National Animal Disease Referral Expert System

(NADRES V2)

NADRES V2 - Complete R Programming Codes for Livestock Disease Risk Prediction

(Data Capturing, Data Processing, Data Modeling, Data Communication)

ICAR – NATIONAL INSTITUTE OF VETERNIRY EPIDOMOLOGY AND DISEASE INFORMATICS (ICAR – NIVEDI)

PUBLISHED BY:
DIRECTOR
@ ICAR-NIVEDI

Citation: Suresh K P (2024). NADRES V2-Complete R programming Codes for Livestock Disease Risk Prediction, ICAR- NIVEDI, Bengaluru, 1-105.

Year: 2024

Published by: Director, ICAR- National Institute of Veterinary Epidemiology and Disease Informatics (NIVEDI), Yelahanka, Bengaluru-560064.

PME NUMBER: F.No.11/NIVEDI/PMEC/RPS/2021-22/2425-26

Development Team:
Dr. K. P. Suresh, Nodal officer, NADRES V2
Ms. Sushma R, YP-II, NADCP CSF-CP Project
Ms. Raaga R, SRF, NICRA Project
Mr. Dheeraj R, JRF, DBT-AdMac-I Project

Manual Prepared by:

Dr. K. P. Suresh, Nodal officer, NADRES V2

Disclaimer

The forewarnings are based on the retrospective disease data available in the NADRES database. Hence, for those states wherein data is limited/less, the forewarning may not be realistic. Further the forewarning will not take into consideration the control measures that are *in situ*.

Acknowledgements

I would like to acknowledge the constant support and inspiration from Hon'ble Secretary, DARE and Director General, ICAR, Government of India, New Delhi.

I would like to express sincere everlasting gratitude to Hon'ble Deputy Director General (Animal Science) for his constant encouragement, support and guidance.

I would also like to express sincere gratitude to the Department of Animal Husbandry and Dairying (DAHD), Ministry of Agriculture and Farmers Welfare, Government of India for providing the livestock population data for use in model building.

Animal Husbandry Departments of state governments and also NADEN centres are gratefully acknowledged for the timely submission of reports of livestock disease outbreak data. I am thankful to all the scientific and technical staff of ICAR-NIVEDI for their feedback and support.

I would like to extend my heartfelt gratitude to the National Animal Disease Control Programme (NADCP) and the National Innovations on Climate Resilient Agriculture (NICRA) projects for their invaluable financial support and manpower contributions in the preparation of this document.

I would like to acknowledge The R Foundation for Statistical Computing, c/o Institute for Statistics and Mathematics 1020 Vienna, Austria for providing the opportunity to work in open source.

Furthermore, I would also like to acknowledge with much appreciation the and support received from the scientists, Dr. Rajeswari Shome, Dr. D. Hemadri, Dr. S.S. Patil, Dr. P. Krishnamoorthy, Dr. S.J. Siju and SRF's, JRF's, Young Professionals and other contractual staff working in Disease Informatics Lab/Spatial Epidemiology Lab in preparing this R code document.

K. P. Suresh, Ph.D
Principal Scientist

TABLE OF CONTENT

1. ABOUT THE DOCUMENTATION…

This document provides a comprehensive, systematic approach to livestock disease prediction and management in India, focusing on the application of R programming. It presents advanced forecasting techniques, machine learning models, and data analytics to predict risks associated with 15 economically significant livestock diseases. Through detailed methodologies and R programming codes, it offers a robust framework for disease prevention and control. Structured across 105 pages, the documentation ensures a logical progression from data collection to forecasting results, emphasizing scientific rigor and practical application.

Forecasting in livestock health involves quantitative predictions based on historical data, enabling decision-makers to anticipate disease outbreaks and implement timely interventions. Initially applied in fields like agriculture and meteorology, forecasting has evolved to incorporate advanced methods such as time series analysis, econometric models, and machine learning algorithms. In livestock disease management, these techniques are crucial for analyzing complex data patterns, including environmental factors, animal populations, and historical disease incidence. The ability to predict disease dynamics allows for effective disease prevention, minimizing economic losses and supporting rural livelihoods.

R programming plays a central role in developing and applying these forecasting models. As an open-source platform, R offers extensive tools for statistical analysis, machine learning, and data visualization. Specialized R packages like forecast, tsibble, prophet, and caret provide an environment for constructing robust models that predict livestock disease risks. The integration of machine learning models within R enhances forecasting accuracy by identifying non-linear relationships and patterns that traditional models may miss. Additionally, R's visualization tools, such as ggplot2 and shiny, enable the creation of dynamic and interactive visualizations, making complex forecasting results more accessible to stakeholders.

The efficiency of machine learning (ML) models in handling high-dimensional, complex datasets lies in their ability to automatically select relevant features, model intricate relationships, and process large volumes of data with minimal human intervention. Models such as random forests, support vector machines (SVM), neural networks, gradient boosting machines (GBM), k-nearest neighbors (KNN), decision trees, and ensemble methods have demonstrated superior performance in livestock disease forecasting. These models excel in both classification tasks, where random forests, SVMs, and neural networks accurately categorize disease occurrences, and in forecasting tasks, where models like recurrent neural networks (RNN) and long short-term memory (LSTM) predict disease outbreaks based on temporal data. Their ability to capture non-linear interactions and complex relationships within datasets significantly enhances prediction accuracy compared to traditional statistical approaches. As more data becomes available, ML models continue to refine their predictions, offering real-time, data-driven insights that are essential for timely and effective disease intervention and control strategies.

The document outlines a four-stage workflow for livestock disease forecasting:

Data Capturing: The process of data capturing involves the systematic collection of comprehensive and high-quality data from sources such as veterinary records, environmental databases, and historical disease trends. Accurate and detailed data is the foundation for any predictive model, as it enables the detection of underlying patterns and relationships that drive disease outbreaks. High-quality data ensures that models can identify complex, non-linear interactions between variables, such as the influence of environmental factors on disease spread. Without accurate data capturing, the subsequent steps in the modelling process are compromised, leading to biased or unreliable predictions.

Data Processing: Data processing involves cleaning, normalizing, and aggregating raw data to ensure consistency and reliability across datasets. This step is crucial for addressing missing values, handling outliers, and performing necessary transformations such as scaling or encoding variables. Through techniques like imputation for missing data and normalization to standardize variables, the dataset becomes suitable for machine learning and statistical models. Processing also involves feature engineering, where new variables are derived to enhance the model's predictive power. Proper data processing ensures that the model inputs are accurate and that the model is capable of learning meaningful relationships without being influenced by noise or outliers.

Data Modelling: Once the data is processed, both statistical and machine learning models are applied to predict future disease risks. This step involves selecting the appropriate models based on the nature of the data and the prediction task. Statistical models like generalized linear models (GLMs) capture linear relationships, while machine learning models such as random forests, support vector machines (SVM), and neural networks identify complex, non-linear patterns. By training on historical data, these models can forecast future disease outbreaks, accounting for interactions between environmental, biological, and historical variables. The accuracy of these predictions is heavily dependent on the quality of data and the appropriateness of the model chosen.

Data Communication: After the model produces forecasts, data communication translates the results into actionable insights through reports, visualizations, and interactive dashboards. This step is critical for ensuring that stakeholders, such as veterinarians, policy makers, and disease control agencies, can understand and interpret the results. Effective data communication not only presents the forecast but also conveys uncertainty, confidence intervals, and risk factors associated with the predictions. Clear, data-driven communication allows for timely and informed decision-making, enabling the implementation of necessary interventions to mitigate disease risks and improve animal health outcomes.

Why R Programming?

R programming is exceptionally suited for livestock disease forecasting due to its robust computational capabilities and extensive suite of statistical tools. The efficiency of R in managing large, complex datasets is underpinned by its advanced memory management system, which allows for effective in-memory data manipulation. This capability is further enhanced by specialized packages like data.table and bigmemory, which provide optimized data structures and algorithms to handle extensive

datasets without overwhelming system memory. R's storage capacity is augmented through these external memory packages, enabling it to manage and analyse datasets that exceed the physical RAM limits of a machine. This is achieved through techniques such as data chunking and virtual memory management, which ensure that operations on large datasets remain feasible and efficient.

Additionally, R's integration with parallel and distributed computing frameworks allows for the scaling of computations across multiple processors and nodes, optimizing both performance and memory usage. This is crucial for processing and analysing voluminous data typical in livestock disease forecasting. R's support for GPU acceleration via packages like keras and tensorflow further extends its computational capabilities. By leveraging Graphics Processing Units (GPUs), R accelerates the training and execution of complex machine learning models, particularly deep learning algorithms that require substantial computational resources.

These features collectively make R an indispensable tool for livestock disease forecasting. Its ability to handle large datasets, manage memory efficiently, and utilize advanced computational resources ensures that it can address the intricate demands of modelling, forecasting, and visualizing disease trends with precision and effectiveness.

NIVEDI's Contribution to Livestock Disease Management

The ICAR - National Institute of Veterinary Epidemiology and Disease Informatics (NIVEDI) plays a critical role in addressing economically significant livestock diseases in India. NIVEDI's success in eradicating diseases like Rinderpest demonstrates the effectiveness of predictive strategies. Building on this, NIVEDI has focused on 15 priority livestock diseases, developing a comprehensive database that supports the National Animal Disease Referral Expert System (NADRESv2). This system, powered by advanced forecasting methodologies and R programming, provides monthly disease forewarnings, disseminated through bulletins to national and state-level animal husbandry departments. These forewarnings enable veterinarians to take preventive measures, reducing the likelihood of disease outbreaks and protecting the livestock sector.

Through its integration of predictive modelling, data analytics, and R programming, NIVEDI has developed a scientifically robust system that enhances livestock health management in India. The methodologies outlined in this document not only advance the technical precision of disease forecasting but also contribute to the broader goal of sustaining rural economies and improving public health.

SUMMARY

This document serves as a detailed guide to the scientific methods and R programming codes used in forecasting livestock diseases in India. By combining advanced statistical techniques, machine learning models, and comprehensive data handling capabilities, the framework presented here equips researchers, veterinarians, and policymakers with the tools needed for effective disease prediction and control. NIVEDI's efforts, supported by NADRESv2, underscore the importance of accurate forecasting in safeguarding the livestock industry and enhancing rural livelihoods.

3. INTRODUCTION TO NADRES v2

The geographic and seasonal distribution of many infectious diseases are associated with climate and therefore the possibility of using seasonal climate forecasts as predictive indicators in disease early warning system (EWS) became imminent. In this context, ICAR-NIVEDI, in its quest for achieving better livestock health, had developed an interactive web portal named "National Animal Disease Referral Expert System (NADRES)" during early part of the first decade of the millennium. The web portal, which was developed from the financial support of National Agricultural Technology Project, was launched in the year 2005. The portal which is interactive, allows the user/stakeholder to access livestock disease forewarning (n=13) at the district level for entire country two months in advance. The portal which was initially built on oracle platform was later changed to MySQL platform to store the administrator provided disease information and other relevant meteorological and risk factor information. However, with the availability of remote sensed satellite images and the advancement in information technology and statistical algorithms, the upgradation of NADRES became inevitable. To this end, a newer version of NADRES (NADRES *V2*) has been developed.

How it is different from previous version?

In brief, it can be said that NADRES $_{V2}$ underwent a sea change not only in its internal structure but also in its physical design. As a result, now the central menu bar consists of Home, about us, Risk factors, Analysis, Livestock disease, post prediction validation and contact details. Risk factors menu comprises of details on resolution, time interval, units and source of 18 meteorological and 5 remote sensing parameters. Analytics menu has various analysis options. The newly created livestock disease menu has the details regarding species affected, clinical signs and preventive measures to be adopted for the 15 economically important diseases. Post prediction validation menu contains the outbreak reports vs prediction. The menu bar on the RHS tabs include online GIS, state wise Livestock disease forecast, district wise Livestock disease forecast, Epi-calculator, download links for mobile app, etc. The website now hosts disease maps in the form of choropleth maps for 15 diseases in two time periods (1990-2000 and 2000-2018). Similarly, disease trends plots exhibit periodic regression plots providing future trend for the disease. On the LHS, Login menu is provided for authorized persons to login and enter disease details and other related parameters. Disease maps provide choropleth maps for 15 diseases in two time periods (1990-2000 and 2000-2018) is presented. Disease trends- Periodic regression plots are exhibited for prediction of the diseases. Auto-messaging option has been created to send the reminders in the form of text messages to concerned PI's and Co-PI's of NADEN centers for submission of outbreak reports. This message is sent weekly to all the concerned officials. Additionally, a message is sent to the concerned veterinary officers in Karnataka for initiation of preventive measures for the forewarned diseases at the block level. Plans are in place to incorporate farmers' and local vets' mobile numbers in to the list so that they may be asked to initiate preventive measures for the forewarned diseases.

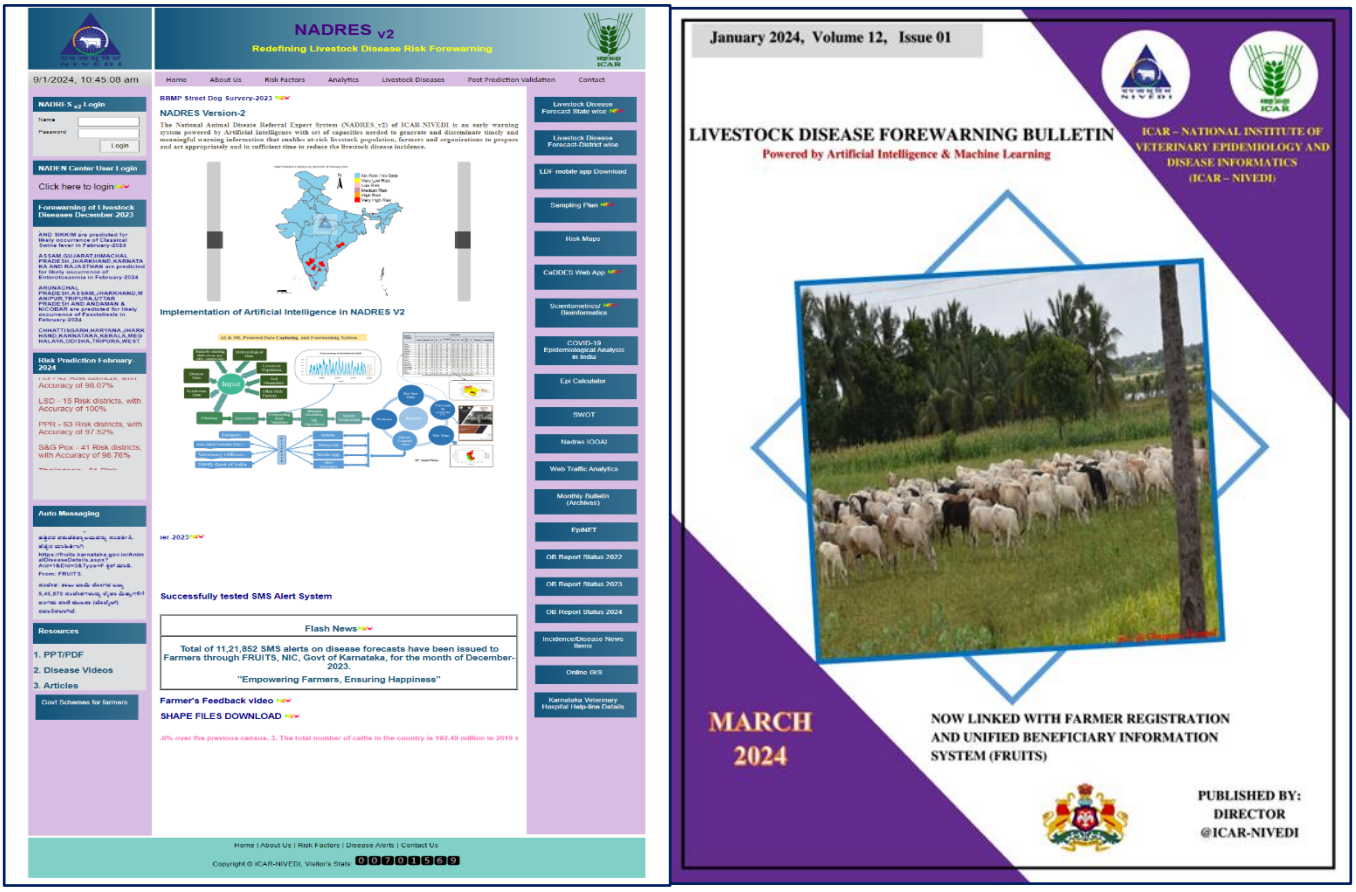


Fig 3.1. NADRES V_2 Home page

The forewarning methodology used is unique and has not been used earlier for livestock disease forewarning in India. Following few paragraphs describe about the forewarning methodology used. It is a well-known fact that weather plays an important role in the precipitation of many diseases and therefore, the climatic parameters such as land surface temperature (LST), precipitation, wind velocity, humidity etc are considered as risk parameters. These parameters along with other non-climatic parameters such as livestock population, density, Normalized Differential Vegetation Index (NDVI), soil moisture constitute the overall risk parameters. A total of 23 such parameters are collected/generated at village level and then aggregated to district level before these are used for analysis.

In addition to the output provided at interactive web portal, the NADRES output are also published in the form of monthly livestock disease forewarning bulletins. The prediction results come with a disclaimer that forewarnings do not take into account of the control measures that already in situ and also may not be realistic for those regions where the data is either unavailable or limited. This bulletin provides the likely occurrence of the 15 shortlisted diseases two months in advance at the district level, disease forewarning maps, prediction accuracy, details on diseases, species affected, clinical signs and its preventive measures.

In summary, it can be said that NADRES $_{V2}$ has underwent substantial changes not only in its internal structure but also in its physical design and can be a useful tool for visitors of the website, farmers, vets, policy makers etc.

4. Forewarning Methodology

Preamble

NADRES v2 is an early warning system powered by Artificial Intelligence with set of capacities needed to generate and disseminate timely and meaningful warning information that enables at-risk livestock population, farmers and organizations to prepare and act appropriately and in sufficient time to reduce the livestock disease incidence.

Objectives

- Development of forecasting model for the major livestock diseases and predicting the risk of livestock diseases in advance of two months.
- Development of state of art of communication models to communicate risk of livestock diseases to the stake holders.

I. Materials and data acquisition

Livestock disease data

Previous 10 years' livestock disease outbreak data retrieved from the NADRES database linked with Risk factors data.

Livestock population data

The population data at village level for five major livestock species viz., cattle, buffalo, sheep, goat and pigs were obtained from 20th Livestock census (2019) from Department of statistics, DAHD, GOI.

Species-wise & Category-wise Livestock Population (in thousands)					
Sl No	**Species**	**Category**	**Population in 2012**	**Population in 2019**	**% Change**
1	Cattle	Exotic	39732	51356	29.3
		Indigenous	151172	142106	-6
		Total	190904	193462	1.3
2	Buffalo	Total	108702	109852	1.1
3	Sheep	Exotic	3781	4088	8.1
		Indigenous	61288	70172	14.5
		Total	65069	74260	14.1
4	Goat	Total	135173	148885	10.1
5	Pig	Exotic	2456	1897	-22.8
		Indigenous	7837	7159	-8.7
		Total	10293	9056	-12
6	Yaks	Total	77	58	-24.7
7	Mithuns	Total	298	386	29.5
8	Horses & Ponies	Total	625	342	-45.3
9	Mules	Total	196	84	-57.1
10	Donkeys	Total	319	124	-61.1
11	Camels	Total	400	252	-37
Total Livestock			**512056**	**536761**	**4.8**

Meteorological and Remotely Sensed Data:

The parameters such as air temperature (^{0}C), perceptible water (mm), Precipitation rate (mm), pressure (millibar), relative humidity (%), Vwind (m/s), U wind (m/s), Soil temperature (^{0}C), Vapour pressure(hPa), Wet day frequency, and sea level pressure (millibar) were extracted from National Centre for environmental prediction (NCEP). The parameters such as potential evapotranspiration (PET), Enhanced Vegetation Index (EVI), Leaf Area Index (LAI), Land Surface Temperature (LST), Normalised Difference Vegetation Index (NDVI) were extracted from remote sensed images from MODIS website (https://modis.gsfc.nasa.gov/). In brief, the MODIS products from NASA-TERRA satellite was downloaded for the Indian locations by specifying the tiles (H24V5, H25V6, H24V6, H24V7, H25V7, H25V8, H26V7, H26V6) from 2001 to till date.

The details are given below;

PRODUCT	Science Data Sets (HDF Layers)
MOD15A2H	Lai_500m (Leaf area index) 8 days average
MOD16A2	PET_500m (Total Potential Evapotranspiration) 8 days average
MOD11A2	LST_Day_1km (Daytime Land Surface Temperature) 8 days average
MOD13A1	i. 500m 16 days NDVI (Normalized Difference Vegetation Index) ii. Enhanced Vegetation Index (EVI) 16 days average

The downloaded HDF files (Datasets, which are multidimensional arrays (layers) of a homogeneous type) were converted to GeoTIFF files (single layer data) using R packages, which were later used to extract the parameters by linking it with the sinusoidal values of the Indian villages. The scale factors were multiplied for the extracted values as specified by the MODIS data products to get the values of the parameters. As shown above, the atmospherically corrected NDVI was collected on 16-day interval at 250-meter resolution using MODIS product MOD13A1 and LST was collected on 8-day interval using MOD11A2 at 1 KM resolution.

The parameters such as rainfall, Soil moisture (%), and Wind speed (m/s) were obtained from Global Land Data Assimilation System of NASA (https://disc.gsfc.nasa.gov). The remaining parameters were downloaded from climatic research unit (CRU) of University of East Anglia website. It is worth mentioning that the entire process of extraction, assimilation, processing and aligning have been done using R programming language and R environment. After aligning the climatic and non-climatic data with the disease and the livestock population data (aggregated at the district level), the statistical analysis was performed in the R environment.

Static and Dynamic set

Two distinct sets of data regarding risk variables have been established: a static dataset comprising the average weather parameters spanning a decade, from 2011 to 2022, and a dynamic dataset containing recent years' data, specifically from 2023 and 2024. This categorization facilitates a scientific approach to analyzing temporal patterns, trends, and variations in the designated risk factors, allowing for comprehensive assessment and prediction of potential impacts.

Delta-Weather parameter

In the context of the National Animal Disease Referral and Export System (NADRES), made significant strides by enhancing database capabilities. original "Static Set" covered a 10-year average of 23 weather parameters from 2011 to 2022. Now, took a bold step forward, introducing a more intricate analysis. The upgraded "Static Set" remains the foundation but with a notable addition. Alongside the initial 23 parameters, we have included another set of 23 parameters known as "Delta" variables. These represent the differences between corresponding weather parameters from 2001 to 2021. This detailed differencing process offers valuable insights into long-term trends in meteorological conditions, especially relevant to understanding animal disease dynamics. In addition, we have introduced the "Dynamic Set" bringing another layer of sophistication. This set focuses on a more recent timeframe, using 2-year averaged parameter values for 2023 and 2024. Derived from data spanning from 2018 to 2023, the Dynamic Set also includes 23 "Delta" parameters, reflecting ongoing changes in climatic trends.

This dynamic approach ensures that NADRES stays updated with the latest meteorological patterns, crucial for timely responses in the early detection of animal diseases. With a total of 46 parameters now, this upgraded database is a cornerstone for NADRES. The integration of static and dynamic perspectives solidifies our position in meteorological analysis. NADRES is a reliable tool for animal disease management and a vital resource for informed decision-making in the intricate landscape of animal disease referral and export processes.

Initially, two regression models and Seventeen machine learning models were applied to test their suitability to fit the data and in all, Fourteen models; two regression model (Generalized Linear Model (GLM), Generalized Additive Model (GAM), and twelve machine learning models, *viz.,* Gradient Boosting Machine Learning Algorithm (GBM), Random Forest (RF), Multivariate Adaptive Regression Splines (MARS), Extreme Gradient Boosting Machine (XGM), Support Vector Machine (SVM), Decision Trees (Tree_prob), Least Absolute Shrinkage and Selection Operator (Lasso), Functional Data Analysis (FDA), Gaussian Process (GP), Neural Network (NN), Multinomial Logistic Regression (Multinom_probs), and Kernel Support Vector Machine (KSVM) which fitted to data well were incorporated for the purpose of disease prediction.

The models were trained using the case and control data available at ICAR-NIVEDI. Validation of the models were done by dividing the total observations for a particular disease into marker samples and validation samples and accuracy was tested in terms of discrimination power, which was done using Receiving Operating Characteristics (ROC), Cohen Kappa (Heildke Skill Score) and True Skill statistics (TSS) Accuracy, Error Rate, Precision, Sensitivity, Specificity, F1 score, Log Loss, and Gini-Coefficient. Once the models produce the probability value, it was used for categorizing the risk. Briefly, when all the models produce the p value of more than 0.5, then the highest p value is used for determining the high-risk category. If all the models or any one model produces the p value of less than 0.5, then the lowest p value was used for categorizing lower risk. This was done to minimize the false alert. Thus, the risk predictions based on the probability values ranging from 0-1 are made as follows; Very High Risk (p=0.81-1.0), High Risk (p=0.61-0.80), Moderate Risk (p=0.41-0.60), Low Risk (p=0.21-0.40), Very Low Risk (p=0.0-0.20) and No Risk (p=0.0) for the occurrence of a said disease. It is believed that categorizing districts in to various risk categories will help the stake holders to effectively utilize the available resources (money and manpower).

II. NADRES v2 Data Flow and Data Processing Diagram

A) Data Flow Diagram:

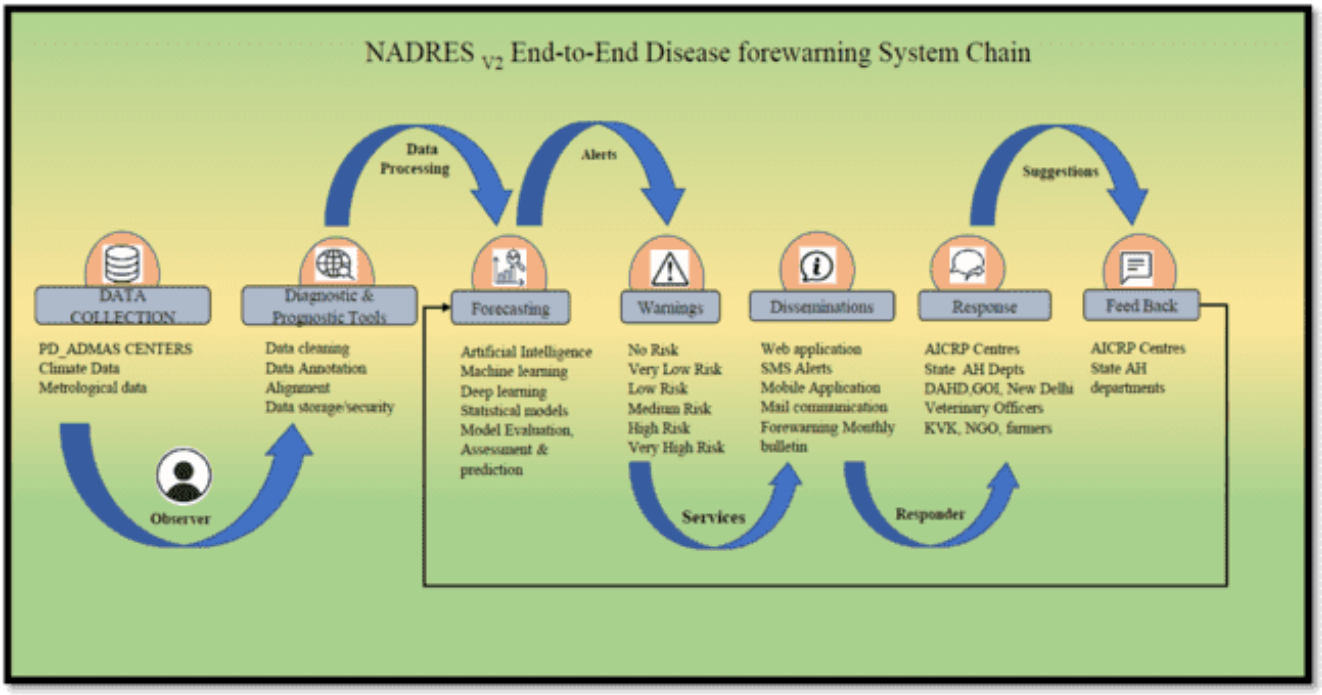


Fig 4.1. NADRES V2 Data Flow Diagram.

