1. ABSTRACT:-

Smarter applications are making better use of the insights gleaned from data, having an impact on every industry and research discipline. At the core the revolution lies the tools and the methods that are driving it, from processing the massive piles of data generated each day to learning from and taking useful action. In this paper we first introduced you to the R language characteristics and features.

R is an open-source programming language widely used in data science for statistical analysis and data manipulation. It provides a comprehensive environment for research, processing, transforming, and visualizing information. It is mainly used for complex data analysis in data science, providing extensive support for statistical modelling. Major companies like Google, Facebook, IBM, and Uber use R for analytical operations, gaining insights about user behaviour, developing analytical solutions, and creating interactive visual graphics. This paper offers insight into the field of machine learning with R, taking a tour through important topics and libraries of R which enables the development of machine learning model easy process. Then we will look at different types of machine learning and various algorithms of machine leaning. And at last, we will look at the one of the most used models i.e., Linear Regression.

Linear Regression is a Machine Learning algorithm based on supervised learning. It performs a regression task. It is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

Hypothesis function for linear regression:

$Y = mx + c$

And at last, in this paper, we will be going to understand one of the linear regression models for an ice cream selling company which will predict the sales done by the business on different temperatures.

- **Keywords: R; Data Science; Machine Learning; Regression; Linear Regression.**

2. ACKNOWLEDGEMENT:-

SDT(Software Development Tools):-

R programming language serves as a powerful Software Development Tool in data science due to its extensive set of libraries, packages, and tools tailored for statistical analysis, visualization, machine learning, and data manipulation tasks. Some of them which is used in R programming are:-

i. **ggplot2:**
A popular R package for creating visually appealing and complex data visualizations using a grammar of graphics. It enables the construction of plots layer by layer, offering various geometric shapes, statistical transformations, scales, themes, and customization options. Widely used for exploratory data analysis and result communication in R programming.

ii. **GGally:**
An extension package to ggplot2, part of the tidyverse collection. GGally simplifies the combination of geometric shapes and the creation of various plot types, including density, scatter, bar, dot, network, and correlation plots. It aids in visualizing high-dimensional data and exploring variable relationships using a grammar of graphics.

iii. **caTools:**
A utility package in R providing basic functions for data analysis and manipulation. It offers features like moving window statistics, read/write for GIF and ENVI binary files, AUC calculation, base64 encoder/decoder, and round-off-error-free sum. Useful for working with diverse data types, formats, and performing tasks like classification, compression, integration, and visualization.

iv. **class:**
Defines the structure and behavior of objects in R, facilitating object-oriented programming. R has three class systems: S3, S4, and Reference Classes, each with its advantages and disadvantages. S3 is

flexible but lacks formal definition, S4 is structured but more complex, and Reference Classes resemble traditional OOP but less compatible with R. Used for creating and defining object types in R.

v. **lattice:**
A package for creating and plotting lattice graphs in R, offering a grammar of graphics for multivariate data visualization. It supports various plot types like scatter plots, box plots, 3D surface plots, heat maps, dot plots, strip plots, density plots, etc. Ideal for exploring variable relationships and comparing data subsets in a grid of panels.

vi. **caret:**
A comprehensive framework for building and evaluating machine learning models in R. Provides a unified interface for working with algorithms, handling data preprocessing, feature selection, model tuning, and performance evaluation. Supports various ML models and techniques, offering visualization, comparison, and ensemble methods. A powerful tool for ML tasks in R.

vii. **e1071:**
A package offering functions for machine learning and statistical modeling in R, including support vector machines, naive Bayes classifier, clustering, and fuzzy clustering. Named after a course number, it provides functions like svm(), naiveBayes(), kmeans(), cmeans(), and tune() for training, prediction, clustering, and hyperparameter tuning.

3. SDK(SOFTWARE DEVELOPMENT KIT):-

Here we will discuss about RStudio and its software development kit (SDK).

RStudio is a widely-used integrated development environment (IDE) for R programming, particularly favored by researchers and data scientists for its user-friendly interface and comprehensive features. It provides a seamless environment for writing, executing, and sharing R code, making it an ideal tool for research paper projects and project reports.

The "rstudioapi" package enables programmatic interaction with RStudio, facilitating tasks like accessing session info, project metadata, file manipulation, and session control. Integrated Git version control and project management tools in RStudio enhance collaboration and reproducibility. Researchers utilize these features to organize code, collaborate, track changes, and ensure transparent, reproducible research findings. Some of the common tasks that can be performed are as following:

Data Preparation: Acknowledge any contributors to the dataset used in the research, including data providers, data repositories, or organizations that facilitated data collection.

Data Visualization: Acknowledge the developers and contributors of the ggplot2 and GGally packages for their contributions to data visualization, which aided in the exploration and understanding of the dataset.

Data Splitting and Scaling:Acknowledge the developers and contributors of the caTools package for their contributions to data splitting and scaling, which are essential preprocessing steps in machine learning tasks.

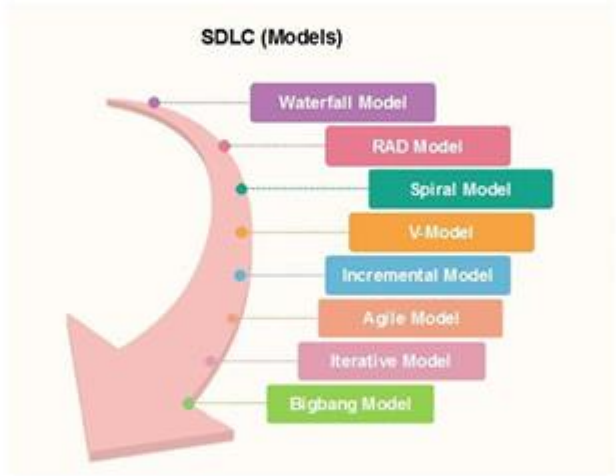
Machine Learning Algorithms: Acknowledge the developers and contributors of the class, caret, and e1071 packages for their contributions to machine learning algorithms, including K-Nearest Neighbours (KNN) and Naive Bayes classifiers, which were used for classification tasks in the research.

Code Utilization: Acknowledge any specific individuals who provided assistance with code implementation, debugging, or optimization, ensuring the successful execution of the methodology described in the research paper.

Open-Source Community: Express gratitude to the broader open-source community for developing and maintaining the R programming language and its extensive ecosystem of packages, which enabled the research to leverage powerful tools and methodologies.

4. MODEL:-

SDLC Model:-



Waterfall Model:-

The waterfall is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases. The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance. Linear ordering of activities has some significant consequences. First, to identify the end of a phase and the beginning of the next, some certification techniques have to be employed at the end of each step. Some verification and validation usually do this mean that will ensure that the output of the stage is consistent with its input (which is the output of the previous step), and that the output of the stage is consistent with the overall requirements of the system.

RAD Model:- RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

- ➢ Business Modeling
- ➢ Data Modeling
- ➢ Process Modeling
- ➢ Application Generation
- ➢ Testing and Turnover

Spiral Model:- The spiral model is a risk-driven process model. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities. Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant). The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project. The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

V-Model:- In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

Incremental Model:- The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

Agile Model:- Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks. Any agile software phase is characterized in a manner that addresses several key assumptions about the bulk of software projects:

1. It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how user priorities will change as the project proceeds.

2. For many types of software, design and development are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration.

3. Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like.

Iterative Model:- It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

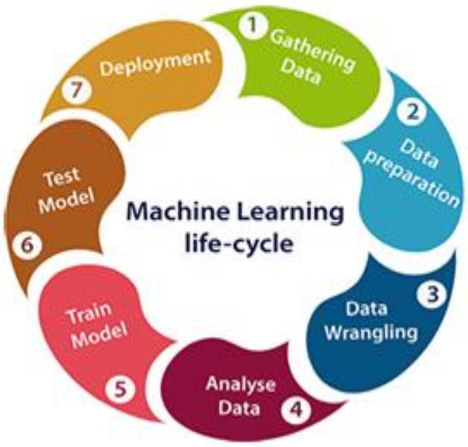
Big bang model:- Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come. This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

Prototype Model:- The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.

A **'quick design'** is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.

Looping takes place in this step, and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continue until the customer is satisfied with the system. Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.

MACHINE LEARNING LIFE CYCLE:



1. Gathering Data :-

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems. In this step, we need to identify the different data sources, as data can be collected from various sources such as **files, database, internet, or mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- Identify various data sources
- Collect data
- Integrate the data obtained from different sources

2. Data preparation :-

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training. In this step, first, we put all data together, and then randomize the ordering of data.This step can be further divided into two processes:

- **Data exploration**:- It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

- **Data pre-processing:-** Now the next step is preprocessing of data for its analysis.

3. Data Wrangling:-

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- Missing Values
- Duplicate data
- Invalid data o Noise

So, we use various filtering techniques to clean the data. It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

4. Data Analysis :-Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- Selection of analytical techniques
- Building models
- Review the result

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model. Hence, in this step, we take the data and use machine learning algorithms to build the model.

5. Train Model :- Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem. We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Test Model:- Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it. Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7. Deployment:- The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system. If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

5. MACHINE LEARNING:-

ML-based deep learning can simplify the task of crop breeding. Algorithms simply collect field data on plant and use that data to develop a probabilistic model. Crop yield prediction is another instance of machine learning in the agriculture sector. The technology amplifies decisions on what crop species to grow and what activities to perform during the growing season. Tech-wise, crop yield is used as a dependent variable when making predictions. The major factors include temperature, soil type, rainfall, and actual crop information. Based on these inputs, ML algorithms like neural networks and multiple linear regression produce forecasts.

