

REAL TIME RESEARCH PROJECT ON
PREDICTIVE CROP YIELD ESTIMATION: MACHINE
LEARNING COMPARISON CLASSIFIER COMPARISON
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Submitted By

GUNDA VIDYA SRI – 22RA1A6693

RANDHI RAM KRIAN-22RA1A6659

AELLA VISHNU VARDHAN – 22RA1A6695

MADUGU YUVARAJ – 22RA1A6697

Under the guidance of

MR. T. RAVINDER

ASSISTANT PROFESSOR



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)



DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING (ARTIFICIAL INTELLIGENCE AND
MACHINE LEARNING)

KOMMURI PRATAP REDDY INSTITUTE OF
TECHNOLOGY (UGC -AUTONOMOUS ,

AFFILIATED TO JNTUH, GHANPUR(V),
GHATKESAR(M),
MEDCHAL(D)-501301)

2023- 2024

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(UGC - Autonomous Affiliated to JNTUH, GHANPUR (V), GHATKESAR (M), MEDCHAL (D)-501301)



CERTIFICATE

This is to certify that the project work entitled **“PREDICTIVE CROP YIELD ESTIMATION: MACHINE LEARNING COMPARISON CLASSIFIER COMPARISON”** is submitted by M s . G U N D A VIDYA SRI 22RA1A6693, M r .RANDHI RAM KRIAN-22RA1A6659, M r .ARELLA VISHNU VARDHAN – 22RA1A6695Mr. MADUGU YUVA RAJ – 22RA1A6697 in B-Tech II-II semester **Computer Science and Engineering –AI&ML** is a recorded bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Examiner

HOD

Co-ordinator

External Examiner

DECLARATION

We here by M s . GUNDA VIDYASRI 22RA1A6693, M r .RANDHI RAM KRIAN -22RA1A6659, M r .ARELLA VISHNU VARDHAN – 22RA1A6695Mr. MADUGU YUVARAJ – 22RA1A6697 by declare that the results embodied in this project dissertation entitled “PREDICETIVE CROP YIELD ESTIMATION : MACHINE LEARNING COMPARISION CLASSIFIER COMPARISION” is carried out by us during the year 2024 in partial fulfillment of the award of Bachelor of Technology in Computer Science & Engineering(AI&ML) from Kommuri pratap Reddy Institute of Technology. It is an authentic record carried out by us under the guidance of T.Ravinder , Associate Professor, Department of COMPUTER SCIENCE & ENGINEERING (AI&ML).

Date:

Place:

BY

M s . G U N D A VIDYA SRI- 22RA1A6693

Mr.RANDHI RAM KRIAN-22RA1A6659

Mr.A.VISHNU VARDHAN – 22RA1A6695

Mr. MADUGU YUVARAJ – 22RA1A6697

ACKNOWLEDGEMENT

It gives us immense pleasure to acknowledge with gratitude, the help and support extended throughout the project report from the following:

We will be very much grateful to almighty, parents who made us capable of carrying out our job.

We express our profound gratitude to **Dr.E.RAVINDRA, Principal of Kommuri Pratap Reddy Institute of Technology**, who has encouraged in completing our project report successfully.

We are grateful to **Dr. J . SRIKANTH** who is our **Head of the Department, AI&ML** for his amiable ingenious and adept suggestions and pioneering guidance during the project report.

We express our gratitude and thanks to the Project Coordinator Dr. S.Sankar ganesh Assistant Professor of our department for his contribution for making it success with in the given time duration.

We express our deep sense of gratitude and thanks to **Internal Guide, T.Ravinder , Assistant Professor**, for his guidance during the project report.

We are also very thankful to our **Management, Staff Members** and all **Our Friends** for their valuable suggestions and timely guidance without which we would not have been completed it.

BY

M s . GUNDA VIDYA SRI- 22RA1A6693

Mr . RANDHI RAM KRIAN-22RA1A6659

Mr. ARELLA VISHNU VARDHAN – 22RA1A6695

Mr. MADUGU YUVARAJ – 22RA1A6697



KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VISION OF THE INSTITUTE

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the Nation.