B) Artificial Intelligence enabled Data Capturing and Forewarning System:

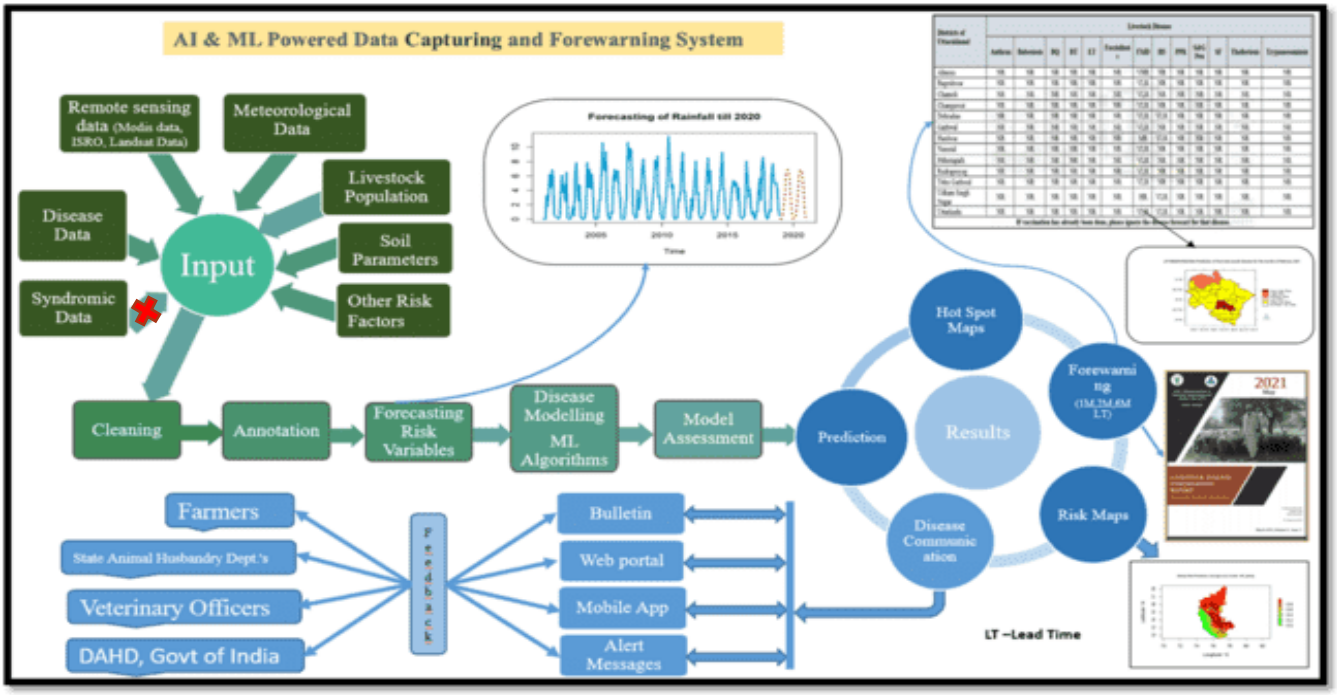


Fig 4.2. Data Capturing and Forewarning system

III. Weighted Outbreak Score

The outbreak data for the month of forecasting is extracted from NADRES database for the period of 10 years from current year. Outbreak data of 15 important livestock diseases are considered. The data is aggregated at district level and the weighted score is defined based on the number of outbreaks for each district in each month considering last 10 years. The weightage score was assigned as 0 for less than three number of outbreaks in the last 10 years for selected month, score 1 for 3–6 number of outbreaks and 2 for more than 6 outbreaks. This weightage score for each district is labelled as risk variable in building the models and risk maps.

IV. Feature Extraction and Data Engineering

Data collection from different sources could be internal and or external to satisfy the objectives of forewarning requirements, data can be of any format, CSV, XML, JSON etc. In this processing of data and feature engineering, we focus mainly on understanding the specified data set and cleaning the dataset, a better understanding of features and their relationships, extracting essential variables, handling missing values and human error, identifying outliers, transforming features if there are outliers so that either truncates a data above a threshold or transform the data using log or any other transformation, scaling the features extracted. This process would be maximising the insights into a dataset.

V. Forecasting of Weather Parameters

Weather forecasting has been one of the most challenging problems around the world because of both its practical value in meteorology and the popular sphere for scientific research. Weather forecast systems are among the most complex equation systems that computer has to solve. A great quantity of data, coming from satellites, ground stations and sensors located around our planet send daily information that must be used to foresee the weather situation in next hours and days all around. Weather forecasts provide critical information about future weather. There are various techniques involved in weather forecasting, from relatively simple observation of the sky to highly complex computerized mathematical models. Further, forecast products by Indian Metrological department were used for validation of our forecasts (https://mausam.imd.gov.in/imd_latest/contents/extendedrangeforecast.php).

Following are the basic steps of forecasting process:

1. Determine the forecast's purpose
2. Establish a time horizon
3. Select a forecasting technique
4. Gather and analyse data
5. Perform the forecast
6. Monitor the forecast and use it in prediction of disease

Statistical Models used for forecasting of weather and remotely sensed variables

ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is also known as Box-Jenkins approach. Box and Jenkins claimed that non-stationary data can be made stationary by differencing the series, Y_t. The general model for Y_t is written as,

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} \ldots \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots \theta_q \epsilon_{t-q}$$

Where, Y_t is the differenced time series value, ϕ and θ are unknown parameters and ϵ are independent identically distributed error terms with zero mean. Here, Y_t is expressed in terms of its past values and the current and past values of error terms.

The ARIMA Model combines three basic Methods:

- Auto Regression (AR) – In auto-regression the values of a given time series data are regressed on their own lagged values, which is indicated by the "p" value in the model.
- Differencing (I-for Integrated) – This involves differencing the time series data to remove the trend and convert a non-stationary time series to a stationary one. This is indicated by the "d" value in the model. If d = 1, it looks at the difference between two-time series entries, if d = 2 it looks at the differences of the differences obtained at d =1, and so forth.
- Moving Average (MA) – The moving average nature of the model is represented by the "q" value which is the number of lagged values of the error term.

This model is called Autoregressive Integrated Moving Average or ARIMA (p, d, q) of Y_t. We will follow the steps enumerated below to build our model. ARIMA models were run in 18 combinations of p, d, q. Based on the minimum AIC value, the order of ARIMA model was selected. This order was used for the prediction of all the weather parameters used in developing disease forewarning models.

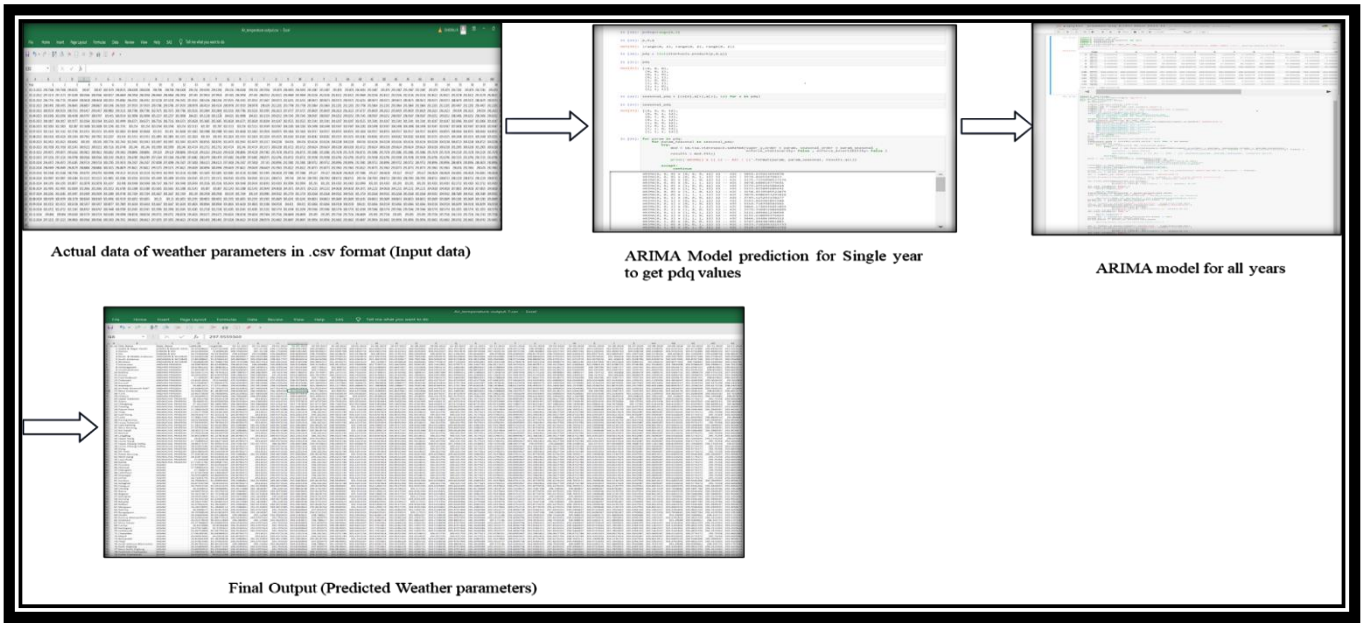


Fig. 4.3. Risk parameter Prediction using ARIMA model in Python

VI. Implementation of Principal Component Analysis

Large datasets are gradually common and are often difficult to interpret. Principal Component Analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing the interpretability but at the same time, minimizing the information loss. The PCA is employed in NADRES v2 by creating new uncorrelated variables that successively maximize the variance. This means that ` preserving as much variability as possible` translates into finding new variables that are linear functions of those in the original dataset, that successively maximize variance and that are uncorrelated with each other. Determining such new variables, the principal components (PCs) reduce to solve an eigenvalue/eigenvector problem. PCA can be based on either covariance matrix or the correlation matrix and the main use of PCA are descriptive. In the present study, all the meteorological and remote sensing variables are considering for PCA, with correlation matrix, the final output of principal components which are independent of each were considered for further ML modelling and risk estimation.

VII. Machine Learning Models

Disease outbreak data were aligned with generated risk variables to the respective latitude and longitude, which were subjected to climate-disease modelling. A number of models were fit to aligned data and tested for accuracy in terms of discrimination power. Two regression models, Generalized Linear Models (GLM) and Generalized Additive Models (GAM) and Seventeen machine learning algorithms, i.e. Random Forest (RF), Boosted Regression Tree (BRT), Artificial Neural Network (ANN), Multiple Adaptive Regression Spline (MARS), Flexible Discriminant Analysis (FDA), Classification Tree Analysis (CTA), Extreme Gradient Boosting Machine (XGM), Support Vector Machine (SVM), Decision Trees (Tree_prob), Least Absolute Shrinkage and Selection Operator (Lasso), Functional Data Analysis (FDA), Gaussian Process (GP), Neural Network (NN), Multinomial Logistic Regression (Multinom_probs), Kernel Support Vector Machine (KSVM), Ridge Regression, Naive Bayes were employed for disease modelling. Different modelling methods return different types of 'model object' and all these model objects could be used for the predict function to make predictions for any combinations of values of independent variables. Response plots were created to explore and understand model predictions.

The fitted models were assessed for their discriminating power using Receiving Operating Characteristic (ROC) curve, Cohen's Kappa (Heildke Skill Score), True Skill Statistics (TSS) Accuracy, Error_rate, Precision, Sensitivity, Specificity, F1 score, Loglass, Gini-Coefficient. These measures were used to evaluate the quality of predictions based on presence-absence data. Raster Stack was used to combine the results of individual predictions by different model methods. All the models were assessed for overfitting.

The outcome of best fitted models was in probability of disease occurrence and was categorised into 6 risk levels as No risk (NR), Very low risk (VLR), Low risk (LR), Moderate risk (MR), High risk (HR) and Very high risk (VHR) for enabling the stakeholders to take appropriate control measures by suitably allocating available resources.

5. ACCURACY OF PREDICTION

Serial No.	Diseases	Accuracy (%)
1.	**Anthrax**	**100**
2.	**Babesiosis**	**99.45**
3.	**Black quarter**	**98.49**
4.	**Bluetongue**	**98.21**
5.	**Classical swine fever**	**99.59**
6.	**Enterotoxaemia**	**98.90**
7.	**Fasciolosis**	**99.73**
8.	**Foot and mouth disease**	**99.18**
9.	**Hemorrhagic septicemia**	**96.15**
10.	**Lumpy skin disease**	**94.09**
11.	**Peste des petits ruminants**	**99.86**
12.	**Sheep & Goat pox**	**97.25**
13.	**Theileriosis**	**97.39**
14.	**Trypanosomiasis**	**98.76**

Aggregation and prediction of livestock diseases at district level leading to higher accuracy.

- **Formula Used:** The **Accuracy** of disease prediction was calculated using the following formula.

$$\frac{TP + TN}{Total} * 100$$

TP-True Positive Observations, TN-True Negative Observations, Total- Total observations.

- Internal Accuracy was performed using 10 years of data. Accuracy obtained was >90% for all the diseases predicted.
- Despite the power of climate and disease risk models, considerable uncertainties remain, identifying these uncertainties, highlighting importance of improved data may improve the model accuracy, realism, confidence, together with translating uncertainties in model inputs into uncertainties in model outputs, are important benefits of modelling.

6. MORAN'S I FOR CLUSTERING OF LIVESTOCK DISEASES

Moran's I is a tool that measures spatial autocorrelation (feature similarity) based on both feature locations and feature values simultaneously. Given a set of features and an associated attribute, it evaluates whether the pattern expressed is clustered, dispersed, or random. The tool calculates the Moran's I Index value and both a Z score and p-value evaluating the significance of that index. In general, a Moran's Index value near +1.0 indicates clustering while an index value near -1.0 indicates dispersion.

Autocorrelation tool, the null hypothesis states that there is no spatial clustering of the values associated with the geographic features in the study area". When the p-value is small and the absolute value of the Z score is large enough that it falls outside of the desired confidence level, the null hypothesis can be rejected. If the index value is greater than 0, the set of features exhibits a clustered pattern. If the value is less than 0, the set of features exhibits a dispersed pattern.

7. R SOFTWARE

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R is a simple and effective programming language, which includes conditionals, loops, user defined recursive functions and input and output facilities. R statistical software version 3.6.2 (version 3.6.2, R Foundation for Statistical Computing, Dark and Stormy Night. https://www.R-project.org/) was used as an integrated suite for data mining, calculation and graphical display. Several R packages like *openxlx, raster, RMySQL, rgdal, RColorBrewer, sqldf, sp, spdep, xlsx, plyr, randomFores, dismo, SDMTool, dplyr, tmapand data table*were used for data extraction, data alignment, annotation, analysis, modelling and risk mapping.

8. FOREWARNING OF LIVESTOCK DISEASE FOR EVERY MONTH

I. DISEASES, SPECIES AFFECTED CLINICAL SIGNS AND ITS PREVENTIVE MEASURES.

Sl No.	Disease	Species Affected	Clinical Signs	Preventive Measures
1.	African Swine Fever (ASF)	Primarily affects domestic and wild pigs	High Fever, Lethargy and Weakness, Red or Blue Skin Discoloration, Pig display respiratory signs such as coughing, difficulty breathing, and nasal discharge, causes digestive symptoms, including diarrhoea and vomiting.	involves strict measures like isolating new pigs, regularly disinfecting facilities, and controlling the movement of pigs and pig-related items. Early detection through surveillance, educating farm personnel, and managing wild boar populations are crucial, while proper disposal of infected material and quick reporting of suspected cases contribute to effective prevention and control. Collaboration among authorities, farmers, and the public, along with clear communication about ASF and its preventive measures, plays a key role in safeguarding pig populations and the swine industry.
2.	Anthrax (AX)	Most of the mammals and ruminants are highly susceptible. Pigs and Horses are moderately susceptible. Carnivores are relatively resistant.	Convulsion and sudden death with oozing of blood from natural orifices such as rectum and nose prior to death. Occasionally oedema develops in the throat and shoulder over a period of one week before death.	Ring vaccination and reporting of the disease is advised. Vaccination to be done in consultation with the veterinarians and as decided by state animal husbandry authorities. Strict biosecurity measures may be followed. Carcass may be disposed by deep burying covered with lime powder. Contaminated area may be disinfected with 4% formalin or 10% caustic soda. Grazing area may be restricted.
3.	Babesiosis (BA)	Cattle. Cross breeds are more susceptible.	High temperature, jaundice like symptoms, yellowish mucosal membrane of eye, rectum and coffee colour urine.	Periodical application of acaricides in and around the animal shed and on the animals. For therapeutic application, Diaminazine or Imidocarb can be useful.

4.	Black Quarter (BQ)	Common disease of cattle and sheep, but occasionally goats and pigs also suffer from the disease.	High fever and lameness followed by swelling in the neck, shoulder, lumbar, gluteal and sacral regions. Skin over the affected area become dark and crepitate on palpation. Loss of feed intake, colic, lateral recumbency, dyspnoea and death.	Affected animals may be treated with suitable antibiotics. Vaccination to be done in consultation with the veterinarians and as decided by state animal husbandry authorities. Strict biosecurity measures may be followed. Grazing area may be restricted. Carcass may be disposed hygienically.
5.	Bluetongue (BT)	Sheep are more susceptible than goats.	Fever, swelling of face, neck, eyelids respiratory distress, nasal discharge, Salivation, necrotic ulcers on tongue, dental pad, gum, lips hyperaemia of muzzle and may bleed at muco-cutaneous junction. Affected tongue may become swollen, cyanotic and purple blue in colour – 'bluetongue'.	Vector control using insecticides and good water management. Vaccination of susceptible animals preferably in the month of May. Do not shear sheep during winter months. Restriction in animal movement, segregation of affected animals and symptomatic treatment. Strict biosecurity measures.
6.	Enterotoxaemia (ET)	Common disease of sheep and goats especially among the young animals.	Dullness, opisthosomas, convulsions, coma and sudden death. Affected adult sheep, which survive for several days May show diarrhoea and staggering.	Affected animals may be treated with suitable antibiotics. Vaccination to be done in consultation with the veterinarians and as decided by State Animal Husbandry Authorities. Strict biosecurity measures may be followed. Carcass may be disposed hygienically. Grazing area to be restricted, stall fed, vitamins and probiotics may be provided.

7.	Fasciolosis (FA)	Cattle, buffalo, sheep and goats.	Progressive anaemia, pale mucous membrane, sub-mandibular oedema (Bottle jaw), loss of appetite, weakness, isolated from flock while grazing, loss in production.	The animal should not be allowed to graze in water stagnant fields or submerged fodder should not be given directly to the animals. The submerged fodder can be processed through hay/silage preparation in order to destroy the meta cercariae. The affected animals can be treated with Carbon tetrachloride/ Rafoxanide/Nitroxinils/ Niclofolan/Closantel/Oxyclozanide, under the strict supervision of veterinarian.
8.	Foot and Mouth Disease (FMD)	Cattle, buffalo, sheep, goats and pigs are often affected domesticated species, but the disease is more severe in cattle and pigs.	Fever, loss of feed intake, drop in milk production, drooling of saliva like ropey string, vesicles develop on the tongue, lips, gums, and palate and eventually rupture. Concurrent to oral lesions, vesicles also appear in inter digital skin and coronary band of the feet. The animal may open and close its mouth with a characteristic smacking sound. Sheep and goats may show lameness. In pigs, lesions may be seen on snout and also on the feet.	Regular vaccination and seromonitoring. Disinfection with sodium carbonate (4%) or 10% washing soda and strict biosecurity measures to be followed and animal movement may be controlled.

9.	Haemorrhagic septicaemia (HS)	Common disease for cattle and buffaloes, but can also occur among other species such as pigs, sheep, goats and many wild animals.	The disease starts with high fever, respiratory distress and haemorrhages maybe seen on the mucous membranes. There is lachrymation, nasal discharge, drop in milk production and anorexia. As the disease progress ear droops and the animals will be prostrated with cyanosis of mucous membranes. There may be oedema along the head, neck, thorax, vulva and anal areas. Sudden death occurs within few hours of clinical signs.	Affected animals may be treated with suitable antibiotics. Vaccination to be done in consultation with the veterinarians and as decided by state animal husbandry authorities. Strict biosecurity measures may be followed. Carcass may be disposed hygienically and stress factors may be reduced by following good animal husbandry practices.
10.	Lumpy skin disease (LSD)	Common disease for Cattle, Buffalo and other domestic animals	Fever, reduced milk production and skin nodules. Mastitis, swelling of peripheral lymph nodes, loss of appetite, increased nasal discharge and watery eyes are also common. Temporary or permanent infertility occur among infected cows and bulls	Vaccination of susceptible animals of above 3 months old age. Restriction on animal movement, strict biosecurity measures and proper disposal of carcass.
11.	Peste des Petits Ruminants (PPR)	Goats and sheep are most affected domestic animals.	Fever, nasal and ocular discharge, respiratory distress, necrotic lesions in buccal mucosa, gum, dental pad, palate, tongue and diarrhoea. Animals may die because of dehydration and pneumonia.	Vaccination of susceptible animals of above 3 months old age. Restriction on animal movement, strict biosecurity measures and proper disposal of carcass.

12.	Sheep and Goat pox (SGP)	Sheep and Goats	Respiratory distress and pock lesions over the non-hairy parts of body, more common in teat, udder, scortum, head, neck, ear, perineum, inner aspect of thighs and under tail.	Vaccination of susceptible animals of above 3 months old age. Symptomatic treatment of affected animals. Restriction on animal movement, strict biosecurity measures and proper disposal of carcass.
13.	Classical Swine Fever (CSF)	Pigs	Fever, Conjunctivitis, purplish discolouration of snout, ears, abdomen, inner side of the legs and staggering gait.	Vaccination of susceptible animals. Restriction on animal movement, strict biosecurity measures and proper disposal of carcass.
14.	Theileriosis (TE)	Large Ruminants. Crossbreed cattle are more vulnerable.	High temperature, yellowish eye, sometime eyes may be heavily swollen, icteric mucosal membrane of rectum, dark yellowish urine, sometime may reach to coffee colour. Antibiotic is of no use to check the fever.	Periodical application of acaricides in and around the animal shed and on the animals. Therapeutic treatment with Buparvaquone can be useful in both early and advanced stages of the infection.
15.	Trypanosomiasis (TR)	Domestic and wild carnivores and herbivores including cattle, buffalo, horse, donkey, camel, dog and cats. Buffaloes are known as carriers.	Fluctuating high fever which is not responded by antibiotics, swollen lymph gland, chronic emaciation and weakness, loss of appetite, gradual loss of production.	The affected animal should be treated with Diaminazine compounds or chloride and sulphate salts of Quinapyramine. Periodical spray of insecticide in and around animal shed to remove the flies.

II. SIGNIFICANT WEATHER PARAMETERS FOR LIVESTOCK DISEASE TABLE USING DISCRIMINANT FUNCTION ANALYSIS

SI. No	Disease Names	ML Derived Significant Parameters
1	**African Swine Fever**	Air Temperature, Delta-Precipitation, Delta-Leaf Area Index
2	**Anthrax**	WET, Maximum Temperature, Wind Velocity, Soil Temperature
3	**Babesiosis**	Delta-Relative Humidity, Precipitable-water, Delta-Minimum Temperature
4	**Black quarter**	Wind Velocity, Potential Evapotranspiration, Air Temperature
5	**Bluetongue**	Wind Velocity, Soil Temperature
6	**Classical Swine fever**	Air Temperature, Enhanced Vegetation Index, Relative Humidity
7	**Enterotoxaemia**	Soil Temperature, Precipitable water, Delta-Precipitable water
8	**Fascioliasis**	Precipitable Water, Air Temperature, Precipitation-rate
9	**Foot and mouth disease**	Delta-Temperature Minimum, Precipitation-rate, Precipitable water, Soil Temperature
10	**Hemorrhagic septicemia**	Precipitable Water, Delta-Pressure
11	**Lumpy Skin Diseases**	Delta-Vapour Pressure, Minimum Temperature, WET, Delta- Enhanced Vegetation Index, Pressure
12	**Peste des petits ruminants**	Delta-Minimum Temperature, Precipitable water, Delta-Relative Humidity
13	**Sheep & Goat pox**	WET, Enhanced Vegetation Index, Relative Humidity
14	**Theileriosis**	Precipitable Water, Delta- Minimum Temperature, Precipitation-rate
15	**Trypanosomiasis**	WET, Precipitable-water, Delta-soil moisture

Table: Significant weather parameters govern the Livestock disease incidence (forecast)

9. DATA STORAGE, SECURITY AND VISUALIZATION

I. DATA STORAGE AND SECURITY

Before proceeding to the modeling phase, livestock data must be meticulously cleaned and stored in a database to ensure data integrity and reliability. This critical step involves a systematic approach to handling and preparing the data for subsequent analytical tasks. Cleaning the data encompasses various processes, including the removal of duplicates, handling missing values, and correcting inconsistencies or errors. These steps are essential to eliminate any biases or inaccuracies that could affect the outcomes of the modeling process.

Once the data is cleaned, it is essential to store it in a robust and efficient database management system such as MySQL. MySQL offers a reliable platform for data storage with powerful features for data manipulation, querying, and maintenance. Using MySQL queries, the cleaned data can be efficiently inserted into appropriate database tables, structured in a way that facilitates easy retrieval and analysis. The process involves defining suitable schemas that reflect the data's structure and relationships, ensuring optimal storage and access patterns. By leveraging MySQL's capabilities, researchers can ensure that the livestock data is not only securely stored but also readily available for advanced modeling techniques, ultimately contributing to more accurate and insightful analytical outcomes .**CSV File:** Data is initially stored or received in CSV format.

- **MySQL Database:** The data from the CSV file is imported into a MySQL database for storage and management.
- **TLS/SSL Encryption:** The connection between the application and the server is secured using TLS/SSL encryption, ensuring data privacy and integrity during transmission.
- **Server:** The server hosts the MySQL database and serves as the endpoint for data retrieval and storage
- **Data Encryption:** Protects data both in transit and at rest using tools like OpenSSL for TLS/SSL encryption and MySQL Enterprise Encryption for data-at-rest encryption.
- **Access Control:** Ensures only authorized users access data, managed through MySQL's user management and enhanced with MySQL Enterprise Edition features.
- **Auditing and Logging:** Tracks user activities for security monitoring, facilitated by MySQL Enterprise Audit plugin and Audit beat.
- **Data Masking:** Obscures sensitive information within the database, leveraging MySQL's built-in functions and enhanced with tools like DataMasker.
- **Data Integrity Checks:** Verifies data integrity using checksum functions in MySQL and automated tools like Checksum for MySQL.
- **Patch Management:** Keeps the database updated with security patches obtained from MySQL's official website and managed using tools like Patch Manager Plus.

- **Incident Response:** Preparedness for security incidents with documented playbooks and aided by log analysis tools such as Splunk for real-time monitoring and response.
- **Real-time Data Files Capture for Admins:** Database Integration via PHP: As administrators, we've integrated PHP to seamlessly interact with our database system. This enables us to efficiently retrieve essential file data, including names, upload dates, types, and unique identifiers.
- **Dynamic Table Population:** Through PHP scripting, our system dynamically populates an organized table structure within the HTML framework. This ensures that every file entry is accurately represented, providing admins with a comprehensive overview of available data files.
- **JavaScript Search Enhancement:** Our implementation includes real-time search functionality powered by JavaScript, specifically tailored for administrative use. Admins can efficiently locate specific files by entering relevant keywords, streamlining the data retrieval process and enhancing overall system usability.