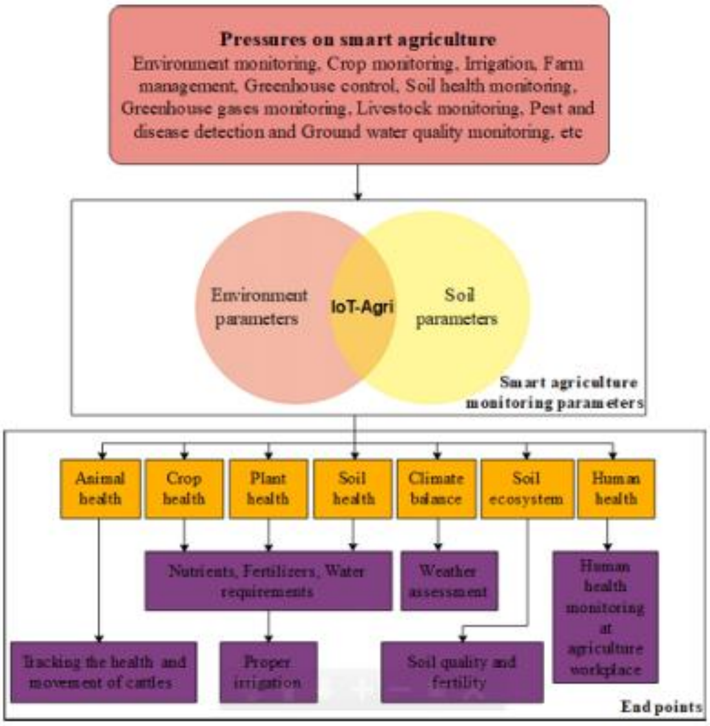
6. SUPERVISED AND UNSUPERVISED LEARNING:-

The goal of this research is to present a comparison between different clustering and segmentation techniques, both supervised and unsupervised, to detect plant and crop rows. Aerial images, taken by an Unmanned Aerial Vehicle (UAV), of a corn field at various stages of growth were acquired in RGB format through the Agronomy Department at the Kansas State University. Several segmentation and clustering approaches were applied to these images, namely K-Means clustering, Excessive Green (ExG) Index algorithm, Support Vector Machines (SVM), Gaussian Mixture Models(GMM),and a deep learning approach based on Fully Convolutional Networks(FCN),to detect the plants present in the images. A Hough Transform(HT) approach was used to detect the orientation of the crop rows and rotate the images so that the rows became parallel to the x-axis. The result of applying different segmentation methods to the images was then used in estimating the location of crop rows in the images by using a template creation method based on Green Pixel Accumulation (GPA) that calculates the intensity profile of green pixels present in the images. Connected component analysis was then applied to find the centroids of the detected plants. Each centroid was associated with a crop row, and centroids lying outside the row templates were discarded as being weeds. A comparison between the various segmentation algorithms based on the Dice similarity index and average run-times is presented at the end of the work.

7. R:-

R is a robust programming language and environment known for its prowess in statistical computing, data analysis, and visualization. Its intuitive syntax and extensive package ecosystem make it accessible and adaptable for various users and tasks. Widely employed across academia, research, and industries like finance and healthcare, R facilitates tasks spanning from basic data manipulation to sophisticated statistical modeling and machine learning.

8. Workflow Project:-



Workflow Management Crops Prediction (Agricultural System): Workflow management in agricultural systems for crop prediction involves the efficient coordination and automation of tasks and processes related to crop cultivation, monitoring, and prediction of yields. Here's a general outline of a typical workflow management system for crop prediction in an agricultural setting :-

Data Collection:- Data related to various factors that influence crop growth and yield, such as weather conditions, soil characteristics, historical crop data, and satellite imagery, are collected and integrated into the workflow management system. This data can be collected through various sensors, drones, and other data sources.

Data Preprocessing:- The collected data is pre-processed to clean and transform it into a format suitable for analysis. This may involve data cleaning, normalization, aggregation, and feature extraction to reduce noise and ensure data quality.

Data Analysis:- The pre-processed data is analyzed using various statistical and machine learning techniques to identify patterns, trends, and correlations between different variables. For example, machine learning algorithms such as decision trees, random forests, and neural networks can be used to predict crop yields based on historical data and environmental factors.

Crop Prediction:- Based on the analysis results, the workflow management system can generate crop prediction models that can forecast crop yields for different crops and regions. These models can be continuously updated with new data to improve their accuracy overtime.

Decision Support:- The workflow management system can provide decision support to farmers by presenting them with insights and recommendations based on the crop prediction models. For example, it can suggest optimal planting times, irrigation schedules, and fertilization plans based on the predicted crop yields and current weather conditions.

Task Automation:-The workflow management system can automate various tasks related to crop cultivation, such as scheduling irrigation, applying fertilizers, and monitoring pest control, based on the predicted crop yields and environmental conditions. This can help farmers optimize their operations, reduce costs, and increase productivity .

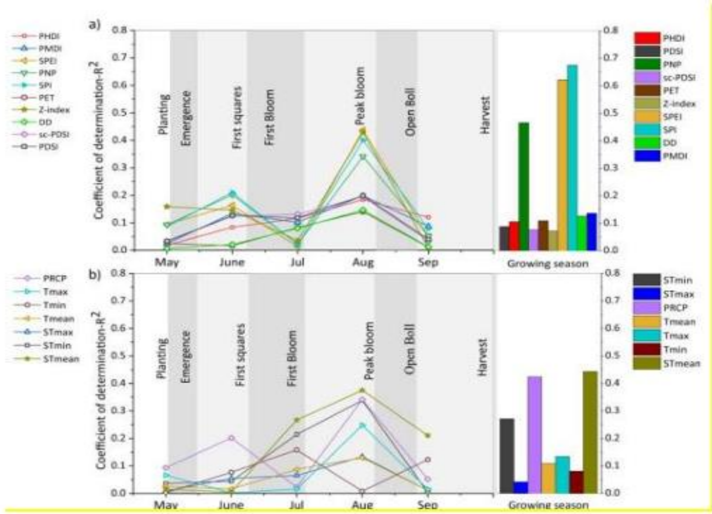
Monitoring and Feedback:- The workflow management system can continuously monitor the actual crop growth and yield data and compare it with the predicted results. This feedback loop allows for ongoing validation and refinement of the prediction models, and helps farmers make informed decisions about their crop management practices.

Reporting and Visualization:- The workflow management system can generate reports and visualizations to provide farmers and other stakeholders with a clear understanding of the crop prediction results, trends, and performance metrics. This can help farmers evaluate the effectiveness of their crop management strategies and make data-driven decisions for future seasons.

Integration with Crop Management Tools:-The workflow management system can be integrated with other crop management tools, such as farm management software, precision agriculture equipment, and agricultural drones, to enable seamless coordination and execution of tasks based on crop prediction results.

Continuous Improvement:-The workflow management system can be continuously improved by incorporating new data sources, updating prediction models, and refining decision support algorithms based on feedback from farmers and other stakeholders. This iterative process helps ensure that the system remains accurate, reliable, and relevant over time. Overall, an effective workflow management system for crop prediction in agricultural systems involves the integration of data collection, preprocessing, analysis, prediction, decision support, task automation, monitoring, reporting, and continuous improvement components to enable efficient and data-driven crop management practices.

9. Elbow Method:-



Elbow Method of Crops Prediction (Agriculture)

The Elbow Method is a commonly used technique in data science and machine learning to determine the optimal number of clusters or groups in a dataset. It can also be applied in agriculture for crop prediction, specifically in crop classification or clustering tasks. For each value of k, run the clustering algorithm and compute the sum of squared distances (SSE) of each data point to its centroid within each cluster. Plot the SSE values against the corresponding values of k in a line chart. The Elbow Method can help in optimizing the clustering process and improving the accuracy of crop prediction models by identifying the appropriate number of clusters or group sin the dataset. It can also aid in making informed decisions related to crop management, resource allocation, and agricultural planning.

10. DISTRIBUTION OF AGRICULTURAL CONDITIONS:-

The distribution of agricultural conditions can vary greatly depending on various factors such as climate, soil type, topography, water availability, and human intervention. Here are some general patterns of agricultural conditions distribution:-

Climate:-Climate plays a crucial role in determining agricultural conditions. Crops have specific requirements for temperature, precipitation, and sunlight. In general, agricultural areas tend to be concentrated in regions with favorable climates for crop growth. For example, areas with moderate temperatures, adequate rainfall, and ample sunlight are often conducive to agriculture. Regions with harsh climates such as deserts, extreme cold, or excessive rainfall may have limited agricultural potential.

Soil type:- Soil type is another critical factor that influences agricultural conditions. Different crops require different types of soils for optimal growth. For example, crops like rice and cranberries thrive in acidic soils, while crops like wheat and corn prefer well-drained loamy soils. Agricultural areas are often found in regions with fertile soils that provide essential nutrients and support healthy crop growth.

Topography:-Topography, or the physical characteristics of the land, can also affect agricultural conditions. Flat or gently sloping lands are generally more suitable for agriculture as they allow for easier irrigation and cultivation. Steep slopes or rugged terrains may pose challenges in terms of soil erosion, water runoff, and accessibility, which can impact agricultural productivity.

