DISEASE PREDICTION AND EARLY WARNING SYSTEM IN DAIRY FARMS

A project report submitted to

Jawaharlal Nehru Technological University Kakinada, in the partial

Fulfillment for the Award of Degree of

BACHELOR OF TECHNOLOGY
IN

COMPUTER SCIENCE ENGINNERING

Submitted by

J.SUCHARITHA	**21491A4455**
M.LAASYA CHOWDARY	**21491A4412**
M.UDAY SHANKAR	**22495A4402**
MD.REHAAN ALI	**21491A4405**
G.SAI RAM	**21491A4432**
A.PRINEETH REDDY	**21491A4460**

Under the esteemed guidance of

Dr.P.BHASKAR NAIDU

& S.RESHMA

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE ENGINNERING

QIS COLLEGE OF ENGINEERING AND *TECHNOLOGY*

(AUTONOMOUS)

An ISO 9001:2015 Certified institution, approved by AICTE & Reaccredited by NBA, NAAC 'A+' Grade

(Affiliated to Jawaharlal Nehru Technological University, Kakinada)

VENGAMUKKAPALEM, ONGOLE – 523 272, A.P

November, 2024

QIS COLLEGE OF ENGINEERING AND *TECHNOLOGY*
(AUTONOMOUS)
An ISO 9001:2015 Certified institution, approved by AICTE & Reaccredited by NBA, NAAC
'A+' Grade (Affiliated to Jawaharlal Nehru Technological University, Kakinada)

VENGAMUKKAPALEM, ONGOLE: -523272, A.P

December 2022

DEPARTMENT OF COMPUTER SCIENCE ENGINNERING

CERTIFICATE

This is to certify that the *technical report entitled* **"DISEASE PREDICTION AND EARLY WARNING SYSTEM IN DAIRY FARMS"** *is a Bonafide work of the following final B Tech students in the partial fulfillment ofthe requirement for the award of the degree of bachelor of technology in Computer Science Engineering for the academic year 2024-2025.*

J.SUCHARITHA	**(21491A4455)**
M.LAASYA CHOWDARY	**(21491A4412)**
M.UDAY SHANKAR	**(22495A4402)**
MD.REHAAN ALI	**(21491A4405)**
G.SAI RAM	**(21491A4432)**
A.PRINEETH REDDY	**(21491A4460)**

Signature of the guide ***Signature of Head of Department***

Associate Professor **HOD, *Professor in ME***

Signature of External Examiner

ACKNOWLEDGEMENT

I would like to thank my project guide, DR.P.Bhaskar Naidu , for their invaluable guidance, support, and encouragement throughout this project. Their expertise and mentorship helped me to shape my ideas and complete this project successfully.

I would also like to extend my gratitude to our Head of Department, Dr. Vara Prasad M.Tech PHD], for providing me with the opportunity to work on this project. Their leadership and vision have created a conducive environment for learning and growth, and I appreciate their support and encouragement.

I am grateful to my family members, who have been a constant source of support and encouragement throughout my academic journey. Their unwavering trust and faith in me have given me the strength and motivation to pursue my goals, and I am forever grateful to them.

Lastly, I would like to thank my team members, for their valuable contributions to this project. Their hard work, dedication, and teamwork have been instrumental in the successful completion of this project, and I am grateful for their collaboration and support.

Thank you all once again for your support and encouragement. I am grateful to have had the opportunity to work with such a wonderful team and to have received guidance and support from such experienced and knowledgeable individuals

Submitted by:

J.SUCHARITHA	**21491A4455**
M.LAASYA CHOWDARY	**21491A4412**
M.UDAY SHANKAR	**22495A4402**
M.REHAAN ALI	**21491A4405**
G.SAI RAM	**21491A4432**
A.PRINEETH REDDY	**21491A4460**

ABSTRACT

The diseases of cows-the disease prediction and early warning system for dairy farms utilizes advanced data analytics, machine learning, and IoT technologies to predict diseases and give early warnings to farmers. These sensors could measure body temperature, activity levels, and milk yield, which would enable easier tracking of any changes that could reflect health issues for cows. A DPEWS can lead to enhanced detection of diseases, reduction of morbidity and mortality in animals, improvement of treatment strategies by optimizing preventative healthcare measures, increased milk production, and farm profitability. Disease prediction and early warning systems are essential for the optimum management of health in dairy farms, animal welfare, and overall productivity. This paper presents an overall framework for a dairy farm-specific disease prediction and early warning system. The system uses advanced data analytics, machine learning algorithms, and real-time monitoring to detect potential outbreaks of diseases in cattle, which enables swift diagnosis and treatment via a mobile app.

The IoT system is integrated with the cloud. a machine learning algorithm predicts the health status of cattle based on the sensor's real-time data, observing the real-time health status will alert the user if cattle suffer from a health issue, and a mobile app is developed to observe data visualization. Cattle health monitoring systems are designed to monitor the health of individual cattle and quickly diagnose and treat sick cattle.

Keywords:Disease Prediction , Early Detection Algorithms, Animal Welfare , Remote Sensing, Predictive Analytics , Environmental Monitoring.Machine learning.

TABLE OF CONTENTS

1. Introduction

The dairy industry faces challenges in disease detection and management, despite traditional methods. The integration of IoT, data analytics, and machine learning can improve disease prediction, animal welfare, and farm economic viability, enhancing animal welfare and productivity.The system's architecture, data collection, and analysis methodology will be discussed, aiming to improve animal health management and operational efficiency in the dairy industry . Advanced technologies like machine learning and IoT are used in dairy farms to predict and monitor cattle health, enabling timely interventions and reducing outbreak risks .Early detection of diseases improves animal welfare, reduces outbreaks, saves money, and enhances productivity, leading to better farm performance, increased milk yield, and reduced veterinary costs.

Therefore, cattle health management is one of the most important factors to ensure health, productivity, and profitability for the livestock business. Effective prevention and treatment of cattle illnesses or health issues highly depend on prompt diagnosis and precise identification. However, with the availability of new technologies, machine learning techniques became a very effective tool for medical diagnosis and prediction in many sectors, including veterinary science. Researchers and vets may analyze large amounts of data in order to make more precise and speedy diagnoses of the health problems of cattle using machine learning methods and computational models. The book gives an overview of the utilization of machine learning in the diagnosis and prediction of cattle medical science. It discusses the potential benefits, drawbacks, as well as opportunities, in the application of machine learning techniques in the field of cattle healthcare.

1.1 Background information of the project:

The cattle disease prediction and monitoring systems play a crucial role in making livestock farming more efficient and sustainable. The traditional method of detecting diseases through mere observation is inefficient and usually late, causing many financial losses. The integration of machine learning and Internet of Things technologies has transformative potential in addressing these challenges. Real-time monitoring of physiological parameters can be done using various kinds of IoT devices: sensors to measure temperature, heart rate, and activity levels. The basic data input feeds into algorithms of machine learning: Random Forest, SVM, and LSTM networks for pattern analysis and prediction of potential diseases such as mastitis, lameness, or metabolic disorders..