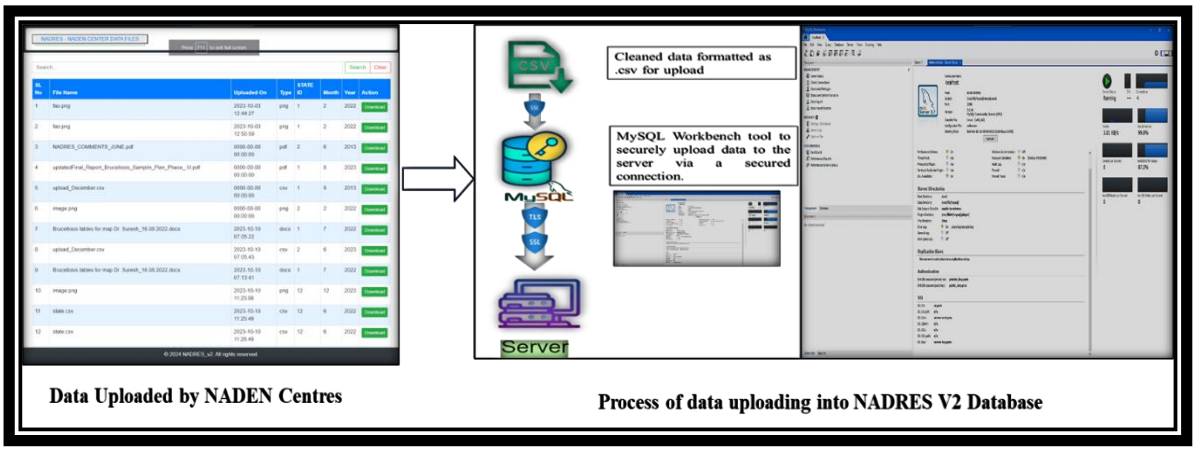


Fig. 9.1. Data Uploading to Database

II. DATA VISUALIZATION (INTERACTIVE LINE GRAPH)

Data visualization, particularly through interactive line graphs, plays a pivotal role in elucidating trends and patterns within complex datasets. These graphs allow for dynamic exploration, enabling users to zoom in on specific time periods, hover over data points for detailed information, and toggle various data series on and off. Such interactivity enhances the user's ability to interpret and analyse data, making it an invaluable tool in both exploratory data analysis and the communication of results. By transforming raw data into visual insights, interactive line graphs facilitate a deeper understanding of temporal changes and relationships within the dataset.

Frontend:

- HTML: Markup language for structuring the document.
- JavaScript: Scripting language for dynamic behavior and interaction.
- Chart.js: JavaScript library for creating interactive charts.

Backend:

- PHP: Server-side scripting language for generating dynamic content.
- MySQL: Relational database management system for storing and managing data.
- MySQLi (MySQL Improved): PHP extension for interacting with MySQL databases using improved methods.

Tools and Libraries:

- Fetch API: Web API used in JavaScript for making asynchronous HTTPS requests to the server.
- Chart.js: JavaScript library used for creating various types of charts, including line charts in this case.

Development Steps:

- Utilize the fetch API to retrieve data from the server.
- Extract pertinent information from the fetched data through data processing.
- Generate canvas elements and contexts for each chart dynamically.
- Employ Chart.js to instantiate charts programmatically.
- Apply Chart.js configurations for animation and styling effects.
- Append canvas and chart information to the chart container.
- Update the index to consecutively render the next chart in a loop.
- Enhance data processing speed through efficient backend API utilization and utilization of more technical development language.

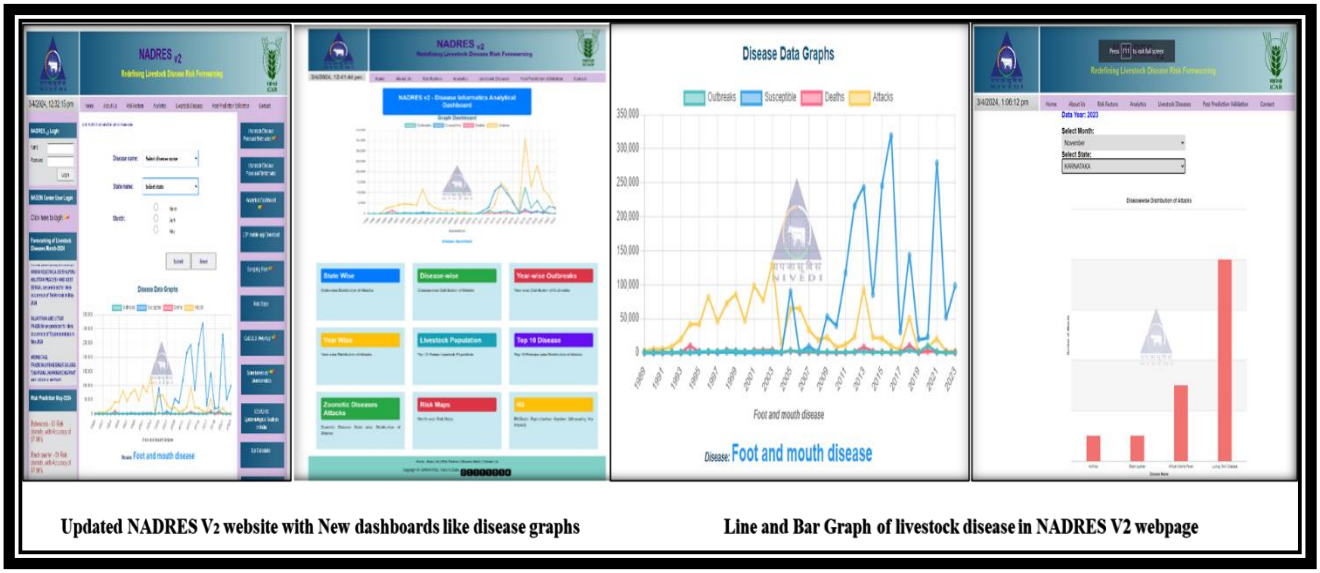


Fig. 9.2. Updated NADRES V2 website with interactive graphs

10. R PROGRAMMING SCRIPTS ALIGNED WITH THE METHODOLOGY

Step 1: Convert extracted data to CSV format (Weather parameter)

Information taken from several websites (GLDAS, MODIS, CRU, NCEP) must be converted from a specific format into CSV format. The extracted risk variable data from GLDAS, NOAA and MODIS will be in HDF format, which needs to be converted to GeoTIFF format, then convert from TIFF format to CSV for analysis. There are five remote sensing variables (LST, NDVI, EVI, PET, LAI), and those will be in hdf format. Separate codes for each variable are available, and provided below.

Procedure of extracting risk parameters from GLDAS, NOAA and MODIS

Extracting risk parameters data from GLDAS, NOAA, and MODIS involves a systematic and scientifically rigorous approach. For GLDAS (Global Land Data Assimilation System), the process begins by identifying relevant datasets that correspond to the risk parameters needed, such as soil moisture or surface temperature. These datasets can be found on the NASA Goddard Earth Sciences Data and Information Services Center (GES DISC) website. Once the appropriate datasets are identified, users must define the temporal and spatial extents for their analysis. Data can be downloaded in formats such as NetCDF or HDF using tools like Giovanni or direct FTP/HTTP access. Preprocessing steps include converting data formats if necessary, regridding spatially and temporally, and handling missing values. The data is then ready for analysis using statistical or modelling tools, with visualization performed through software like R libraries.

For NOAA data, the process similarly begins with identifying the necessary datasets related to risk parameters such as precipitation or drought indices. These datasets can be accessed through NOAA's Climate Data Online (CDO) or the Operational Model Archive and Distribution System (NOMADS). Users specify the temporal and spatial resolution needed and download the data in suitable formats like NetCDF or CSV. Preprocessing involves converting and formatting the data, cleaning it by handling outliers and missing values, and aggregating or disaggregating as required. Analysis is conducted using statistical methods or machine learning algorithms, with visualizations created using tools like R or GIS software.

Extracting data from MODIS (Moderate Resolution Imaging Spectroradiometer) involves identifying datasets relevant to risk parameters such as land surface temperature or vegetation indices. These datasets can be accessed through NASA’s Earth data portal or the LP DAAC (Land Processes Distributed Active Archive Center). Users define the temporal and spatial scope, ensuring the temporal resolution matches their analysis needs. Data is downloaded in formats like HDF or GeoTIFF using tools provided by Earth data or LP DAAC’s Data Pool. Preprocessing includes converting HDF files using tools like GDAL, reprojecting data to a common spatial reference system, and applying quality filters. Analysis is then conducted using spatial analysis techniques, with remote sensing software like ENVI or open-source tools such as QGIS and R libraries employed for data visualization.

Each step in these processes requires meticulous attention to detail and methodological rigor to ensure accurate extraction and analysis of risk parameters from GLDAS, NOAA, and MODIS datasets. Step by step processes showed in figures.

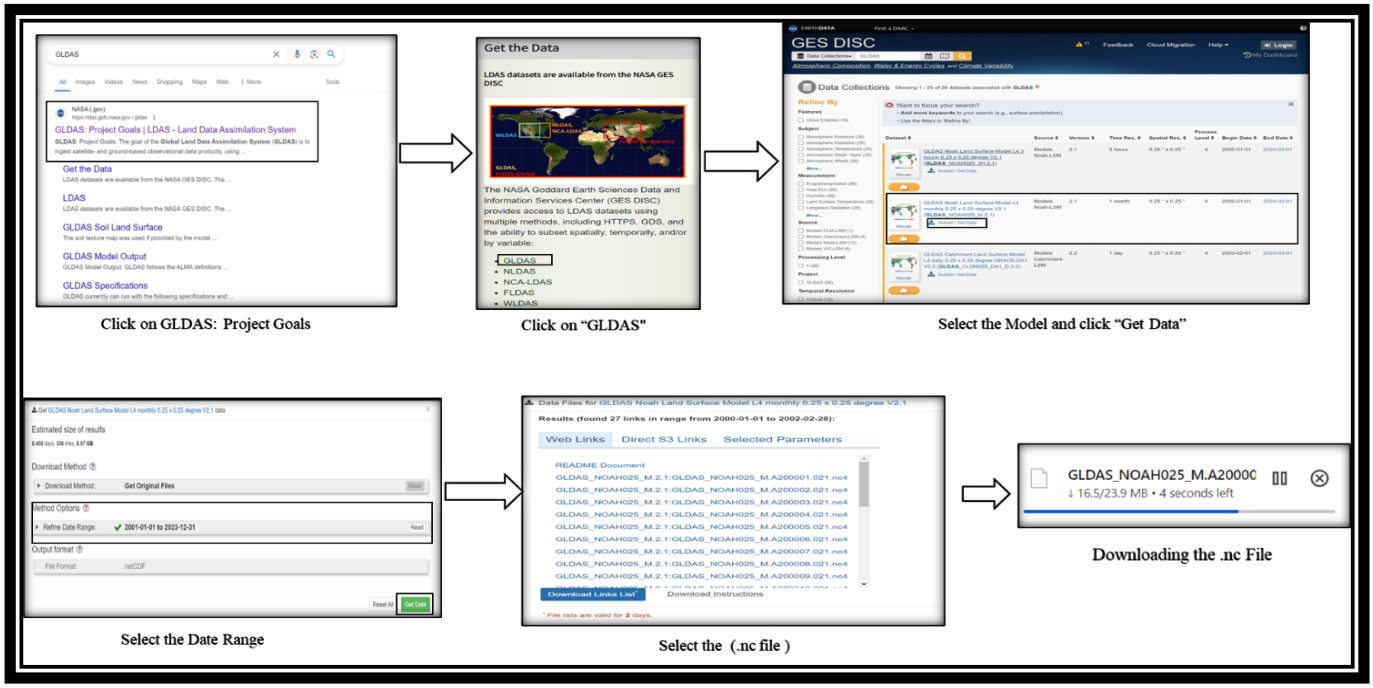


Fig. 10.1. Data extraction procedure from GLDAS

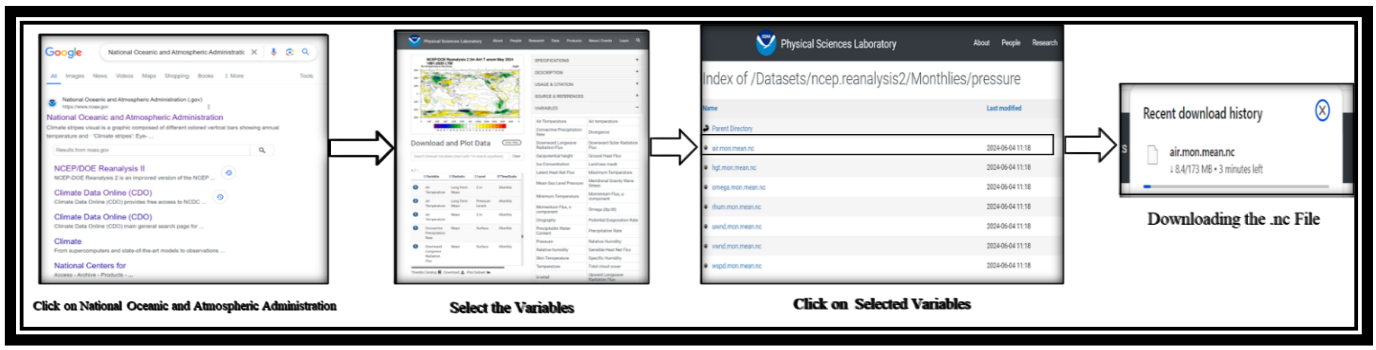


Fig. 10.2. Data extraction procedure from NOAA

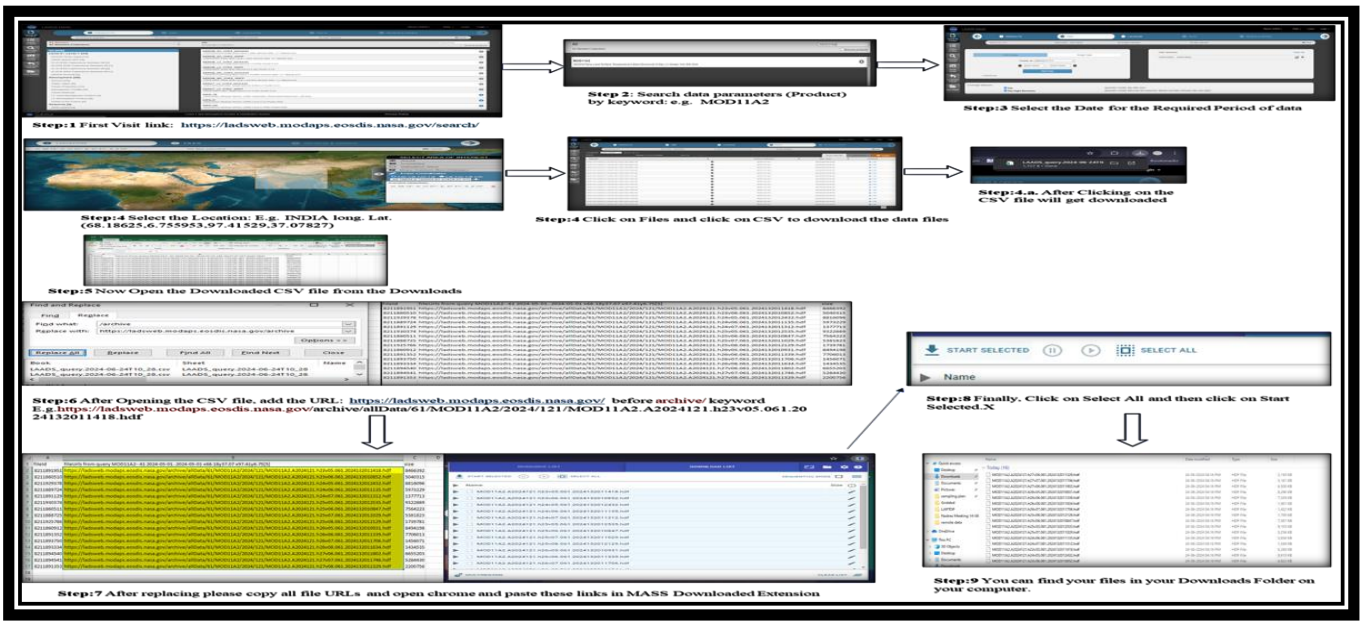


Fig. 10.3. Data extraction procedure from MODIS

I. Code for Converting HDF Files to TIFF Format for Land Surface Temperature (LST) Data

Load required libraries
```
library(gdalUtils)    # For converting HDF to TIFF
library(MODIS)        # For extracting dates from filenames
library(qdap)         # For string manipulation
```

The gdalUtils library facilitates the conversion of HDF files to TIFF format. The MODIS library is used to extract date information from filenames. The qdap library assists with string manipulation tasks.

List all HDF files in the "LST" directory
```
files <- list.files(path = "LST/", pattern = ".hdf", full.names = TRUE)
```

The list.files function retrieves a list of all HDF files within the specified directory, including the full file paths.

Count the total number of files
```
j <- length(files)
```

The total number of HDF files is determined by calculating the length of the list of file paths. This value is used to control the iteration in the subsequent processing loop.

Extract date values from filenames for output naming
```
date <- extractDate(files, asDate = TRUE)
```

The extractDate function extracts date information from the filenames. This information is used to create meaningful output filenames for the TIFF files.

Generate output filenames for the TIFF files
```
filename <- paste0("LST/", substr(files, 23, 28), date$inputLayerDates,
".tif")
```

Output filenames for the TIFF files are generated by combining parts of the original filename with the extracted date values.

Initialize loop index
```
i <- 1
```

The loop index i is initialized to start the processing from the first file.

Loop through each HDF file to convert it to TIFF
```
while (i <= j) {
 # Retrieve subdatasets within the HDF file
  sds <- get_subdatasets(files[i])
```

```
 # Convert the first subdataset (LST) to TIFF format
 gdal_translate(sds[1], dst_dataset = paste0("processed_", filename[i]))

 # Move to the next file
  i <- i + 1
}
```

Each HDF file is processed in a loop. Within the loop:
get_subdatasets(files[i]) retrieves subdatasets from the current HDF file.
gdal_translate(sds[1], dst_dataset = filename[i]) converts the first subdataset (assumed to be the Land Surface Temperature data) to TIFF format and saves it using the generated filename.
The loop index i is incremented to proceed to the next file.

II. Code for Converting HDF Files to TIFF Format for Normalized Difference Vegetation Index (NDVI) Data

```
files <- list.files(path="NDVI/",pattern = ".hdf",full.names = T)
```

The list.files function retrieves all HDF files within the "NDVI" directory, including their full file paths.

```
j<-length(files)
```

The total number of HDF files is determined by calculating the length of the list of file paths. This count is used for controlling the iteration in the processing loop.

```
date=extractDate(files,asDate = T)
```

The extractDate function extracts date information from the filenames, which is necessary for creating meaningful output filenames for the TIFF files.

```
filename <- paste0("NDVI/",
substr(files,24,29),date$inputLayerDates,".tif")
```

Output filenames for the TIFF files are created by combining parts of the original filename with the extracted date values.

```
i <-1
```
The loop index i is initialized to start processing from the first file.

```
while(i<=j){ sds <- get_subdatasets(files[i]);
# sds[1] NDVI
gdal_translate(sds[1], dst_dataset = filename[i]);
i<-i+1;
}
```

In this loop:

get_subdatasets(files[i]) retrieves the subdatasets from the current HDF file.

gdal_translate(sds[1], dst_dataset = filename[i]) converts the first subdataset (assumed to be NDVI data) to TIFF format and saves it using the generated filename.

The loop index i is incremented to process the next file.

III. Code for Converting HDF Files to TIFF Format for Enhanced Vegetation Index (EVI) Data

```
files <- list.files(path="NDVI/",pattern = ".hdf",full.names = T)
```

The list.files function retrieves all HDF files located in the "NDVI" directory, including their full file paths.

```
j<-length(files)
```

The total number of HDF files is determined by calculating the length of the file path list. This value is used to control the iteration in the processing loop.

```
date=extractDate(files,asDate = T)
```

The extractDate function extracts date information from the filenames, which is used to generate appropriate output filenames for the TIFF files.

```
filename <- paste0("EVI/",
substr(files,24,29),date$inputLayerDates,".tif")
```

Output filenames for the TIFF files are created by combining parts of the original filename with the extracted date values. Note the output directory is set to "EVI" instead of "NDVI."

```
i <-1
```

The loop index i is initialized to start processing from the first file.

```
while(i<=j){
sds <- get_subdatasets(files[i]);
#View(sds)
# sds[2] EVI
gdal_translate(sds[1], dst_dataset = filename[i]);
i<-i+1;
}
```

In this loop:
get_subdatasets(files[i]) retrieves subdatasets from the current HDF file.
gdal_translate(sds[2], dst_dataset = filename[i]) converts the second subdataset (assumed to be EVI data) to TIFF format and saves it using the generated filename.
The loop index i is incremented to process the next file.

IV. Code for Converting HDF Files to TIFF Format for Potential Evapotranspiration (PET) Data

```
files <- list.files(path="PET/",pattern = ".hdf",full.names = T)
```

The list.files function retrieves all HDF files located in the "PET" directory, including their full file paths.

```
j<-length(files)
```

The total number of HDF files is determined by calculating the length of the list of file paths. This value is used to control the iteration in the processing loop.

```
date=extractDate(files,asDate = T)
```

The extractDate function extracts date information from the filenames, which is necessary for generating meaningful output filenames for the TIFF files.

```
filename <- paste0("PET/",
substr(files,23,28),date$inputLayerDates,".tif")
```

Output filenames for the TIFF files are created by combining parts of the original filename with the extracted date values. The output directory is set to "PET."

```
i <-1
```
The loop index i is initialized to start processing from the first file.

```
while(i<=j){
sds <- get_subdatasets(files[i]);
#sds[3] PET
gdal_translate(sds[3], dst_dataset = filename[i]);
i<-i+1;
}
```

In this loop:

- get_subdatasets(files[i]) retrieves subdatasets from the current HDF file.
- gdal_translate(sds[3], dst_dataset = filename[i]) converts the third subdataset (assumed to be PET data) to TIFF format and saves it using the generated filename.
- The loop index i is incremented to proceed to the next file.

V. Code for Converting HDF Files to TIFF Format for Leaf Area Index (LAI) Data

```
files <- list.files(path="LAI/",pattern = ".hdf",full.names = T)
```

The list.files function retrieves all HDF files located in the "LAI" directory, including their full file paths.

```
j<-length(files)
```

The total number of HDF files is determined by calculating the length of the list of file paths. This value is used to control the iteration in the processing loop.

```
date=extractDate(files,asDate = T,pos1 = 11,pos2 = 18)
```

The extractDate function extracts date information from the filenames. The pos1 and pos2 parameters specify the positions in the filename from which to extract the date. This information is used to generate output filenames for the TIFF files.

```
filename <- paste0("LAI/",
substr(files,24,29),date$inputLayerDates,".tif")
```

Output filenames for the TIFF files are created by combining parts of the original filename with the extracted date values. The output directory is set to "LAI."

```
i <-1
```
The loop index i is initialized to start processing from the first file.

```
while(i<=j){
 sds <- get_subdatasets(files[i]);
 # sds[2] LAI
 gdal_translate(sds[2], dst_dataset = filename[i]);
 i<-i+1;
}
```

In this loop:

- get_subdatasets(files[i]) retrieves subdatasets from the current HDF file.
- gdal_translate(sds[2], dst_dataset = filename[i]) converts the second subdataset (assumed to be LAI data) to TIFF format and saves it using the generated filename.
- The loop index i is incremented to proceed to the next file.

VI. Code for Converting TIFF Files into CSV Format

```
library(raster)
library(data.table)
library(qdap)
```

The raster, data.table, and qdap libraries are loaded to handle raster data, process data tables, and perform string operations, respectively.

```
files_latlong=c('./India_districts_latlong1.csv')
filename1 <- list.files(path="./LAI/",pattern = ".tif",full.names = T)
v = paste(2023:2024, collapse = "|")
filename2 = grep(v, filename1, value = T)
```

- files_latlong specifies the path to the CSV file containing latitude and longitude data.
- filename1 retrieves all TIFF files from the "LAI" directory.
- filename2 filters these files to include only those from the years 2023 and 2024.

```
j=1

for(j in 1:length(files_latlong)){
  df_total = data.frame()
  merge_data = data.frame()
  ss<-fread(files_latlong[j],header=T,check.names=F,data.table = F)
  x<-ss$lat
  y<-ss$long
  data<-data.frame(y,x)
  latlon1<-CRS('+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0')
  coordinates1 = SpatialPoints(data,latlon1)
  sinus1 = CRS("+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +a=6371007.181
+b=6371007.181 +units=m +no_defs")
  coordinates_sinus1 = spTransform(coordinates1,sinus1)
  df_total<-NULL
```

- df_total and merge_data are initialized as empty data frames for storing results.
- Latitude and longitude data are read using fread and converted into a spatial object.
- Coordinate reference systems are defined and used to transform the spatial coordinates.

```
  for(i in 1:length(filename2))
  {
    my<-raster(filename2[i])
    my<-stack(my)
    dd <- extract(my, coordinates1)
    tempf = which(is.na(dd))
```

```
  if(length(tempf) != nrow(dd))
  {
    df_total <- rbind(df_total, dd)
  }
}
df_total = as.matrix(df_total)
df_total=df_total*0.1
df_total =as.data.frame(df_total)
col_names= gsub("\\.","-",colnames(df_total))
df_total=setNames(df_total,substr(col_names,8,17))
merge_data<-cbind(ss,df_total)
f_name=paste(beg2char(files_latlong[j],".",2),"_LAI_2023.csv",sep="")
fwrite(merge_data,f_name,col.names = T,row.names = F,sep=",")}
```

- Each TIFF file is processed to extract raster values at the specified spatial coordinates.
- Missing values are handled, and the extracted data is normalized and converted to a data frame.
- Column names are adjusted, and the results are merged with the original latitude and longitude data.
- The merged data is saved to a CSV file with a name based on the original file.

VII. Code for Converting Extracted Data from CRU (NetCDF Format) to CSV Format

Start by going to the Climatic Research Unit's (CRU) data portal using a web browser. Look for the section labelled "Climate Data." Once there, pick the desired dataset, such as temperature, precipitation, or humidity. Each dataset comes with a description that explains its coverage, resolution, and variables. Read the documentation to understand the data format, variable names, and any preprocessing that was done. Follow the steps shown in the accompanying image for a visual guide. Go to the download page and select the NetCDF file format, which works well with many analysis tools. Choose the time period and area needed, then download the .nc file, ensuring there is enough storage space and a good internet connection. After downloading, check the file size to ensure its correct, and unpack it if compressed using tools like tar or unzip. Open the NetCDF file in preferred software, such as R. Examine the data structure, including dimensions, variables, and attributes. Perform any necessary preprocessing, like masking, interpolation, or aggregation, to fit the data to specific research needs. By following these steps, it is easy to download and use climate data from the CRU website for analysis, aiding in a better understanding of climate patterns.