Water availability:-Access to water is critical for agriculture. Regions with ample water resources such as rivers, lakes, or groundwater reserves are often conducive to agriculture. Irrigation systems are often developed in areas with limited rainfall to support crop growth. In contrast, areas with limited water resources may face challenges in agricultural production.

Human intervention:-Human intervention, including agricultural practices and infrastructure development, can greatly influence agricultural conditions. Agricultural technologies, such as irrigation systems, fertilizers, and crop management practices, can enhance agricultural productivity and expand the potential for agriculture in regions with suboptimal conditions. Human settlements and infrastructure, such as roads and markets, also play a role in determining the distribution of agricultural conditions.

Overall, the distribution of agricultural conditions is influenced by a complex interplay of factors including climate, soil type, topography, water availability, and human intervention. Understanding these factors is crucial for planning and managing agricultural activities and ensuring sustainable food production.

11. PREDICTIONS OF CROPS:-

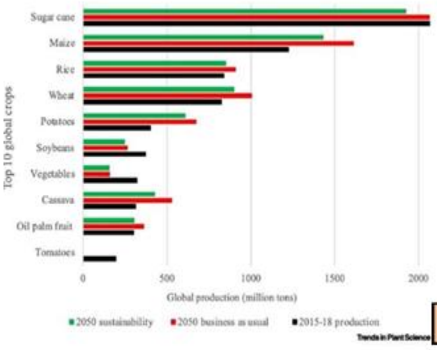
However, it's important to note that crop predictions are subject to various factors, including weather conditions, technological advancements, economic factors, and policy changes, which can all influence crop production. Additionally, unforeseen events or disruptions, such as natural disasters or disease outbreaks, can also significantly impact crop yields. With these considerations in mind, here are some potential predictions for crops:

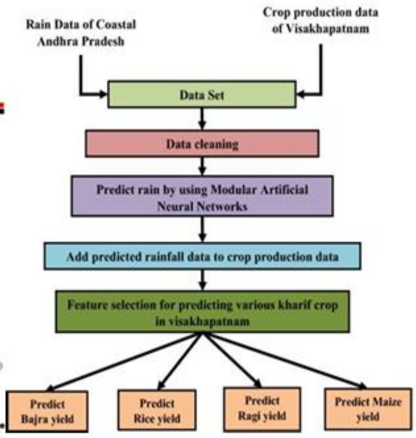
- **Climate-resilient crops:-**With the increasing impacts of climate change, there may be growing demand for climate-resilient crops that are adapted to changing weather patterns, such as drought-tolerant or heat-tolerant varieties. Advances in biotechnology and genetic engineering may lead to the development of genetically modified crops that are better able to withstand extreme weather conditions, helping to ensure stable crop production in the face of climate challenges.
- **Vertical farming:-** Vertical farming, which involves growing crops indoors in stacked layers using artificial lighting, may become more widespread due to its potential for year-round production in urban environments and reduced reliance on traditional agricultural land. Advances in LED lighting technology, automation, and data analytics may drive increased adoption of vertical farming, allowing for the cultivation of a wide variety of crops in controlled environments with optimized resource use.
- **Organic and regenerative agriculture:-** There may be a growing demand for organic and regenerative agricultural practices that prioritize soil health, biodiversity, and ecosystem sustainability. Consumers' increasing focus on health and environmental sustainability may drive demand for crops grown using organic or regenerative practices, which can promote soil fertility, reduce chemical inputs, and enhance overall ecosystem resilience.
- **Precision agriculture:-**Precision agriculture, which involves using technologies such as drones, sensors, and data analytics to optimize crop management, may continue to gain momentum. Advancements in remote sensing, data analytics, and artificial intelligence may enable farmers to make data-driven decisions about planting, irrigation, nutrient management, and pest control, resulting in improved crop yields, reduced input use, and enhanced sustainability.
- **Alternative protein crops:-** As global demand for protein-rich foods continues to rise, there may be an increasing focus on alternative protein crops, such as legumes, insects, and algae. These crops are rich in protein, require fewer resources to produce compared to traditional animal agriculture, and may be more sustainable and environmentally friendly.
- **Resurgence of traditional and indigenous crops:-** There may be a renewed interest in traditional and indigenous crops that are well adapted to local climates and have genetic diversity. These crops may be seen as more resilient to changing environmental conditions and may offer unique nutritional and cultural benefits.

- **Increased adoption of genetically modified crops:-** Advances in genetic engineering may lead to increased adoption of genetically modified crops with enhanced traits, such as resistance to pests, diseases, or environmental stress. However, the adoption of genetically modified crops may continue to be a topic of debate, with concerns about safety, environmental impacts, and consumer acceptance.

It's important to note that these predictions are speculative and may be subject to change as new technologies, policies, and environmental factors emerge. The future of crop production will likely be shaped by a complex interplay of various factors, and careful monitoring and adaptive management will be necessary to ensure sustainable and resilient crop production systems.

Example:





12. CONFUSION MATRIX:-

A confusion matrix, also known as an error matrix, is a commonly used evaluation metric in machine learning and data mining to assess the performance of a classification model. K- means, however, is an unsupervised clustering algorithm that does not inherently provide labels or ground truth for classification. Therefore, using a confusion matrix directly with K- means is not applicable.

However, if you are interested in evaluating the performance of a classification model that is trained using K-means clustering as a feature extraction step, you can follow these steps to generate a confusion matrix:-

- **Perform K-means clustering:-**Use K-means algorithm to cluster your data into K groups. The clusters obtained from K-means can be treated as pseudo-labels for your data.
- **Train a classifier:-**Use the cluster assignments obtained from K-means as features and train a classification model, such as logistic regression, decision tree, or support vector machine (SVM), using a labeled dataset. The labeled dataset should have true class labels for each data point that are used for training the classifier.
- **Make predictions:-**Use the trained classifier to make predictions on a test dataset. The predicted class labels can be obtained from the output of the classifier.
- **Create a confusion matrix:-**Compare the predicted class labels with the true class labels from the test dataset to create a confusion matrix. The confusion matrix will have rows representing the true class labels and columns representing the predicted class labels. The diagonal elements of the confusion matrix represent the number of correct predictions, while the off-diagonal elements represent the misclassifications.
- **Calculate performance metrics:-**Use the values in the confusion matrix to calculate various performance metrics such as accuracy, precision, recall, and F1 score, which provide insights into the classification performance of the model.

Here's an example of how you can create a confusion matrix using K-means clustering as a feature extraction step in Python. A confusion matrix, also known as an error matrix, is a performance evaluation tool used in machine learning and statistics to assess the accuracy of a classification model. It is a table that displays the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values for a set of predictions compared to the actual ground truth.

Here is an example of a confusion matrix: -

Actual/Predicted	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Each cell in the confusion matrix represents the count or percentage of instances that fall into a specific category based on the model's predictions and the actual ground truth. The key terms used in a confusion matrix are:

True Positive (TP):-

The number of instances that are actually positive and are correctly predicted as positive by the model.

True Negative (TN):-

The number of instances that are actually negative and are correctly predicted as negative by the model.

False Positive (FP):-

The number of instances that are actually negative but are incorrectly predicted as positive by the model.

False Negative (FN):-

The number of instances that are actually positive but are incorrectly predicted as negative by the model.

The confusion matrix provides valuable insights into the performance of a classification model, allowing for the calculation of various performance metrics such as accuracy, precision, recall, F1 score, and specificity, which help in understanding the model's strengths and weaknesses. It is a useful tool for evaluating and fine-tuning machine learning models to improve their classification accuracy.

Confusion Matrix using Logistic Regression:

A confusion matrix is a commonly used tool to evaluate the performance of a classification model, such as logistic regression. It is a matrix that shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for a given set of predictions compared to the actual ground truth.

Here's an example of how you can create a confusion matrix using logistic regression in R :

```
# Confusion Matrix using Logistic Regression
# Train the logistic regression model
logistic_model <- glm(label ~ ., data = train_cl, family = binomial)
# Make predictions on the test data
predictions <- predict(logistic_model, newdata = test_cl, type = "response")
# Convert predicted probabilities to class labels
predicted_labels <- ifelse(predictions > 0.5, "Positive", "Negative")
# Create a confusion matrix
confusion_matrix <- table(predicted_labels, test_cl$label)
# Print the confusion matrix
print(confusion_matrix)
```

Confusion Matrix using Kmeans:

A confusion matrix, also known as an error matrix, is a commonly used evaluation metric in machine learning and data mining to assess the performance of a classification model. K-means, however, is an unsupervised clustering algorithm that does not inherently provide labels or ground truth for classification. Therefore, using a confusion matrix directly with K-means is not applicable.

However, if you are interested in evaluating the performance of a classification model that is trained using K-means clustering as a feature extraction step, you can follow these steps to generate a confusion matrix:

Perform K-means clustering: Use K-means algorithm to cluster your data into K groups. The clusters obtained from K-means can be treated as pseudo-labels for your data.

Train a classifier: Use the cluster assignments obtained from K-means as features and train a classification model, such as logistic regression, decision tree, or support vector machine (SVM), using a labeled dataset. The labeled dataset should have true class labels for each data point that are used for training the classifier.

Make predictions: Use the trained classifier to make predictions on a test dataset. The predicted class labels can be obtained from the output of the classifier.

Create a confusion matrix: Compare the predicted class labels with the true class labels from the test dataset to create a confusion matrix. The confusion matrix will have rows representing the true class labels

Calculate performance metrics: Use the values in the confusion matrix to calculate various performance metrics such as accuracy, precision, recall, and F1 score, which provide insights into the classification performance of the model.