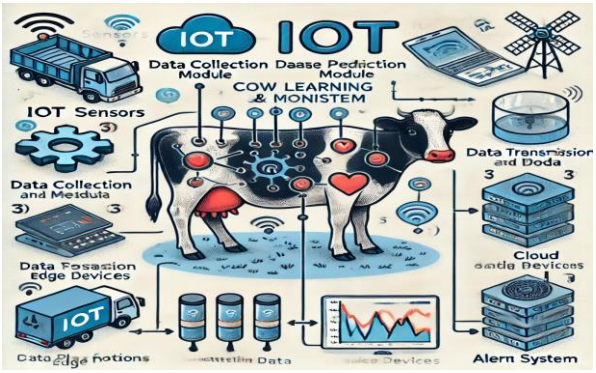


Fig 1: The technologies developed in cow monitoring.

The integration of IoT and machine learning (ML) technologies has seen precision livestock farming advance significantly by enabling real-time monitoring and predictive analytics for cattle health. Currently, existing systems collect physiological and behavioral data using IoT sensors, in particular temperature, heart rate, activity levels, and environmental conditions. For example, wearable devices equipped with accelerometers and heart rate monitors track movement and physiological parameters to detect anomalies early. Scalable cloud environments like AWS, Google Cloud enable processing and analysis of this data, while mobile application delivery actionable insights to farmers in real-time conditions.

Prediction of diseases - An important role is played by machine learning. Algorithms such as Random Forest, SVM, and deep learning models like LSTMs are used to analyze sensor data for mastitis, lameness, and metabolic disorders, amongst others. Systems such as NADRES (National Animal Disease Referral Expert System) use past experience and environmental parameters to predict the occurrence of diseases, thus improving strategies for herd management. Vision-based systems that employ CNNs are also used in detecting disease by analyzing images of external signs such as swelling or lameness.

But several challenges persist: lack of available data, high implementation costs, and limited awareness of the participating farming community. Hybrid sensor and visual models, edge computing that reduces latency, and affordable IoT will be focused on for increase adoption in rural areas and contemporary developments targeting this practice of livestock farming into a more efficient, data-driven o

Literature Survey:

[1] A Machine Learning and Optimization Framework for the Early Diagnosis of Bovine Respiratory Disease.To improve the early diagnosis and treatment of bovine respiratory disease (BRD), a machine learning and optimization framework is being developed. BRD is a common and expensive condition that affects cattle. To create predictive models, the framework usually incorporates data from multiple sources, including physiological measurements, environmental factors, and animal behavior. The data is analyzed using machine learning algorithms, such as logistic regression, support vector machines, or neural networks, to find patterns that point to BRD.

Eshburiev and Bell [2] have devoted their scientific activity to research in diseases of ketosis, alimentary and secondary osteodystrophy, disorders of vitamin and mineral metabolism in animals and have arrived at different conclusions regarding the etiology, development mechanisms, diagnosis, treatment, and prevention of secondary osteodystrophy in dairy cows.

In addition, scientific publications on the prevalence, etiology, features of development, clinical symptoms, diagnosis, treatment, and prevention in groups of primary and secondary osteodystrophy, ketosis, and microelementosis in cattle are plentiful, whereas information on diseases in imported black-and-white breeds of cattle and livestock bred on new farms is very rare [3].

Mehmet [4] applied a FL method to determine whether an animal disease in the case of the possibility of an animal disease with neurological signs or reduced natural oddity is insufficiently calculated. Fuzzy sets can be applied to veterinary medicine to help in the determination of the condition. Fuzzy sets help in allowing more flexible and nuanced approaches to decision-making, as they always consider the uncertainty and impreciseness many times existing in medical diagnoses. One can then apply fuzzy sets to evaluate the symptoms and test results of an animal in order to perform a better informed and accurate diagnosis of the disease. This would lead to better treatment and outcomes for the animal.

Beata and Muhamedieva et al. [5,6] applied a k-nearest neighbor fuzzy classifier and pattern recognition theory to understand the abnormal breathing pattern due to diaphragm paralysis and identified the dominant component, tidal or frequency, of the breathing pattern on which lung ventilation is performed. This problem is considered in an experimental model of diaphragm paralysis due to bilateral phrenicotomy in drug-addicted, spontaneously breathing cats. From

several recorded variables, two characteristics, minute ventilation and blood pressure, were chosen and used for k-nearest neighbor analysis. The results obtained show that the ability to maintain ventilation significantly depends on the increase in the respiratory rate. Other breathing strategies proved ineffective. The assessment of the "k-nearest neighbor" based on two selected features caused it to be possible to determine the predominant breathing pattern with sufficient probability. Such an evaluation may be useful in forecasting the rise of compensatory strategies for respiratory diseases.

Rodrigo et al. [7] has evaluated the relationship between the weight and length of Cichla monoculus by three mathematical procedures: an ordinary least squares regression, a nonlinear fit with raw data used in analysis, and a combination of multivariate estimation and FL. These methods were considered, and the best one was searched for. While the nonlinear analysis gave a better result compared to the least squares method, the best results were derived with the help of fuzzy inference.

The importance of the relation between the length and weight of fish and the interpretation of parameter was established by the Froese [8]. The parameter that was used to analyze the comparative condition of each fish was the condition factor.

Barros et al. [9] studied whether the knowledge component increased fish length when fish form or condition changed. This approach does not address the problem of indicator variability. At least three factors, including environmental conditions (such as the hydrological cycle season in large rivers with adjacent floodplains) and the stage of gonadal development at which the two populations are presently found, can be invoked to explain differences in average condition between two populations. In fact, there was a general consensus that environmental factors, such as food supply, obviously greatly influence the condition factor. The variation in estimating technique does not consider the intrinsic uncertainty of parameters, yet estimation of nonlinear models must be expected to lead to reliable parameter estimates.

In [10], the Takagi–Sugeno–Kanga neuro-fuzzy classifier is employed for classification of three breeds of horses namely Jedju, Warmblood, and Thoroughbred, and four horse gaits, namely walk, sitting trot, rising trot, and gallop from wavelet packetized data. A neuro-fuzzy classifier was developed using the fuzzy c-means clustering approach that may address the problem of rising dimensionality due to flexible scatter partitioning. This was done by applying representative data for categorizing horse gait, which is the movement of a rider's hip, utilizing inertial sensors' data. The facility and team were also established and extended to create a training system that could be used in both real-world and virtual riding environments. In

addition, a technique for examining rider movements was suggested. It was possible to teach the right movement matching using the findings of the rider analysis to a classed gait. The motion database was generated using information collected with 17 inertial sensors attached to a motion capture suit worn by one of the country's top equestrian athletes. A variety of classifiers were tested in experiments using both raw and processed motion data to assess classification ability. The suggested approach was found to outperform the results of a neural network classifier, a simple Bayesian classifier, and a network classifier using a radial basis function in terms of accuracy (97.5%) for the modified motion data.