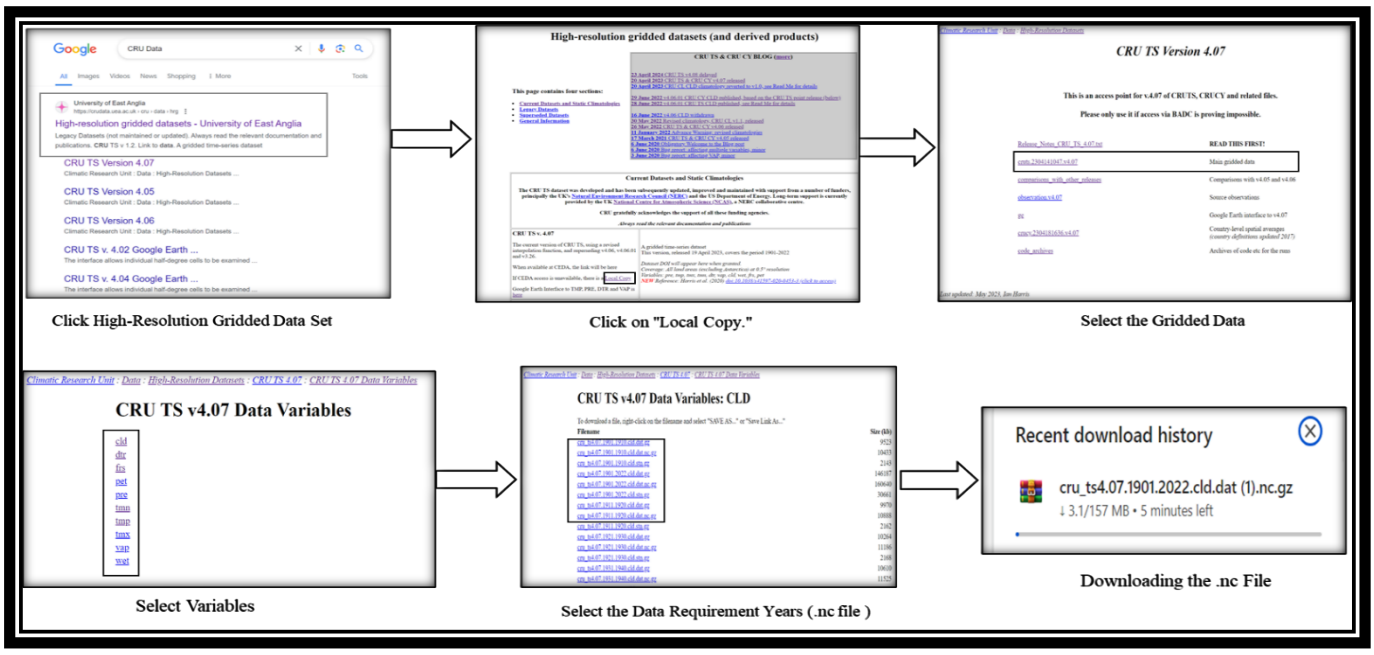


Fig. 10.4. Data extraction processes from CRU (NetCDF format data)

After obtaining the NetCDF files, use the following R code to convert them to csv format for easier data handling and analysis.

```
#Load the necessary package
library(ncdf4)
library(raster)
library(data.table)
library(qdap)
#install.packages("qdap")
library(sqldf)
```

```
library(reshape)
```

These libraries are loaded to manage NetCDF files, manipulate raster data, handle data tables, perform string operations, and reshape data.

```
# Define file paths for NetCDF files
files_nc <- list.files( path = "C:\\Users\\ICAR
 Bangalore\\Desktop\\Aj\\Weather Data\\NOAA (Ncep-Ncar)\\Precipitable
 water\\", pattern = "*.nc$", full.names = TRUE, recursive = TRUE )
# Define file paths for CSV files containing latitude and longitude

files <- "India_Dist_2021_new.csv"
# Define parameters for the data

pars <- c("Precipitable water")
```

- files_nc lists all NetCDF files in the specified directory.
- files specifies the path to the CSV file containing latitude and longitude data.
- pars defines the parameters being processed, in this case, "Precipitable water".

```
m <- 1
k <- 1

for (m in 1:length(files_nc)) {
for (k in 1:length(files)) {

# Read latitude and longitude data from CSV

ss1 <- read.csv(files[k], sep = ",", header = TRUE)
    lg_lt1 <- cbind(ss1$long, ss1$lat)
    latlon1 <- CRS('+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0')
    coordinates1 <- SpatialPoints(lg_lt1, latlon1)
    sinus1 <- CRS("+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +a=6371007.181
+b=6371007.181 +units=m +no_defs")
    coordinates_sinus1 <- spTransform(coordinates1, sinus1)

  # Load NetCDF file and get number of bands
    banddata <- raster(files_nc[m+1])
    z <- nbands(banddata)

  # Initialize data frame to store results
  df_total <- data.frame(c(1:(nrow(ss1) + 1)), stringsAsFactors = FALSE)
```

```
    for (i in 1:z) {
      # Extract raster data for each band
      mydata <- raster(files_nc[m], i)
      crs(mydata) <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0"
      ff <- extract(mydata, coordinates_sinus1)

   # Extract and format date information
   dd <- substr(getZ(mydata), 1, 7)
   ff <- c(as.character(dd), ff)
   df_total <- cbind(df_total, ff)
   gc()  # Clean up memory
    }

  # Prepare final data frame for export
  df_total <- df_total[, 2:ncol(df_total)]
  colnames(df_total) <- as.character(unlist(df_total[1, ]))
  final_df <- cbind(ss1, df_total[2:nrow(df_total), ])

  # Generate filename for output

filename <- paste0(beg2char(files[k], "."), "_", pars[m], "Precipitable
water_india 1979-2024.csv")
print(filename)

# Write final data to CSV
fwrite(final_df, filename, row.names = FALSE, col.names = TRUE, sep =
",")
  }
}
```

- Latitude and longitude data are read from the CSV and converted to spatial points.
- Each NetCDF file is processed to extract raster data for all bands.
- The extracted data is combined with the latitude and longitude data into a final data frame.
- The final data frame is saved to a CSV file, with a name based on the parameter and input file.

Step 2: Dimensionality Reduction of Risk Variables Using Principal Component Analysis (PCA)

Apply Principal Component Analysis (PCA) to reduce dimensionality, retaining essential weather patterns and relationships. This process helps identify impactful weather variables efficiently for risk assessment.

Load Required Libraries
```
library(stats)  # Provides functions for statistical calculations
including PCA
library(dplyr)  # Used for data manipulation
```

These libraries are loaded to utilize statistical functions and data manipulation capabilities in R.

Read the NADRES parameters file
```
mydata <- read.csv("Inputfile.csv")
```

Extract column names of specific components (columns 18 to end)
```
c_names <- colnames(mydata[18:ncol(mydata)])
```

Extract specific components for PCA analysis
```
mydata_pca <- mydata[, c(18:ncol(mydata))]
```

Extract non-PCA components (columns 1 to 17)
```
mydata2 <- mydata[, c(1:17)]
```

- mydata contains the dataset with NADRES parameters.
- c_names captures the names of the PCA components.
- mydata_pca isolates the columns used for PCA.
- mydata2 contains other relevant variables excluding the PCA components.

Convert Soil moisture values to millimeters by multiplying by 86400
```
mydata_pca$Soil_moisture <- mydata_pca$Soil_moisture * 86400
```

Define a function to replace zero values with a small non-zero value (0.0001)
```
replace_zeros <- function(x) {
  ifelse(x == 0, 0.0001, x)
}
```

Apply the function to each column of the dataset
```
mydata_pca <- lapply(mydata_pca, replace_zeros)
```

Convert the list back to a data frame
```
mydata_pca <- as.data.frame(mydata_pca)
```

- Soil moisture values are scaled to millimeters.
- replace_zeros function ensures no zero values are present in the dataset, substituting them with a minimal value (0.0001).
- The function is applied to all columns of mydata_pca, and the result is converted back to a data frame.

Perform PCA on the dataset
```
pca_mydata <- princomp(mydata_pca)
```

Display number of observations
```
pca_mydata$n.obs
```

Display PCA scores
```
pca_mydata$scores
```

- princomp function performs PCA on the mydata_pca dataset.
- The number of observations and PCA scores are accessed for analysis.

Combine PCA scores with the other variables
```
pca_results <- cbind(mydata2, pca_mydata$scores)
```

Rename columns of PCA components
```
colnames(pca_results)[18:ncol(pca_results)] <- c_names
```

- PCA scores are combined with non-PCA variables from mydata2.
- Column names for the PCA components are updated to match c_names.

Write the PCA results to a new CSV file
```
write.csv(pca_results, "PCAappliedfile.csv")
```

The combined dataset with PCA results is saved to a CSV file for further use.

Step-3: Modelling Approaches (Risk Prediction)

Part 1: Analysis, forecasted results and disease maps at the state level

In this step, Data on risk factors and disease incidences were systematically integrated from a comprehensive database. After merging these datasets, 16 distinct machine learning algorithms were applied for predictive modelling. The accuracy of these models was rigorously evaluated using ten different validation techniques, ensuring robust assessment. The results were meticulously documented in an Excel spreadsheet, categorized by state for clarity and ease of interpretation. Furthermore, geospatial maps were created to visualize the predictive outcomes across states, enhancing data interpretability and enabling more precise state-level disease risk predictions. This methodological approach significantly improves the accuracy and reliability of the predictions.

```
#"C:\Program Files\R\R-3.3.3\bin\Rscript" nadres_final_mod.R 2018 2008 7
```

This line indicates how to run the R script nadres_final_mod.R from the command line, with arguments specifying the forecast month and range of years.

```
month_number= forecasting for which month(number example: 01 or 02 or 04
or 06) ; year_number=2015 (from which year considering data );
current_year=2024 (upto which year you are going to forecast);
```

First, you must specify which month you are forecasting using how many years' worth of data (year_number to Current_year).

```
nadres_func=function(current_year,year_number,month_number)
```

Defines a function nadres_func that takes three parameters: current_year, year_number, and month_number. This function handles data processing and forecasting tasks.

```
print(current_year)
print(year_number)
print(month_number)

library(RMySQL)
library(rgdal)
library(RColorBrewer)
library(sqldf)
library(data.table)
library(reshape2)
library(imputeMissings)
require(sp)
require(spdep)
```

```
require(rms)
library(xlsx)
library(plyr)
library(dplyr)
library(randomForest)
library(dismo)
library(psych)
library(pROC)
library(SDMTools)
library(BIOMOD)
library(ROCR)
library(caret)
library(MLmetrics)
```

Loads various libraries required for data manipulation, spatial analysis, machine learning, and performance metrics.

##Database Connection and Data Import

```
df_total<-NULL
mydb = dbConnect(MySQL(), user="base user id", password='Password',
dbname='databasename', host='hostid')
month_name=data.frame(
month=c(1:12),
month_names=c("January","February","March","April","May","June","July","A
ugust","September","October","November","December")
)
```

Connects to a MySQL database and imports data from a CSV file. month_name data frame maps month numbers to names.

##Data Preparation

```
ss_final<-fread(file="PCAappliedInputfile.csv",header=T,check.names = F)
ss=ss_final[ss_final$month==month_number,]

#write.csv(ss,"ss_dec_data.csv")
ss_lag=ss_final[ss_final$month==month_number-1,]
colnames(ss_lag)[18:63]=paste0(colnames(ss_lag)[18:63],"_lag")
ss=cbind(ss,ss_lag[,18:63])
names(ss)
col_pars=names(ss)
vars= paste(col_pars[7:ncol(ss)],collapse = "+")
vars=paste0(vars,"+rating")
options(verbose = F)
```

Filters data for the current month and the previous month, then combines them. Lag variables are created by appending suffixes to column names.

Database Query and Fetching Data

```
rs<-dbSendQuery(mydb,"SELECT
state.state_name,state.state_id,district.district_id,
district.district_name, dval_ob_district_final.year_id,
dval_ob_final.month_id,species.species_name, disease.disease_id,
disease.disease_name, dval_ob_district_final.number_outbreak,
dval_ob_district_final.number_susceptible,
dval_ob_district_final.number_attack, dval_ob_district_final.number_death
FROM dval_ob_district_final
INNER JOIN  state on state.state_id=dval_ob_district_final.state_id
INNER JOIN  district on
district.district_id=dval_ob_district_final.district_id
INNER JOIN  disease on
disease.disease_id=dval_ob_district_final.disease_id
INNER JOIN species on
species.species_id=dval_ob_district_final.species_id")
data1 = fetch(rs, n=-1)

colnames(data1)=c("state_name","state_id","district_id","district_name","
year","month","species_name","disease_id","disease_name",
"number_of_outbreaks","number_susceptible","number_of_attacks","number_of
_deaths")
```

Sends a SQL query to fetch disease-related data from the database and stores it in data1.

Import or read shapefile

```
k1=readOGR("shapefile")
k1@data
names(k1)[1]<-"DISTRICT"
names(k1)[2]<-"ST_NM"
names(k1)[3]<-"ST_CEN_CD"
names(k1)[4]<-"DT_CEN_CD"
ll_coord=data.frame(coordinates(k1))
final_eval=NULL
```

Reads a shapefile for spatial data of districts and states in India

##write disease data file to external folder for further analysis

```
fwrite(data1,paste0("dist_out_nadres_",gsub("\\:","_",Sys.time()),".csv")
)
```

determine diseases / assign ID to the unique disease, and then determine the diseases you want to analyze.

```
for(disease in c(8,10,11,12,31,35,37,48,60,65,70,72,79,146,189))
# The livestock diseases are arranged alphabetically, and IDs are
assigned to them in the database.

{
  k=k1
  rat_df=c(1:nrow(k))
  df=data1
```

This loop iterates over a predefined list of disease IDs. For each disease, it sets up the necessary variables and prepares to process the data related to that disease.

```
if(disease==12)
{
d1=df[df$year>=1987 & df$year<=current_year & df$disease_id==disease &
df$month==month_number,]
  } else  d1=df[df$year>=year_number & df$year<=current_year &
df$disease_id==disease & df$month==month_number,]
d2=d1[,c("state_name","district_name","disease_name","month","year")]
```

Filters the df data frame to include records for the current disease, within the specified year and month range. The filtered data is used for further analysis.

assigning rating to the years to remove noise

```
yr_rt=data.frame(year=c(year_number:current_year),rating=c(1:10))
st_dt=data.frame(k@data[,c("ST_NM","DISTRICT")])
d2$state_name=toupper(d2$state_name)
st_dt$rating=0
for (i in 1:nrow(st_dt)) {

rt=yr_rt[yr_rt$year==max(d2[which(d2$state_name==as.character(st_dt[i,"ST
_NM"]) &

d2$district_name==as.character(st_dt[i,"DISTRICT"])),"year"]),"rating"]
    if(sum(rt)!=0)
      st_dt[i,"rating"]=rt
  }
rat_df=cbind(rat_df,st_dt)
colnames(rat_df)[c(2,3)]=c("state_name","district_name")
```

Assigns ratings to each year to minimize noise in the data. Ratings are used to evaluate the relevance of the data based on the most recent year of data available.

```
data<-subset(data1,data1$year>=year_number & data1$disease_id==disease)
df<-sqldf("SELECT
state_id,state_name,district_id,district_name,disease_id,disease_name,mon
th,sum(number_of_outbreaks)as outbreak FROM data GROUP BY
state_id,district_id,state_name,district_name,month,disease_id,disease_na
me",drv="SQLite")
ss1<-subset(ss,ss$disease_id==disease)
```

Aggregates the data by summing up the number of outbreaks for each state, district, and month. This step ensures that data is compiled in a format suitable for analysis.

```
dd<-merge(ss1, df, by =
c("state_id","district_id","disease_id","month"),all.x=TRUE)
attach(dd,warn.conflicts = F)
out<-data.frame(outbreak)
out<-ifelse(outbreak>=1,1,0)
out[is.na(out)]<-0
final<-cbind(dd,out)
```

Merges the PCA-applied data (ss1) with aggregated disease data (df). Creates a binary outcome variable out based on whether outbreaks occurred or not, and prepares the final dataset for modelling.

```
final1<-final[which(final$disease_id==disease &
final$month==month_number),]
cat("For disease: ",as.character(unique(ss1[,"disease_name"])),"\n")
ncs= ncol(final1)-5
temp = data.frame(final1[,8:ncs])
for(i in 1:ncol(temp)){
temp[is.na(temp[,i]), i] <- mean(temp[,i], na.rm = TRUE)
  }
```

Selects the relevant rows for the current disease and month, and handles missing values in the data by replacing them with column means.

```
final2<-
cbind(final1$state_id,final1$state_name.x,final1$district_id,final1$distr
ict_name.x,final1$disease_id,final1$disease_name.x,final1$out,final1$mont
h,temp)
setnames(final2,old=c("final1$state_id","final1$state_name.x","final1$dis
trict_id","final1$district_name.x","final1$disease_id","final1$disease_na
me.x","final1$out","final1$month"),new=c("state_id","state_name","distric
t_id","district_name","disease_id","disease_name","out","month"))
final2=join(data.frame(final2),rat_df,match="full",type="left")
formula=paste("out ~",vars)
formula=as.formula(formula)
```

```
final2[is.na(final2$rating),"rating"]=0
```

Constructs the final dataset final2 by combining various columns and ensuring proper naming. Also joins with the rating data and prepares the formula for modeling.

```
kappa=roc=tss=array()
sensitivity<- numeric(1)
specificity<- numeric(1)
precision <- numeric(1)
recall <- numeric(1)
f1_score <- numeric(1)
mcc <- numeric(1)
accuracy <- numeric(1)
error_rate <- numeric(1)
model<-glm(formula,data = final2, family
=binomial(link="logit"),maxit=20)
prediction_glm<-predict(model,type="response")
tmp_df=cbind(final2,prediction_glm)
glkappa<-cohen.kappa(data.frame(final2$out,prediction_glm))
kappa[1]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,prediction_glm)
roc[1]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,prediction_glm)
max_thres=max(glvv[[8]])
glxx<-confusion.matrix(final2$out,prediction_glm,max_thres)
tss[1]<-round(TSS.Stat(glxx),3)
binary_predictions <- ifelse(prediction_glm > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[1] <- TP / (TP + FN)
specificity[1] <- TN / (TN + FP)
precision[1] <- TP / (TP + FP)
f1_score[1] <- 2 * (precision * sensitivity) / (precision +sensitivity)
mcc[1] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP)  *
(TN + FN))
accuracy[1] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[1] <- 1 - accuracy
```

Fits a generalized linear model (GLM) using logistic regression. Evaluates the model's performance by calculating various metrics such as kappa, ROC AUC, TSS, sensitivity, specificity, precision, recall, F1 score, MCC, accuracy, and error rate.

```
# library(randomForest)
n2 <- randomForest(as.formula(formula), data = final2, ntree = 8000, mtry
= 100, maxdepth = 900)
prediction_rf<-predict(n2,type="response")
tmp_df=cbind(final2, prediction_rf)
glkappa<-cohen.kappa(data.frame(final2$out,prediction_rf))
kappa[2]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,prediction_rf)
roc[2]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,prediction_rf)
max_thres=max(glvv[[8]])
glxx<-confusion.matrix(final2$out,prediction_rf,max_thres)
tss[2]<-round(TSS.Stat(glxx),3)
binary_predictions <- ifelse(prediction_rf > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[2] <- TP / (TP + FN)
specificity[2] <- TN / (TN + FP)
precision[2] <- TP / (TP + FP)
f1_score[2] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[2] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[2] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[2] <- 1 - accuracy[2]
```

The code trains a Random Forest model with specified parameters and uses it to predict outcomes. It then integrates these predictions with the original dataset to evaluate model performance. Cohen's Kappa is calculated to measure agreement between observed and predicted values, while ROC AUC assesses the model's classification performance. The optimal prediction threshold is determined, and the True Skill Statistic (TSS) is computed. Finally, various performance metrics, including sensitivity, specificity, precision, F1 score, Matthews Correlation Coefficient (MCC), accuracy, and error rate, are calculated to assess the model's overall effectiveness.

```
library(mgcv)
model_gam <- gam(formula, data = final2, family = binomial(link =
"logit"))
predicted_gam<- predict(model_gam, type = "response")
tmp_df1=cbind(final2,predicted_gam)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_gam))
kappa[4]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_gam)
roc[4]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_gam)
```

```
max_thres=max(glvv[[8]])
glgam<-confusion.matrix(final2$out,predicted_gam,max_thres)
tss[4]<-round(TSS.Stat(glgam),3)
binary_predictions <- ifelse(predicted_gam > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[4] <- TP / (TP + FN)
specificity[4] <- TN / (TN + FP)
precision[4] <- TP / (TP + FP)
f1_score[4] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[4] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[4] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[4] <- 1 - accuracy[4]
```

The code trains a Generalized Additive Model (GAM) with a logistic link function using the mgcv package and makes predictions based on this model. It combines the predictions with the original dataset for further evaluation. Cohen's Kappa is computed to measure agreement between the observed and predicted values, while ROC AUC evaluates the model's performance. The optimal prediction threshold is identified, and the True Skill Statistic (TSS) is calculated. Performance metrics such as sensitivity, specificity, precision, F1 score, Matthews Correlation Coefficient (MCC), accuracy, and error rate are then calculated to assess the model's effectiveness.

```
##install.packages("earth")
library(earth)
mars_model <- earth(formula, data = final2)
predicted_mars <- predict(mars_model, newdata = final2)
tmp_df2=cbind(final2,predicted_mars)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_mars))
kappa[5]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_mars)
roc[5]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_mars)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_mars,max_thres)
tss[5]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_mars > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
```

```
sensitivity[5] <- TP / (TP + FN)
specificity[5] <- TN / (TN + FP)
precision[5] <- TP / (TP + FP)
f1_score[5] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[5] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[5] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[5] <- 1 - accuracy[5]
```

The MARS model was applied to predict binary outcomes based on the features in the final2 dataset. It uses a non-parametric approach to model complex, nonlinear relationships. After training the model, predictions were generated, and performance was evaluated using metrics such as Cohen's kappa, ROC AUC, and confusion matrix statistics. The model's effectiveness was quantified through various metrics, including sensitivity, specificity, and F1 score, revealing its ability to handle non-linearities in the data.

```
#install.packages("cubist)
library(Cubist)
param_grid <- expand.grid(committees = c(3, 7, 15), neighbors = c(5, 6,
7))
cubist_grid <- train( x = final2[, -c(1:7)],  # Features
y = final2$out,         # Response variable
method = "cubist",      # Cubist method
trControl = trainControl(method = "cv", number = 5),  # 5-fold cross-
validation
      tuneGrid = param_grid   # Parameter grid
    )
predicted_cubist <- predict(cubist_grid, newdata = final2, method =
"response")
tmp_df2=cbind(final2,predicted_cubist)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_cubist))
kappa[5]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_cubist)
roc[5]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_cubist)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_cubist,max_thres)
tss[5]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_cubist > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[5] <- TP / (TP + FN)
specificity[5] <- TN / (TN + FP)
```

```
precision[5] <- TP / (TP + FP)
f1_score[5] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[5] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[5] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[5] <- 1 - accuracy[5]
```

The Cubist model, a rule-based approach that combines regression trees with instance-based learning, was used to model the binary outcome variable. Hyperparameter tuning was performed through a grid search, optimizing the number of committees and neighbors. Performance metrics, including Cohen's kappa and ROC AUC, were calculated to assess the model's predictive accuracy. The confusion matrix was used to derive additional performance metrics such as sensitivity, specificity, and F1 score, providing insights into the model's classification effectiveness.

```
#install.packages("xgboost")
library(xgboost)
X <- as.matrix(final2[, -(1:7)])  # Predictor variables
y <- final2[, "out"]              # Response variable
params <- list(
booster = "gbtree",               # Tree-based model
objective = "binary:logistic",    # Binary classification
max_depth = 6,                    # Maximum tree depth
eta = 0.3,                        # Learning rate
nrounds = 20         # Number of boosting rounds (similar to maxit)
  )
xgb_model <- xgboost(data = X, label = y, params = params, nrounds =
params$nrounds)
predicted_xgb <- predict(xgb_model, X)
tmp_df3=cbind(final2,predicted_xgb)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_xgb))
kappa[6]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_xgb)
roc[6]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_xgb)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_xgb,max_thres)
tss[6]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_xgb > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[6] <- TP / (TP + FN)
specificity[6] <- TN / (TN + FP)
precision[6] <- TP / (TP + FP)
```

```
f1_score[6] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[6] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[6] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[6] <- 1 - accuracy[6]
```

The XGBoost model, known for its gradient boosting capabilities, was used for binary classification. The model was configured with parameters suited for boosting, such as learning rate and maximum tree depth. Predictions were made, and performance metrics including Cohen's kappa, ROC AUC, and confusion matrix statistics were computed. The model's predictive power was evaluated using accuracy, sensitivity, and other performance measures, highlighting its effectiveness in handling large datasets and complex patterns.

```
library(e1071)
svm_model <- svm(formula, data = final2, kernel = "radial", cost = 1)
# Adjust kernel and cost as needed
predicted_svm <- predict(svm_model, final2, type = "response")
tmp_df4=cbind(final2,predicted_svm)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_svm))
kappa[7]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_svm)
roc[7]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_svm)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_svm,max_thres)
tss[7]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_svm > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[7] <- TP / (TP + FN)
specificity[7] <- TN / (TN + FP)
precision[7] <- TP / (TP + FP)
f1_score[7] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[7] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[7] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[7] <- 1 - accuracy[7]
```

An SVM model with a radial basis function kernel was applied to classify the binary outcome. The model was trained with specific cost parameters to balance margin and error. Predictions were evaluated using Cohen's kappa, ROC AUC, and other classification metrics. The confusion matrix provided detailed insights into the model's sensitivity, specificity, precision, and F1 score, demonstrating its capability in separating classes in high-dimensional space.

```
library(rpart)
tree_model <- rpart(formula, data = final2, method = "class")
predicted_tree_prob <- predict(tree_model, final2, type = "prob")[, "1"]
tmp_df5=cbind(final2,predicted_tree_prob)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_tree_prob))
kappa[8]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_tree_prob)
roc[8]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_tree_prob)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_tree_prob,max_thres)
tss[8]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_tree_prob > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[8] <- TP / (TP + FN)
specificity[8] <- TN / (TN + FP)
precision[8] <- TP / (TP + FP)
f1_score[8] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[8] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[8] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[8] <- 1 - accuracy[8]
```

The decision tree model, implemented with the rpart package, was used to predict the binary outcome variable. This model generates a tree-like structure to make decisions based on feature values. Predictions were assessed using Cohen's kappa, ROC AUC, and confusion matrix metrics. Performance evaluation included sensitivity, specificity, and accuracy measures, illustrating the model's effectiveness in capturing decision rules from the data.

```
library(glmnet)
X <- as.matrix(final2[, -(1:7)])  # Predictor variables
y <- final2[, "out"]  # Response variable
lasso_cv <- cv.glmnet(X, y, alpha = 1)
optimal_lambda <- lasso_cv$lambda.min
lasso_model <- glmnet(X, y, alpha = 0, lambda = optimal_lambda)
predicted_lasso <- predict(lasso_model, newx = X)
final2$predicted_lasso <- predicted_lasso
tmp_df6=cbind(final2,predicted_lasso)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_lasso))
kappa[9]<-round(glkappa[[2]],3)
```

```
glroc<-roc(final2$out,predicted_lasso)
roc[9]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_lasso)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_lasso,max_thres)
tss[9]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_lasso > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[9] <- TP / (TP + FN)
specificity[9] <- TN / (TN + FP)
precision[9] <- TP / (TP + FP)
f1_score[9] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[9] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[9] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[9] <- 1 - accuracy[9]
```

The Lasso regression model, implemented using glmnet, was used for feature selection and prediction in binary classification. By applying L1 regularization, the model penalizes less important features, enhancing model interpretability. Predictions were evaluated with metrics such as Cohen's kappa, ROC AUC, and confusion matrix statistics. The model's performance in terms of accuracy, sensitivity, and F1 score was assessed, demonstrating its ability to handle high-dimensional data and perform feature selection.

```
#install.packages("fda")
library(fda)
fda_model <- lm(formula, data = final2)
predicted_fda <- predict(fda_model)
final2$predicted_fda <- predicted_fda
tmp_df8=cbind(final2,predicted_fda)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_fda))
kappa[10]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_fda)
roc[10]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_fda)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_fda,max_thres)
tss[10]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_fda > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
```

```
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[10] <- TP / (TP + FN)
specificity[10] <- TN / (TN + FP)
precision[10] <- TP / (TP + FP)
f1_score[10] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[10] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[10] <- (TP + TN) / (TP + TN + FP + FN)
error_rate <- 1 - accuracy[10]
```