MISSION OF THE INSTITUTE

Mission	Statement
IM₁	To have holistic approach in curriculum and pedagogy through industry interface to meet the needs of Global Competency.
IM₂	To develop students with knowledge, attitude, employability skills, entrepreneurship, research potential and professionally Ethical citizens.
IM₃	To contribute to advancement of Engineering & Technology that would help to satisfy the societal needs.
IM₄	To preserve, promote cultural heritage, humanistic values and Spiritual values thus helping in peace and harmony in the society.



KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VISION OF THE DEPARTMENT

To Provide Quality Education in Computer Science for the innovative professionals to work for the development of the nation.

MISSION OF THE DEPARTMENT

Mission	Statement
DM1	Laying the path for rich skills in Computer Science through the basic knowledge of mathematics and fundamentals of engineering
DM2	Provide latest tools and technology to the students as a part of learning Infrastructure
DM3	Training the students towards employability and entrepreneurship to meet the societal needs.
DM4	Grooming the students with professional and social ethics



**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO'S	Statement
PEO1	Demonstrate technical skills, competency in AI&ML and exhibit team management capability with proper communication in a job environment.
PEO2	Support the growth of economy of a country by starting enterprise with a lifelong learning attitude.
PEO3	Carry out research in the advanced areas of AI&ML and address the basic needs of the society.
PEO4	To develop research attitude among the students in order to carry out research in cutting edge technologies, solve real world problems and provide technical consultancy services.

PROGRAM OUTCOMES

PO1	Engineering Knowledge: Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex Engineering problems.
PO2	Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data , synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethicsand responsibilities and norms of the engineering practice.

PO9	Individual and team network: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make Effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work as a member and leader in a team, to manage projects and in multidisciplinary environment.
PO12	Life-Long learning: Recognize the need for, and have the preparation and able to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1	Demonstrate the knowledge of human cognition, Artificial Intelligence, Machine Learning and data engineering for designing intelligence systems.
PSO2	Apply computational knowledge and project development skills to provide innovative solutions.
PSO3	Use tools and techniques to solve problems in AI&ML.



**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

TABLE OF CONTENTS

ABSTRACT	1
1. INTRODUCTION.....	2
HISTORY	
PROBLEM STATEMENT	
.....	
BACKWARDS OF TELEMEDICONE SYSTEMS.....	
LIMITATIONS OF MACHINE LEARNING ALGORITHMS	
APPLICATIONS ON TELEMEDICINE	
DEPENDENCY ON STATIC DESCRIPTIONS	
HUMAN SUBJECTIVITY IN CATEGORIZATION	
NEED FOR REAL-TIME PRECISION	
UNDERUTILIZATION OF TECHNOLOGICAL ADVANCES	
RISK TO CUSTOMER SATISFACTION	
RESEARCH MOTIVATION	
APPLICATIONS	
PRECISION AGRICULTURE.....	
RISK MANAGEMENT.....	
SUPPLY CHAIN MANAGEMENT.....	
POLICY FORMULATION.....	
RESEARCH AND DEVELOPMENT	
2. LITERATURE SURVEY	6
3. EXISITING SYSTEMS.....	9
ORTHOGONALMACHINE PERSUIT (OMP)	
DRAWBACKS OF OMP	
4. PROPOSED METHODOLOGY	13
OVERVIEW	
DATA IMPORT AND PROCESSING.....	

MODEL TRAINING	
MODEL EVALUATION.....	
MODEL PERSISTANCE	
PERFORMANCE COMPARISION	
MODEL PREDICTION ON TEST DATA.....	
BLOCK DIAGRAM OF PROPOSED SYSTEM	
DATA PREPROCESSING	
GETTING THE DATASET	
IMPORTING LIBRARIES.....	
IMPORTING DATASETS	
FINDING MISSING DATA.....	
ENCODING CATAGEORICAL DATA	
SPITTING DATASET INTO TRAINING AND TEST SET	
ADVANTAGES OF CALIBRATEDCLASSIFIER.....	
IMPROVED DECISION-MAKING.....	
ENHANCED MODEL TRUSTWORTHINESS.....	
BETTER PERFORMACE METRICS.....	
APPLICABILITY ACROSS MODELS	
UNITY IN POST-PREPROCESSING	
VERSATILITY.....	
5. SOFTWARE ENVIRONMENT.....	24
PROGRAMMING LANGUAGES USED	
PROGRAMMING LIBRARIES USED.....	
ENVIRONMENT CREATION	
6.SYSTEM REQRIMENTS.....	38
SOFTWARE REQRIMENT	
HARDWARE RWQRIMENT	
7. FUNCTIONAL REQRIMENT.....	39
OUTPUT DESIGN	
OUTPUT DEFINATION	
USER INTERFACE	
8. SOURCE CODE	44
SOURCE CODE	