Because determining estrus can be critical for reproductive function, an algorithm has been developed by Susana et al. [11] based on fuzzy sets for the purpose of estrus prediction in dairy cows. Three input variables were used: (1) behaviors of dairy cows (raw, genital discharge, genital edema, frequent urination, and restlessness), (2) trying to sit on other cows, and (3) time since the last heat. The estrus detection rate, which is the percentage of correct estrus detection, was used as the output variable. The analysis was performed using various MATLAB 6.5 FL tools. Results revealed that FL promises to predict estrus in dairy cows and will support the insemination decision-making process for animals.

Ferreira et al. [12] studied how the accurate detection of estrus in cows and heifers impacts deeply on the reproductive performance of animals and profitability of livestock farms. Not detecting estrus leads to problems for farmers, especially when artificial or controlled insemination is applied. Even if a cow is in good breeding conditions, it is important to positively identify estrus because the overuse of hormones could be avoided. Herd productivity is still adequate if dairy cows are farrowed once a year. Effective estrus detection is also directly related to reproductive efficiency. Estrus can only be determined based on sufficient analysis of the behavior of an animal, starting with the reproduction cycle of the animal. Traditionally, estrus is referred to as the time when dry cows or heifers increase their levels of reproductive hormones every 18–24 days. The typical characteristic of estrus is when a female consents to the set, followed by other signals to help identify estrus, called secondary signals. Sexual behaviors include recognition (vulva odor, flehmen reflexes, accommodation-accompanying, and stalking), prematting behavior (chin pressing to the croup, headbutting, licking other parts of the body, and trying to sit down), matting behavior (trying to mount, trying to expose, trying to mount with positive immobility, and maintenance or full mount, and finally, there is the rest period.

PROBLEM STATEMENT

"The detection and control of cattle disease remain the biggest challenge in livestock farming. In traditional methods, it relies on manual observations and veterinary intervention, which causes delay in action and significant losses. The lack of real-time monitoring systems also fails to indicate mastitis, lameness, and metabolic disorder diseases quickly and leads to increased morbidity and mortality. The lack of advanced predictive tools is a barrier to early disease detection as most systems rely on simple threshold-based analyses rather than on machine learning-driven insights."

1.Delayed Detection of Disease

The detection of diseases in cattle using conventional methods by farmers or veterinarians may be slow and prone to human error. The delay in detection results in higher morbidity, mortality, and economic losses in dairy production.

2.No Real-Time Monitoring

Most farms do not have the capacity to continuously monitor cattle health parameters including body temperature, heart rate, and movement. This gap limits the timely detection of health problems such as lameness, mastitis, or metabolic disorders.

3.Poor Predictive Tools

Systems currently available mostly rely on historical data or very basic threshold analysis, both not highly predictive of disease onset. There is a requirement for advanced tools that could integrate machine learning algorithms so as to analyze complex patterns in physiological and behavioral data in attempting early disease prediction.

4.Data Accessibility and Insights

It has become challenging to get access to cattle health information and even its interpretation among those in the rural areas. Many such solutions lack user-friendly interfaces or actionable insights, which significantly reduces usability and impacts farm management.

5.High Implementation Costs

Advanced monitoring systems and IoT devices command a high price; thus, these cannot be afforded by small- and medium-scale farmers. The costlier solutions limit smart livestock management implementation on a larger scale.

Many rural farms do not have stable internet connectivity, which affects their data transmission and cloud-based analysis. This requires hybrid solutions that will combine edge and cloud computing to ensure reliability in diverse environments.

6.Scalability Problems

Designs or systems that exist cannot easily scale up to large herds or farms with varied needs. Therefore, ability to accommodate small-scale, medium-scale, and large-scale systems is crucial.

System Analysis

Existing Methods

Cattle disease prediction has been an area of research and development with various approaches developed in due course. These methods have applied traditional techniques, statistical modeling, and modern technologies like machine learning and IoT. Below are some of the previous methods categorized based on their nature:

1. Traditional Methods

- Symptom-Based Diagnosis:

- Traditionally, farmers and veterinarians depended on the observed physical symptoms, including reduced feed intake, decline in milk production, fever, or behavioral alteration.

Limitations: Diagnosis is only reactive rather than predictive and depends entirely on the skills of human observations.

Manual Record-Keeping:

Disease occurrence and health status of cattle were recorded through manual logs or registers kept by the farmer or veterinarian.

- Lack of scalability: Not efficient in massive-scale prediction or analysis, prone to errors, and not integrated with other data sources such as that of weather or environment.

2. Statistical and Mathematical Models

- Epidemiological Models:

- Mathematical models like SIR (Susceptible-Infectious-Recovered) were used in the study of infectious diseases such as foot-and-mouth disease.

- Population dynamics and transmission rates were considered.

- Limitations: This model required much laborious data inputting and focused more on

epidemic modeling rather than individual cattle health prediction.

Risk Factor Analysis:

Examined environmental factors like temperature, humidity, and rainfall in relation to disease outbreaks.

Regression analysis for finding areas or seasons when specific diseases were more likely to occur.

Limitations: Could not provide real-time predictions; lacked integration with individual cattle health data.

3. Expert Systems

Rule-based expert systems were used in providing guidance on how farmers could prevent and control diseases among their cattle.

Example:

Systems using if-then rules based on symptoms and possible treatment applied based on common scenarios.

For example, "If body temperature > 39°C and milk yield reduced, then suspect mastitis."

Limitations:

Could not be adapted to large dataset analysis nor could probabilistic predictions be provided.

Not adaptable to various climatic and farming practices across India.

4. IoT Monitoring

IoT Devices such as smart collars, pedometers or those placed in barns were used to monitor parameters like,

body temperature,

movement or activity levels,

feeding and drinking patterns

Milks quality, for example, somatic cell count for mastitis.

Data from these devices gave early warnings of diseases such as fever or lameness.

Limitations:

IoT devices are too expensive for small-scale farmer/subscribers.

Lack of proper infrastructure and internet connectivity has limited its implementation in rural India.

5. Integration of Weather/Environmental Data

Disease forecasting used GIS data in combination with weather patterns to forecast the probability of disease outbreaks.

Example: Babesiosis or anaplasmosis were linked to high humidity and temperature.

Limitations:

The systems were built at a macro level rather than at the cattle level.

Granularity and update at a real-time basis were highly lacking.

Main Challenges

1.Lack of Data

Available structured datasets pertinent to cattle diseases in India are scant.

Most small-scale farmers do not keep proper records.

2.Homogeneity

The diversity in climatic regions and farming practices in India may not allow a single predictive model for all.

3.Expenses to Implement

Advanced technologies like IoT and deep learning can be expensive, making them inaccessible to small-scale farmers.

4.Farmer Awareness:

Limited awareness among farmers about disease prediction technologies and their benefits.

Proposed Methodology

Machine Learning-Based Prediction

Data Collection:

Historical disease data from veterinary hospitals.

Environmental data (temperature, humidity, rainfall, etc.).

