. (2014). Agricultural Crop Yield Prediction Using Artificial. International Journal of Innovative Research in Electrical,Electronic, 1 (1).

9. Obua, W. O. (2011). Machine Learning Classification Technique for Famine Prediction.Proceedings of the World Congress on Engineering, 2.