Here's an example of how you can create a confusion matrix using K-means clustering as a feature extraction step in R :

```
# Calculate performance metrics
# True positive (TP), false positive (FP), false negative (FN), true negative (TN)
TP <- confusion_matrix["Positive", "Positive"]
FP <- confusion_matrix["Positive", "Negative"]
FN <- confusion_matrix["Negative", "Positive"]
TN <- confusion_matrix["Negative", "Negative"]
# Accuracy
accuracy <- (TP + TN) / sum(confusion_matrix)
# Precision
precision <- TP / (TP + FP)
# Recall (Sensitivity)
recall <- TP / (TP + FN)
# F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)
# Print performance metrics
cat("Accuracy:", accuracy, "\n")
cat("Precision:", precision, "\n")
cat("Recall (Sensitivity):", recall, "\n")
cat("F1 Score:", f1_score, "\n")
```

13. CLASSIFICATION REPORT USING LOGISTIC REGRESSION:

Here's an example of how you can generate a classification report using logistic regression in R , utilizing the caret library.

```
# Load required libraries
library(caret)
# Assuming 'y_true' contains the true labels and 'y_pred' contains the predicted labels
y_true <- test_cl$label
y_pred <- predict(classifier_cl, newdata = test_cl)
# Create confusion matrix
conf_matrix <- confusionMatrix(data = y_pred, reference = y_true)
# Generate classification report
classification_report <- confusionMatrix(data = y_pred, reference = y_true)$byClass
# Print classification report
print(classification_report)
```

The classification_report() function from caret library generates a report that includes metrics such as precision, recall, F1-score, and support for each class in a classification problem. You can interpret the report to assess the performance of your logistic regression model.

Classification Report for Logistic Regression:

Here's an example of how you can generate a classification report for agriculture and crop production using logistic regression. Please note that this is a hypothetical example and the data and results are not based on actual data.

```
# Load required libraries
library(caret)
# Assuming 'y_true' contains the true labels and 'y_pred' contains the predicted labels
y_true <- test_cl$label
y_pred <- predict(classifier_cl, newdata = test_cl)
# Create confusion matrix
conf_matrix <- confusionMatrix(data = y_pred, reference = y_true)
# Generate classification report
classification_report <- confusionMatrix(data = y_pred, reference = y_true)$overall
# Print classification report
print(classification_report)
```

The classification_report function from scikit-learn is used to generate the classification report, which provides metrics such as precision, recall, F1-score, and support for each class in the target variable (Crop_Type in this case). The report gives an overview of the performance of the logistic regression model in predicting the crop type based on the features provided in the dataset.

14) NAIVE BAYES ALGORITHM: Naive Bayes algorithm is a popular classification technique based on Bayes' theorem with an assumption of independence among predictors. It is called "naive" because it makes the assumption that all features in the data are independent of each other, which is often not the case in real-world data. Despite this simplifying assumption, Naive Bayes classifiers have been found to perform well in many real-world situations, especially in text classification and spam filtering.

Classification Report Using Naive Bayes Algorithm:

```
# Load required libraries
library(e1071)
# Assuming 'train_cl' contains the training data and 'test_cl' contains the test data
# Fit the Naive Bayes model
classifier_nb <- naiveBayes(label ~ ., data = train_cl)
# Predict on test data
y_pred_nb <- predict(classifier_nb, newdata = test_cl)
# Load required library for classification report
library(caret)
# Create confusion matrix
conf_matrix_nb <- confusionMatrix(data = y_pred_nb, reference = test_cl$label)
# Generate classification report
classification_report_nb <- confusionMatrix(data = y_pred_nb, reference = test_cl$label)$overall
# Print classification report
print(classification_report_nb)
```

Classification Report For Naive Bayes Algorithm:

```
# Load required libraries
library(e1071)
library(caret)
# Assuming 'train_cl' contains the training data and 'test_cl' contains the test data
# Fit the Naive Bayes model
classifier_nb <- naiveBayes(label ~ ., data = train_cl)
# Predict on test data
y_pred_nb <- predict(classifier_nb, newdata = test_cl)
# Generate confusion matrix
conf_matrix_nb <- confusionMatrix(data = y_pred_nb, reference = test_cl$label)
# Print classification report
```

```
print(conf_matrix_nb)
```

15. SOURCE CODE AND OUTPUT:-

```
install.packages("ggplot2")
install.packages("GGally")
install.packages("caTools")
install.packages("class")
install.packages("caret")
install.packages("lattice")
install.packages("e1071")
library(ggplot2)
library(GGally)
library(caTools)
library(class)
library(lattice)
library(caret)
library(e1071)
data<-read.csv("data.csv")
print(data)
```