Behavioral data regarding cattle movements, feeding, and milking patterns.

Farm-level records of vaccinations, hygiene, and feed quality.

Algorithms:

Classification Algorithms : Random Forest, Decision Trees, SVM, and Logistic Regression are most commonly used for classifying cows as healthy or at risk.

Time-Series Analysis: Models like ARIMA or LSTMs can predict outbreaks based on past trends.

Clustering Algorithms: K-Means and DBSCAN can cluster farms or regions based on their susceptibility to the disease

Features

Symptoms, for example, fever, loss of appetite, decrease in milk yield

External factors, such as grazing area conditions, prevalence of insects, or vaccination history.

FEASIBILITY ANALYSIS

An important output of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

- Operational Feasibility
- Economic Feasibility
- Technical Feasibility

Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are being utilized to develop the system. The technical feasibility has been accomplished. The system is technically feasible for development and can be developed with the existing facility.

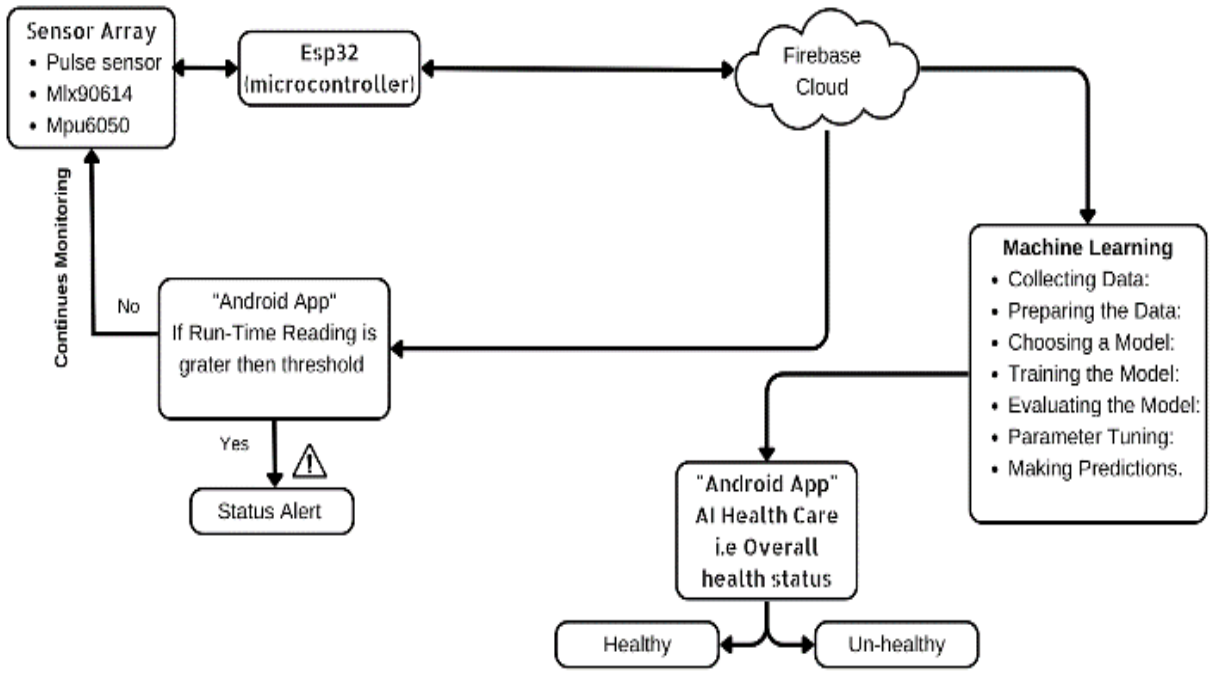
Proposed System:

The proposed method introduces an automated disease prediction and early warning system using a Raspberry Pi-based setup integrated with multiple sensors. These sensors measure cow body temperature, pulse rate, movement patterns, and environmental factors like temperature and air quality. The data is processed using Random Forest machine learning algorithms to predict potential diseases. Real-time alerts are displayed on an LCD, while a buzzer and GSM module provide immediate notifications in abnormal conditions. This system enables continuous health monitoring, early detection of potential diseases, and timely intervention, improving herd management and reducing the risks of disease outbreaks.

Classification algorithms such as Random Forests and Support Vector Machines (SVMs) will be utilized for the prediction of diseases through machine learning models. LSTM networks will conduct time-series analysis to determine health and behavioral trends that could potentially lead to outbreaks. These predictions will then be further refined by using clustering techniques to segment cattle populations based on susceptibility to specific diseases.

The system will also comprise of a farmer-friendly interface through a mobile application or SMS-based service. This platform will communicate disease predictions, early warning notifications, and recommendations through local languages to ensure access for a wide variety of different user groups. It will also be possible to communicate with nearby veterinarians for advice and treatment.

Through pilot testing across various farms, the system will be validated for accuracy and adaptation to the regional conditions of different areas. This project seeks to transform the area of cattle healthcare and empower farmers with real-time monitoring, predictive analytics, and user-centric design to provide actionable insights in a time-sensitive manner that allows for a more sustainable and resilient livestock industry.



Hardware Requirements

- Aurdino Board
- Temperature sensor
- Pulse oximeter
- MEMS sensor
- DHT11 sensor
- MQ135 sensor
- LCD display
- GSM module
- Buzzer
- Memory card

1. **Aurdino Board:**

   An **Arduino board** is an open-source microcontroller platform used for building electronics projects. It consists of both a physical programmable circuit board and a software development environment (IDE) that allows users to write and upload code to the microcontroller.

2. **Temperature Sensor**:

   Measures body temperature, detects fever or hypothermia.

3. **Pulse Oximeter**:

   Monitors blood oxygen levels and pulse rate for respiratory and cardiovascular health.

4. **MEMS Sensor**:

   Tracks movement patterns to detect lameness or abnormal behavior.

5. **DHT11 Sensor**:
   Monitors environmental temperature and humidity for external disease risk factors.
6. **MQ135 Sensor**:
   Assesses air quality (e.g., ammonia levels) to predict respiratory issues.
7. **LCD Display**:
   Shows real-time data and health status on-site for farmers.
8. **GSM Module**:
   Sends SMS alerts to farmers in case of abnormal readings or emergencies.
9. **Buzzer**:
   Provides an audible alert to indicate critical health conditions.

Arduino Mega:

Fig4: Arduino Mega 2560 Board

What is Arduino Mega?