9. RESULT DESCUSION	51
SOURCE CODE	
10. CONCLUSION AND FUTURE SCOPE	62
FUTURE SCOPE	
REFERENCES	

PREDICTIVE FOR CROP YIELD ESTIMATION: MACHINE LEARNING CLASSIFIER COMPARISION

ABSTRACT

The aims to enhance agricultural productivity through advanced predictive modeling. Leveraging machine learning techniques, the project seeks to analyze and compare various classifiers to accurately estimate crop yields. By harnessing historical data, weather patterns, soil conditions, and other relevant factors, the goal is to develop a robust framework capable of forecasting crop yields with high precision. Traditional methods of crop yield estimation often rely on manual observation and historical trends. However, these approaches are often time-consuming, labor-intensive, and prone to inaccuracies due to the complex and dynamic nature of agricultural systems. Additionally, they may not fully utilize the available data or adapt to changing environmental conditions, leading to suboptimal predictions. The need for a more efficient and accurate crop yield estimation system that can leverage the power of machine learning algorithms. It aims to overcome the limitations of traditional methods by developing a predictive model capable of analyzing large datasets, identifying patterns, and making precise yield predictions based on various factors such as weather, soil conditions, and agricultural practices. The motivation behind this project lies in the potential impact it can have on agriculture. Accurate crop yield predictions enable farmers to make informed decisions regarding planting, harvesting, and resource allocation, leading to improved productivity and resource utilization. By harnessing the capabilities of machine learning, the project seeks to empower farmers with valuable insights for better crop management and yield optimization. The proposed system involves the integration of machine learning classifiers to analyze historical agricultural data and predict crop yields. By leveraging advanced algorithms, the system aims to improve the accuracy and efficiency of crop yield estimation, providing valuable insights to farmers and stakeholders. Additionally, the system will facilitate decision-making processes, ultimately contributing to sustainable agricultural practices and food security.

CHAPTER 1

INTRODUCTION

1.1 History:

Crop yield estimation has long been a critical aspect of agricultural planning and management. Traditionally, farmers relied on manual observation, historical data, and basic statistical methods to predict crop yields. However, these approaches were often limited in accuracy and efficiency, as they could not fully capture the complex and dynamic nature of agricultural systems. With the advent of technology, particularly in the field of data science and machine learning, there emerged a new opportunity to revolutionize crop yield estimation.

In recent years, the application of machine learning techniques in agriculture has gained significant attention. Researchers began exploring the potential of leveraging advanced algorithms to analyze vast amounts of data and extract valuable insights for crop yield prediction. This marked a shift towards more data-driven and predictive approaches, moving away from reliance on traditional methods.

As machine learning algorithms evolved and computing power became more accessible, the feasibility of developing sophisticated predictive models for crop yield estimation increased. Researchers started experimenting with various machine learning techniques, such as decision trees, random forests, support vector machines, and neural networks, to determine their effectiveness in predicting crop yields accurately.

Furthermore, the availability of large-scale agricultural datasets, including information on weather patterns, soil characteristics, crop types, and agricultural practices, provided researchers with valuable resources for training and validating predictive models. These datasets served as the foundation for building robust frameworks capable of generating precise crop yield forecasts.

1.2 Research Motivation:

The motivation behind the project "Predictive For Crop Yield Estimation: Machine Learning Classifier Comparison" stems from the pressing need to enhance agricultural productivity and

sustainability. Agriculture plays a crucial role in global food security and economic development, yet it faces numerous challenges, including climate change, population growth, resource constraints, and environmental degradation. In this context, accurate crop yield estimation is essential for informing decision-making processes and optimizing resource allocation.