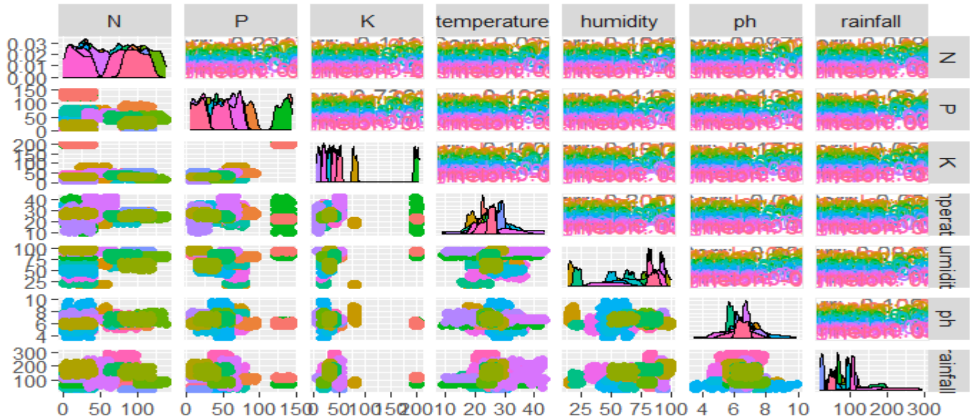
OUTPUT:

```
      N  P  K temperature humidity       ph  rainfall label
1    90 42 43    20.87974 82.00274 6.502985 202.93554  rice
2    85 58 41    21.77046 80.31964 7.038096 226.65554  rice
3    60 55 44    23.00446 82.32076 7.840207 263.96425  rice
4    74 35 40    26.49110 80.15836 6.980401 242.86403  rice
5    78 42 42    20.13017 81.60487 7.628473 262.71734  rice
6    69 37 42    23.05805 83.37012 7.073454 251.05500  rice
7    69 55 38    22.70884 82.63941 5.700806 271.32486  rice
8    94 53 40    20.27774 82.89409 5.718627 241.97419  rice
9    89 54 38    24.51588 83.53522 6.685346 230.44624  rice
10   68 58 38    23.22397 83.03323 6.336254 221.20920  rice
11   91 53 40    26.52724 81.41754 5.386168 264.61487  rice
12   90 46 42    23.97898 81.45062 7.502834 250.08323  rice
13   78 58 44    26.80080 80.88685 5.108682 284.43646  rice
14   93 56 36    24.01498 82.05687 6.984354 185.27734  rice
15   94 50 37    25.66585 80.66385 6.948020 209.58697  rice
16   60 48 39    24.28209 80.30026 7.042299 231.08633  rice
17   85 38 41    21.58712 82.78837 6.249051 276.65525  rice
18   91 35 39    23.79392 80.41818 6.970860 206.26119  rice
19   77 38 36    21.86525 80.19230 5.953933 224.55502  rice
20   88 35 40    23.57944 83.58760 5.853932 291.29866  rice
21   89 45 36    21.32504 80.47476 6.442475 185.49747  rice
22   76 40 43    25.15746 83.11713 5.070176 231.38432  rice
23   67 59 41    21.94767 80.97384 6.012633 213.35609  rice
24   83 41 43    21.05254 82.67840 6.254028 233.10758  rice
25   98 47 37    23.48381 81.33265 7.375483 224.05812  rice
26   66 53 41    25.07564 80.52389 7.778915 257.00389  rice
27   97 59 43    26.35927 84.04404 6.286500 271.35861  rice
28   97 50 41    24.52923 80.54499 7.070960 260.26340  rice
29   60 49 44    20.77576 84.49774 6.244841 240.08106  rice
30   84 51 35    22.30157 80.64416 6.043305 197.97912  rice
31   73 57 41    21.44654 84.94376 5.824709 272.20172  rice
32   92 35 40    22.17932 80.33127 6.357389 200.08828  rice
33   85 37 39    24.52784 82.73686 6.364135 224.67572  rice
34   98 53 38    20.26708 81.63895 5.014507 270.44173  rice
35   88 54 44    25.73543 83.88266 6.149411 233.13214  rice
36   95 55 42    26.79534 82.14809 5.950661 193.34740  rice
37   99 57 35    26.75754 81.17734 5.960370 272.29991  rice
38   95 39 36    23.86330 83.15251 5.561399 285.24936  rice
39   60 43 44    21.01945 82.95222 7.416245 298.40185  rice
40   63 44 41    24.17299 83.72876 5.583370 257.03436  rice
41   62 42 36    22.78134 82.06719 6.430010 248.71832  rice
42   64 45 43    25.62980 83.52842 5.534878 209.90020  rice
43   83 60 36    25.59705 80.14509 6.903986 200.83490  rice
44   82 40 40    23.83067 84.81360 6.271479 298.56012  rice
45   85 52 45    26.31355 82.36699 7.224286 265.53559  rice
46   91 35 38    24.89728 80.52586 6.134287 183.67932  rice
47   76 49 42    24.95878 84.47963 5.206373 196.95600  rice
48   74 39 38    23.24114 84.59202 7.782051 233.04535  rice
49   79 43 39    21.66628 80.70961 7.062779 210.81421  rice
50   88 55 45    24.63545 80.41363 7.730368 253.72028  rice
51   60 36 43    23.43122 83.06310 5.286204 219.90483  rice
52   76 60 39    20.04541 80.34776 6.766240 208.58102  rice
53   93 56 42    23.85724 82.22573 7.382763 195.09483  rice
54   65 60 43    21.97199 81.89918 5.658169 227.36370  rice
```

```
55    95 52 36    26.22917 83.83626 5.543360 286.50837  rice
56    75 38 39    23.44677 84.79352 6.215110 283.93385  rice
57    74 54 38    25.65553 83.47021 7.120273 217.37886  rice
58    91 36 45    24.44345 82.45433 5.950648 267.97619  rice
59    71 46 40    20.28019 82.12354 7.236705 191.95357  rice
60    99 55 35    21.72383 80.23899 6.501698 277.96262  rice
61    72 40 38    20.41447 82.20803 7.592491 245.15113  rice
62    83 58 45    25.75529 83.51827 5.875346 245.66268  rice
63    93 58 38    20.61521 83.77346 6.932400 279.54517  rice
64    70 36 42    21.84107 80.72886 6.946210 202.38383  rice
65    76 47 42    20.08370 83.29115 5.739175 263.63722  rice
66    99 41 36    24.45802 82.74836 6.738652 182.56163  rice
67    99 54 37    21.14347 80.33503 5.594820 198.67309  rice
68    86 59 35    25.78721 82.11124 6.946636 243.51204  rice
69    69 46 41    23.64125 80.28598 5.012140 263.11033  rice
70    91 56 37    23.43192 80.56888 6.363472 269.50392  rice
71    61 52 41    24.97670 83.89181 6.880431 204.80018  rice
72    67 45 38    22.72791 82.17069 7.300411 260.88751  rice
73    79 42 37    24.87301 82.84023 6.587919 295.60945  rice
74    78 43 42    21.32376 83.00320 7.283737 192.31975  rice
75    75 54 36    26.29465 84.56919 7.023936 257.49149  rice
76    97 36 45    22.22870 81.85873 6.939084 278.07918  rice
77    67 47 44    26.73072 81.78597 7.868475 280.40444  rice
78    73 35 38    24.88921 81.97927 5.005307 185.94614  rice
79    77 36 37    26.88445 81.46034 6.136132 194.57666  rice
80    81 41 38    22.67846 83.72874 7.524080 200.91332  rice
81    68 57 43    26.08868 80.37980 5.706943 182.90435  rice
82    72 45 35    25.42978 82.94683 5.758506 195.35745  rice
83    61 53 43    26.40323 81.05636 6.349606 223.36719  rice
84    67 43 39    26.04372 84.96907 5.999969 186.75368  rice
85    67 58 39    25.28272 80.54373 5.453592 220.11567  rice
86    66 60 38    22.08577 83.47038 6.372576 231.73650  rice
87    82 43 38    23.28617 81.43322 5.105588 242.31706  rice
88    84 50 44    25.48592 81.40634 5.935344 182.65494  rice
89    81 53 42    23.67575 81.03569 5.177823 233.70350  rice
90    91 50 40    20.82477 84.13419 6.462392 230.22422  rice
91    93 53 38    26.92995 81.91411 7.069172 290.67938  rice
92    90 44 38    23.83510 83.88387 7.473134 241.20135  rice
93    81 45 35    26.52873 80.12267 6.158377 218.91636  rice
94    78 40 38    26.46428 83.85643 7.549874 248.22565  rice
95    60 51 36    22.69658 82.81089 6.028322 256.99648  rice
96    88 46 42    22.68319 83.46358 6.604993 194.26517  rice
97    93 47 37    21.53346 82.14004 6.500343 295.92488  rice
98    60 55 45    21.40866 83.32932 5.935745 287.57669  rice
99    78 35 44    26.54348 84.67354 7.072656 183.62227  rice
100   65 37 40    23.35905 83.59512 5.333323 188.41367  rice
101   71 54 16    22.61360 63.69071 5.749914  87.75954 maize
102   61 44 17    26.10018 71.57477 6.931757 102.26624 maize
103   80 43 16    23.55882 71.59351 6.657965  66.71995 maize
104   73 58 21    19.97216 57.68273 6.596061  60.65171 maize
105   61 38 20    18.47891 62.69504 5.970458  65.43835 maize
106   68 41 16    21.77689 57.80841 6.158831 102.08617 maize
107   93 41 17    25.62172 66.50415 6.047907 105.46547 maize
108   89 60 19    25.19192 66.69029 5.913665  78.06640 maize
109   76 44 17    20.41683 62.55425 5.855442  65.27798 maize
110   67 60 25    24.92162 66.78627 5.750255 109.21623 maize
111   70 44 19    23.31689 73.45415 5.852607  94.29713 maize
112   90 49 21    24.84017 68.35846 6.472523  74.05475 maize
113   62 52 16    22.27527 58.84016 6.967058  63.87021 maize
114   92 44 16    18.87751 65.76816 6.082974  94.76189 maize
115   66 54 21    25.19009 60.20017 5.919046  72.12376 maize
116   63 58 22    18.25405 55.28220 6.204748  63.72358 maize
117   70 47 17    24.61291 70.41624 6.600827 104.16261 maize
118   61 41 17    25.14206 65.26185 6.021902  76.68456 maize
119   66 53 19    23.09348 60.11594 6.033550  65.49731 maize
120   74 55 19    18.05034 62.89367 6.288868  84.23613 maize
121   77 57 21    24.93216 73.80435 6.550564  79.74079 maize
122   99 50 15    18.14710 71.09445 5.573286  88.07754 maize
123   74 56 22    18.28362 66.65953 6.829199  80.97573 maize
124   83 45 21    18.83344 58.75082 5.716223  79.75329 maize
125 100 48 16    25.71896 67.22191 5.549902  74.51491 maize
 [ reached 'max' / getOption("max.print") -- omitted 2075 rows ]
```

```
ggpairs(data,columns = 1:7,aes(colour=label))
split <- sample.split(data, SplitRatio = 0.7)
train_cl <- subset(data, split == "TRUE")
test_cl <- subset(data, split == "FALSE")
```

OUTPUT :



 # Feature Scaling
train_scale <- train_cl[, 1:7]
test_scale <- test_cl[, 1:7]

```
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$label,
                      k = 3)
length(classifier_knn)
length(test_cl$label)
```

<u>OUTPUT</u> :

```
> train_scale <- train_cl[, 1:7]
> test_scale <- test_cl[, 1:7]
>
> classifier_knn <- knn(train = train_scale,
+                       test = test_scale,
+                       cl = train_cl$label,
+                       k = 3)
> length(classifier_knn)
[1] 825
> length(test_cl$label)
[1] 825
```

```
yt <- as.factor( test_cl$label )
levels(classifier_knn)
```

<u>OUTPUT:</u>

```
> yt <- as.factor( test_cl$label )
> levels(classifier_knn)
 [1] "apple"       "banana"      "blackgram"   "chickpea"    "coconut"     "coffee"      "cotton"
 [8] "grapes"      "jute"        "kidneybeans" "lentil"      "maize"       "mango"       "mothbeans"
[15] "mungbean"    "muskmelon"   "orange"      "papaya"      "pigeonpeas"  "pomegranate" "rice"
[22] "watermelon"
```

```
levels(yt)
misClassError <- mean(classifier_knn != test_cl$label)
print(paste('Accuracy =', 1-misClassError))
```

OUTPUT:

```
> levels(yt)
 [1] "apple"       "banana"      "blackgram"   "chickpea"    "coconut"     "coffee"      "cotton"
 [8] "grapes"      "jute"        "kidneybeans" "lentil"      "maize"       "mango"       "mothbeans"
[15] "mungbean"    "muskmelon"   "orange"      "papaya"      "pigeonpeas"  "pomegranate" "rice"
[22] "watermelon"
> misClassError <- mean(classifier_knn != test_cl$label)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.974545454545454"
```