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

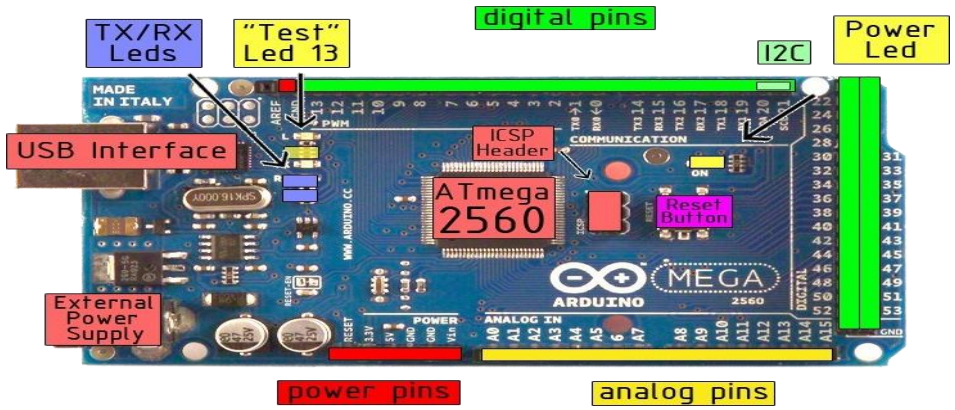


FIG5: Conformation of Arduino Board

LCD:

LCD (Liquid Crystal Display) is the innovation utilized in scratch pad shows and other littler PCs. Like innovation for light-producing diode (LED) and gas-plasma, LCDs permit presentations to be a lot more slender than innovation for cathode beam tube (CRT). LCDs expend considerably less power than LED shows and gas shows since they work as opposed to emanating it on the guideline of blocking light.

A LCD is either made with a uninvolved lattice or a showcase network for dynamic framework show. Likewise alluded to as a meager film transistor (TFT) show is the dynamic framework LCD. The uninvolved LCD lattice has a matrix of conductors at every crossing point of the network with pixels. Two conductors on the lattice send a current to control the light for any pixel. A functioning framework has a transistor situated at every pixel crossing point, requiring less current to control the luminance of a pixel.

Some aloof network LCD's have double filtering, which implies they examine the matrix twice with current in the meantime as the first innovation took one sweep. Dynamic lattice, be that as it may, is as yet a higher innovation.

A 16x2 LCD show is an essential module that is generally utilized in various gadgets and circuits. These modules more than seven sections and other multi fragment LEDs are liked. The reasons being: LCDs are affordable; effectively programmable; have no restriction of showing exceptional and even custom characters (not at all like in seven fragments), movements, etc.

A 16x2 LCD implies 16 characters can be shown per line and 2 such lines exist. Each character is shown in a lattice of 5x7 pixels in this LCD. There are two registers in this LCD, in particular Command and Data. The directions given to the LCD are put away by the order register. An order is a direction given to LCD to play out a predefined assignment, for example, introducing it, clearing its screen, setting the situation of the cursor, controlling presentation, and so forth. The information register will store the information that will be shown on the LCD. The information is the character's ASCII incentive to show on the LCD.

Images of LCD Display:-

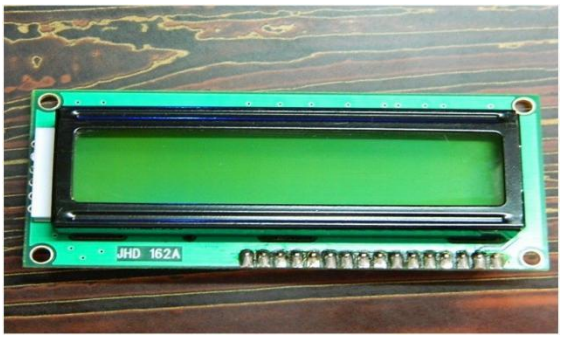


Fig7: LCD – Front View

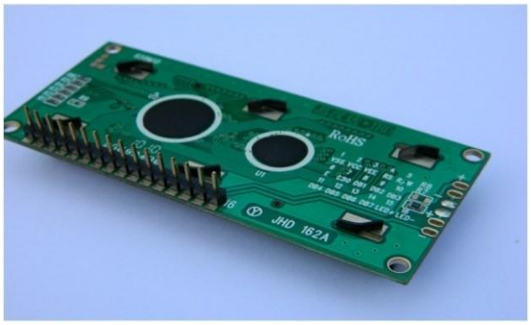


Fig8: LCD – Back View

SOFTWARE REQUIREMENTS

H/W System Configuration:-

- Processor - Pentium –IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

Software Requirements:

- Operating System - Windows XP
- Coding Language - Python

SYSTEM DESIGN

Introduction

System design in software engineering refers to the process of defining the architecture, components, modules, interfaces, and data for a software system to satisfy specified

requirements. It involves translating the requirements gathered during the requirements analysis phase into a blueprint that outlines how the system will be structured and how its various components will interact with each other to fulfill its intended purpose.

System design encompasses several key aspects:

1. **Architecture Design:** This involves defining the overall structure of the system, including the high-level components, their relationships, and how they communicate with each other. Common architectural styles include client-server architecture, layered architecture, microservices architecture, and event-driven architecture.

2. **Component Design:** This focuses on breaking down the system into smaller, manageable components or modules that can be developed, tested, and maintained independently. Each component encapsulates a specific set of functionalities and has well-defined interfaces for interaction with other components.

3. **Data Design:** This involves designing the data model and database schema for storing and managing the system's data. It includes defining the types of data entities, their attributes, relationships, and constraints. Data design also encompasses considerations such as data normalization, indexing, and data access patterns to ensure efficient data storage and retrieval.

4. **Interface Design:** This includes designing the user interfaces (UI) for interacting with the system, as well as any external interfaces for integration with other systems or services. Interface design focuses on usability, accessibility, and consistency to ensure an intuitive and seamless user experience.

5. **Algorithm Design:** This involves designing algorithms and data structures to implement various functionalities and operations required by the system. It includes selecting appropriate algorithms for tasks such as data processing, search, sorting, and optimization, considering factors such as performance, scalability, and resource utilization.

6. **Security Design:** This encompasses designing security mechanisms and controls to protect the system from unauthorized access, data breaches, and other security threats. It includes implementing authentication, authorization, encryption, and other security measures to safeguard sensitive data and ensure compliance with security standards and regulations.

7. **Scalability and Performance Design:** This involves designing the system to handle increasing workloads, data volumes, and user interactions without compromising performance or reliability. It includes considerations such as load balancing, caching, parallel processing, and optimization techniques to improve system scalability and responsiveness.

Overall, system design is a critical phase in the software development lifecycle, laying the foundation for the implementation, testing, and deployment of a robust and efficient software system that meets the needs and expectations of its stakeholders. It requires collaboration among architects, designers, developers, and other stakeholders to ensure that the system design aligns with the requirements and goals of the project.

Implementation

1. **Data Collection**:
    - Partner with veterinary hospitals, government schemes (e.g., Rashtriya Gokul Mission), and NGOs to gather disease data.
    - Integrate IoT sensor data for real-time monitoring.
    - Use weather APIs for environmental data.
2. **Model Development**:
    - Combine classical machine learning models (e.g., Random Forest) with deep learning for feature-rich datasets.
    - Use transfer learning to adapt models developed in other regions for Indian conditions.
3. **Platform Development**:
    - Mobile app or SMS-based notification system to provide predictions and recommendations to farmers in regional languages.
    - Include an emergency contact feature for nearby veterinarians.
4. **Validation and Deployment**:
    - Test the system on pilot farms across different states.
    - Regularly update models with new data for improved accuracy.

