AGRICROP PERFORMANCE ANALYSIS: FERTILIZER & ENVIRONMENTAL FACTORS

Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Technology in the field of Computer Science and Engineering

BY
RUMPA PAUL(123211003120)
RIYA PAL(123211003117)
RAJDEEP PAL(123211003107)
RIDDHI DHARA(123211003113)
RANGAN NATH(123211003110)

Under the supervision
of
Prof. Chandrima Sinha Roy

Department of Computer Science and Engineering JIS College of Engineering

Block-A, Phase-III, Kalyani, Nadia, Pin-741235
West Bengal, India
Dec, 2023

JIS College of Engineering
Block 'A', Phase-III, Kalyani, Nadia, 741235
Phone: +91 33 2582 2137, Telefax: +91 33 2582 2138
Website: www.jiscollege.ac.in, Email: info@jiscollege.ac.in

CERTIFICATE

This is to certify that **Rumpa Paul (123211003120)**, **Riya Pal (123211003117)** , **Rajdeep Pal (123211003107)**, **Riddhi Dhara(123211003113)**, **Rangan Nath (123211003110)** have completed their project entitled **Agricorp Performance Analysis: Fertilizer & Environmental Factors**, under the guidance of **Prof. Chandrima Sinha Roy** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Science and Engineering** from **JIS college of Engineering** (An Autonomous Institute) is an authentic record of their own work carried out during the academic year 2023-2024 and to the best of our knowledge, this work has not been submitted elsewhere as part of the process of obtaining a degree, diploma, fellowship or any other similar title.

---------- ----------

Signature of the Supervisor **Signature of the HOD**

Place:

Date:

ACKNOWLEDGEMENT

The analysis of the project work wishes to express our gratitude to **Prof. Chandrima Sinha Roy** for allowing the degree attitude and providing effective guidance in development of this project work. Her conscription of the topic and all the helpful hints, she provided, contributed greatly to successful development of this work, without being pedagogic and overbearing influence.

We also express our sincere gratitude to **Dr. Bikramjit Sarkar** Head of the Department of Computer Science and Engineering of JIS College of Engineering and all the respected faculty members of Department of CSE for giving the scope of successfully carrying out the project work.

Finally, we take this opportunity to thank to **Prof. (Dr.) Partha Sarkar**, Principal of JIS College of Engineering for giving us the scope of carrying out the project work.

Date:

…….

Rumpa Paul
B.TECH in Computer Science and Engineering
3rdYEAR/5thSEMESTER
Univ Roll--123211003120

…….

Riya Pal
B.TECH in Computer Science and Engineering
3rdYEAR/5th SEMESTER
Univ Roll—123211003117

…….

Rajdeep Pal
B.TECH in Computer Science and Engineering
3rdYEAR/5thSEMESTER
Univ Roll—123211003107

…….

Riddhi Dhara
B.TECH in Computer Science and Engineering
3rdYEAR/5thSEMESTER
Univ Roll—123211003113

…….

Rangan Nath
B.TECH in Computer Science and Engineering
3rdYEAR/5thSEMESTER
Univ Roll—123211003110

List of figures

CONTENTS

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 python programming characteristics and features. Python is one of the most preferred languages for scientific computing, data science, and machine learning, boosting both performance and productivity by enabling the use of low-level libraries. This paper offers insight into the field of machine learning with python, taking a tour through important topics and libraries of python which enables the development of machine learning model a 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: - Python; Machine Learning; Artificial Intelligence, Regression, Linear Regression.

2. Acknowledgment: -

SDT (Software Development Tools): -Machine learning involves using algorithms to allow computer software programs to 'learn' different tasks making use of the available data. ML programs become more accurate the more they train on the relevant test data. Many types of Software Development Tools are there. Some of them which is used in Programming are: -

2.1 NumPy: -

Work:-In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. Language: -Python, C and C++.

2.2 Pandas: -

Work: -Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

Language:-Python

2.3 Matplotlib: -

Work: -Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc. Language:-Python.

2.4 Pyplot: -

Work:-pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
Language:-Python.

2.5 Seaborn: -

Work:-Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.
Language:-Graphics, Python.