cm <- confusionMatrix(classifier_knn, yt)
cm

OUTPUT:

```
> cm <- confusionMatrix(classifier_knn, yt)
> cm
Confusion Matrix and Statistics

             Reference
Prediction    apple banana blackgram chickpea coconut coffee cotton grapes jute kidneybeans lentil maize
  apple          38      0         0        0       0      0      0      0    0           0      0     0
  banana          0     37         0        0       0      0      0      0    0           0      0     0
  blackgram       0      0        38        0       0      0      0      0    0           0      0     0
  chickpea        0      0         0       37       0      0      0      0    0           0      0     0
  coconut         0      0         0        0      37      0      0      0    0           0      0     0
  coffee          0      0         0        0       0     37      0      0    0           0      0     0
  cotton          0      0         0        0       0      0     38      0    0           0      0     1
  grapes          0      0         0        0       0      0      0     37    0           0      0     0
  jute            0      0         0        0       0      1      0      0   31           0      0     0
  kidneybeans     0      0         0        0       0      0      0      0    0          38      0     0
  lentil          0      0         0        0       0      0      0      0    0           0     37     0
  maize           0      0         0        0       0      0      0      0    0           0      0    37
  mango           0      0         0        0       0      0      0      0    0           0      0     0
  mothbeans       0      0         0        0       0      0      0      0    0           0      0     0
  mungbean        0      0         0        0       0      0      0      0    0           0      0     0
  muskmelon       0      0         0        0       0      0      0      0    0           0      0     0
  orange          0      0         0        0       0      0      0      0    0           0      0     0
  papaya          0      0         0        0       0      0      0      0    0           0      0     0
  pigeonpeas      0      0         0        0       0      0      0      0    0           0      0     0
  pomegranate     0      0         0        0       0      0      0      0    0           0      0     0
  rice            0      0         0        0       0      0      0      0    6           0      0     0
  watermelon      0      0         0        0       0      0      0      0    0           0      0     0
             Reference
Prediction    mango mothbeans mungbean muskmelon orange papaya pigeonpeas pomegranate rice watermelon
  apple           0         0        0         0      0      0          0           0    0          0
  banana          0         0        0         0      0      0          0           0    0          0
  blackgram       0         2        0         0      0      0          0           0    0          0
  chickpea        0         0        0         0      0      0          0           0    0          0
  coconut         0         0        0         0      0      0          0           0    0          0
  coffee          0         0        0         0      0      0          0           0    0          0
  cotton          0         0        0         0      0      0          0           0    0          0
  grapes          0         0        0         0      0      0          0           0    0          0
  jute            0         0        0         0      0      2          0           0    5          0
  kidneybeans     0         0        0         0      0      0          2           0    0          0
  lentil          0         2        0         0      0      0          0           0    0          0
  maize           0         0        0         0      0      0          0           0    0          0
  mango          38         0        0         0      0      0          0           0    0          0
  mothbeans       0        34        0         0      0      0          0           0    0          0
  mungbean        0         0       37         0      0      0          0           0    0          0
  muskmelon       0         0        0        37      0      0          0           0    0          0
  orange          0         0        0         0     37      0          0           0    0          0
  papaya          0         0        0         0      0     36          0           0    0          0
  pigeonpeas      0         0        0         0      0      0         35           0    0          0
  pomegranate     0         0        0         0      0      0          0          38    0          0
  rice            0         0        0         0      0      0          0           0   32          0
  watermelon      0         0        0         0      0      0          0           0    0         38

Overall Statistics

               Accuracy : 0.9745
                 95% CI : (0.9614, 0.9842)
    No Information Rate : 0.0461
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9733

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: apple Class: banana Class: blackgram Class: chickpea Class: coconut
Sensitivity               1.00000       1.00000          1.00000         1.00000        1.00000
Specificity               1.00000       1.00000          0.99746         1.00000        1.00000
Pos Pred Value            1.00000       1.00000          0.95000         1.00000        1.00000
Neg Pred Value            1.00000       1.00000          1.00000         1.00000        1.00000
Prevalence                0.04606       0.04485          0.04606         0.04485        0.04485
Detection Rate            0.04606       0.04485          0.04606         0.04485        0.04485
Detection Prevalence      0.04606       0.04485          0.04848         0.04485        0.04485
Balanced Accuracy         1.00000       1.00000          0.99873         1.00000        1.00000
                     Class: coffee Class: cotton Class: grapes Class: jute Class: kidneybeans
Sensitivity                0.97368       1.00000       1.00000     0.83784            1.00000
Specificity                1.00000       0.99873       1.00000     0.98985            0.99746
Pos Pred Value             1.00000       0.97436       1.00000     0.79487            0.95000
Neg Pred Value             0.99873       1.00000       1.00000     0.99237            1.00000
Prevalence                 0.04606       0.04606       0.04485     0.04485            0.04606
```

```
Detection Rate            0.04485      0.04606      0.04485     0.03758          0.04606
Detection Prevalence      0.04485      0.04727      0.04485     0.04727          0.04848
Balanced Accuracy         0.98684      0.99936      1.00000     0.91384          0.99873
                     Class: lentil Class: maize Class: mango Class: mothbeans Class: mungbean
Sensitivity               1.00000      0.97368      1.00000          0.89474          1.00000
Specificity               0.99746      1.00000      1.00000          1.00000          1.00000
Pos Pred Value            0.94872      1.00000      1.00000          1.00000          1.00000
Neg Pred Value            1.00000      0.99873      1.00000          0.99494          1.00000
Prevalence                0.04485      0.04606      0.04606          0.04606          0.04485
Detection Rate            0.04485      0.04485      0.04606          0.04121          0.04485
Detection Prevalence      0.04727      0.04485      0.04606          0.04121          0.04485
Balanced Accuracy         0.99873      0.98684      1.00000          0.94737          1.00000
                     Class: muskmelon Class: orange Class: papaya Class: pigeonpeas Class: pomegranate
Sensitivity                   1.00000       1.00000       0.94737           0.94595            1.00000
Specificity                   1.00000       1.00000       1.00000           1.00000            1.00000
Pos Pred Value                1.00000       1.00000       1.00000           1.00000            1.00000
Neg Pred Value                1.00000       1.00000       0.99747           0.99747            1.00000
Prevalence                    0.04485       0.04485       0.04606           0.04485            0.04606
Detection Rate                0.04485       0.04485       0.04364           0.04242            0.04606
Detection Prevalence          0.04485       0.04485       0.04364           0.04242            0.04606
Balanced Accuracy             1.00000       1.00000       0.97368           0.97297            1.00000
                     Class: rice Class: watermelon
Sensitivity              0.86486           1.00000
Specificity              0.99239           1.00000
Pos Pred Value           0.84211           1.00000
Neg Pred Value           0.99365           1.00000
Prevalence               0.04485           0.04606
Detection Rate           0.03879           0.04606
Detection Prevalence     0.04606           0.04606
Balanced Accuracy        0.92863           1.00000
```