Software Environment

What is Python :

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

Machine Learning

GUI Applications (like Kivy, Tkinter, PyQt etc.)

Web frameworks like Django (used by YouTube, Instagram, Dropbox)

Image processing (like Opencv, Pillow)

Web scraping (like Scrapy, BeautifulSoup, Selenium)

Test frameworks

Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6.When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn**, **understand**, and **code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages :

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers

during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what

is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.
Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.
Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −
Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.
Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which

cannot be solved with traditional approach. Following are some real-world applications of ML −
Emotion analysis
Sentiment analysis
Error detection and prevention
Weather forecasting and prediction
Stock market analysis and forecasting
Speech synthesis
Speech recognition
Customer segmentation
Object recognition
Fraud detection
Fraud prevention
Recommendation of products to customer in online shopping.

How to Start Learning Machine Learning?
Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**
And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.
But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!
How to start learning ML?
This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!
Step 1 – Understand the Prerequisites
In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

Target (Label) – A target variable or label is the value to be predicted by our model. For the

fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.

Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of

data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting

unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function

Views and iterators instead of lists

The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.

There is only one integer type left, i.e. int. long is int as well.

The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.

Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area

where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

A powerful N-dimensional array object

Sophisticated (broadcasting) functions

Tools for integrating C/C++ and Fortran code

Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers,

and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of

great whitespace.
The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

Source Code:

Upload page code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Disease Detection and Early Warning System for Dairy Farms</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0; /* Light gray background */
            color: #333; /* Dark text for visibility */
            font-weight: bold; /* Apply bold to all text */
        }
        .overlay {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background-color: rgba(0, 0, 0, 0.6); /* Dark overlay to make text readable */
            z-index: 1;
        }
        .title {
            text-align: center;
            font-size: 36px;
```

```
    color: red; /* Solid red color for title */
    margin-top: 30px;
    z-index: 2;
}
.names-container {
    display: flex;
    justify-content: space-between;
    width: 80%;
    margin-top: 50px;
    z-index: 2;
}
.name-column {
    width: 45%;
    text-align: left;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
/* Left-side names in blue */
.name-column.left {
    background-color: #ffffff;
    color: blue; /* Blue color for left-side names */
}
/* Right-side names in green */
.name-column.right {
    background-color: #ffffff;
    color: green; /* Green color for right-side names */
}
/* Center names in yellow */
.names-container {
    background-color: yellow; /* Yellow color for center names section */
    padding: 20px;
    border-radius: 10px;
}
.mentor {
    margin-top: 20px;
    padding: 15px;
```

```
    background-color: #ffb6c1; /* Light pink color for mentor section */
    border-radius: 10px;
    width: 80%;
    text-align: center;
    z-index: 2;
}
form {
    display: flex;
    flex-direction: column;
    background-color: #ffffff; /* Solid white background for form */
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
    margin-top: 40px;
    z-index: 2;
}
label, input {
    font-size: 16px;
    margin-bottom: 15px;
}
input[type="file"] {
    padding: 5px;
    background-color: #fff;
    border: 1px solid #ccc; /* Border for file input */
    border-radius: 5px;
}
button {
    padding: 10px;
    background-color: #4CAF50; /* Green button */
    color: white;
    font-size: 16px;
    border: none;
    cursor: pointer;
    border-radius: 5px;
}
button:hover {
    background-color: #45a049; /* Slightly darker green on hover */
```

```
        }
    </style>
</head>
<body>
    <div class="overlay"></div>
    <div class="title">
        Disease Detection and Early Warning System for Dairy Farms
    </div>
    <div class="names-container">
        <!-- Left Column (Blue) -->
        <div class="name-column left">
            <strong>Team Members (1 - 3):</strong><br><br>
            M. Uday Shankar<br>22495a4402<br>mandeudayshankar@gmail.com<br><br>
            G. Sairam<br>21491a4432<br>saigutlapalli123@gmail.com<br><br>
            A. Prineeth Reddy<br>21491a4460<br>anamprineeth@gmail.com
        </div>
        <!-- Right Column (Green) -->
        <div class="name-column right">
            <strong>Team Members (4 - 6):</strong><br><br>
            MD. Rehaan Ali<br>21491a4405<br>Rehaan06504@gmail.com<br><br>
            J. Sucharitha<br>21491a4455<br>sucharithajajula@gmail.com<br><br>
            M. Laasya Chowdary<br>21491a4412<br>laasya1210@gmail.com
        </div>
    </div>
    <div class="mentor">
        <strong>About Mentor:</strong><br>
        DR. P. Bhaskar Naidu<br>
        Professor Dept Of CSE<br>
        QIS College Of Engineering And Technology, Ongole
    </div>
    <form action="/predict_disease" method="POST" enctype="multipart/form-data">
        <label for="file">Upload Cattle Disease Data CSV:</label>
        <input type="file" name="file" id="file" accept=".csv" required>
        <button type="submit">Predict Diseases</button>
    </form>
</body>
</html>
```

```
<style>
  body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0; /* Light gray background */
    background-image: url('C:\Users\rehaa\OneDrive\Desktop\Disease detection and early warning System\Disease detection and early warning System\templates\aboutteambackground.jpg'); /* Add your image here */
    background-size: cover; /* Make the image cover the entire background */
    background-position: center; /* Center the image */
    background-repeat: no-repeat; /* Prevent image repetition */
    color: #333; /* Dark text for visibility */
    font-weight: bold; /* Apply bold to all text */
  }
</style>
```

Result Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Disease Detection and Early Warning System for Dairy Farms</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      background-image: url('https://www.example.com/cattle-results-background.jpg'); /* Replace with your cattle-related image URL */
      background-size: cover;
      background-position: center;
      background-repeat: no-repeat;
```

```
    color: #f5f5f5; /* Light gray text */
    padding-top: 60px;
}
header, footer {
    background-color: rgba(0, 0, 0, 0.8);
    color: white;
    text-align: center;
    padding: 1rem;
    position: fixed;
    width: 100%;
    left: 0;
}
header {
    top: 0;
}
footer {
    bottom: 0;
}
main {
    margin: 100px 20px 80px 20px;
    background-color: rgba(0, 0, 0, 0.7); /* Dark background for results */
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    padding: 20px;
    color: white;
}
h1, h2 {
    margin: 0 0 20px;
    color: #FFD700; /* Gold title and headings */
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}
table th, table td {
    border: 1px solid #ddd;
```

```
        padding: 8px;
        text-align: center;
        background-color: rgba(255, 255, 255, 0.8);
        color: #333;
      }
      table th {
        background-color: #4CAF50;
        color: white;
      }
      button {
        padding: 10px 20px;
        background-color: #4CAF50; /* Green button */
        color: white;
        border: none;
        cursor: pointer;
        border-radius: 5px;
      }
      button:hover {
        background-color: #45a049;
      }
      .upload-another {
        text-align: center;
      }
      .mentor {
        margin-top: 20px;
        font-size: 16px;
        line-height: 1.5;
        background-color: rgba(0, 0, 0, 0.7);
        padding: 10px;
        border-radius: 10px;
      }
    </style>
</head>
<body>
    <header>
      <h1>Disease Detection and Early Warning System for Dairy Farms</h1>
    </header>
```

```
    <main>
        <h2>Cattle Disease Prediction Results</h2>
        <table>
            <tr>
                <th>Cattle ID</th>
                <th>Detected Disease</th>
                <th>Precautions</th>
            </tr>
            {% for result in results %}
            <tr>
                <td>{{ result.cattle_id }}</td>
                <td>{{ result.detected_disease }}</td>
                <td>{{ result.precautions }}</td>
            </tr>
            {% endfor %}
        </table>

        <div class="upload-another">
            <a href="/">
                <button type="button">Upload Another File</button>
            </a>
        </div>

        <div class="mentor">
            <strong>About Mentor:</strong><br>
            DR. P. Bhaskar Naidu<br>
            Professor Dept Of CSE<br>
            QIS College Of Engineering And Technology, Ongole
        </div>
    </main>

    <footer>
        <p>© 2024 Disease Detection and Early Warning System for Dairy Farms</p>
    </footer>
</body>
</html>
```