Traditional methods of crop yield estimation often rely on manual observation and historical trends, which are labor-intensive, time-consuming, and prone to inaccuracies. These methods may not fully leverage the available data or adapt to changing environmental conditions, leading to suboptimal predictions. Moreover, the increasing complexity of agricultural systems necessitates more sophisticated analytical tools capable of processing large volumes of data and identifying intricate patterns.

Machine learning offers a promising solution to these challenges by enabling the development of predictive models that can analyze diverse datasets and generate accurate forecasts. By harnessing advanced algorithms, researchers can extract valuable insights from complex agricultural data, including information on weather, soil conditions, crop characteristics, and management practices. These insights empower farmers and policymakers to make informed decisions regarding planting, harvesting, irrigation, fertilization, pest control, and resource allocation, ultimately improving productivity and sustainability.

The motivation behind this project lies in the potential impact it can have on agriculture. By comparing and evaluating different machine learning classifiers for crop yield estimation, the project aims to identify the most effective approaches for accurately predicting yields across various crops and regions. This research has the potential to advance the field of agricultural analytics and contribute to the development of more efficient and scalable solutions for addressing the challenges faced by farmers worldwide.

1.3 Problem Statement:

The problem statement for the project "Predictive For Crop Yield Estimation: Machine Learning Classifier Comparison" revolves around the need to develop accurate, efficient, and scalable methods for predicting crop yields. Traditional methods of crop yield estimation are often limited in accuracy and scalability, relying on manual observation and historical trends. These methods may not fully leverage the available data or adapt to changing environmental conditions, leading to suboptimal predictions.

Moreover, the increasing complexity of agricultural systems, coupled with the impacts of climate change and resource constraints, further exacerbates the challenges associated with crop yield estimation. Farmers and policymakers require more sophisticated analytical tools capable of processing large volumes of data and generating precise forecasts to inform decision-making processes and optimize resource allocation.

Machine learning presents a promising solution to these challenges by enabling the development of predictive models that can analyze diverse datasets and extract valuable insights for crop yield estimation. However, the effectiveness of machine learning algorithms in this context depends on various factors, including the choice of classifier, feature selection, data preprocessing techniques, and model evaluation metrics.

The problem addressed by this project is twofold: first, to compare and evaluate different machine learning classifiers for crop yield estimation, and second, to develop a robust framework capable of accurately predicting yields across various crops and regions. By addressing these challenges, the project aims to advance the field of agricultural analytics and contribute to the development of more efficient and scalable solutions for addressing the challenges faced by farmers worldwide.

1.4 Applications:

- **Precision Agriculture:** Accurate crop yield estimation enables precision agriculture practices, where farmers can optimize resource allocation, including water, fertilizers, and pesticides, based on predicted yield patterns. This leads to improved efficiency, reduced costs, and minimized environmental impact.
- **Risk Management:** Predictive crop yield models help farmers and stakeholders mitigate risks associated with weather fluctuations, pests, diseases, and market variability. By identifying potential yield losses in advance, farmers can implement proactive strategies to minimize adverse impacts on their livelihoods.
- **Supply Chain Management:** Accurate crop yield forecasts facilitate better supply chain management, allowing stakeholders to anticipate fluctuations in crop availability and plan logistics accordingly. This leads to smoother operations, reduced waste, and enhanced market stability.
- **Policy Formulation:** Policymakers can use crop yield predictions to formulate evidence-based policies related to agricultural subsidies, food security, land use

planning, and climate change adaptation. By understanding future yield trends, policymakers can implement interventions to support sustainable agriculture and rural development.

- **Research and Development:** Crop yield estimation models serve as valuable tools for agricultural research and development, enabling scientists to study the impacts of different factors on crop productivity and explore innovative farming techniques. This fosters continuous improvement and innovation within the agricultural sector.