The Functional Data Analysis (FDA) model was applied to analyze and predict binary outcomes based on functional data. This approach fits a model using functional data representation, which is suitable for capturing complex patterns over a continuum. By predicting outcomes based on these functional features, the model enhances interpretability and handles data variability effectively. Model performance was evaluated using Cohen's kappa, ROC AUC, and confusion matrix metrics, assessing its accuracy, sensitivity, and overall predictive capability.

```
library(caret)
probit_model <- train(formula, data = final2, method = "glm", family =
binomial(link = "probit"))
predicted_probit = predict(probit_model)
final2$predicted_probit <- predicted_probit
tmp_df8=cbind(final2,predicted_probit)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_probit))
kappa[10]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_probit)
roc[10]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_probit)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_probit,max_thres)
tss[10]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_probit > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[10] <- TP / (TP + FN)
specificity[10] <- TN / (TN + FP)
precision[10] <- TP / (TP + FP)
f1_score[10] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[10] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[10] <- (TP + TN) / (TP + TN + FP + FN)
error_rate <- 1 - accuracy[10]
```

The Probit regression model was utilized for binary classification by estimating probabilities using a cumulative normal distribution. This model addresses non-linear relationships between predictors and binary outcomes. Its performance was evaluated through Cohen's kappa, ROC AUC, and confusion matrix, highlighting its accuracy, sensitivity, and ability to manage classification tasks effectively.

```
library(kernlab)
gp_model <- gausspr(formula, data = final2)
predicted_gp <- predict(gp_model, newdata = final2)
gp_model <- gausspr(formula, data = final2, kernel = "rbfdot", kpar =
list(sigma = 0.1), cost = 1)
predicted_gp <- predict(gp_model, newdata = final2)
tmp_df9=cbind(final2,predicted_gp)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_gp))
kappa[11]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_gp)
roc[11]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_gp)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_gp,max_thres)
tss[11]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_gp > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[11] <- TP / (TP + FN)
specificity[11] <- TN / (TN + FP)
precision[11] <- TP / (TP + FP)
f1_score[11] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[11] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[11] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[11] <- 1 - accuracy[11]
```

The Gaussian Process (GP) model was employed to predict binary outcomes by leveraging a distribution over functions. This model is advantageous for handling complex, non-linear relationships in high-dimensional data. Performance metrics such as Cohen's kappa, ROC AUC, and confusion matrix were used to evaluate the model's accuracy, sensitivity, and classification effectiveness.

```
#install.packages("neuralnet")
library(neuralnet)
nn_model <- neuralnet(formula, data = final2, hidden = c(5, 2),
linear.output = FALSE)
predicted_nn <- predict(nn_model, final2)
tmp_df10=cbind(final2,predicted_nn)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_nn))
kappa[12]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_nn)
roc[12]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_nn)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_nn,max_thres)
tss[12]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_nn > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[12] <- TP / (TP + FN)
specificity[12] <- TN / (TN + FP)
precision[12] <- TP / (TP + FP)
f1_score[12] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[12] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[12] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[12] <- 1 - accuracy[12]
```

The Neural Networks (NN) model was employed for binary classification by learning complex patterns through interconnected layers. This model can capture non-linear relationships effectively. Performance was measured using Cohen's kappa, ROC AUC, and confusion matrix metrics, which assessed the model's accuracy, sensitivity, and overall classification performance.

```
library(nnet)
multinom_model <- multinom(formula, data = final2)
predicted_multinom_probs <- predict(multinom_model, final2, type =
"probs")
final2$predicted_multinom_probs <- predicted_multinom_probs
tmp_df11=cbind(final2,predicted_multinom_probs)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_multinom_probs))
kappa[13]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_multinom_probs)
roc[13]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_multinom_probs)
max_thres=max(glvv[[8]])
```

```
glmars<-confusion.matrix(final2$out,predicted_multinom_probs,max_thres)
tss[13]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_multinom_probs > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[13] <- TP / (TP + FN)
specificity[13] <- TN / (TN + FP)
precision[13] <- TP / (TP + FP)
f1_score[13] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[13] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[13] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[13] <- 1 - accuracy[13]
```

The Multinomial Regression model was used to predict outcomes with more than two categories by estimating probabilities for each category. This model extends logistic regression to handle multi-class classification problems. Performance evaluation through Cohen's kappa, ROC AUC, and confusion matrix provided insights into the model's accuracy, sensitivity, and classification performance for multi-class data.

```
library(kernlab)
ksvm_model <- ksvm(formula, data = final2, type = "C-svc", kernel =
"rbfdot")
predicted_ksvm <- predict(ksvm_model, final2)
tmp_df13=cbind(final2,predicted_ksvm)
glkappa<-cohen.kappa(data.frame(final2$out,predicted_ksvm))
kappa[14]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,predicted_ksvm)
roc[14]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,predicted_ksvm)
max_thres=max(glvv[[8]])
glmars<-confusion.matrix(final2$out,predicted_ksvm,max_thres)
tss[14]<-round(TSS.Stat(glmars),3)
binary_predictions <- ifelse(predicted_ksvm > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[14] <- TP / (TP + FN)
specificity[14] <- TN / (TN + FP)
precision[14] <- TP / (TP + FP)
f1_score[14] <- 2 * (precision * sensitivity) / (precision + sensitivity)
```

```
mcc[14] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[14] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[14] <- 1 - accuracy[14]
```

The k-SVM model was applied for binary classification by utilizing kernel functions to map input features into a higher-dimensional space. This model is effective for handling non-linearly separable data. Performance was assessed using Cohen's kappa, ROC AUC, and confusion matrix metrics, highlighting the model's accuracy, sensitivity, and classification capabilities.

```
if(disease_id!=12)
{
gbm_model=gbm.step(data=final2, gbm.x = 8:grep("PET",names(final2)),
gbm.y = 7, family = "bernoulli", tree.complexity = 1, learning.rate =
0.01,
bag.fraction = 0.65, n.trees = 5,keep.fold.fit=T,tolerance.method="fixed"
, step.size = 5,n.folds = 10)
prediction_gbm<-
predict(gbm_model,n.trees=gbm_model$gbm.call$best.trees,type="response")
glkappa<-cohen.kappa(data.frame(final2$out,prediction_gbm))
kappa[3]<-round(glkappa[[2]],3)
glroc<-roc(final2$out,prediction_gbm)
roc[3]<-round(as.numeric(glroc$auc),3)
glvv<-optim.thresh(final2$out,prediction_gbm)
max_thres=max(glvv[[8]])
glxx<-confusion.matrix(final2$out,prediction_gbm,max_thres)
tss[3]<-round(TSS.Stat(glxx),3)
binary_predictions <- ifelse(prediction_gbm > 0.5, 1, 0)
conf_matrix <- table(final2$out, binary_predictions)
TP <- sum(binary_predictions == 1 & final2$out == 1)
TN <- sum(binary_predictions == 0 & final2$out == 0)
FP <- sum(binary_predictions == 1 & final2$out == 0)
FN <- sum(binary_predictions == 0 & final2$out == 1)
sensitivity[3] <- TP / (TP + FN)
specificity[3] <- TN / (TN + FP)
precision[3] <- TP / (TP + FP)
f1_score[3] <- 2 * (precision * sensitivity) / (precision + sensitivity)
mcc[3] <- (TP * TN - FP * FN) / sqrt((TP + FP) * (TP + FN) * (TN + FP) *
(TN + FN))
accuracy[3] <- (TP + TN) / (TP + TN + FP + FN)
error_rate[3] <- 1 - accuracy[3]

  }
```

In this section of the code, a Gradient Boosting Machine (GBM) model is trained using the gbm.step function. The model is built with features from the dataset final2, excluding those with "PET" in their names, and the response variable is specified. Key parameters for the GBM model, including tree complexity, learning rate, and the number of trees, are set to control model complexity and training. Cross-validation with 10 folds is employed to optimize the model's performance, and predictions are generated using the best number of trees. Evaluation metrics such as Cohen's Kappa, ROC AUC, True Skill Statistic (TSS), and other performance measures are then computed to assess the model's accuracy and effectiveness in classifying outcomes. This process helps in understanding the model's predictive power and its alignment with actual data.

```
eval<-cbind(kappa, roc, tss, sensitivity, specificity, precision,
f1_score, mcc, accuracy, error_rate)
model_names =
c("glm","rf","gam","mars","xgb","svm","tree_model","lasso_model","fda","g
p_model","nn_model","multinomial","ksvm","gbm")

eval=data.frame(eval)
eval$disease=as.character(unique(ss1[,"disease_name"]))
eval <- cbind(eval, model_names)
final_eval=rbind(final_eval,eval)
# fname=paste(dir_name,"/",dir_name,"_evaluation.csv",sep="")
# write.csv(final,fname)

prediction=numeric()
for (i in 1:length(prediction_glm)) {
prediction[i]=min(prediction_glm[i],prediction_rf[i],prediction_gbm[i],pr
edicted_gam[i],predicted_mars[i],predicted_xgb[i] >= 0.5)
      if (
      prediction_glm[i] >= 0.5 ||
      prediction_rf[i] >= 0.5 ||
      prediction_gbm[i] >= 0.5 ||
      predicted_gam[i] >= 0.5 ||
      predictedmars[i] >= 0.5 ||
      predicted_xgb[i] >= 0.5 ||
      predicted_svm[i] >= 0.5 ||
      predicted_tree_prob[i] >= 0.5 ||
      predicted_lasso[i] >= 0.5 ||
      predicted_fda[i] >= 0.5 ||
      predicted_gp[i] >= 0.5 ||
      predicted_nn[i] >= 0.5 ||
      predicted_multinom_probs[i] >= 0.5 ||
      predicted_ksvm[i] >= 0.5
    ) {
      prediction[i] = pmax(
        prediction_glm[i],
        prediction_rf[i],
```

```
        predictiongbm[i],
        predicted_gam[i],
        predicted_mars[i],
        predicted_xgb[i],
        predicted_svm[i],
        predicted_tree_prob[i],
        predicted_lasso[i],
        predicted_fda[i],
        predicted_gp[i],
        predicted_nn[i],
        predicted_multinom_probs[i],
        predicted_ksvm[i]
      )
    } else {
      prediction[i] = pmin(
        prediction_glm[i],
        prediction_rf[i],
        prediction_gbm[i],
        predicted_gam[i],
        predicted_mars[i],
        predicted_xgb[i],
        predicted_svm[i],
        predicted_tree_prob[i],
        predicted_lasso[i],
        predicted_fda[i],
        predicted_gp[i],
        predicted_nn[i],
        predicted_multinom_probs[i],
        predicted_ksvm[i]
      )
    }
```

This code block combines predictions from multiple models. For each prediction, it evaluates the maximum or minimum prediction value among all models based on whether any model's prediction exceeds 0.5. This approach helps in aggregating predictions from various models to potentially improve overall performance.

```
# summary(prediction)
  vv<-round(prediction,2)

# names(final2)
  df1<-cbind(final2,vv)
  df_total<-rbind(df_total, df1)
  gc()
}
```

The above codes combine the original dataset (final2) with the newly computed predictions (vv). The combined data is appended to df_total, and garbage collection (gc()) is performed to free up memory.

##define risk level by giving predicted values

```
f=function(m){
  if(m<=0.0 | is.na(m)) i=1
  else if(m>=0.1 && m<=0.19) i=2
  else if(m>=0.20 && m<=0.40) i=3
  else if(m>=0.41 && m<=0.60) i=4
  else if(m>=0.61 && m<=0.80) i=5
  else i=6
}
```

This function f assigns a risk category based on the predicted value m. The risk levels are defined as follows:

1: No Risk (NR) if the value is 0.0 or NA.
2: Very Low Risk (VLR) for values between 0.1 and 0.19.
3: Low Risk (LR) for values between 0.20 and 0.40.
4: Medium Risk (MR) for values between 0.41 and 0.60.
5: High Risk (HR) for values between 0.61 and 0.80.
6: Very High Risk (VHR) for values above 0.80.

```
df_total$cate=factor(mapply(f,df_total$vv),levels=1:6,labels=c("NR","VLR"
,"LR","MR","HR","VHR"))
fwrite(df_total,"outputfile.csv",row.names = F)
```

The df_total dataset is updated with a new column cate, which represents the risk category for each prediction using the function f. The risk categories are converted to factors with appropriate labels. The updated dataset is then saved to a CSV file named "outputfile.csv".

Function to get the highest category

```
get_highest_category <- function(categories) {
risk_levels <- c("VHR", "HR", "VLR", "MR", "LR", "NR")
max_risk_level <- risk_levels[which.max(risk_levels %in% categories)]
return(max_risk_level)
}
```

The code starts with a function named get_highest_category. This function is designed to identify the highest risk category from a list of categories. It operates by first defining a vector called risk_levels that contains the risk categories ordered from highest to lowest priority: "VHR" (Very High Risk), "HR" (High Risk), "VLR" (Very Low Risk), "MR" (Moderate Risk), "LR" (Low Risk), and "NR" (No Risk). The function then checks which of these risk levels are present in the input vector categories. Using which.max, it finds the index of the highest risk level in the predefined order and returns this level as the output. This function ensures that the most critical risk level is selected from the given categories.

Group by state, district, and disease, and summarize to get the highest category for each group

```
df_total <- df_total %>%
group_by(state_id,state_name,district_id,district_name,disease_id,
disease_name,month,out,cattle,buaffalo,goat,sheep,pig) %>%
summarise(cate = get_highest_category(cate),vv=max(vv))
```

The code groups the data frame df_total by multiple columns and summarizes it by applying the get_highest_category function to determine the highest risk category (cate) and calculating the maximum value of the vv column for each group.

```
write.csv(df_total,"filteredRisk.csv")
```

This line saves the summarized data frame df_total to a CSV file named filteredRisk.csv for further use or sharing.

#Results Import and Directory Creation

```
df_total=fread("outputfile.csv",header=T,check.names=F,data.table = F)
dir.create(path = paste(month_name[month_number,2],current_year))
df_poa=df_total
df_poa$cate=factor(mapply(f,df_poa$vv),levels=1:6,labels=c(0,0,0,0,1,1))
df_poa=df_poa[which(df_poa$month==month_name[month_number,1]),]
df_p=df_poa[,c("disease_name","out","cate")]
df_poa_acc=sqldf("select df_p.disease_name,count(df_p.out) as
Outbreak_count,count(df_p.cate) as Predicted_count,df_p.out,df_p.cate
                 from df_p group by
df_p.disease_name,df_p.out,df_p.cate")
df_tp_tn=sqldf("select
df_poa_acc.disease_name,sum(df_poa_acc.Outbreak_count) as Outbreak_count
from df_poa_acc
             where df_poa_acc.out=df_poa_acc.cate
             group by df_poa_acc.disease_name")
df_tot_res=sqldf("select
df_poa_acc.disease_name,sum(df_poa_acc.Outbreak_count) as Total_count
from df_poa_acc
             group by df_poa_acc.disease_name")

df_acc=cbind(data.frame(c(1:nrow(df_tot_res))),data.frame(df_tp_tn[,1]),(
df_tp_tn[,2]/df_tot_res[,2])*100)
df_acc=setNames(df_acc,c("No","Disease","Accuracy"))
dis_acc=paste(paste(month_name[month_number,2]," ",current_year,"/",sep =
""),"Disease Accuracy ",month_name[month_number,2],"
",current_year,".csv",sep="")
write.csv(df_acc,dis_acc,row.names = F)
```

This section begins by importing the data from the outputfile.csv file into a data frame and creating a directory for the specified month and year. The data is then processed to convert vv values into binary categories using a custom function. The script filters the data for the current month and calculates the accuracy of predictions by comparing predicted categories to actual outcomes. The results are aggregated and saved to a CSV file named with the current month and year.

```
# Risk Count Calculation and Transformation
qry=sprintf("select
df_total.state_name,df_total.disease_name,count(df_total.cate) as
Risk_count from df_total where df_total.cate in('VHR','HR') and
month='%s' group by
df_total.state_name,df_total.disease_name",month_number)
df_qr=sqldf(qry)
df_cast_risk=dcast(df_qr,state_name~disease_name,value.var =
"Risk_count")
df_cast_risk=df_cast_risk[,!(names(df_cast_risk) %in% drops)]
df_colmean <- data.frame("Total_Disease_District_Predicted",
t(colSums(as.data.frame(df_cast_risk[, 2:ncol(df_cast_risk)]), na.rm =
TRUE)))
df_rowmean=data.frame(as.integer(rowSums(df_cast_risk[2:ncol(df_cast_risk
)],na.rm = T)))
df_rowmean=setNames(df_rowmean,"Total_Outbreaks_predicted")
sum_all <- as.integer(sum(colSums(as.data.frame(df_cast_risk[,
2:ncol(df_cast_risk)]), na.rm = TRUE)))
df_cast_risk2=cbind(df_cast_risk,df_rowmean)
df_colmean=cbind(df_colmean,sum_all)
df_colmean= setNames(df_colmean,names(df_cast_risk2))
df_cast_risk2=rbind(df_cast_risk2,df_colmean)
risk_fname=paste(paste(month_name[month_number,2],"
",current_year,"/",sep = ""),"Risk Count StateWise
",month_name[month_number,2]," ",current_year,".csv",sep="")
write.csv(df_cast_risk2,risk_fname,row.names = F,na = "0")
```

This part of the code calculates the count of high-risk categories ("VHR" and "HR") by state and disease for the specified month. It reshapes the data to a wide format with states as rows and diseases as columns, excluding specific diseases from the analysis. The script then computes summary statistics, including total counts of outbreaks predicted and overall counts for each state and disease. The results are saved to a CSV file named with the current month and year.

```
#Outbreak Counts Calculation
df_nd=df_total
df_nd=df_nd[,c("out","month")]
df_count0=sqldf("select count(df_nd.out) as Out_count0,df_nd.month from
df_nd
                where df_nd.out=0
                group by df_nd.month")
```

```
df_count1=sqldf("select count(df_nd.out) as Out_count1,df_nd.month from
df_nd
                where df_nd.out=1
                group by df_nd.month")

outb_df=merge(df_count0,df_count1,all=T)
fnm=paste(paste(month_name[month_number,2]," ",current_year,"/",sep =
""),"Outbreak Counts ",month_name[month_number,2],"
",current_year,".csv",sep="")
write.csv(outb_df,fnm)
```

The script begins by selecting the columns "out" and "month" from the df_total data frame. It then calculates the count of outbreaks where out is 0 and 1, grouped by month, storing these counts in separate data frames (df_count0 and df_count1). These counts are merged into a single data frame (outb_df), and the combined data is saved to a CSV file named with the current month and year, detailing the outbreak counts.

Directory Creation and Library Loading

```
Forecastingtable_dir=paste(paste(month_name[month_number,2],"
",current_year,"/",sep=""),month_name[month_number,2]," ",current_year,"
T",sep="")
dir.create(path = table_dir)
detach("package:xlsx", unload=TRUE)
library(xlsx)
wb = createWorkbook()
```

This section starts by creating a directory to store the Excel files. The directory name is based on the current month and year. It then ensures the xlsx package is unloaded if previously loaded and reloads it. A new Excel workbook (wb) is initialized for exporting data.

```
#Data Processing and Export to Excel
st=sort(as.character(unique(df_total[,"state_name"])))
print("Exporting to Excel for each state")
i=1
for(i  in 1:36)
{
df_temp=df_total[,c("state_id","state_name","district_id","district_name"
,"disease_id","disease_name","month"),]
df_temp=df_temp[which(df_temp$month==month_name[month_number,1] &
df_temp$state_name==st[i]),]
state_name =unique(as.character(df_temp[,"state_name"]))
df_cast=dcast(df_temp,state_name+district_name~disease_name)
drops=c("Contagious caprine pleuro pneumonia","Rabies")
df_disease_final=df_cast[,!(names(df_cast) %in% drops)]
```

```
#colnames(df_disease_final)=c(paste("State Name","Districts
of",state_name),"Anthrax","Babesiosis","BQ","BT","ET","Fascioliasis","FMD
","HS","PPR","S&G Pox","SF","Theileriosis","Trypanosomiasis")
colnames(df_disease_final)=c("State Name",paste("Districts
of",state_name),"African Swine
Fever","Anthrax","Babesiosis","BQ","BT","Classical Swine
fever","ET","Fascioliasis","FMD","HS","Lumpy skin disease","PPR","S&G
Pox","Theileriosis","Trypanosomiasis")
title <- paste(" District wise Livestock Disease forewarning for ",
as.character(month_name[month_number, 2]), " ", current_year, " : ",
state_name, sep="")
sheet <- createSheet(wb, state_name)
v_df <- data.frame(title)
# Add an extra row to match the number of columns in df_disease_final
v_df <- rbind(v_df, rep(NA, ncol(df_disease_final)))
# Set names to match the title
names(v_df) <- title
setColumnWidth(sheet, colIndex = c(1:ncol(df_disease_final)), colWidth =
15)
addDataFrame(v_df, sheet = sheet, startColumn = 7, startRow = 1,
row.names = FALSE)
addDataFrame(df_disease_final, sheet = sheet, startColumn = 1, startRow =
2, row.names = FALSE)
}
```

This section processes data for each state. It loops through all states, filters the data for the current month and state, and reshapes it to show disease counts by district. Unwanted diseases are removed, and columns are renamed for clarity. Each state's data is added to a new sheet in the Excel workbook with a descriptive title. Column widths are adjusted for readability, and the data is added to the sheet.

#Saving the Workbook

```
file_xls= paste(table_dir,"/","State Forecasting
",as.character(month_name[month_number,2]),"
",current_year,".xlsx",sep="")
saveWorkbook(wb, file_xls)
```

This section constructs the filename for the Excel file based on the current month and year. It then saves the workbook (wb) to the specified directory with the constructed filename, completing the export process.

#Directory Creation and Library Loading

```
library(tmap)
i=1
```

```
plot_dir=paste(paste(month_name[month_number,2],"
",current_year,"/",sep=""),month_name[month_number,2]," ",current_year,"
N",sep="")
dir.create(path = plot_dir)
```

This section loads the tmap library required for thematic mapping. It then creates a directory to store the plot files. The directory name includes the current month and year to organize the output files.

#Data Preparation and Plotting

```
disease = c(8,10,11,12,31,35,37,48,60,65,70,72,79,146,189)
state.sp=readOGR("shapefile")
while(i<=length(disease)){
kar=k1
cols=as.character(unique(df_total[df_total$disease_id==disease[i],"diseas
e_name"]))
df_disease=df_total[which(df_total$month==month_name[month_number,1] &
df_total$disease_id==disease[i]),]
Available",disease[i],0.00))
df_disease=df_disease[,c("state_id","state_name","district_id","district_
name","disease_id","vv")]
df_disease=setNames(df_disease,c("ST_CEN_CD","state_name","DT_CEN_CD","di
strict_name","disease_id","vv"))
kar@data=join(data.frame(kar@data),data.frame(df_disease),by=c("ST_CEN_CD
","DT_CEN_CD"),type="left",match="first")

#write.csv(kar@data, "merged_data.csv", row.names = FALSE)
colours<-c("darkgrey","#FFFF00","#FFC1C1","#FF7150","#FF8500","darkred")
kar$vv[is.na(kar$vv)]<-0
kar$lb=factor(mapply(f,kar$vv),levels=1:6,labels=c("No Risk / No
Data","Very Low Risk","Low Risk","Medium Risk","High Risk","Very High
Risk"))
cols=gsub("&", "and",cols)
disname= gsub("\\."," ",cols)
if(disname=="Enterotoxaemia")
{
    disname="Enterotoxemia"
  }
cat("Plot for disease:",disname,"\n")
plot_loc=paste(plot_dir,"/",disname,"/",sep="")
dir.create(plot_loc)
file_name=paste(plot_loc,disname,"_resmod.png",sep="")
plot_title= paste(" Risk Prediction of ",disname," for the month of
",month_name[month_number,2]," ",current_year," ",sep="")
#plot_title= paste(disname," risk
prediction(",month_name[month_number,2]," ",current_year,")",sep="")
```

```
#plot_title= paste(disname)
png(file_name,width = 6, height = 4, units = 'in',res = 200)


#print(spplot(obj = kar,c("lb"),col.regions=colours,main =
list(plot_title,cex=0.8),key.space=list(x=0.2,y=0.9,corner=c(0,1)),scales
=list(draw = TRUE)))
  t_map=tm_shape(kar, unit = "km") +
    tm_polygons(col = "lb", style = "jenks",
                border.alpha = 0, title = "", palette =
c("skyblue","yellow","pink","pink3","orange","#FF0000")) +
    tm_scale_bar(breaks = c(0, 100, 200), size = 1, position=c("left",
"bottom")) +
    tm_compass(type = "arrow", position = c("right","top")) +
    tm_layout(main.title = plot_title,
              main.title.size = 0.55, frame = FALSE)

  t_map1=t_map+tm_shape(state.sp, unit = "km") + tm_borders()
  print(t_map1)
  dev.off()
  i=i+1
}
```

This section iterates through a list of disease IDs, processes data related to each disease, and generates maps to visualize risk predictions. For each disease, it prepares a dataset and merges it with spatial data. It then creates a color-coded map using tmap, specifying risk levels with distinct colors. Each map is saved as a PNG file in a directory named after the disease, with appropriate titles and legends.

#Data Table Export

```
df_tot=df_total
df_tot$Outcome=factor(mapply(f,df_tot$vv),levels=1:6,labels=c("No Risk /
No Data","Very Low Risk","Low Risk","Medium Risk","High Risk","Very High
Risk"))
df_tot$month_letter=month.name[month_number]
load("cs_NDR_table.RData")
cs_ndr_final
df_tot=df_tot[,cs_ndr_final]
write.csv(df_tot,paste0("upload_",month.name[month_number],".csv"),row.na
mes = F)

final_eval12=final_eval
fwrite(final_eval,paste0(paste(month_name[month_number,2],current_year),"
/Eval.csv"))
}
nadres_func(current_year,year_number,month_number)
```

This section prepares a data table for uploading to a website. It converts the numeric risk values to categorical outcomes and adds a column for the month. It then loads a predefined table structure (cs_NDR_table.RData), filters the columns accordingly, and writes the updated data to a CSV file. Additionally, it saves another evaluation file (final_eval) to a CSV format for future use.