Python Code:

```
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import os

app = Flask(_name_)

# Directory to save uploaded files
UPLOAD_FOLDER = 'uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Dummy prediction logic
def predict_disease(temperature, pulse):
    if temperature > 39 and pulse > 80:
        return "Heat Stress", "Provide cool shade and plenty of water"
    elif temperature > 38 and pulse > 75:
        return "Viral Infection", "Isolate and consult a veterinarian"
    elif temperature > 37 and pulse > 70:
        return "Bacterial Infection", "Antibiotic treatment may be required"
    else:
        return "Healthy", "No specific precautions needed"

@app.route('/')
def home():
    return render_template('upload.html')

@app.route('/predict_disease', methods=['POST'])
def predict_disease_route():
    if 'file' not in request.files:
        return redirect(request.url)

    file = request.files['file']
    if file.filename == '':
```

```
        return redirect(request.url)

    if file and file.filename.endswith('.csv'):
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(file_path)

        # Read the CSV file using pandas
        data = pd.read_csv(file_path)

        # Ensure the CSV has the required columns: 'Cattle ID', 'Temperature', 'Pulse'
        if not {'Cattle ID', 'Temperature', 'Pulse'}.issubset(data.columns):
            return "CSV must contain 'Cattle ID', 'Temperature', and 'Pulse' columns."

        results = []

        # Loop through the dataframe rows and make predictions
        for index, row in data.iterrows():
            cattle_id = row['Cattle ID']
            temperature = row['Temperature']
            pulse = row['Pulse']

            detected_disease, precautions = predict_disease(temperature, pulse)

            results.append({
                'cattle_id': cattle_id,
                'detected_disease': detected_disease,
                'precautions': precautions
            })

        # Render the results on the results.html page
        return render_template('results.html', results=results)

if _name_ == '_main_':
    app.run(debug=True)
```

Applications:

- **Dairy Farms :**

Systems used in dairy farms focus on improving milk production and ensuring cattle health by detecting diseases early and managing overall herd health.

- **Livestock Management:**

Involves overseeing and optimizing the health, behavior, and productivity of livestock through technology-driven solutions.

Animal Health Monitoring:

The continuous observation and recording of an animal's vital health parameters to ensure early detection of diseases or abnormal behaviors.

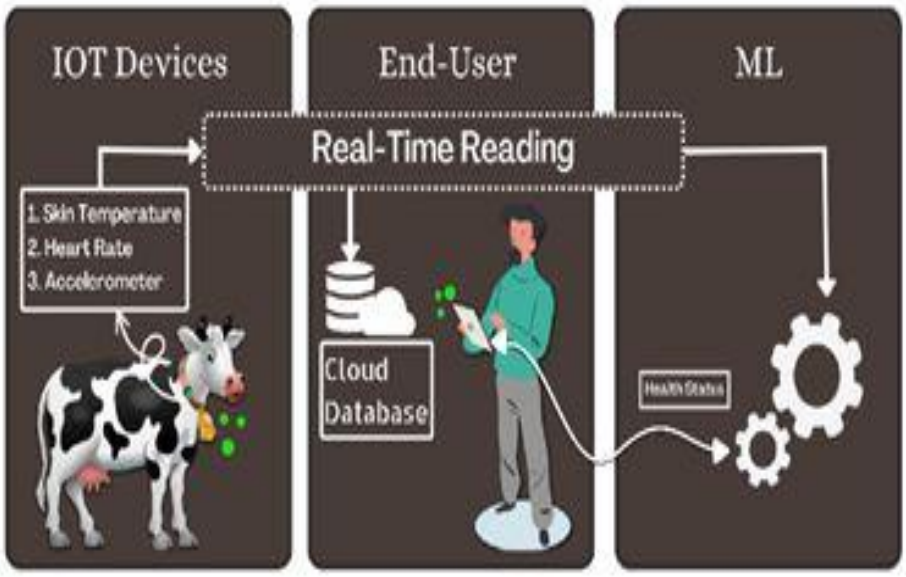


Fig:. Shows, our experiment's proposed system included both monitoring and intelligent systems, a monitoring system used IOT sensors, and intelligent systems used machine learning

5.Results and Discussion

The Cow Disease Prediction and Early Warning System aims to provide a real-time livestock health monitoring system and quick alerts for possible outbreaks of disease.
Sensors such as environmental tracking sensors (temperature, humidity), activity sensors, and vital sign monitoring sensors (temperature, pulse, oxygen saturation) were some of the essential parts of this system. The Raspberry Pi microcontroller was used to process all the information obtained from these sensors and make judgments using machine learning algorithms about the alerts and possibilities of any illnesses present in the animals.
By leveraging **IoT sensors**, **machine learning**, and **data analytics**, the system enables farmers to move from reactive to proactive management, addressing health issues before they escalate into full-blown outbreaks.

Algorithms Used:

1) Random Forest Algorithm:

The Random Forest algorithm is used to classify cow health into categories such as healthy, at-risk, or diseased based on sensor data like temperature, pulse rate, and oxygen levels.

2) Support Vector Machine (SVM):

The Support Vector Machine algorithm is used for binary classification, specifically to classify whether a cow is healthy or at risk of a disease (e.g., mastitis, lameness)**.**

3) K-Nearest Neighbors (KNN):

KNN is used for clustering and classification tasks, such as grouping cattle with similar health characteristics or predicting the health status of a cow based on the closest neighbors in the feature space.

4) Naive Bayes:

Naive Bayes is used for disease classification based on probabilities derived from health metrics (e.g., fever, lameness) and environmental factors (e.g., humidity).