CHAPTER 2

LITERATURE SURVEY

Aquet et al. [21] proposed two separate Machine Learning (ML) algorithms to evaluate the yield of crops. The algorithms Support Vector Regression (SVR) and Linear Regression (LR) have been well suited for validating variable parameters in continuous changeable estimation with 140 data points. The mentioned parameters are important determinants for crop yield. The error rate has been calculated for using Coefficient of Determination (R^2) and Mean Square Error (MSE) with MSE yielding around 0.005 and R^2 yielding around 0.86. The same dataset has been used to compare the performance of the algorithms quickly. Nishant et al. [22] proposed web application to forecast the yield of nearly all types of crops grown in India. This study is unique because it uses simple parameters such as state, district, season, and region to predict crop yields in any year the user desires. To predict the yield, advanced regression techniques such as Kernel Ridge, Lasso, and ENet algorithms are used in the paper, as well as the principle of Stacking Regression to improve the algorithms. The root means the square error is the output metric that used in this work. The models when have been implemented individually, ENet had an error of about 4%, Lasso had an error of about 2% and Kernel Ridge had an error of about 1%, and after stacking it was less than 1%. Kalimuthu et al. [23] suggested a smartphone application for Android, which uses machine learning, which is one of the most advanced crops prediction technologies, to direct beginner farmers in sowing the appropriate crops. The naive Bayes algorithm proposes a method for doing so. The data seed of the crops is collected, along with the suitable parameters such as humidity, moisture, and temperature content, that is aids the crops' development. To begin the prediction process, users are encouraged to input parameters such as their location and temperature this application will take it automatically to start the process of prediction. Y. J. N. Kumar et al. [24], implemented prediction system on crop production from the collecting of past data. Crop yield is estimated using data mining techniques. They used the Random Forest algorithm to forecast the highest yield crop as a product. Crops yield predictions are often appropriate in the agricultural sector. The higher the accuracy, the higher the benefit on the crop yield. Farmers will use the proposed technique to help them decide which crop to plant in their fields. Under this system would cover the widest range of crops possible. Farmers in India can benefit from accurate forecasting of various crops across various districts. Gupta et al. [25] used IoT and data mining in monitoring applications to make smart farming possible. Smart farming is a method of providing all of the services

needed for a specific time. Soil moisture, light intensity, relative humidity, soil pH reading, and ambient temperature are all resources that are needed. They demonstrated the transformation of a device capable of gathering data from sensors using IoT in the agriculture field. This device successfully senses data and sends it locally to the thing speak cloud, which the user can then access via his or her custom website. The data mining techniques such as SVM, KNN and Random forest are used to crop-producing with the correct number of resources so that the farmer still has the upper hand, and by comparing the current pattern to the previous one, will be able to determine whether or not a parameter is right. All of these agricultural process values are monitored on a user-defined platform. Terliksiz & Altıylar [26] used a 3D CNN model that leverages spatiotemporal features, a soybean yield prediction for Lauderdale County, Alabama, USA has been presented. From 2003 to 2016, the yield has been taken from the USDA NASS Fast Stat tool. Google Earth Engine was used to gather satellite data from (NASA's MODIS) land products surface reflectance and land surface temperature. For comparison of the results, the root means squared error (RMSE) has been used as the measurement metric. When 64x64 data frames with more than 20% cropland coverage were used without dropout, the best result for Lauderdale County, Alabama was obtained. The RMSE is 0.81 and the error is 2.70% in this situation. Khosla et al., [27] The Kharif crops yield were estimated in 2 steps; first, the rainfall has been estimated by using a modular artificial neural network MANN, and then the yield was predicted by using support vector regression SVR. The dataset of experimental included data from the year 2000 to 2016 and generated results for the years 2018 and 2019. Nigam et al. [28] utilized machine learning techniques such as KNN classifier, Random Forest, Artificial Neural Network, liner Regression and XGboost to predict yield crop. Based on the Mean Absolute Error MAE, the results of these techniques are compared. An algorithm of a machine learning algorithm can be help farmers to determine which crops planting to get the best yield by taking into account factors such as temperature, rainfall, area, ...etc. When all parameters are combined, the results show that the best classifier is Random Forest. S & R [29] determined most of the important points for accurate Crop's yield Prediction. For improved accuracy, the algorithm of machine learning namely Support Vector Machine, KNN, Regression, Random Forest and Artificial Neural Network have been proposed. The agricultural dataset contains 745 instances, 30% of data is used to test the prediction performance of the models, while 70% of the data is chosen at random and used to train the model. The results show that, by using the same farming training data, the RF algorithm achieves maximum precision employing its error analysis values for all the separated feature sub-sets. Kumar et al. [30] used the three supervised techniques SVM, KNN and Least Squared

Support Vector Machine are. It is a comparative study that shows the training proposed model's accuracy and error rate. They have three different datasets: soil, rainfall, and yield. They then combined the datasets and used the techniques to determine the actual approximate cost as well as the 121 accuracy of the techniques used. The training model's accuracy will be higher and the error rate will be low. The technique with the highest accuracy rate (LS SVM =90%) and the lowest error rate (LS SVM=0,0362).