#naiveBayes

classifier_cl <- naiveBayes(label ~ ., data = train_cl)
classifier_cl

<u>OUTPUT:</u>

```
> classifier_cl <- naiveBayes(label ~ ., data = train_cl)
> classifier_cl

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
      apple      banana   blackgram    chickpea     coconut      coffee      cotton      grapes 
 0.04509091  0.04581818  0.04509091  0.04581818  0.04581818  0.04509091  0.04509091  0.04581818 
       jute kidneybeans      lentil       maize       mango   mothbeans    mungbean   muskmelon 
 0.04581818  0.04509091  0.04581818  0.04509091  0.04509091  0.04509091  0.04581818  0.04581818 
     orange      papaya  pigeonpeas pomegranate        rice  watermelon 
 0.04581818  0.04509091  0.04581818  0.04509091  0.04581818  0.04509091 
```

```
Conditional probabilities:
             N
Y                  [,1]     [,2]
  apple        21.00000 12.20788
  banana      101.93548 11.05407
  blackgram    40.74603 12.13362
  chickpea     37.64516 11.94262
  coconut      22.38710 11.97313
  coffee      100.88889 13.31518
  cotton      115.77778 11.63344
  grapes       21.95161 12.82343
  jute         78.96774 11.68713
  kidneybeans  19.84127 10.68343
  lentil       17.48387 11.68541
  maize        78.31746 12.33514
  mango        20.36508 12.20431
  mothbeans    21.31746 12.37170
  mungbean     20.58065 11.16280
  muskmelon   100.48387 11.36826
  orange       19.66129 12.80490
  papaya       50.20635 12.21601
  pigeonpeas   20.43548 11.39258
  pomegranate  17.66667 12.71017
  rice         80.09677 12.19609
  watermelon   98.39683 12.75645

             P
Y                  [,1]     [,2]
  apple       135.07937 8.110729
  banana       81.11290 7.435195
  blackgram    67.61905 7.258873
  chickpea     68.74194 7.328220
  coconut      16.91935 8.566611
  coffee       29.00000 7.242349
  cotton       46.04762 7.245529
  grapes      133.40323 7.069815
  jute         46.83871 7.236505
```

```
  kidneybeans  68.52381 7.407183
  lentil       69.06452 7.458895
  maize        47.92063 7.717390
  mango        26.68254 8.083886
  mothbeans    47.84127 7.773417
  mungbean     47.48387 7.427777
  muskmelon    16.91935 7.145556
  orange       15.91935 7.466424
  papaya       58.39683 7.250227
  pigeonpeas   67.56452 7.453948
  pomegranate  19.76190 7.072697
  rice         48.20968 8.200634
  watermelon   18.20635 7.258274


            K
Y                  [,1]     [,2]
  apple       199.87302 3.154577
  banana       50.35484 3.259774
  blackgram    19.28571 3.244989
  chickpea     80.17742 3.226551
  coconut      30.48387 3.191905
  coffee       30.01587 3.289730
  cotton       19.63492 3.148484
  grapes      200.48387 3.191905
  jute         39.40323 3.138652
  kidneybeans  19.96825 3.222743
   lentil       19.12903 2.922406
  maize        19.82540 3.029637
  mango        29.90476 2.949654
  mothbeans    20.30159 3.088140
  mungbean     19.27419 2.959671
  muskmelon    49.61290 3.189917
  orange       10.16129 3.003787
  papaya       50.12698 3.092614
  pigeonpeas   20.17742 2.837248
  pomegranate  40.09524 2.960570
  rice         39.79032 3.121250
  watermelon   50.47619 3.110114

            temperature
Y                 [,1]      [,2]
  apple       22.69395 0.8555413
  banana      27.34318 1.4452143
  blackgram   29.92332 2.6894990
  chickpea    18.97389 1.1496688
  coconut     27.37994 1.3852449
  coffee      25.48455 1.5428347
  cotton      23.87204 1.0656326
  grapes      25.25124 9.3404687
  jute        24.98164 1.1918672
  kidneybeans 20.35922 2.4774687
  lentil      24.44094 3.2488351
  maize       22.05302 2.7399133
  mango       31.38185 2.6203945
  mothbeans   28.19961 2.1769995
  mungbean    28.56047 0.8311697
  muskmelon   28.69924 0.8968481
  orange      22.68467 7.3133300
  papaya      33.37315 6.3621611
  pigeonpeas  27.23249 5.6197146
  pomegranate 21.54855 2.2372925
  rice        23.76447 2.0313312
  watermelon  25.57891 0.8693995

            humidity
Y                 [,1]     [,2]
  apple       92.36135 1.502565
  banana      80.08031 2.868457
  blackgram   64.64328 2.929916
  chickpea    16.51123 1.694882
  coconut     95.07590 2.945164
  coffee      59.44090 6.234963
  cotton      79.78769 3.037565
  grapes      81.88037 1.146070
  jute        80.57116 5.271143
  kidneybeans 21.55262 2.252967
  lentil      65.13793 2.961578
  maize       64.88437 5.252448
  mango       49.94521 2.708305
  mothbeans   52.96275 7.082344
  mungbean    85.60557 2.969254
  muskmelon   92.26302 1.503896
  orange      92.21760 1.496870
  papaya      92.36465 1.390895
  pigeonpeas  47.19418 9.544135
  pomegranate 90.48408 2.926246
  rice        82.23224 1.428705
  watermelon  85.32071 2.979124

            ph
Y                 [,1]      [,2]
  apple       5.926988 0.2793999
  banana      5.995651 0.2576380
  blackgram   7.133572 0.3956869
  chickpea    7.246491 0.7330141
```

```
  coconut     5.971360 0.2806077
  coffee      6.798624 0.4102132
  cotton      6.907891 0.6111983
  grapes      6.056157 0.2826779
  jute        6.749320 0.4470068
  kidneybeans 5.756508 0.1448290
  lentil      6.976263 0.5913500
  maize       6.249015 0.4254910
  mango       5.787574 0.6968699
  mothbeans   6.682755 1.9082999
  mungbean    6.694990 0.2847135
  muskmelon   6.363872 0.2194133
  orange      7.040106 0.5955312
  papaya      6.714447 0.1403588
  pigeonpeas  5.792904 0.8908812
  pomegranate 6.504529 0.4830742
  rice        6.381090 0.8138597
  watermelon  6.499114 0.2730898

             rainfall
Y                  [,1]      [,2]
  apple       111.90169  6.928346
  banana      104.38323  9.766866
  blackgram    68.21376  4.124942
  chickpea     80.10412  8.071419
  coconut     179.01758 28.861100
  coffee      154.21836 24.475051
  cotton       79.08542 10.998806
  grapes       69.60277  3.067933
  jute        177.48701 14.518082
  kidneybeans 110.13842 25.203074
  lentil       45.67619  5.640025
  maize        85.34757 15.778507
  mango        95.06138  3.376049
  mothbeans    49.59329 12.861902
  mungbean     48.16666  7.075133
  muskmelon    24.75293  2.727313
  orange      110.52744  5.338393
  papaya      141.89851 64.630693
  pigeonpeas  149.83753 32.661746
  pomegranate 107.44752  3.071778
  rice        238.14715 35.149004
  watermelon   50.27813  5.863521
```

 # Predicting on test data
y_pred <- predict(classifier_cl, newdata = test_cl)

misClassError <- mean(y_pred != test_cl$label)
print(paste('Accuracy =', 1-misClassError))
print(train_cl)

OUTPUT:

```
> y_pred <- predict(classifier_cl, newdata = test_cl)
>
> misClassError <- mean(y_pred != test_cl$label)
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.993939393939394"
> print(train_cl)
      N  P  K temperature humidity       ph  rainfall label
2    85 58 41    21.77046 80.31964 7.038096 226.65554  rice
4    74 35 40    26.49110 80.15836 6.980401 242.86403  rice
5    78 42 42    20.13017 81.60487 7.628473 262.71734  rice
6    69 37 42    23.05805 83.37012 7.073454 251.05500  rice
7    69 55 38    22.70884 82.63941 5.700806 271.32486  rice
10   68 58 38    23.22397 83.03323 6.336254 221.20920  rice
12   90 46 42    23.97898 81.45062 7.502834 250.08323  rice
13   78 58 44    26.80080 80.88685 5.108682 284.43646  rice
14   93 56 36    24.01498 82.05687 6.984354 185.27734  rice
15   94 50 37    25.66585 80.66385 6.948020 209.58697  rice
18   91 35 39    23.79392 80.41818 6.970860 206.26119  rice
20   88 35 40    23.57944 83.58760 5.853932 291.29866  rice
21   89 45 36    21.32504 80.47476 6.442475 185.49747  rice
22   76 40 43    25.15746 83.11713 5.070176 231.38432  rice
23   67 59 41    21.94767 80.97384 6.012633 213.35609  rice
26   66 53 41    25.07564 80.52389 7.778915 257.00389  rice
28   97 50 41    24.52923 80.54499 7.070960 260.26340  rice
29   60 49 44    20.77576 84.49774 6.244841 240.08106  rice
30   84 51 35    22.30157 80.64416 6.043305 197.97912  rice
31   73 57 41    21.44654 84.94376 5.824709 272.20172  rice
34   98 53 38    20.26708 81.63895 5.014507 270.44173  rice
36   95 55 42    26.79534 82.14809 5.950661 193.34740  rice
37   99 57 35    26.75754 81.17734 5.960370 272.29991  rice
38   95 39 36    23.86330 83.15251 5.561399 285.24936  rice
39   60 43 44    21.01945 82.95222 7.416245 298.40185  rice
42   64 45 43    25.62980 83.52842 5.534878 209.90020  rice
44   82 40 40    23.83067 84.81360 6.271479 298.56012  rice
45   85 52 45    26.31355 82.36699 7.224286 265.53559  rice
```