3. SDK: -

As an AI language model, I can provide you with information about Google Colab and its software development kit (SDK).
Google Colab is a cloud-based interactive computing environment that provides free access to a Jupyter Notebook environment along with computational resources, including CPU, GPU, and TPU. It allows users to write, run, and share Python code in a collaborative and interactive manner. Colab is hosted on Google Drive, and notebooks can be created, shared, and saved directly in Google Drive.
While Google Colab does not have an official SDK, it provides a Python library called "google.colab" that allows developers to interact with the Colab environment programmatically. The "google.colab" library provides functionality for tasks such as importing and exporting files, installing Python packages, managing Colab sessions, and connecting to external services like Google Drive and Google Sheets.

Importing and exporting files:-The library allows you to upload and download files to and from the Colab environment. For example, you can use the "files.upload()" function to upload files from your local machine to Colab, and the "files.download()" function to download files from Colab to your local machine.

Installing Python packages: -The library provides a way to install Python packages directly from within the Colab environment using the “! pip install" command.

Managing Colab sessions:-The library allows you to manage the lifecycle of a Colab session. You can use functions like "drive.mount()" to mount your Google Drive, "drive.flush_and_unmount()" to flush and unmount the Google Drive, and "os.kill()" to terminate the current session.Connecting to external services: The library provides functionality to connect to external services like Google Drive and Google Sheets, allowing you to read and write data to these services from within a Colab notebook.

Interacting with Colab UI: - The library allows you to interact with the Colab user interface programmatically, for example, by using the "IPython.display" module to display images, videos, and other media in the output of a Colab cell.

Overall, while Google Colab does not have a standalone SDK, the "google.colab" library provides a convenient way to interact with the Colab environment programmatically and automate various tasks within Colab notebooks. You can import the "google.colab" library in your Python code and use its functions to perform operations within the Colab environment.

4. Models: -

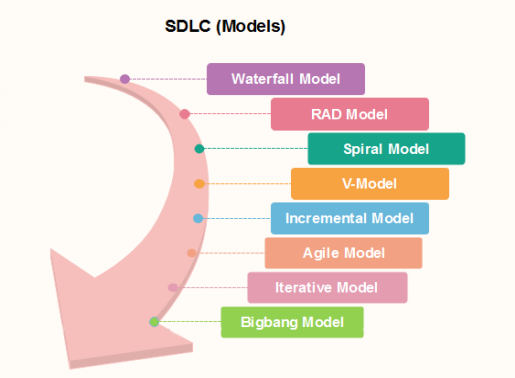

Figure 1

4.1 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.

4.2 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

4.3. 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: -

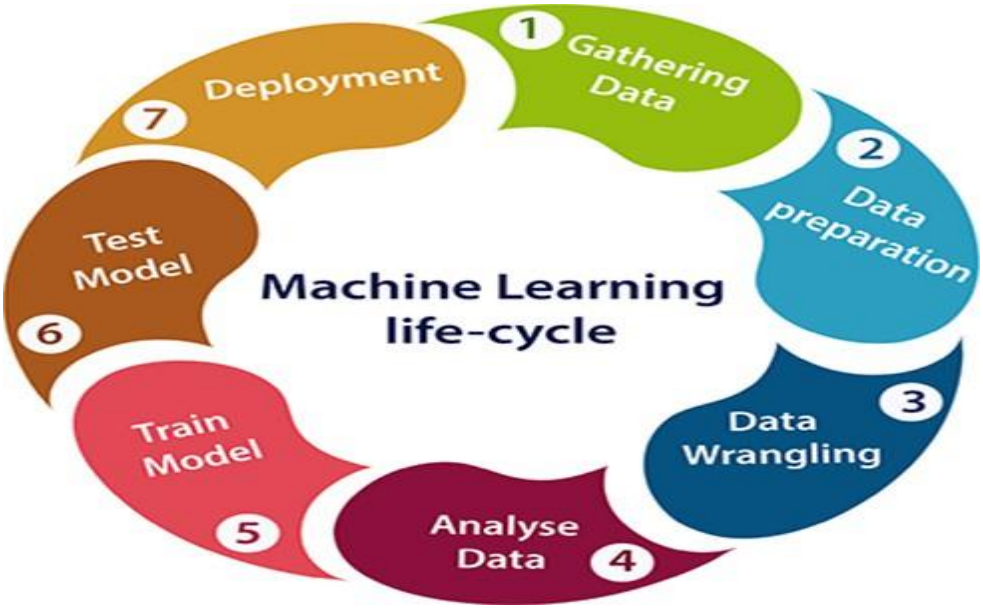

Figure 2

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.