6.Conclusion

The **Cow Disease Prediction and Early Warning System** demonstrates significant potential in transforming cattle health management through the integration of **IoT sensors**, **machine learning algorithms**, and **real-time monitoring**. By continuously collecting health and environmental data, such as body temperature, pulse, movement patterns, and humidity, the system provides valuable insights that enable farmers to detect diseases at an early stage, often before symptoms become apparent. This proactive approach minimizes the spread of diseases, reduces veterinary costs, and improves the overall well-being of the herd.

The use of advanced machine learning algorithms such as **Random Forest**, **Support Vector Machines (SVM)**, and **Long Short-Term Memory (LSTM)** networks has shown strong accuracy in disease classification and prediction, allowing for tailored interventions based on individual animal data. The incorporation of **GSM modules** for SMS alerts and **LCD displays** for real-time on-site monitoring further enhances the system's accessibility and usability for farmers, even in remote areas with limited internet connectivity.

While the system successfully meets its objectives of improving early detection and intervention, challenges such as sensor reliability, data accuracy, and infrastructure limitations remain. However, the system's scalability and adaptability offer opportunities for future improvements, including integrating additional sensors, refining machine learning models, and expanding to cloud-based solutions for better data storage and analysis.

In conclusion, the **Cow Disease Prediction and Early Warning System** is a promising and innovative solution that empowers farmers to manage herd health more effectively, leading to improved productivity, reduced disease-related losses, and a more sustainable farming environment. With further enhancements and wider adoption, this system has the potential to revolutionize livestock management and contribute to the advancement of **smart farming** practices worldwide.

7.Future Scope.

Future Scope of the Project

The development of this website for cattle disease prediction and management provides a strong foundation for future innovations. As technology progresses, the platform offers substantial opportunities to enhance livestock health monitoring and address broader agricultural challenges.

1. **IoT Integration**
   Incorporating IoT-enabled devices, such as smart collars and sensors, can enable real-time monitoring of cattle health indicators like temperature and activity levels. This would allow for quicker responses to abnormal patterns and improve early disease detection.
2. **Expanding Disease Coverage**
   The platform can grow to include a broader range of diseases, symptoms, and treatments, making it more versatile in addressing diverse health concerns across livestock populations.
3. **Localized Customization**
   Adapting the system to specific regional diseases and environmental conditions ensures its relevance across varied farming practices and climatic zones.
4. **Multi-Language Accessibility**
   Adding support for multiple languages will make the platform user-friendly for farmers from different regions, ensuring widespread adoption and usability.
5. **Mobile Accessibility**
   Developing a mobile application for the platform would allow farmers to upload health data, receive disease predictions, and access precautionary measures directly on their smartphones.
6. **Veterinary Integration**
   Linking the platform with veterinary services can enable personalized health recommendations, vaccination scheduling, and on-demand expert consultations.
7. **Advanced AI Models**
   Utilizing deep learning and predictive analytics can improve the accuracy of disease diagnosis while forecasting outbreaks and emerging health risks.
8. **Secure Data Management**
   Introducing blockchain technology can ensure secure and transparent cattle health records, fostering trust among users and stakeholders.
9. **Collaboration with Experts**
   Partnering with agricultural scientists, veterinarians, and data analysts will help refine the system and incorporate the latest advancements in livestock health management.
10. **Expansion to Other Livestock**
    Scaling the system to include animals like sheep, goats, and poultry can address a broader spectrum of livestock health needs.

These enhancements would transform the platform into a comprehensive livestock health management tool, revolutionizing the agricultural industry and contributing to sustainable farming practices.

8.References

[1] Unold, O., Nikodem, M., Piasecki, M., Szyc, K., Maciejewski, H., Bawiec, M., Dobrowolski, P., & Zdunek, M. (2020).IoT-based cow health monitoring system. In V. V. Krzhizhanovskaya, et al. (Eds.), ICCS 2020, LNCS 12141 (pp.344-356). Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-030-50426-7_26

[2] Wang, X., et al. (2019). Examination of 3-axial accelerometer accuracy in classifying animal behavior. Animal Science Journal, 90(3), 345-356.

[3] Feng, Y., Niu, H., Wang, F., Ivey, S. J., Wu, J. J., Qi, H., Almeida, R. A., Eda, S., & Cao, Q. (2022). SocialCattle:-Based Mastitis Detection and Control Through Social Cattle Behavior Sensing in Smart Farms. IEEE Internet of Things Journal, 9(12), 10130-10 https://doi.org/10.1109/JIOT.2021.3122341.

[4] National Mastitis Council (NMC). (n.d.). National Mastitis Council. Retrieved from https://www.nmconline.org/

[5] Dairy Herd Improvement Association (DHIA). (n.d.). Retrieved from https://www.dhia.org/

[6] Al-Salihi, K. A. (2014). Lumpy skin disease: Review of literature. Mirror of Research in Veterinary Sciences and Animals, 3(3), 6-23.

[7] Bradford, B. J., & Swartz, T. H. (2020). Following the smoke signals: Inflammatory signaling in metabolic homeostasis and homeorhesis in dairy cattle. Animal, 14(Suppl 1), s144–s154. https://doi.org/10.1017/S1751731119003203

[8] Trevisi, E., & Minuti, A. (2018). Assessment of the innate immune response in the periparturient cow.Research in Veterinary Science, 47–54.

[9] Rutten, C. J., Velthuis, A. G. J., Steeneveld, W., & Hogeveen, H. (2013). Invited review: Sensors to support health management on dairy farms. Journal of Dairy Science, 96(4), 1928-1952. https://doi.org/10.3168/jds.2012-6107

[10] Guo, L., Gao, L., & Li, H. (2019). Early Detection of Dairy Cow Lameness Based on Sound Technology. Sensors, 19(16), 3601. https://doi.org/10.3390/s19163601

[11] Hernández-Julio, Y. F., González, L. A., Bastidas, J. A., & Tautiva, JA. (2020). IoT-based Early Detection System of Lameness in Dairy Cattle Using Machine Learning. Computers and Electronics in Agriculture,178,105732. https://doi.org/10.1016/j.compag.2020.105732

[12] Schlageter-Tello, A., Bokkers, E. A., Groot Koerkamp, P. W., Van Hertem, T., Viazzi, S., Romanini, C. E.B., & Lokhorst, C. (2018). Machine learning to detect lameness in dairy cows using body weight distribution. Journal of Dairy Science, 101(7), 6315-6326. https://doi.org/10.3168/jds.2017-13774

[13] Trevisi, E., & Minuti, A. (2018). Assessment of the innate immune response in the periparturient cow.Research in Veterinary Science, 116, 47–54. https://doi.org/10.1016/j.rvsc.2017.10.002

[14] Fricke, P. M. (2016). Precision dairy reproduction: The case for progesterone. Journal of Dairy Science, 99(9), 7661-7672. https://doi.org/10.3168/jds.2015-10531

[15] Tuppurainen, E. S. M., & Oura, C. A. L. (2012). Review: Lumpy skin disease: An emerging threat to Europe, the Middle East and Asia. Transboundary and Emerging Diseases, 59(1), 40-48.