46	91	35	38	24.89728	80.52586	6.134287	183.67932	rice
47	76	49	42	24.95878	84.47963	5.206373	196.95600	rice
50	88	55	45	24.63545	80.41363	7.730368	253.72028	rice
52	76	60	39	20.04541	80.34776	6.766240	208.58102	rice
53	93	56	42	23.85724	82.22573	7.382763	195.09483	rice
54	65	60	43	21.97199	81.89918	5.658169	227.36370	rice
55	95	52	36	26.22917	83.83626	5.543360	286.50837	rice
58	91	36	45	24.44345	82.45433	5.950648	267.97619	rice
60	99	55	35	21.72383	80.23899	6.501698	277.96262	rice
61	72	40	38	20.41447	82.20803	7.592491	245.15113	rice
62	83	58	45	25.75529	83.51827	5.875346	245.66268	rice
63	93	58	38	20.61521	83.77346	6.932400	279.54517	rice
66	99	41	36	24.45802	82.74836	6.738652	182.56163	rice
68	86	59	35	25.78721	82.11124	6.946636	243.51204	rice
69	69	46	41	23.64125	80.28598	5.012140	263.11033	rice
70	91	56	37	23.43192	80.56888	6.363472	269.50392	rice
71	61	52	41	24.97670	83.89181	6.880431	204.80018	rice
74	78	43	42	21.32376	83.00320	7.283737	192.31975	rice
76	97	36	45	22.22870	81.85873	6.939084	278.07918	rice
77	67	47	44	26.73072	81.78597	7.868475	280.40444	rice
78	73	35	38	24.88921	81.97927	5.005307	185.94614	rice
79	77	36	37	26.88445	81.46034	6.136132	194.57666	rice
82	72	45	35	25.42978	82.94683	5.758506	195.35745	rice
84	67	43	39	26.04372	84.96907	5.999969	186.75368	rice
85	67	58	39	25.28272	80.54373	5.453592	220.11567	rice
86	66	60	38	22.08577	83.47038	6.372576	231.73650	rice
87	82	43	38	23.28617	81.43322	5.105588	242.31706	rice
90	91	50	40	20.82477	84.13419	6.462392	230.22422	rice
92	90	44	38	23.83510	83.88387	7.473134	241.20135	rice
93	81	45	35	26.52873	80.12267	6.158377	218.91636	rice
94	78	40	38	26.46428	83.85643	7.549874	248.22565	rice
95	60	51	36	22.69658	82.81089	6.028322	256.99648	rice
98	60	55	45	21.40866	83.32932	5.935745	287.57669	rice
100	65	37	40	23.35905	83.59512	5.333323	188.41367	rice
101	71	54	16	22.61360	63.69071	5.749914	87.75954	maize
102	61	44	17	26.10018	71.57477	6.931757	102.26624	maize
103	80	43	16	23.55882	71.59351	6.657965	66.71995	maize
106	68	41	16	21.77689	57.80841	6.158831	102.08617	maize
108	89	60	19	25.19192	66.69029	5.913665	78.06640	maize
109	76	44	17	20.41683	62.55425	5.855442	65.27798	maize
110	67	60	25	24.92162	66.78627	5.750255	109.21623	maize
111	70	44	19	23.31689	73.45415	5.852607	94.29713	maize
114	92	44	16	18.87751	65.76816	6.082974	94.76189	maize
116	63	58	22	18.25405	55.28220	6.204748	63.72358	maize
117	70	47	17	24.61291	70.41624	6.600827	104.16261	maize
118	61	41	17	25.14206	65.26185	6.021902	76.68456	maize
119	66	53	19	23.09348	60.11594	6.033550	65.49731	maize
122	99	50	15	18.14710	71.09445	5.573286	88.07754	maize
124	83	45	21	18.83344	58.75082	5.716223	79.75329	maize
125	100	48	16	25.71896	67.22191	5.549902	74.51491	maize
126	79	51	16	25.33798	68.49836	6.586245	96.46380	maize
127	94	39	18	23.89115	57.48776	5.893093	102.83019	maize
130	87	54	20	25.61707	63.47118	6.576418	108.83038	maize
132	63	43	19	18.51817	55.53128	6.641906	90.98805	maize
133	84	57	25	22.53511	67.99257	6.489040	64.40866	maize
134	64	35	23	23.02038	61.89472	5.680361	63.03843	maize
135	60	46	22	24.89365	65.61419	6.625404	87.92981	maize
138	86	55	21	21.54156	59.64024	6.803932	109.75154	maize
140	76	57	18	18.98027	74.52601	6.092726	94.26249	maize
141	99	56	17	24.10859	73.13112	6.234330	71.07562	maize
142	60	44	23	24.79471	70.04557	5.722580	76.72860	maize
143	74	48	17	21.63163	60.27766	6.430616	69.21803	maize
146	96	46	22	20.58314	69.00129	6.499936	66.29390	maize
148	74	58	18	20.03728	56.35607	6.727303	109.02414	maize
149	74	43	23	25.95263	61.89082	6.325235	99.57981	maize
150	63	43	17	19.28890	65.47051	6.807488	71.31953	maize
151	99	36	20	20.57982	65.34584	6.671086	78.34604	maize
154	60	38	17	18.41933	64.23580	6.474477	76.41312	maize
156	95	38	22	19.84939	61.24500	5.730617	100.76892	maize
157	84	44	21	21.86927	61.91045	5.850440	107.26819	maize
158	77	58	19	22.80560	56.50769	5.791650	101.59528	maize
159	66	44	20	19.07815	69.02299	6.740001	80.72516	maize
162	72	60	25	18.52511	69.02762	5.773455	88.10234	maize
164	86	36	24	26.54986	72.89187	5.787268	73.33636	maize
165	76	48	18	19.29563	69.63481	5.775978	83.21031	maize
166	75	53	18	20.68900	59.43753	6.864794	103.65144	maize
167	81	45	23	19.32666	68.03449	6.192360	84.22969	maize
170	96	54	22	25.70197	61.33450	6.960358	83.20711	maize
172	62	48	20	21.70181	60.47471	6.708447	95.71388	maize
173	86	37	16	20.51717	59.21235	5.561511	67.61014	maize
174	94	50	19	23.30355	73.62548	5.873242	97.59081	maize
175	76	39	24	24.25475	55.64710	6.995844	64.23845	maize
178	81	49	20	18.04186	60.61494	5.513698	104.23216	maize
180	99	38	21	22.88331	71.59722	6.352472	67.72777	maize
181	90	52	25	25.97482	69.36386	6.822587	103.22342	maize
182	68	40	19	26.14384	66.20570	6.655426	107.23614	maize
183	60	57	24	18.66116	61.55327	6.121294	75.03248	maize
186	88	38	15	25.08240	65.92196	6.455117	62.49191	maize
188	78	37	22	25.34217	63.31802	6.330554	74.52082	maize
189	78	58	15	25.00933	67.81657	6.528631	62.91359	maize
190	92	60	23	18.66747	71.51647	5.721667	69.93293	maize
191	79	59	17	20.38000	63.73850	6.644205	108.50544	maize
194	87	48	25	18.65397	61.37880	6.656730	93.62039	maize
196	90	57	24	18.92852	72.80086	6.158860	82.34163	maize
197	67	35	22	23.30547	63.24648	6.385684	108.76030	maize

```
198  60 54 19    18.74827 62.49878 6.417820  70.23402 maize
199  83 58 23    19.74213 59.66263 6.381202  65.50861 maize
 [ reached 'max' / getOption("max.print") -- omitted 1250 rows ]
```

16. CONCLUSION:- In conclusion, machine learning has emerged as a promising tool for predicting crop yields and improving agricultural practices. By leveraging large datasets and sophisticated algorithms, machine learning models can analyze various factors such as weather patterns, soil conditions, historical crop data, and management practices to make accurate predictions about crop yields. One key benefit of crop prediction using machine learning is its potential to optimize agricultural practices. Farmers can use these predictions to make informed decisions about planting schedules, irrigation, fertilization, and pest management, leading to more efficient resource allocation and higher yields. Additionally, machine learning can help farmers identify early warning signs of crop stress or disease outbreaks, allowing for timely interventions and reducing crop losses. Machine learning in crop prediction also has the potential to contribute to sustainable agriculture by optimizing resource use. For example, by predicting crop water requirements, farmers can implement targeted irrigation strategies, minimizing water waste and conserving this precious resource. Similarly, by predicting crop nutrient needs, farmers can apply fertilizers more judiciously, reducing the risk of nutrient runoff and environmental pollution. However, it's important to note that machine learning models for crop prediction are not without limitations. Accurate predictions depend on the availability of reliable data, and in many regions, data may be sparse or inconsistent. Additionally, machine learning models are not immune to biases and may suffer from limitations in generalization, especially when applied to different regions or crop varieties. Therefore, it's crucial to continue refining and validating these models using field data and expert knowledge. In conclusion, machine learning has the potential to revolutionize crop prediction and agricultural practices, leading to improved crop yields, resource optimization, and sustainable agriculture. However, ongoing research, data collection, and model validation are necessary to ensure their reliability and effectiveness in real-world farming scenarios.

17. FUTURE SCOPE:- The future scope of machine learning in crop prediction is promising and holds significant potential for revolutionizing agriculture and improving crop production. Here are some key areas where machine learning can play a significant role in the future:

Precision Agriculture: Machine learning algorithms can analyze a vast amount of data, including soil quality, weather patterns, pest and disease prevalence, and plant growth rates to provide farmers with precise recommendations on planting, fertilization, irrigation, and pest control. This can optimize resource usage, reduce input costs, and increase crop yields.

Crop Disease and Pest Prediction: Machine learning can be used to analyze historical data on crop diseases and pests and create predictive models that can help farmers anticipate disease outbreaks and pest infestations. This can enable early intervention and prevent crop losses, reducing the reliance on chemical pesticides and minimizing environmental impact.

Climate Change Adaption: As climate change continues to impact agriculture, machine learning can help farmers adapt by providing predictive models that take into account changing weather patterns, temperature fluctuations, and rainfall variability. This can enable farmers to make informed decisions about crop selection, planting times, and irrigation strategies.

Crop Yield Forecasting: Machine learning algorithms can analyze data on crop growth, historical yield data, weather patterns, and other factors to create accurate crop yield forecasts. This can help farmers with crop planning, marketing, and financial decision- making.

Crop Breeding and Genetic Improvement: Machine learning can aid in crop breeding programs by analyzing genetic data and identifying optimal combinations of traits for crop improvement. This can accelerate the development of new crop varieties with improved yield, resistance to diseases and pests, and other desirable traits.

Remote Sensing and Satellite Imagery: Machine learning can analyze remote sensing data, including satellite imagery, to monitor crop health, detect stressors such as nutrient deficiencies, water stress, and disease outbreaks. This can help farmers make data-driven decisions about crop management and optimize inputs.

Decision Support System: Machine learning can power decision support systems that provide farmers with real-time recommendations and insights for crop management. These systems can integrate data from various sources and provide personalized recommendations based on the specific needs of each farm.

In conclusion, machine learning has a bright future in crop prediction and agriculture, and it has the potential to significantly improve crop production, optimize resource usage, and contribute to sustainable farming practices. Continued advancements in machine learning algorithms, data collection, and analytics are expected to drive further innovation in this field in the future.

18. BIBLIOGRAPHY:-

[1]S.Veenadhari1 , Dr. Bharat Misra2 , Dr. CD Singh3 "Data mining Techniques for Predicting Crop Productivity – A review article" 1,2-Mahatma Gandhi Gramodaya Vishwavidyalaya, Chitrakoot, Satna, India, 3-Central Institute of Agricultural Engineering, Bhopal, India, March-2011.

[2] Akbar Batcha, Syed Musthafa, Securing data in transit using data-in-transit defender architecture for cloudcommunication, Soft Computing, springer, june 2021

[3] A Syed Musthafa , M Praveenkumar ," E-agricultural system based intelligent predictive analysis and smart farming with digitalized demand and supply utilization to maximize the yield rate of crops using machine learning algorithm, Turkish Journal of Computer and Mathematics Education, Vol.12 No.10 (2021), pp 2036-2041

[4] D Ramesh1 , B Vishnu Vardhan2 "Data Mining Techniques and Applications to Agricultural Yield Data",September 2013.

[5] Syed Musthafa.A, Mohanraj.B, Priyanga.S, Krishnan.C, "High Security Distributed MANETs using Channel De-noiser and Multi-Mobile-Rate Synthesizer", International Journal of Advanced Trends in Computer Science and Engineering, Volume 9, No. 2, ISSN 2278-3091, Page 1346-1351, April 2020