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.

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.

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

5. Train model: -

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: -

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.

5. Machine Learning: -

ML-based deep learning can simplify the task of crop breeding. Algorithms simply collect field data on plant behavior 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.

7. Python: -

Python is also being used for developing the IoT devices. AI is assisting IoT in enabling real-time data analytics to help make informed decisions to farmers. Precision agriculture or smart Agriculture relies on emerging technologies such as AI, ML and data analytics to revolutionize farming practices.

8. Workflow project: -

Workflow management in agricultural systems for crop prediction involves the efficient coordination and automation of tasks and processes related to crop cultivation, monitoring.

8.1. 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.

8.2. 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.

8.3. 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.

8.4. 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: -

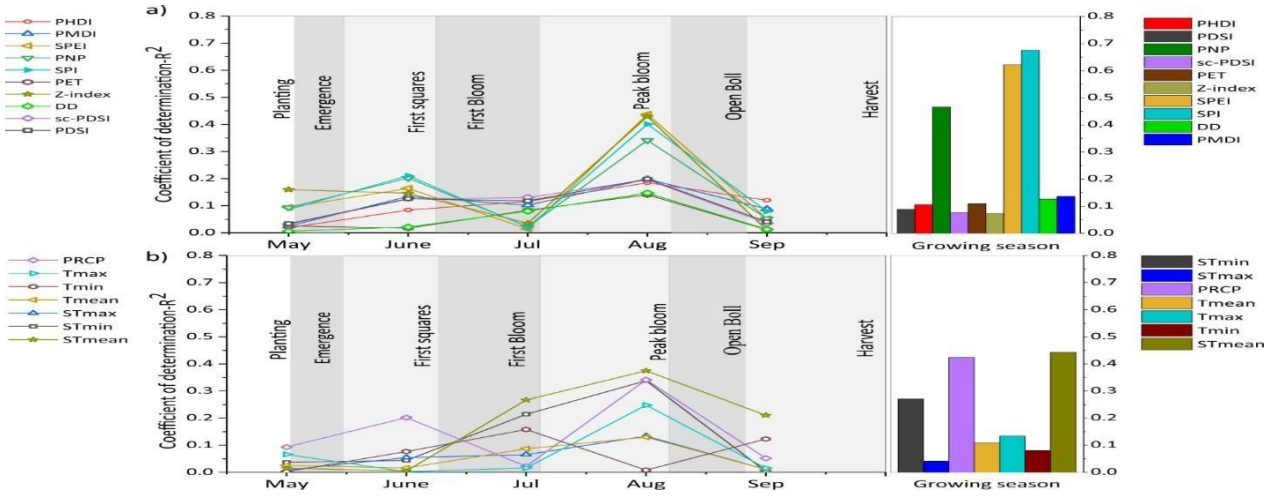

Figure 3

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 groups in 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:-

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	..	..	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

[2200 rows x 8 columns]

10.1. 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.

10.2. 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.

10.3. 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.

10.4. 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.

10.5. 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.

11. Predictions of Crops: -

As an AI language model, I can provide predictions about potential trends and factors that may impact crop production in the future. 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: -

11.2. Vertical farming: -

Vertical farming, which involves growing crops indoors in stacked layers 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:-

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

12.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.

12.1. 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.

12.2. 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.

12.3. 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.

12.4. 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/Predict	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.