Part-2: To obtain district level disease maps

The nadres_func_state_maps function begins by setting the current year, year number, and month number, and then loads a suite of R libraries for data processing, spatial analysis, and machine learning. It initializes a mapping of month numbers to names, reads outbreak data from a CSV file, and categorizes the data into risk levels. The function creates a directory for saving plot outputs, loads a shapefile of Indian districts, and prepares the data by joining disease and spatial information. For each disease and state, it generates color-coded risk maps using spplot, assigning risk levels based on predefined thresholds, and saves these maps as PNG files with appropriate titles and labels. Finally, the function is executed with the specified parameters to produce the visualizations.

```
# Function Definition and Library Loading
month_number=month number; year_number= from year; current_year=
predicted year;
nadres_func_state_maps=function(current_year,year_number,month_number)
{
  print(current_year)
  print(year_number)
  print(month_number)

  library(RMySQL) # database connection
  library(rgdal) # to read shapefile
  library(RColorBrewer) # color palette
  library(sqldf) # to execute sql queries
  library(data.table) # to read csv files

  library(reshape2) # melt or dcast
  library(imputeMissings) # to fill missing values
  require(sp) # spatial data
  require(spdep) # spatial weights matrix
  require(rms)
  library(xlsx) #to read xlsx files
  library(plyr) #to join dataframes
  library(randomForest) # random forest model
  library(dismo) # gbm
  library(xgboost) #xgboost
  library(mgcv)
  library(earth)
  library(e1071)
```

```
  library(rpart)
  library(glmnet)
  library(fda)
  library(kernlab)
  library(neuralnet)
  library(nnet)
  library(kernlab)
  library(psych)
  library(pROC) # to calculate roc value
  library(SDMTools) # kappa
  library(BIOMOD)# TSS
```

The function nadres_func_state_maps begins by defining and printing input parameters for the current year, year number, and month number. It then loads a series of R libraries used for various data processing tasks, including database connections, spatial data handling, machine learning, and data manipulation.

```
#Month Name and Data Preparation
month_name=data.frame( month=c(1:12),
month_names=c("January","February","March","April","May","June","July","A
ugust","September","October","November","December")
  )

# import predicted data
  df_total=fread("outputfile.csv",header=T,check.names=F,data.table = F)
  f=function(m){
    if(m<=0.0 | is.na(m)) i=1
    else if(m>=0.0 && m<=0.20) i=2
    else if(m>=0.21 && m<=0.40) i=3
    else if(m>=0.41 && m<=0.60) i=4
    else if(m>=0.61 && m<=0.80) i=5
    else i=6
  }

df_total$cate=factor(mapply(f,df_total$vv),levels=1:6,labels=c("NR","VLR"
,"LR","MR","HR","VHR"))
```

This part sets up the month_name data frame to map month numbers to names. It then reads a CSV file containing outbreak data into df_total. A function f is defined to categorize the risk levels into six groups based on the vv values. These categories are then applied to the df_total data frame.

```
#Plot Directory Creation and Shapefile Loading
plot_dir=paste(paste(month_name[month_number,2],"
",current_year,"/",sep=""),month_name[month_number,2]," ",current_year,"
N",sep="")
dir.create(path = plot_dir)
```

```
disease = c(8,10,11,12,31,35,37,48,60,65,70,72,79,146,189)

# India district shapefile
kar1=readOGR(dsn = "shapefile",verbose = FALSE)
names(kar1)[1]<-"DISTRICT"
names(kar1)[2]<-"ST_NM"
names(kar1)[3]<-"ST_CEN_CD"
names(kar1)[4]<-"DT_CEN_CD"
kar1@data
st=unique(as.character(kar1$ST_NM))
```

This section creates a directory for storing plots and loads the shapefile of Indian states and districts. The shapefile is renamed to match the expected column names, and the unique state names are extracted from the data.

#Disease and State Loop for Plot Generation

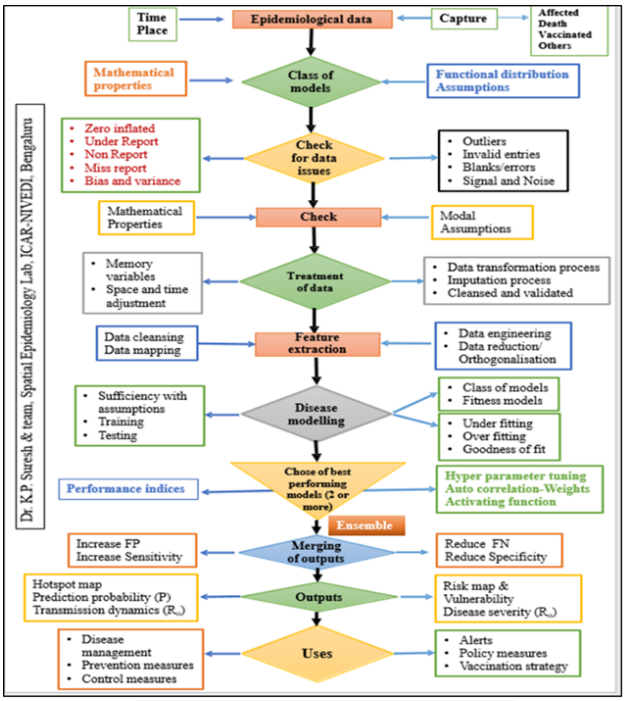
```
i=j=1
i=1
while(i<=length(disease)){
for (j in 1:length(st)) {
kar=kar1[kar1$ST_NM==st[j],]

cols=as.character(unique(df_total[df_total$disease_id==disease[i],"diseas
e_name"]))
df_disease=df_total[which(df_total$month==month_name[month_number,1] &
df_total$disease_id==disease[i]),]
df_disease=df_disease[,c("state_id","state_name","district_id","district_
name","disease_id","vv")]
df_disease=setNames(df_disease,c("ST_CEN_CD","state_name","DT_CEN_CD","di
strict_name","disease_id","vv"))
kar@data=join(data.frame(kar@data),data.frame(df_disease),by=c("ST_CEN_CD
","DT_CEN_CD"),type="left",match="first")
colours<-c("white","#FFFF00","#FFC1C1","#FF7150","#FF8500","darkred")
kar$vv[is.na(kar$vv)]<-0
kar$lb=factor(mapply(f,kar$vv),levels=1:6,labels=c("No Risk / No
Data","Very Low Risk","Low Risk","Medium Risk","High Risk","Very High
Risk"))

cols=gsub("&", "and",cols)
disname= gsub("\\."," ",cols)
  if(disname=="Enterotoxaemia")
      {
        disname="Enterotoxaemia"
      }
      cat("Plot for disease:",disname,st[j],"\n")
      plot_loc=paste(plot_dir,"/",sep="")
```

```
file_name=paste(plot_loc,disname,"_",st[j],"_",month_name[month_number,2]
,".png",sep="")
plot_title= paste(st[j]," Risk Prediction of ",disname," for the month of
",month_name[month_number,2]," ",current_year," ",sep="")
png(file_name,width = 6, height = 4, units = 'in',res = 200)
print(spplot(obj = kar,c("lb"),col.regions=colours,main =
list(plot_title,cex=0.8),key.space=list(x=0.2,y=0.9,corner=c(0,1))
                    ,sp.layout = list("sp.text",
coordinates(kar),kar$district_name,cex=0.5),scales=list(draw = TRUE)))
dev.off()
    }
    i=i+1}}
```

The function iterates through each disease and each state. For each combination, it filters and merges disease data with spatial data, assigns risk levels, and generates a plot using spplot. Each plot is saved as a PNG file in the previously created directory. The color scheme is defined, and the risk levels are labelled.

```
nadres_func_state_maps(current_year,year_number,month_number)
```

Finally, the function nadres_func_state_maps is called with the specified parameters to execute the plotting process.



Disease Modelling Processing

Models Trained in NADRES (V_2)

Step-4: Watermarking for maps created at the district and state levels

After generating all district- and state-level maps, it is crucial to incorporate a watermark to ensure the authenticity and ownership of the visualizations. Specifically, the inclusion of the NIVEDI logo on each map before publication on the NADRES website is essential. This watermarking process not only helps in branding and protecting intellectual property but also maintains the integrity of the maps by clearly identifying the source. By embedding the logo in the maps, the final output is safeguarded against unauthorized use, and the association with the National Institute of Veterinary Epidemiology and Disease Informatics (NIVEDI) is prominently displayed, reinforcing credibility and transparency in the presented data.

I. National level maps (visual, state-by-state)

```
library(magick)
im=list.files(path="maps/",pattern = ".png",full.names = T,recursive = F)
img1=image_read('logo.png')
i=1
for (i in 1:length(im)) {
img=image_read(im[i])
img2=image_composite(img, image_scale(img1, "x120"), offset = "+470+300")

image_write(image = img2,paste0("mapsnew/",i,".png"))

}
```

The provided R code snippet utilizes the magick library to watermark a series of map images with a logo. It begins by listing all PNG files in the maps/ directory and reading a logo image (logo.png). For each map image, the code reads the image, overlays the logo at a specific position with a scaled size, and then saves the watermarked image to a new directory (mapsnew/). This process ensures that each map is branded with the logo, enhancing ownership and preventing unauthorized use. The watermark is positioned at a defined offset to avoid obscuring important map details.

II. Watermarking for District wise maps

The script leverages the magick package in R to systematically add a watermark, such as a logo, to a series of district-wise map images for 15 different diseases across various states. It begins by defining the file paths for both the map images and the watermark. For each map image, the script reads the image and the watermark using the image_read() function. It then resizes the watermark to a specified width using image_resize() to maintain consistency across all images. The watermark is overlaid onto each map image with the image_composite() function, where the position of the watermark is adjusted using specified offsets to ensure optimal placement and visibility. This step is repeated for each disease and state, ensuring that the watermark is appropriately positioned and does not obstruct important map details. Finally, the watermarked images are saved to a designated output directory with filenames reflecting their respective states and diseases. This automated approach allows for efficient processing and branding of a large number of map images, ensuring uniformity and professionalism across the dataset.

```
# install.packages("magick")
library(magick)
img1=image_read('LOGO')
im=list.files(path="State maps/",pattern =
glob2rx("*_ANDAMAN*.png"),full.names = T,recursive = F)
im
i=1
# for (i in 1:length(im)) {

for (i in 1:length(im)) {
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+400+500")
  image_write(image = img2,paste0("1/",basename(im[i])))
}

img1=image_read('LOGO.png')
im=list.files(path="State maps/",pattern =
glob2rx("*_ANDHRA*.png"),full.names = T,recursive = F)
i=1
img=image_read(im[i])
#img2=image_composite(img, image_scale(img1, "x100"), offset =
"+400+500")
#img2
#img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+700")
#img2
img2=image_composite(img, image_scale(img1, "x100"), offset = "+400+500")
img2
for (i in 1:length(im)) {
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+400+500")
  img2
  image_write(image = img2,paste0("1/",basename(im[i])))
}

im=list.files(path="State maps/",pattern =
glob2rx("*_ARUNACHAL*.png"),full.names = T,recursive = F)
i=1
img=image_read(im[i])
img2=image_composite(img, image_scale(img1, "x100"), offset = "+600+500")
img2
# img2=image_composite(img, image_scale(img1, "x100"), offset =
"+800+500")
# img2
```

```
# img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+500")
# img2
for (i in 1:length(im)) {
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+500")
  img2
  image_write(image = img2,paste0("1/",basename(im[i])))
}

img1=image_read('LOGO.png')
im=list.files(path="State maps/",pattern =
glob2rx("*_ASSAM*.png"),full.names = T,recursive = F)
i=1
img=image_read(im[i])
img2=image_composite(img, image_scale(img1, "x100"), offset = "+450+500")
img2
for (i in 1:length(im)) {
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+500")
  img2
  image_write(image = img2,paste0("1/",basename(im[i])))
}

img1=image_read('LOGO.png')
im=list.files(path="State maps/",pattern =
glob2rx("*_BIHAR*.png"),full.names = T,recursive = F)
i=1
img=image_read(im[i])
img2=image_composite(img, image_scale(img1, "x100"), offset = "+650+500")
img2
{for (i in 1:length(im)) {
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+500")
  img2
  image_write(image = img2,paste0("1/",basename(im[i])))
}

  im=list.files(path="State maps/",pattern =
glob2rx("*_CHANDIGARH*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
```

```
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+500")
  # img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+800+500")
  # img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+900+500")
  # img2
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_CHHATTISGARH*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+750+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_DADRA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
```

```
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_DAMAN*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+350")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+500")
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+500")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+350")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_GOA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+500")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
```

```
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_GUJARAT*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_HARYANA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+400+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+400+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_HIMACHAL*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
```

```
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_JAMMU*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_JHARKHAND*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+700+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+700+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_KAR*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }
```

```
  im=list.files(path="State maps/",pattern =
glob2rx("*_KER*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+800+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_LAKSHAD*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_MADHYA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
```

```
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_MAHARA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+350+550")
  img2

  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+350+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }


  im=list.files(path="State maps/",pattern =
glob2rx("*_MANIPUR*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+575+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+575+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_MEGHALA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
```

```
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+400")
  # img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+500")
  # img2
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+750+480")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+750+480")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_MIZORA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+480")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+800+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_NAGALAND*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
  img2
  for (i in 1:length(im)) {
```

```
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+450+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_DEL*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }


  im=list.files(path="State maps/",pattern =
glob2rx("*_ODISHA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+450")
  img2
  # img2=image_composite(img, image_scale(img1, "x100"), offset =
"+850+550")
  # img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+450")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_PUDUC*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
```

```
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_PUNJAB*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_RAJAS*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }


  im=list.files(path="State maps/",pattern =
glob2rx("*_SIKKIM*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
```

```
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }


  im=list.files(path="State maps/",pattern =
glob2rx("*_TAMIL*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }


  im=list.files(path="State maps/",pattern =
glob2rx("*_TELANGANA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+500+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_TRIPURA*.png"),full.names = T,recursive = F)
  i=1
```

```
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+500+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+550+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_UTTAR*.png"),full.names = T,recursive = F)
  im=list.files(path="State maps/",pattern = glob2rx("*_UTTAR
PRA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+650+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_UTTARA*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+625+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+625+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_WEST*.png"),full.names = T,recursive = F)
```

```
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+550")
    img2
    image_write(image = img2,paste0("1/",basename(im[i])))
  }

  im=list.files(path="State maps/",pattern =
glob2rx("*_LADAKH*.png"),full.names = T,recursive = F)
  i=1
  img=image_read(im[i])
  img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+500")
  img2
  for (i in 1:length(im)) {
    img=image_read(im[i])
    img2=image_composite(img, image_scale(img1, "x100"), offset =
"+600+500")
    img2
    image_write(image = img2,paste0("1/",basename(im[i]))) }}
```

Step-5: Data file preparation for warnings and alerts uploading to the website

To obtain warnings and alerts information: Based on the risk variable for a specific disease, warnings will be issued to districts identified as Very High and High Risk, indicating a high likelihood of disease outbreak. Conversely, alerts will be sent to districts predicted to be at Moderate Risk, signifying a lower but still notable risk of disease transmission. This approach ensures targeted responses tailored to the varying levels of risk, allowing for more effective allocation of resources and implementation of preventive measures.

#Environment Setup and Data Reading

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jdk1.8.0_201\\')
library(qdap)
vp=beg2char(Sys.Date(),"-",2) #c("2018-11")
month_number= as.numeric(substr(vp,6,7))+2  #"11"
month_number=ifelse(month_number>12,month_number-12,month_number)
mch=month.name[month_number]
year=beg2char(vp,"-")
```

This section sets up the Java environment and loads the qdap library. It calculates the month and year for the report, adjusting for month transitions if necessary.

```
#Reading Data
#d=read.csv(paste0("../NADRES_PCA_Delta_March_2024.csv"))
d=read.csv(paste0("../","upload_",mch,".csv"))
```

The script reads a CSV file corresponding to the current month, which contains the risk data.

```
#Filtering Data for Risk Levels
d1=d[d$Outcome=="High
Risk",c("month_letter","disease_name","district_name")]
d2=d[d$Outcome=="Very High
Risk",c("month_letter","disease_name","district_name")]
d3=rbind(d1,d2)
d3=d3[!duplicated(d3),]
d4=d[d$Outcome=="Medium
Risk",c("month_letter","disease_name","district_name")]
```

This segment filters the data for high, very high, and medium risk levels, removing duplicates to ensure unique entries.

```
#Generating Alerts for Medium Risk
dis=as.character(unique(d3$disease_name))
i=1
tmp1=d4
d4_h=c()
```

```
for (i in 1:length(dis)) {
  d4=tmp1[tmp1$disease_name==dis[i],]
  v=paste0("<tr class='row100 body'>
  <td class='cell100 column1 alert'>Alert</td>
    <td class='cell100 column2'>",mch,"</td>",
           "<td class='cell100
column3'>",paste0(d4$district_name,collapse = ","),"</td>",
           "<td class='cell100 column4'>",dis[i],"</td>")
  d4_h=append(d4_h,v)
}
write.table(d4_h,file = "NDR_SOL_MR.CSV",row.names = F,eol =
"</tr>",sep="\n")
tmp1=d3
d4_h=c()
for (i in 1:length(dis)) {
  d4=tmp1[tmp1$disease_name==dis[i],]
  v=paste0("<tr class='row100 body'>
           <td class='cell100 column1 warning'>Warning</td>
           <td class='cell100 column2'>",mch,"</td>",
           "<td class='cell100
column3'>",paste0(d4$district_name,collapse = ","),"</td>",
           "<td class='cell100 column4'>",dis[i],"</td>")
  d4_h=append(d4_h,v)
}
write.table(d4_h,file = "NDR_SOL_hr_vhr.CSV",row.names = F,eol =
"</tr>",sep="\n")
```

The script generates HTML-like alert strings for districts at medium risk, formatting them into a CSV file NDR_SOL_MR.CSV.

```
#Generating Warnings for High and Very High Risk
ds=d[d$Outcome==c("Medium Risk","High Risk","Very High
Risk"),c("month_letter","disease_name","state_name")]
ds=ds[!duplicated(ds),]
dis=as.character(unique(d$disease_name))
i=1
tmp1=ds
d4_h=c()
for (i in 1:length(dis)) {
  d4=tmp1[tmp1$disease_name==dis[i],]

  v=paste0(d4$state_name,collapse = ",")
  d4_h=append(d4_h,v)
}
d4_h=data.frame(dis,d4_h)
```

```
write.table(d4_h,file = "1 state_NDR_SOL_hr_vhr.CSV",row.names = F)
```

This part of the script generates a state-wise report of predicted diseases for various risk levels and saves it to 1 state_NDR_SOL_hr_vhr.CSV, providing a comprehensive overview of disease predictions across states.

Step- 6: To assess geographic correlation, used the Moran-I index

The Moran-I index is used to analyze spatial autocorrelation, which assesses the degree of similarity in values between neighbouring locations within a geographic dataset. By conducting Moran-I index analysis, we can identify spatial patterns, clusters, or spatial outliers in the data, helping us understand if there are significant spatial trends or dependencies present. This analysis aids in various fields such as urban planning, epidemiology, and environmental studies by informing decision-making processes related to resource allocation, policy formulation, and spatial targeting of interventions.

The goal of the script is to analyse the spatial distribution of disease outbreaks across various districts in India by calculating Moran's I statistic. Moran's I measures spatial autocorrelation, indicating whether high or low values (e.g., disease outbreaks) are clustered together in space. The analysis is carried out for a specific month and year range, and the results are saved in a CSV file.

```
#Data Preparation and Filtering
month_number = month number  #Predicted Month
year_number = from year
current_year = predicted year

# function Moran I --------
moran_ndr = function(month_number, year_number, current_year)

library(data.table)
library(plyr)
library(rgdal)
library(spdep) # for neighbouring list, weighted matrix
library(plyr)
library(reshape2)

disease = c(8,10,11,12,31,35,37,48,60,65,70,72,79,146,189)
df_dat = fread("dist_out_nadres_2024-03-04 11_20_54.csv",
               header = T,
               check.names = F,
               data.table = F
)
dis = df_dat[df_dat$disease_id %in% disease, c("disease_id",
"disease_name")]
dis = dis[!duplicated(dis), ]

# filter data for month nad ten years data
d = df_dat[df_dat$month == month_number &
             df_dat$year >= year_number &
             df_dat$year <= current_year &
             df_dat$number_of_outbreaks != 0, ]
```

```
d = na.omit(d)
d$state_name = toupper(d$state_name)
d = d[!d$state_name == "ANDAMAN & NICOBAR ISLANDS", ]
d[d$state_name == "ARUNANCHAL PRADESH", "state_name"] = "ARUNACHAL
PRADESH"
```

This section prepares the data by setting parameters for the month and year of interest, loading necessary libraries, and reading the data from a CSV file. It then filters the data for relevant diseases and time periods, cleans it by removing NA values and standardizing state names.

```
#Shapefile Processing and Spatial Analysis
ka1 = readOGR("~/1shp/2011_Dist.shp")
df = NULL
j = 1
for (j in 1:length(disease)) {

d2 = d[d$disease_id == disease[j],]
st = as.character(unique(d2$state_name))
i = 1
s = c() # to store state name
it = c() # to store moran I
for (i in 1:length(st)) {
ka = ka1[ka1$ST_NM == st[i], ]
as.character(unique(ka1$ST_NM))
d1 = d2[d2$state_name == st[i], ]
colnames(d1)[c(1, 4)] = c("ST_NM", "DISTRICT")
if (length(unique(d1$district_id)) > 2) {
      ka@data = join(data.frame(ka@data),
                     d1,
                     match = "first",
                     type = "left")
 #View(ka@data)
 # generate neighbouring list for district polygons
 wgt = poly2nb(ka, row.names(ka))
 # check for non neighbouring polygons
 nghb.list = unlist(lapply(wgt, sum))
 ind_non_nb = which(nghb.list == 0)
 # if non neighbouring polygons exist, remove it
 if (length(ind_non_nb) >= 1 & nrow(ka)!=length(ind_non_nb))
      {
        ka = ka[-ind_non_nb,]
        # neighbour index list
        wgt = poly2nb(ka, row.names(ka))
        # neighbour matrix as 0 or 1
        wm <- nb2mat(wgt, style = 'B', zero.policy = T)
        # weighted matrix list
```

```
      ww <-  nb2listw(wgt, style = 'B', zero.policy = T)
      # set NA to 0
      ka$number_of_attacks[is.na(ka$number_of_attacks)] = 0
      # convert to 1
      ka[ka$number_of_attacks > 0, "number_of_attacks"] = 1
      # calculate moran I
      v = moran(
        ka$number_of_attacks,
        ww,
        n = length(ww$neighbours),
        S0 = Szero(ww)
      )
      # if moran I is NULL
      if (is.nan(v$I)) {
        it = append(it, 0)
      } else{
        it = append(it, v$I)
      }
      # add state, moran I
      s = append(s, st[i])
    } else {
      wm <- nb2mat(wgt, style = 'B', zero.policy = T)
      wgt = poly2nb(ka, row.names(ka))
      wm <- nb2mat(wgt, style = 'B', zero.policy = T)
      ww <- nb2listw(wgt, style = 'B', zero.policy = T)
      ka$number_of_attacks[is.na(ka$number_of_attacks)] = 0
      ka[ka$number_of_attacks > 0, "number_of_attacks"] = 1

      # calculate moran I
      v = moran(
        ka$number_of_attacks,
        ww,
        n = length(ww$neighbours),
        S0 = Szero(ww)
      )
      # if moran I is NULL
      if (is.nan(v$I)) {
        it = append(it, 0)
      } else{
        it = append(it, v$I)
      }
      # add state, moran I
      s = append(s, st[i])
    } }   }
# when df is empty
if (j == 1) {
```

```
    df = cbind(df, s, it, disease[j])
  }    else{
    # when df is not empty, append rows
    df = rbind(df, cbind(s, it, disease[j]))
  } }
```

In this section, a shapefile (ka1) of Indian districts is loaded. An empty data frame df is created to store Moran's I values. The code loops through each disease, filters the data accordingly, and processes it by state. For each state, the shapefile data is merged with the disease data, and neighboring districts are identified using poly2nb. Moran's I is computed for states with more than two districts, and results are stored in df. Non-neighboring districts are removed if necessary, and Moran's I values are calculated and appended to df.

```
#Data Formatting and Output
temp = df
colnames(df) = c("State", "Moran_I", "disease_id")
# join disease and moran I data
df1 = join(data.frame(df), dis, match = "first", type = "left")
# remove disease id column
df1 = df1[, -1]
# rename columns
colnames(df1) = c("state", "moran_I", "disease_name")
# cast disease as columns, moran I value in cells
df_cast = dcast(df1,
                formula = as.formula("state~disease_name"),
                value.var = "moran_I")

fwrite(df_cast, paste0("Moran I_", month.name[month_number], ".csv"))
moran_ndr(month_number, year_number, current_year)
```

The final part of the script formats the computed Moran's I values into a data frame, reshaping it so that each state is a row and each disease is a column. The results are saved to a CSV file for easy access and further analysis. This output helps visualize and interpret the spatial distribution of disease outbreaks across different states.

Step- 7: Forecasting weather parameters at state level

The R code conducts a detailed time series analysis of weather parameters using ARIMA models to forecast future values based on historical data. It involves constructing various ARIMA model configurations to identify the most suitable model for each parameter, based on the Akaike Information Criterion (AIC). By converting weather data into time series objects and applying these models, the code generates forecasts for the next five years, transforming predictions back to their original scale if needed. The results are saved in CSV files and visualized through plots that compare historical data with forecasts. Additionally, the root mean squared error (RMSE) is calculated to assess forecast accuracy, providing a robust framework for understanding long-term trends and supporting decision-making processes in fields such as epidemiology.

```
#Loading Required Libraries
library (tseries)
library(forecast)
```

These libraries provide essential functions for time series analysis (tseries), forecasting with ARIMA models (forecast), and numerical analysis (pracma).

```
#Defining ARIMA Model Parameters
p=rep(c(1:3),each=3)
d=rep(c(0),length=9)
q=rep(c(1:3),length=3)
p
pdq_df=data.frame(p,d,q)
p=rep(c(1:3),each=3)
d=rep(c(1),length=9)
q=rep(c(1:3),length=3)
pdq_df1=rbind(pdq_df,cbind(p,d,q))
```

Here, different combinations of ARIMA parameters (p, d, q) are generated for both non-seasonal and seasonal components. pdq_df and pdq_df1 hold these parameter sets, enabling exploration of various ARIMA models.

```
#Reading and Preparing Data
v1= read.csv(file.choose(),header = TRUE,sep="\t")
v1=read.table("clipboard",header = TRUE,sep="\t")
cs=colnames(v1)
cs=cs[-1]
```

This code reads weather data from a file or clipboard and extracts column names, excluding the first column, which is assumed to be a time index.

```
#Model Selection and Forecasting
m=1
```

```
for (m in 1:length(cs))
{
  v2=v1[,cs[m]]
  if(cs[m]=="Rainfall")
  {
    v2=v2*86400
  }

  ts_sh=ts(data=v2,start = 2001,frequency = 12)
  aic_val=c()
  for (i in 1:nrow(pdq_df1 ))
  {
    v=pdq_df1[i,]
    m1=try(arima(log(ts_sh), order = c(as.numeric(v)),
seasonal=list(order=c(as.numeric(v)),period=12)),silent = T)
    if(class(m1)=="Arima")
    {
      aic_val=append(aic_val,m1$aic)
    } else aic_val=append(aic_val,NA)
  }
  ind_order=which(min(aic_val,na.rm = T)==aic_val)
  best=pdq_df1[ind_order,]
  m1.b=arima(log(ts_sh), order = c(as.numeric(best)),
seasonal=list(order=c(as.numeric(best)),period=12))
  p1.b=predict(m1.b,n.ahead = 5*12)
  p2.b = 2.718^p1.b$pred

  p2.df=data.frame(p2.b)
  yr=rep(2021:2025,each=12)
  mn=rep(1:12,length=60)
  year_mn_pred=data.frame(year=yr,month=mn,p2.b)
  colnames(year_mn_pred)[3]=cs[m]
  write.csv(year_mn_pred, file = paste0(cs[m],".csv"),row.names = F)

  png(filename =paste0(cs[m],".png"),width = 6,height = 4,units =
"in",res = 200)
  ts.plot(ts_sh,p2.b,lty=c(1,3),main=paste0("Forecasting of ",cs[m],"
till 2025"),col=c("#008080","#000080"),lwd=3)
  dev.off()
}
```

This section iterates over each weather parameter, adjusts units if necessary, and creates a time series object. It fits ARIMA models using the predefined parameters, selects the model with the lowest AIC, and uses it to forecast the next 5 years. The forecasts are transformed back from the log scale to the original scale, saved to a CSV file, and plotted for visualization.

```
m1
ind_order
best
p2.b
library (pracma)
t=rmserr(p2.b, p1.b$pred,summary=T)
```

The root mean squared error (RMSE) is calculated to evaluate the accuracy of the forecasted values compared to the predicted values, providing a measure of forecast reliability.

The provided R code systematically applies ARIMA models to historical weather data to identify the best forecasting model based on AIC. It generates forecasts for the next five years, saves the results, and visualizes them with plots. The accuracy of these forecasts is assessed using RMSE, ensuring reliable and actionable insights into weather trends.

Step- 8: Finding Significant Weather Parameters for livestock Diseases

The provided R code performs a statistical analysis of weather-related variables across different diseases using Linear Discriminant Analysis (LDA) and Analysis of Variance (ANOVA). The objective is to identify which weather parameters significantly affect the classification of different diseases. This process involves standardizing the data, fitting LDA models to classify diseases based on weather parameters, and conducting ANOVA to assess the significance of these parameters. The results are summarized and saved for further interpretation.

#Loading Libraries and Reading Data

```
library(MASS)
library(data.table)
df=read.csv("inputfile.csv")
```

The code loads the MASS library for LDA functions and the data.table library for efficient data manipulation. It then reads the dataset from a CSV file, which includes weather and disease data.

#Handling Missing Values in dataframe

```
df[is.na(df)] <- 0
```

This line replaces all missing values in the dataframe with zero, ensuring that subsequent analyses are not affected by missing data.