A) 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 Python:-

```
import numpy as np import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
# Load your dataset
```

```
# X is the feature matrix, y is the target variable X, y = load_your_dataset()
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the logistic regression model
logreg =LogisticRegression()
# Train the model logreg.fit(X_train, y_train)
# Make predictions on the test set y_pred = logreg.predict(X_test)
# Create a confusion matrix
cm =confusion_matrix(y_test, y_pred)
# Extract values from the confusion matrix tn, fp, fn, tp = cm.ravel()
# Print the confusion matrix print("Confusion Matrix:")
print("True Negatives (TN):", tn)
print("False Positives (FP):", fp)
print("False Negatives (FN):", fn)
print("True Positives (TP):", tp)
# You can also visualize the confusion matrix using a heatmap
import seaborn as sns
import matplotlib.pyplot as plt
# Create a heatmap of the confusion matrix
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

B) Confusion Matrix using K-means: -

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:-

B.1. 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.

B.2. 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.

B.3. 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.

B.4. 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.

B.5. 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:-

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Step 1: Perform K-means clustering
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_train) # X_train is your training data

# Step 2: Train a classifier
X_train_kmeans = kmeans.transform(X_train)
X_test_kmeans = kmeans.transform(X_test) # X_test is your test data
clf = LogisticRegression()
clf.fit(X_train_kmeans, y_train) # y_train is your true class labels for training data

# Step 3: Make predictions
y_pred = clf.predict(X_test_kmeans)

# Step 4: Create a confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)
# y_test is your true class labels for test data, y_pred is the predicted class labels

# Step 5: Calculate performance metrics
accuracy = (confusion_mat[0, 0] + confusion_mat[1, 1]) / np.sum(confusion_mat)
precision = confusion_mat[1, 1] / (confusion_mat[1, 1] + confusion_mat[0, 1])
recall = confusion_mat[1, 1] / (confusion_mat[1, 1] + confusion_mat[1, 0])
f1_score = 2 * (precision * recall) / (precision + recall)
print("Confusion Matrix:\n", confusion_mat)
print("Accuracy: {:.2f}".format(accuracy))
print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1 Score: {:.2f}".format(f1_score))
```

13. Classification Report using Logistic Regression: -

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

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Load your dataset
# Replace X and y with your own features and target variable X, y = load_your_dataset()
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the logistic regression model
clf = LogisticRegression() clf.fit(X_train, y_train)

# Make predictions on the testing set y_pred = clf.predict(X_test)
# Generate the classification report
report = classification_report(y_test, y_pred)

# Print the classification report
print(report)
```

The classification_report() function from sklearn.metrics 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.

14. Source Code and Output: -

Source Code: -

```
from google.colab import files
uploaded = files.upload()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ipywidgets import interact data=pd.read_csv("data (1) (1).csv")
print(data)
print(data.isnull().sum())
sns.heatmap(data.isnull())
plt.show()
```

Output:-

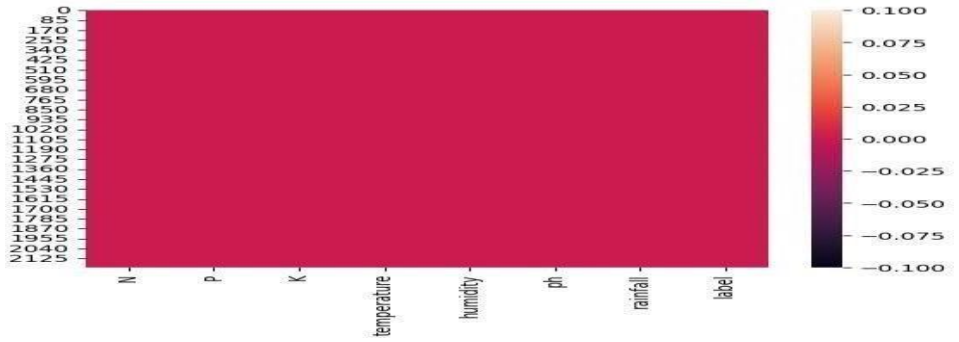

Figure 4

```
print("Avg nitrogen {0:.2f}".format(data["N"].mean()))
print("Avg phosphorus {0:.2f}".format(data["P"].mean()))
print("Avg Potassium {0:.2f}".format(data["K"].mean()))
print("Avg temperature {0:.2f}".format(data["temperature"].mean()))
print("Avg humidity {0:.2f}".format(data["humidity"].mean()))
print("Avg ph {0:.2f}".format(data["ph"].mean()))
print("Avg rainfall {0:.2f}".format(data["rainfall"].mean()))
```

Output:-

Avg nitrogen 50.55

Avg phosphorus 53.36

Avg Potassium 48.15

Avg temperature 25.62

Avg humidity 71.48

Avg ph 6.47

Avg rainfall 103.46

```
@interact
def summary(crops=list(data["label"].value_counts().index)):
x=data[data['label']==crops]
print(x['label'])
print("Min nitrogen required",x["N"].min())
print("Avg nitrogen required",x["N"].mean())
print("Max nitrogen required",x["N"].max())
print("Min phosphorus required",x["P"].min())
print("Avg phosphorus required",x["P"].mean())
print("Max phosphorus required",x["P"].max())
print("Min Potassium required",x["K"].min())
print("Avg Potassium required",x["K"].mean())
print("Max Potassium required",x["K"].max())
print("Min temperature required",x["temperature"].min())
print("Avg temperature required",x["temperature"].mean())
print("Max temperature required",x["temperature"].max())
print("Min ph required",x["ph"].min())
print("Avg ph required",x["ph"].mean())
print("Max ph required",x["ph"].max())
print("Min humidity required",x["humidity"].min())
print("Avg humidity required",x["humidity"].mean())
print("Max humidity required",x["humidity"].max())
print("Min rainfall required",x["rainfall"].min())
print("Avg rainfall required",x["rainfall"].mean())
print("Max rainfall required",x["rainfall"].max())
```

Output:-

```
[ ]        crops | rice
      Maximum N required 99
      Minimum N required 60
      Avg N required 79.89
      ---------------------------------
      --
      Maximum P required 60
      Minimum P required 35
      Avg P required 47.58
      ---------------------------------
      --
      Maximum K required 45
      Minimum K required 35
      Avg K required 39.87
      ---------------------------------
      --
      Maximum temperature required
      26.92995077
      Minimum temperature required
      20.0454142
      Avg temperature required
      23.6893322105
      ---------------------------------
      --
      Maximum rainfall required
      298.5601175
      Minimum rainfall required
      182.5616319
      Avg rainfall required
      236.18111359399998
      ---------------------------------
```

```
      Maximum ph required 7.868474653
      Minimum ph required 5.005306977
      Avg ph required
      6.425470922139999
      ---------------------------------
      --
      Maximum humidity required
      84.96907151
      Minimum humidity required
      80.12267476
      Avg humidity required
      82.27282153889999
      ---------------------------------
```

```
plt.subplot(3,4,1)
sns.histplot(data['N'],color="green")
plt.xlabel("Nitrogen")
plt.grid()

plt.subplot(3,4,2)
sns.histplot(data['P'],color="red")
plt.xlabel("P")
plt.grid()

plt.subplot(3,4,3)
sns.histplot(data['K'],color="yellow")
```

```
plt.xlabel("K")
plt.grid()

plt.subplot(3,4,4)
sns.histplot(data['ph'],color="blue")
plt.xlabel("PH")
plt.grid()

plt.subplot(2,4,5)
sns.histplot(data['temperature'],color="yellow")
plt.xlabel("temperature")
plt.grid()

plt.subplot(2,4,6)
sns.histplot(data['humidity'],color="green")
plt.xlabel("humidity")
plt.grid()

plt.subplot(2,4,7)
sns.histplot(data['rainfall'],color="blue")
plt.xlabel("rainfall")
plt.grid()

#Elbow method
from pandas.core.common import random_state
from sklearn.cluster import KMeans
x=data.drop(['label'],axis=1)
x=x.values wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i,init="k-means++", max_iter=2000,n_init=10,random_state=0)
    km.fit(x)
    wcss.append(km.inertia_)
plt.plot(range(1,11),wcss)
plt.show()
```

```
km=KMeans(n_clusters=4,init="k-means++", max_iter=2000,n_init=10,random_state=0)
y_means=km.fit_predict(x)
a=data['label']
y_means=pd.DataFrame(y_means)
z=pd.concat([y_means,a],axis=1)
z=z.rename(columns={0:'cluster'})

print("Cluster 1",z[z['cluster']==0]['label'].unique())
print("Cluster 2",z[z['cluster']==1]['label'].unique())
print("Cluster 3",z[z['cluster']==2]['label'].unique())
print("Cluster 4",z[z['cluster']==3]['label'].unique())
y=data['label']