```
#Preparing for Analysis
# Get unique disease values
diseases <- unique(df$disease_name)

# Create a list to store results for each disease
results_list <- list()

# Loop through each disease
for (disease in diseases) {
  subset_df <- subset(df, disease_name == disease)
  variables_to_scale <- subset_df[, c("PET", "Vapour_pressure")]
  scaled_variables <- scale(variables_to_scale)

  # Replace the original variables with the scaled ones in the dataframe
  subset_df[, c("PET", "Vapour_pressure")] <- scaled_variables
  cols <- names(subset_df)
  formula <- as.formula(paste("out ~", paste(cols[-1], collapse = " +
")))

  # LDA function call
  lda_result <- lda(formula, data = subset_df)

  # ANOVA function call
```

```
  anova_result <- aov(formula, data = subset_df)
  anova_summary <- data.frame(anova(anova_result))

# Save the results for each disease
  results_list[[disease]] <- list(lda = lda_result, anova =
anova_summary)
}
```

For each disease, the code subsets the data and standardizes the selected weather variables ("PET" and "Vapour_pressure") to ensure comparability. It constructs a formula for LDA and ANOVA based on all variables in the dataset, then performs LDA and ANOVA. The results are stored in a list for each disease, including LDA outputs and ANOVA summaries.

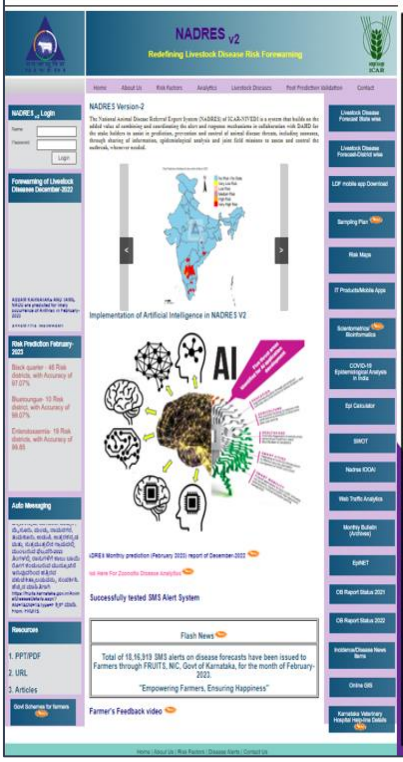
```
# Create a data.table to store the summary
summary_dt <- data.table(Disease = character(),
                         Parameter = character(),
                         P_Value = numeric())

# Loop through each disease and parameter to extract relevant information
for (disease in diseases) {

anova_result <- results_list[[disease]]$anova
# Extract parameters with p-values < 0.05
significant_parameters <- anova_result[anova_result$"Pr..F." < 0.05, ,
drop = FALSE]
if (!is.null(significant_parameters)) {
summary_dt <- rbindlist(l=list(summary_dt,
              data.table(Disease = rep(disease,
nrow(significant_parameters)),
Parameter = rownames(significant_parameters),
P_Value = significant_parameters$"Pr..F.")),
use.names = TRUE, fill = TRUE)
  }
}
```

The code creates an empty data.table to store significant ANOVA results. It iterates through each disease's ANOVA results to extract parameters with p-values less than 0.05, indicating statistical significance. These significant results are aggregated into a summary table.