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=0)
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(np.array([[40,40,40,40,100,7,200]]))
print(y_pred)
y_pred=model.predict(x_test)
from sklearn.metrics
import classification_report
cr=classification_report(y_test,y_pred)
print(cr)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
print(cm)
```

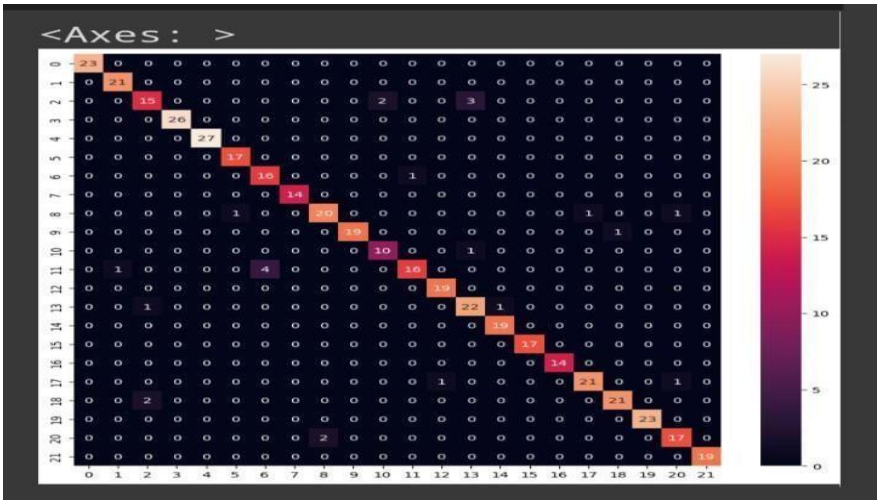

Figure 5

```
from sklearn.cluster import KMeans
x=data.drop(['label'],axis=1)
x=x.values plt.rcParams['figure.figsize']=(10,4)
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i,init='k-means++',max_iter=2000,n_init=10,random_state=0)
    km.fit(x)
    wcss.append(km.inertia_)
plt.plot(range(1,11),wcss)
plt.xlabel("No of cluster")
plt.ylabel("wcss")
plt.show()
```

Output:-

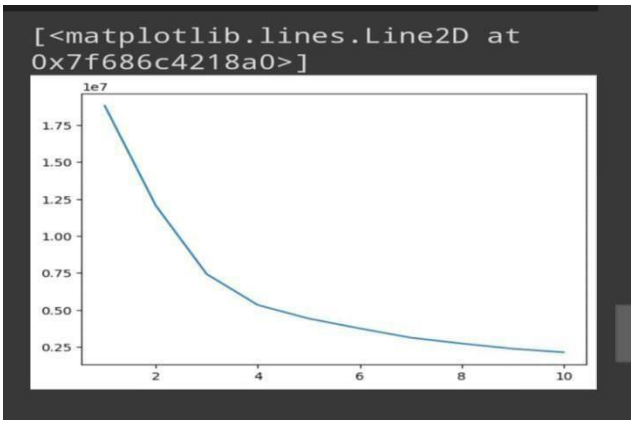

Figure 6

15. 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.

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.

16. Future Work: -

- Enhanced Feature Engineering
- Model Optimization
- Time Series Analysis
- Real-Time Predictions
- User Interface and Accessibility
- Cross-Domain Collaboration

17. Reference: -

Alston, J.M., G.W.Norton, and P.G.Pardey. 1995. Science Under Scarcity: Principles and Practice for Agricultural Research Evaluation and Priority Setting. Ithaca, N.Y.: Cornell University Press.

Ashby, J.A., and L.Sperling. 1995. Institutionalizing participatory, client-driven research and technology development in agriculture. Development and Change 26:753–750.

Batte, M. T., E.Jones, and G. D.Schnitkey. 1990. Computer use by Ohio commercial farmers. American Journal of Agricultural Economics 72(4):935–945.

Abid, S., Nasir, J. , Anwar, M. Z. & Zahid, S. (2018), 'Exponential Growth Model for Forecasting of Area and Production of Potato Crop in Pakistan', Pakistan Journal of Agricultural Research 31(1).

Abidoye, B. O., Herriges, J. A. & Tobias, J. L. (2012), 'Controlling for Observed and Unobserved Site Characteristics in RUM Models of Recreation Demand', American Journal of Agricultural Eco-nomics 94(5), 1070–1093.

Abler, D. (2015), 'Economic Evaluation of Agricultural Pollution Control Options for China', Journal of Integrative Agriculture 14(6), 1045–1056.

Babcock, B. A. (1992), 'The Effects of Uncertainty on Optimal Nitrogen Applications', Review of Agricultural Economics 14(2), 271–280.