```
# Save the summary data.table to a CSV file
fwrite(summary_dt, "ANOVA_Summary.csv", row.names = FALSE)
```

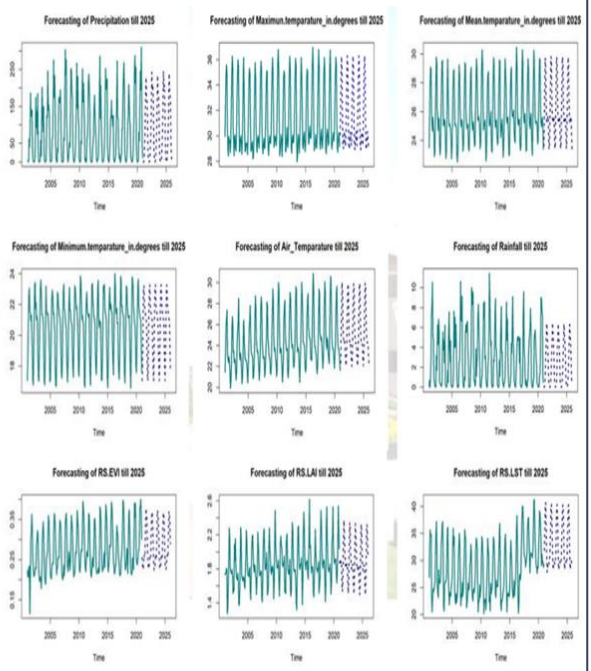
Finally, the summary table containing significant ANOVA results is saved to a CSV file for further review and analysis.

NADRESv2 Home page

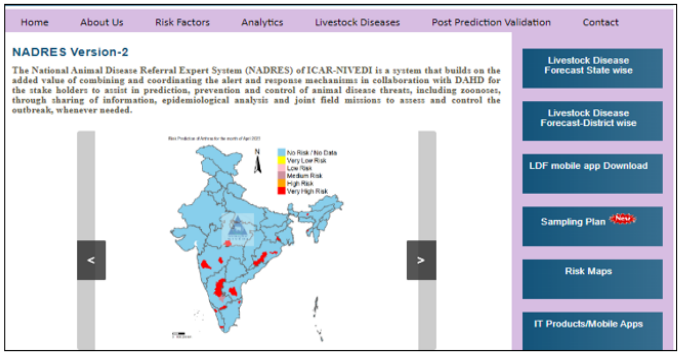


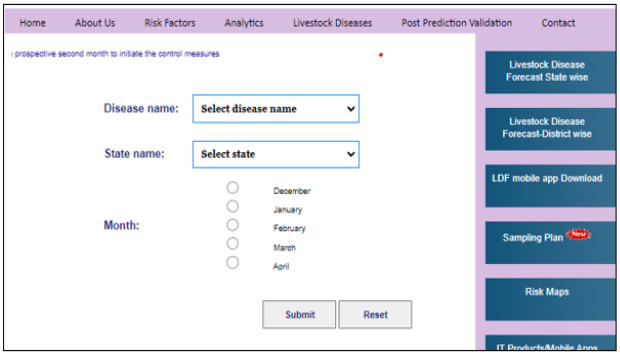
NADRESv2 Monthly Bulletin Cover page

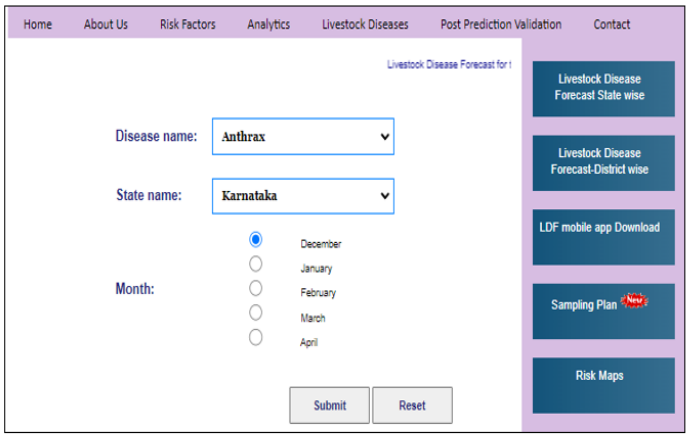
Forecasting of the Weather parameters

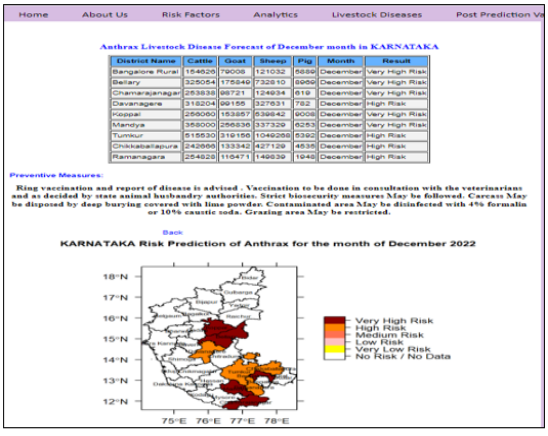


Functionality of Livestock Disease Forecast State wise tab



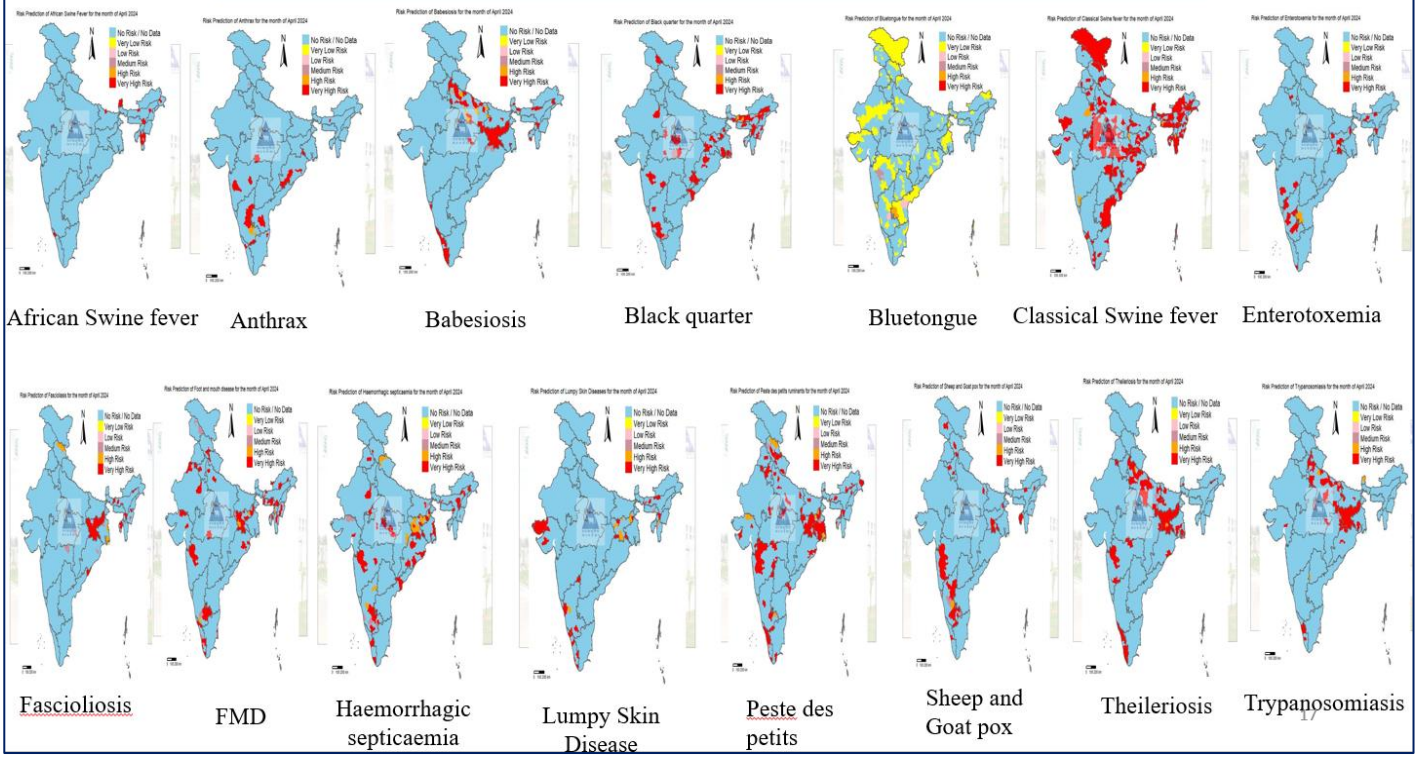






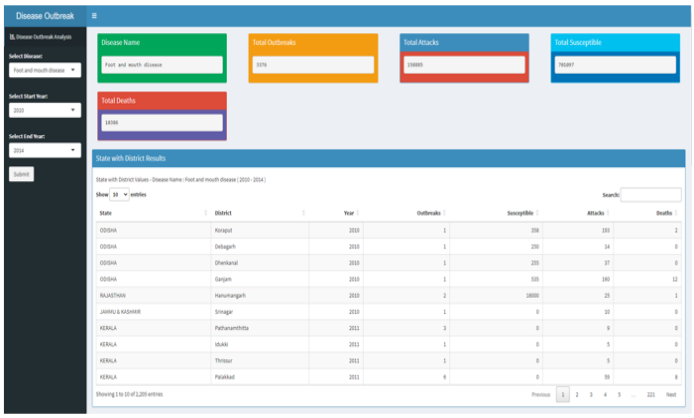
https://www.nivedi.res.in/Nadres_v2/Risk_Map_download.php

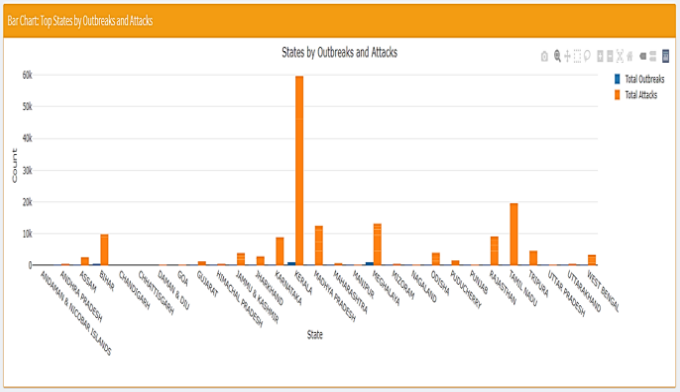
Six level Graded Risk Prediction - Livestock Disease Forewarning Maps (April-2024)

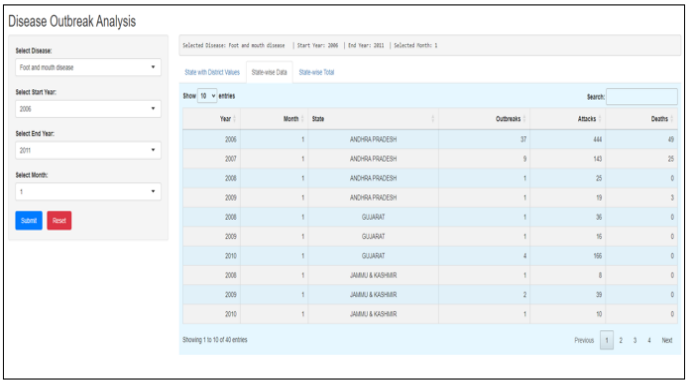


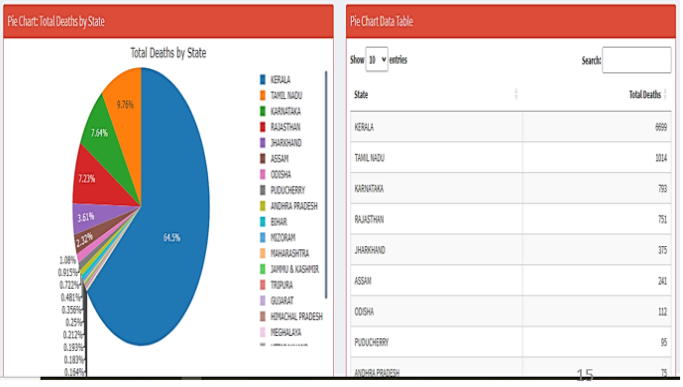
17

Epidemiological Visualization Page









11. DATA COMMUNICATION

I. Email

Effective data communication is crucial for disseminating important information to stakeholders. Each month, we send NADRES_V2 Bulletin and methodology reports between 650 to 700 emails to principals and co-principal investigators of NADEN Centres, Central and State Veterinary Departments, and other relevant stakeholders. These emails are designed to provide timely updates, reports, and critical insights regarding livestock health and disease management. By maintaining regular communication, we ensure that all parties are informed and can take appropriate actions based on the latest data and forecasts.

II. NADRES Website

The NADRES website serves as a central hub for distributing crucial information related to livestock disease forewarning. Every month, we upload the Livestock Disease Forewarning Bulletin and Livestock Disease Forewarning Methodology to ensure that users have access to the most current and accurate data. The website has recently achieved a significant milestone, attracting over 2 million visitors, reflecting its importance and the trust placed in it by the veterinary community and other stakeholders. This online platform enhances accessibility and ensures that valuable information is readily available to those who need it.

III. Mobile Applications

Leveraging mobile technology, we have developed applications to disseminate livestock disease information more effectively. These mobile applications provide real-time updates, alerts, and comprehensive data on livestock health and disease forecasts. By making this information accessible via smartphones, we empower stakeholders, including farmers and veterinary professionals, to make informed decisions promptly. This mobile-centric approach ensures wider reach and engagement, particularly in remote and rural areas were access to traditional

Communication channels might be limited.LDF - Mobile App

Animal Management	Animal Health	Animal Breeding
1. Animal Registration	Animal Treatment & Surgery	Artificial Insemination
2. Owner Registration	Vaccination	Pregnancy Diagnosis
3. Ownership transfer	Deworming	Calving
4. Ear Tag Change	First Aid	ET-Embryo Master
5. Search and Modify Owner	Diseases Testing	ET-Animal Synchronization
6. Search and Modify Animal	Diseases Reporting & Intimation Report	ET-Heat Transaction
7. Latest Tag	Disease Reporting FIR (First Incidence Report)	Bull Master
	Diseases Reporting-Outbreak follow-up	Semen Straw Management
	Post Mortem	
	Campaign Creation	

https://play.google.com/store/apps/details?id=info.androidhive.ldf&hl=en&gl=US&pli=1

Large Language Models

- Farmers can simply voice their queries about diseases, vaccination schedules, or other concerns, and receive accurate and contextually relevant responses instantly
- LLMs empower farmers to proactively manage disease prevention and control by offering timely guidance on identification, prevention strategies, and vaccination protocols

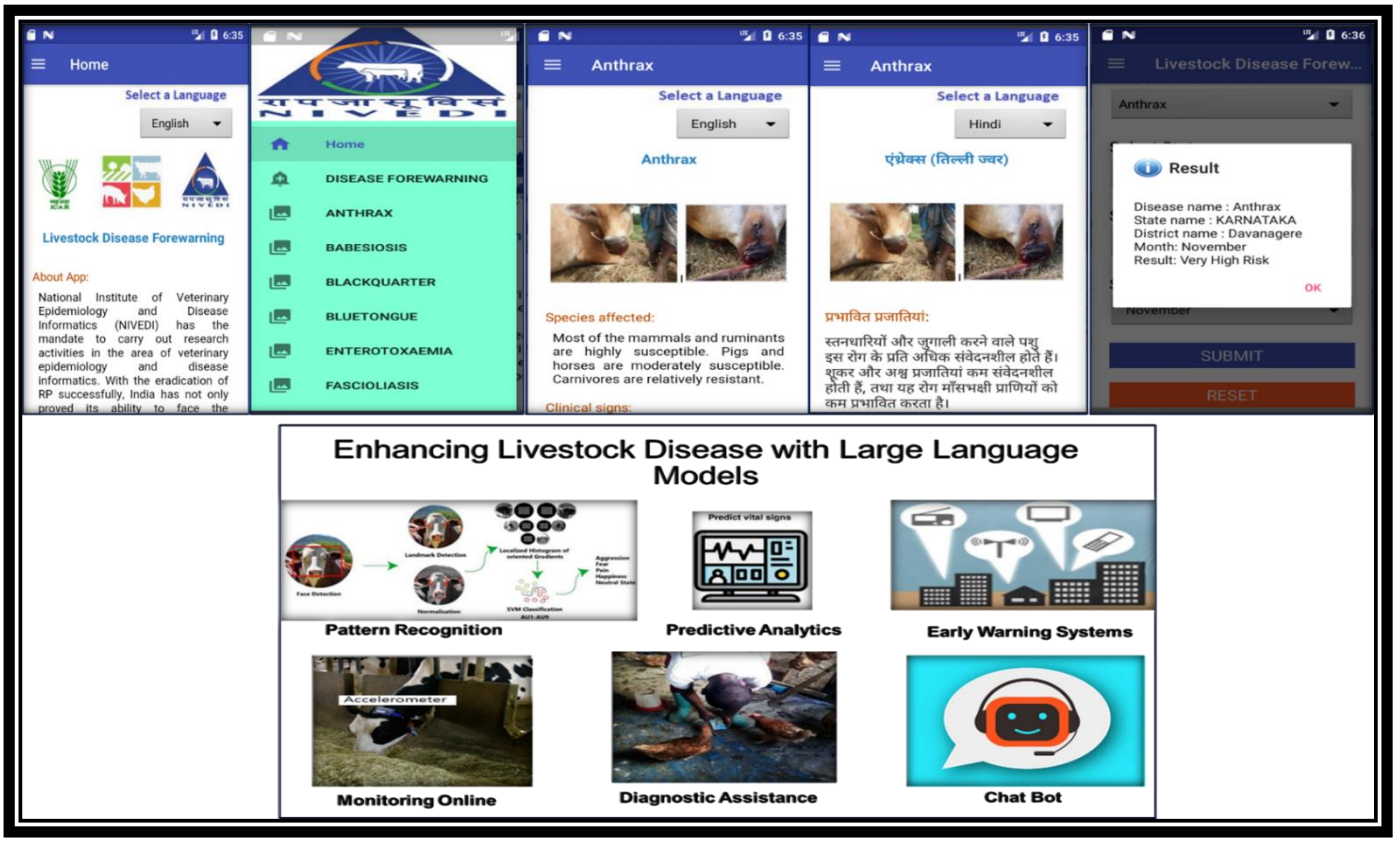


Fig. 11.1. Scientific Integration of Mobile Applications in Livestock Disease Risk Communication

IV. Disease Risk Communication (in collaboration with FRUITS, NIC, and the Government of Karnataka)

In addition to NADRES V2 (The National Animal Disease Referral Expert System), ICAR-NIVEDI collaborated with NIC, Govt. of Karnataka, Karnataka State for sending the SMS alerts directly to the farmers who have registered in FRUITS (Farmers Registration and Unified Beneficiary Information System). The information alerts on risk prediction of livestock diseases were sent through SMS to farmers is presented in Table 11.A. During April 2023 to March 2024, a total of 1,22,31,000 SMS alerts were sent to farmers.

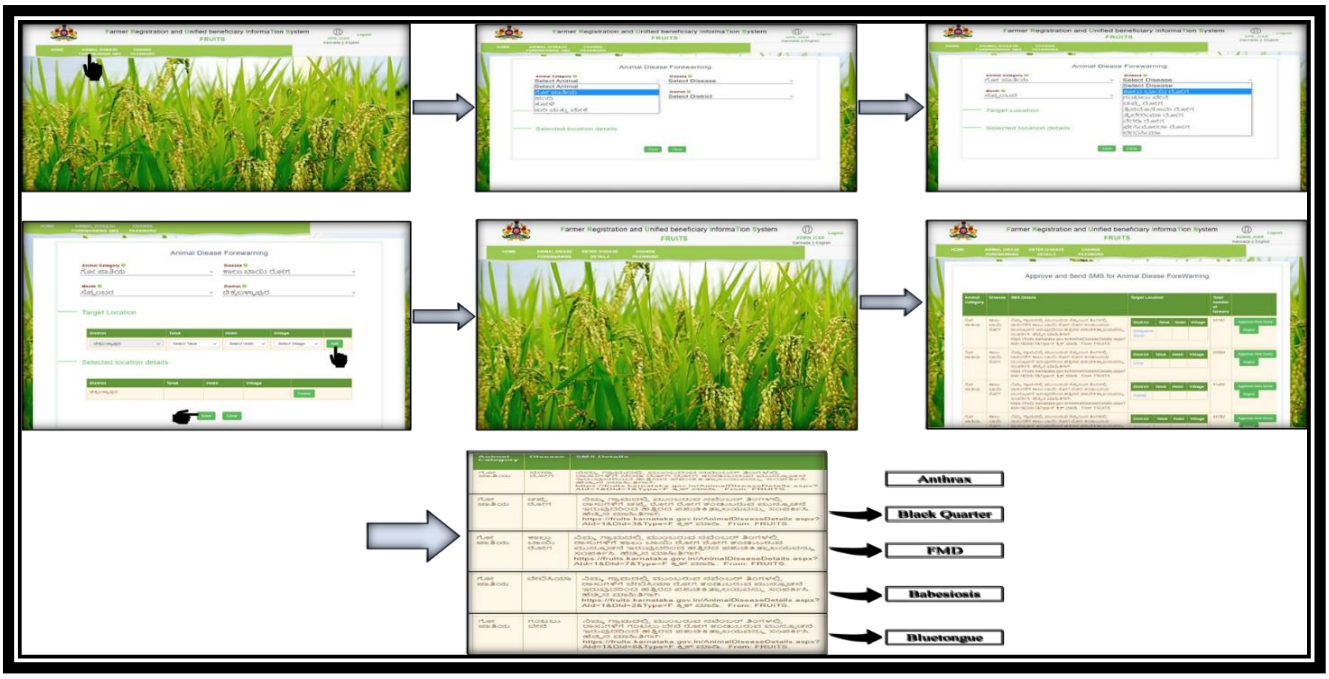


Fig. 11.2. FRUITS Sending Website and Operational Steps

- Farmers in Karnataka were informed of disease risks via SMS notifications.
- A total of 1,22,31,000 SMS notifications were sent between April 2023 to March 2024.

- The notifications covered various animal diseases such as Anthrax, Babesiosis, Black Quarter, Fasciolosis, FMD, Theileriosis, and Trypanosomiasis.
- The SMS notifications were facilitated through the IT, FRUITS a web program of NIC, Government of Karnataka.

Table 11.A. Farmer Registration and Unified beneficiary Information System (FRUITS)

Diseases	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	TOTAL
Anthrax	203184	363983	293834	283765	291256	245002	219650	195055	230841	229575	268477	213959	3038581
Babesiosis	0	0	0	0	0	40011	0	0	0	0	0	0	40011
Black Quarter	308651	130195	491263	267539	126444	297786	201648	542452	344133	346614	314296	350142	3721163
Fascioliasis	25329	251838	360817	30821	149141	0	0	0	0	0	0	0	817946
Foot and Mouth Diseases	671157	0	0	0	0	538519	266820	372625	546878	889240	456273	281573	4023085
Theileriosis	63093	0	0	0	211368	104522	0	0	0	0	0	0	378983
Trypanosomosis	0	0	0	0	207661	0	0	3570	0	0	0	0	211231
Grand Total	12,71,414	7,46,016	11,45,914	5,82,125	9,85,870	12,25,840	6,88,118	11,13,702	11,21,852	14,65,429	10,39,046	8,45,674	1,22,31,000

V. DLT provision

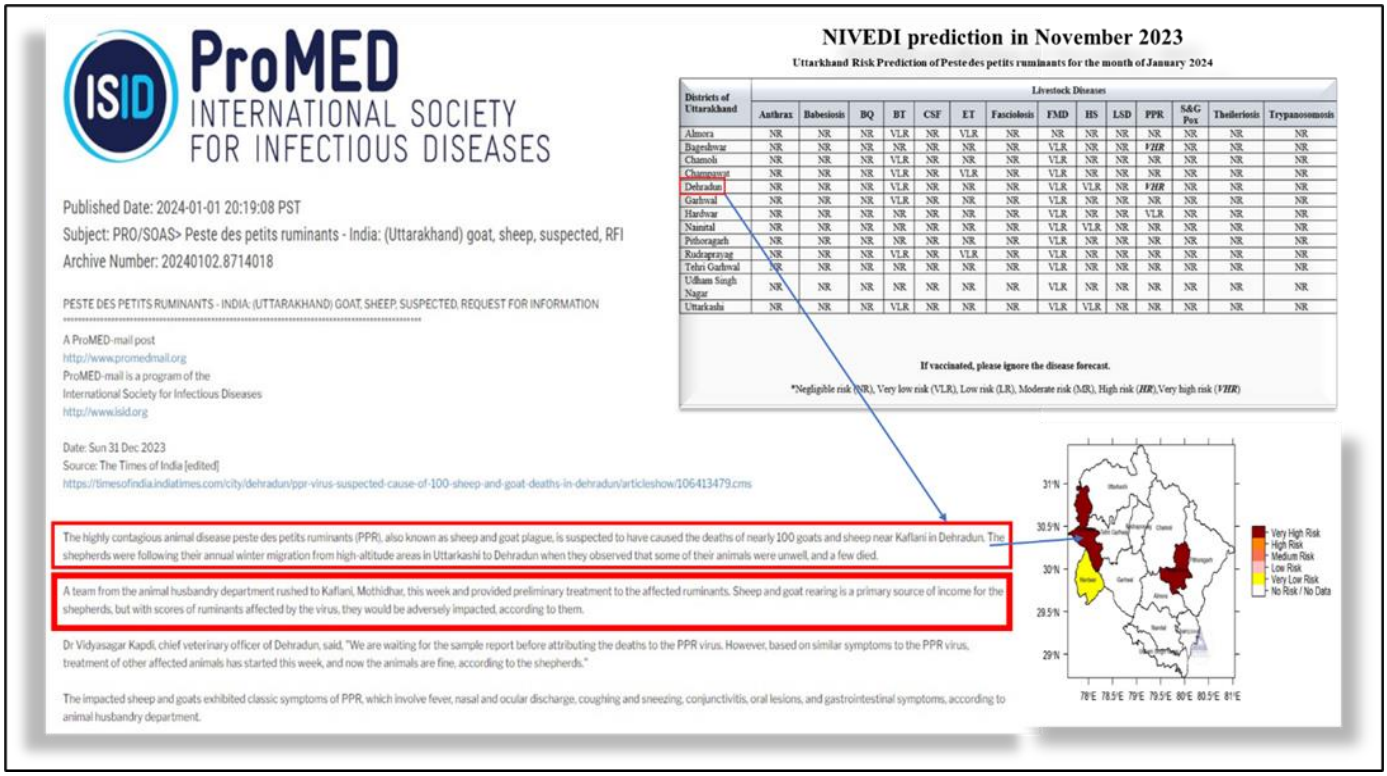
Distributed Ledger Technology (DLT) has been implemented to optimize communication processes, enhancing the efficiency of disseminating information and delivering updates related to livestock disease risk management.

VI. Post-prediction validation

Post-prediction validation is a crucial process in evaluating the accuracy and reliability of predictive models. This involves comparing the model's forecasts against actual outcomes using various sources such as scientific publications and ProMED reports. By systematically validating predictions, researchers can identify discrepancies, refine models, and enhance their predictive performance for future applications.

I. PPR reported in the month of January 2024 in Dehradun district of Uttarakhand

January 2024 ProMED report on Peste des Petits Ruminants (PPR) livestock disease aligns with our November 2023 report of predicting high-risk for January 2024, enhancing forecast accuracy and underscoring the imperative for proactive disease prevention measures.



ProMED
INTERNATIONAL SOCIETY FOR INFECTIOUS DISEASES

Published Date: 2024-01-01 20:19:08 PST
Subject: PRO/SOAS> Peste des petits ruminants - India: (Uttarakhand) goat, sheep, suspected, RFI
Archive Number: 20240102.8714018

PESTE DES PETITS RUMINANTS - INDIA: (UTTARAKHAND) GOAT, SHEEP, SUSPECTED, REQUEST FOR INFORMATION

A ProMED-mail post
http://www.promedmail.org
ProMED-mail is a program of the
International Society for Infectious Diseases
http://www.isid.org

Date: Sun 31 Dec 2023
Source: The Times of India [edited]
https://timesofindia.indiatimes.com/city/dehradun/ppr-virus-suspected-cause-of-100-sheep-and-goat-deaths-in-dehradun/articleshow/106413479.cms

The highly contagious animal disease peste des petits ruminants (PPR), also known as sheep and goat plague, is suspected to have caused the deaths of nearly 100 goats and sheep near Kaflani in Dehradun. The shepherds were following their annual winter migration from high-altitude areas in Uttarkashi to Dehradun when they observed that some of their animals were unwell, and a few died.

A team from the animal husbandry department rushed to Kaflani, Mothidhar, this week and provided preliminary treatment to the affected ruminants. Sheep and goat rearing is a primary source of income for the shepherds, but with scores of ruminants affected by the virus, they would be adversely impacted, according to them.

Dr Vidyasagar Kapdi, chief veterinary officer of Dehradun, said, "We are waiting for the sample report before attributing the deaths to the PPR virus. However, based on similar symptoms to the PPR virus, treatment of other affected animals has started this week, and now the animals are fine, according to the shepherds."

The impacted sheep and goats exhibited classic symptoms of PPR, which involve fever, nasal and ocular discharge, coughing and sneezing, conjunctivitis, oral lesions, and gastrointestinal symptoms, according to animal husbandry department.

NIVEDI prediction in November 2023

Uttarkhand Risk Prediction of Peste des petits ruminants for the month of January 2024

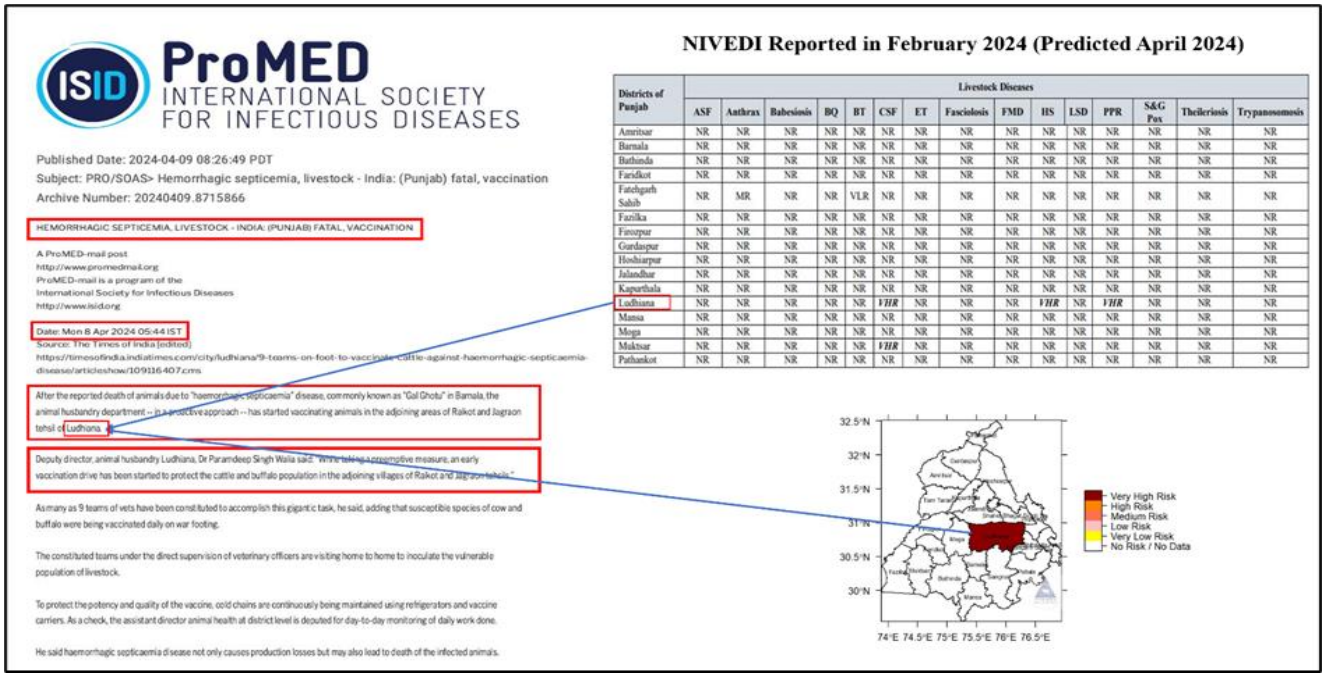
Districts of Uttarakhand	Livestock Diseases													
	Anthrax	Babesiosis	BQ	BT	CSF	ET	Fasciolosis	FMD	HS	LSD	PPR	S&G Pox	Theileriosis	Trypanosomosis
Almora	NR	NR	NR	VLR	NR	VLR	NR	NR	NR	NR	NR	NR	NR	NR
Bageshwar	NR	NR	NR	NR	NR	NR	NR	VLR	NR	NR	*VHR*	NR	NR	NR
Chamoli	NR	NR	NR	VLR	NR	NR	NR	VLR	NR	NR	NR	NR	NR	NR
Champawat	NR	NR	NR	VLR	NR	VLR	NR	VLR	NR	NR	NR	NR	NR	NR
Dehradun	NR	NR	NR	VLR	NR	NR	NR	VLR	VLR	NR	*VHR*	NR	NR	NR
Garhwal	NR	NR	NR	VLR	NR	NR	NR	VLR	NR	NR	NR	NR	NR	NR
Haridwar	NR	NR	NR	NR	NR	NR	NR	VLR	NR	NR	VLR	NR	NR	NR
Nainital	NR	NR	NR	NR	NR	NR	NR	VLR	VLR	NR	NR	NR	NR	NR
Pithoragarh	NR	NR	NR	NR	NR	NR	NR	VLR	NR	NR	NR	NR	NR	NR
Rudraprayag	NR	NR	NR	VLR	NR	VLR	NR	VLR	NR	NR	NR	NR	NR	NR
Tehri Garhwal	NR	NR	NR	NR	NR	NR	NR	VLR	NR	NR	NR	NR	NR	NR
Udham Singh Nagar	NR	NR	NR	NR	NR	NR	NR	VLR	NR	NR	NR	NR	NR	NR
Uttarkashi	NR	NR	NR	VLR	NR	NR	NR	VLR	VLR	NR	NR	NR	NR	NR

If vaccinated, please ignore the disease forecast.

*Negligible risk (NR), Very low risk (VLR), Low risk (LR), Moderate risk (MR), High risk (*HR*),Very high risk (*VHR*)

II. Hemorrhagic Septicemia Reported in Ludhiana District, Punjab, in April 2024

The April 2024 ProMED report on Hemorrhagic Septicemia (HS) in livestock in Ludhiana District, Punjab, corroborates our February 2024 forecast predicting a high risk of HS for April 2024. This alignment between observed disease incidence and our predictive model enhances the accuracy of our forecasts and underscores the critical need for proactive disease prevention measures.



ProMED
INTERNATIONAL SOCIETY FOR INFECTIOUS DISEASES

Published Date: 2024-04-09 08:26:49 PDT
Subject: PRO/SOAS> Hemorrhagic septicemia, livestock - India: (Punjab) fatal, vaccination
Archive Number: 20240409.8715866

HEMORRHAGIC SEPTICEMIA, LIVESTOCK - INDIA: (PUNJAB) FATAL, VACCINATION

A ProMED-mail post
http://www.promedmail.org
ProMED-mail is a program of the
International Society for Infectious Diseases
http://www.isid.org

Date: Mon 8 Apr 2024 05:44 IST
Source: The Times of India [edited]
https://timesofindia.indiatimes.com/city/ludhiana/9-teams-on-foot-to-vaccinate-cattle-against-haemorrhagic-septicaemia-disease/articleshow/109116407.cms

After the reported death of animals due to "haemorrhagic septicaemia" disease, commonly known as "Gal Ghotu" in Barnala, the animal husbandry department -- in a proactive approach -- has started vaccinating animals in the adjoining areas of Raikot and Jagraon tehsil of Ludhiana.

Deputy director, animal husbandry Ludhiana, Dr Paramdeep Singh Walia said, "While taking a preventive measure, an early vaccination drive has been started to protect the cattle and buffalo population in the adjoining villages of Raikot and Jagraon tehsils."

As many as 9 teams of vets have been constituted to accomplish this gigantic task, he said, adding that susceptible species of cow and buffalo were being vaccinated daily on war footing.

The constituted teams under the direct supervision of veterinary officers are visiting home to home to inoculate the vulnerable population of livestock.

To protect the potency and quality of the vaccine, cold chains are continuously being maintained using refrigerators and vaccine carriers. As a check, the assistant director animal health at district level is deputed for day-to-day monitoring of daily work done.

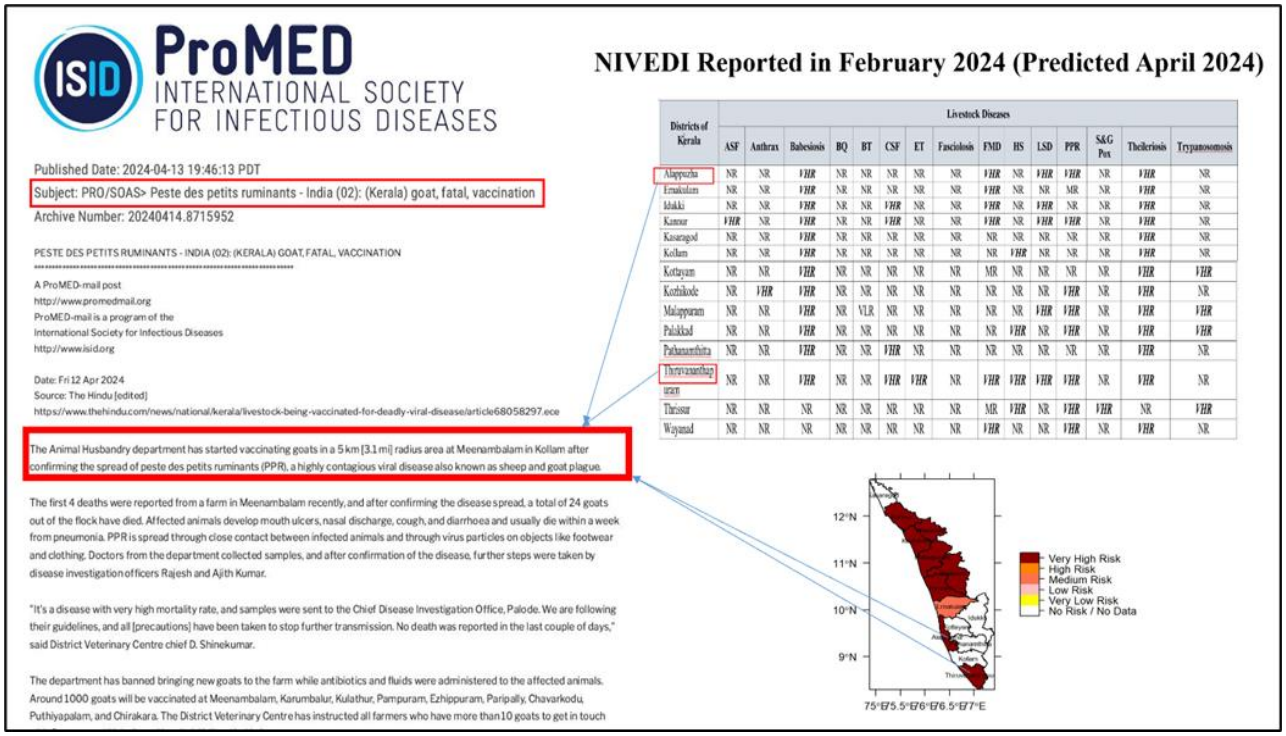
He said haemorrhagic septicaemia disease not only causes production losses but may also lead to death of the infected animals.

NIVEDI Reported in February 2024 (Predicted April 2024)

Districts of Punjab	Livestock Diseases														
	ASF	Anthrax	Babesiosis	BQ	BT	CSF	ET	Fasciolosis	FMD	HS	LSD	PPR	S&G Pox	Theileriosis	Trypanosomosis
Amritsar	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Barnala	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Bathinda	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Faridkot	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Fatehgarh Sahib	NR	MR	NR	NR	VLR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Fazilka	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Firozpur	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Gurdaspur	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Hoshiarpur	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Jalandhar	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Kapurthala	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Ludhiana	NR	NR	NR	NR	NR	*VHR*	NR	NR	NR	*VHR*	NR	*VHR*	NR	NR	NR
Mansa	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Moga	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Muktsar	NR	NR	NR	NR	NR	*VHR*	NR	NR	NR	NR	NR	NR	NR	NR	NR
Pathankot	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR

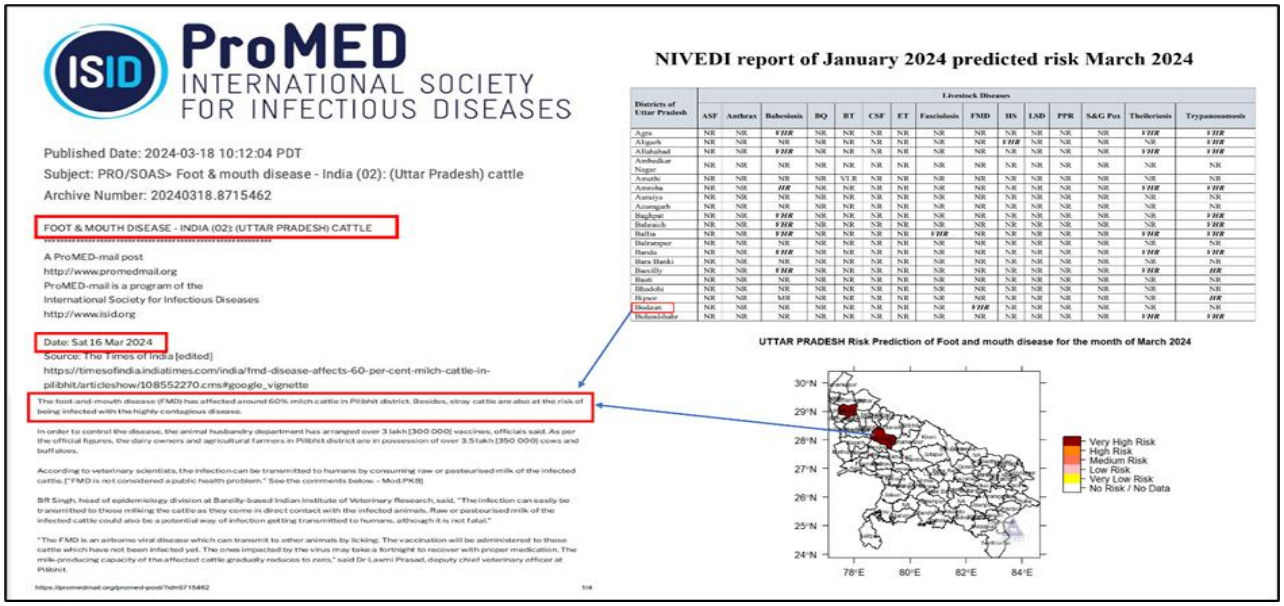
III. PPR reported in the month of April 2024 in Kollam district of Kerala

The April 2024 ProMED report on Peste des Petits Ruminants (PPR) in Kollam District, which is adjacent to the previously identified high-risk area, corroborates our January 2024 forecast predicting a high risk of PPR for March 2024. This alignment of observed data with our predictions enhances forecast accuracy and underscores the necessity for proactive disease prevention measures. The congruence between reported incidence and forecasted risk highlights the robustness of our predictive model and reinforces the need for timely, targeted interventions to mitigate disease spread.



IV. Foot and Mouth Disease Reported in March 2024 in Pilibhit District, Uttar Pradesh, Adjacent to Budaun District

The March 2024 ProMED report on Foot and Mouth Disease (FMD) in Pilibhit District a neighboring district to Badaun District corroborates our January 2024 high-risk prediction for March 2024. This alignment of risk forecasts underscores the accuracy of our predictive models and highlights the critical need for proactive disease prevention measures. The concurrence between the observed disease incidence and our forecasted risk emphasizes the efficacy of our forecasting methodology and reinforces the importance of timely interventions to mitigate disease spread.



APPENDIX
Abbreviations

NADRE : National Animal Disease Referral Expert System
R : R environment for statistical computing
ASF : African Swine Fever
BQ : Black Quarter
BT : Blue Tongue
CSF : Classical Swine Fever
ET : Enterotoxaemia
FS : Fasciolosis
FMD : Foot and Mouth disease
HS : Haemorrhagic Septicaemia
PPR : Peste des Petits Ruminants
SGP : Sheep and Goat pox
hPa : Hectopascals
NR : No risk/ No data available
VLR : Very low risk
LR : Low risk
MR : Moderate risk
HR : High risk
VHR : Very high risk

Reference:

1. Suresh, K. P., Dhemadri, D., Kurli, R., Dheeraj, R., & Roy, P. (2019). Application of Artificial Intelligence for livestock disease prediction. *Indian Farming*, *69*(3).

2. Suresh, K. P., Sengupta, P. P., Jacob, S. S., Sathyanarayana, M. K. G., Patil, S. S., Swarnkar, C. P., & Singh, D. (2022). Exploration of machine learning models to predict the environmental and remote sensing risk factors of haemonchosis in sheep flocks of Rajasthan, India. *Acta Tropica*, *233*, 106542.

3. Suresh, K. P., Bylaiah, S., Patil, S., Kumar, M., Indrabalan, U. B., Panduranga, B. A., & Amachawadi, R. G. (2022). A new methodology to comprehend the effect of El Niño and La Niña oscillation in early warning of anthrax epidemic among livestock. *Zoonotic Diseases*, *2*(4), 267-290.

4. Jayashree, A., Suresh, K. P., Dikshitha, J., Gulati , B. R., Balamurugan , V., Jacob , S. S., Patil , S. S., & Hemadri , D. (2024). Exploring the Impact of Climate Variables on Livestock Anthrax Outbreaks: A Machine Learning Approach. International Journal of Environment and Climate Change, 14(3), 494–507. https://doi.org/10.9734/ijecc/2024/v14i34059.

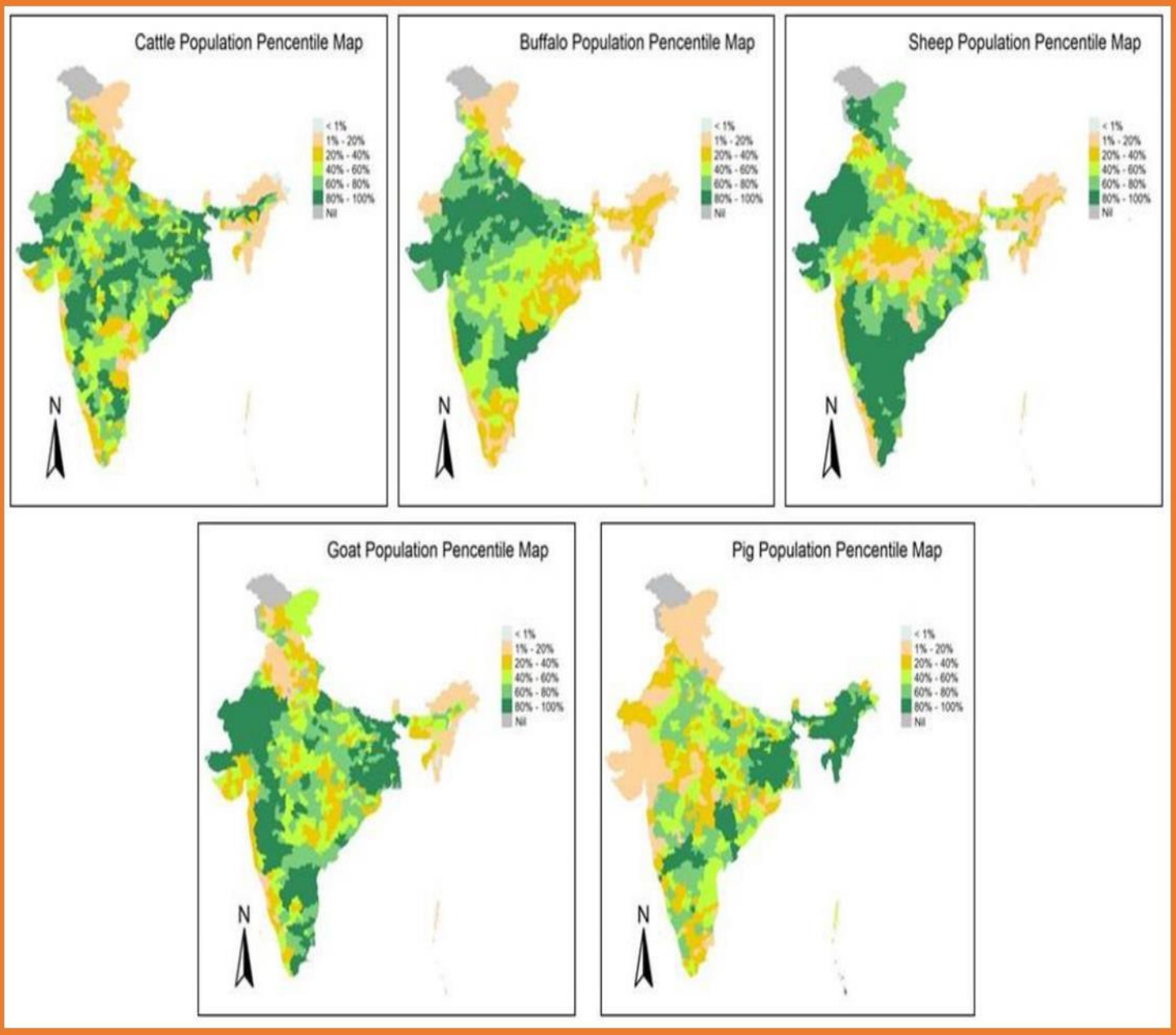
5. Sagar N, Suresh KP, Ramanji RS, Bharath M, Naveesh YB, Ravichandra and Archana CA. Revolutionizing potato crop management: Deep learning-driven potato disease detection with convolutional neural networks. International Journal of Advanced Biochemistry Research 2024; 8(3): 644-653.

6. Jayashree, A., Suresh, K. P., & Raaga, R. (2024). Advancing Coffee Leaf Rust Disease Management: A Deep Learning Approach for Accurate Detection and Classification Using Convolutional Neural Networks. Journal of Experimental Agriculture International, 46(2), 108–118. https://doi.org/10.9734/jeai/2024/v46i22313.

7. KRISHNAMOORTHY, P. ., DHARSHAN, H. V. ., CHANDRASEKHAR, T. M. ., & SURESH, K. P. (2023). Development of Cattle Disease Diagnosis Expert System (CaDDES): A web application for the diagnosis of cattle diseases. The Indian Journal of Animal Sciences, 93(12), 1180–1186. https://doi.org/10.56093/ijans.v93i12.135480.

8. Suresh, K.P., Barman, N.N., Bari, T. et al. Application of machine learning models for risk estimation and risk prediction of classical swine fever in Assam, India. VirusDis. (2023). https://doi.org/10.1007/s13337-023-00847-6.

9. Suresh KP, Sagar N, Jayashree A, Naveesh Y B, Hemadri D, S S Patil, Ramesh Doddamani and Sushma R, EPIDEMIOLOGICAL INSIGHTS INTO THE ANTHRAX OUTBREAK IN MUDDABALLI VILLAGE, KARNATAKA: A CASE REPORT, European Journal of Biomedical and Pharmaceutical sciences, 2023, Volume 10, Issue 11, 112-115.



ICAR-National Institute of Veterinary Epidemiology and Disease Informatics (ICAR-NIVEDI)

P. B. No.6450, Yelahanka, Bengaluru-560064

Phone: +91-80-23093111, Fax: +91-80-23093222, E-mail: director.nivedi@icar.gov.in