CHAPTER 3

EXISTING SYSTEM

3.1 Orthogonal Matching Pursuit (OMP)

Orthogonal Matching Pursuit (OMP) is a greedy algorithm used in signal processing and statistics to solve sparse approximation problems. The goal of OMP is to represent a signal as a sparse linear combination of basis vectors (atoms) from a dictionary. OMP is widely used in applications such as compressed sensing, image processing, and machine learning due to its simplicity and effectiveness in handling high-dimensional data.

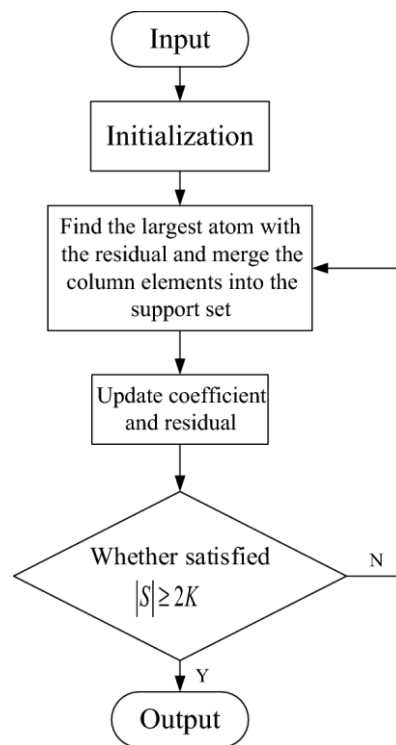


Fig. 1 Orthogonal Matching Pursuit (OMP) algorithm.

How OMP Works

The OMP algorithm iteratively selects the dictionary atoms that best match the residual part of the signal. The process involves the following steps:

1. Initialization:

- Start with an initial residual equal to the signal y .
- Initialize an empty set to store the selected atoms.
- Initialize the solution vector $x=0$.

2. Iteration:

- **Step 1: Compute Correlations:**

- Compute the correlations between the residual and all the atoms in the dictionary \mathbf{D} .
- The correlation vector is given by:

$$\mathbf{C} = \mathbf{D}^T \mathbf{r}$$

- where \mathbf{r} is the current residual.

- **Step 2: Select Atom:**

- Select the atom with the highest absolute correlation. This atom is the one most aligned with the residual.
- Update the set of selected atoms Λ to include the index of the selected atom.

- **Step 3: Solve Least Squares:**

- Solve the least squares problem to find the new coefficients for the selected atoms:

$$\mathbf{x}_\Lambda = \arg \min_{\mathbf{z}} \|\mathbf{y} - \mathbf{D}_\Lambda \mathbf{z}\|_2$$

- where \mathbf{D}_Λ is the submatrix of \mathbf{D} containing the selected atoms, and \mathbf{x}_Λ is the subvector of \mathbf{x} corresponding to the selected atoms.

- **Step 4: Update Residual:**

- Update the residual:

$$\mathbf{r} = \mathbf{y} - \mathbf{D}_\Lambda \mathbf{x}_\Lambda$$

- **Step 5: Check Stopping Criterion:**

- Repeat steps 1-4 until a stopping criterion is met. The stopping criterion can be based on the number of iterations, the norm of the residual, or the sparsity level of the solution.

3. Output:

- The output of the OMP algorithm is the sparse coefficient vector \mathbf{x} that approximates the signal \mathbf{y} as $\mathbf{y} \approx \mathbf{D}\mathbf{x}$.

Principles of OMP

OMP is based on the principles of sparsity and orthogonality. The key ideas can be summarized as follows:

- **Sparsity:** OMP seeks a sparse solution where only a few coefficients in the representation vector \mathbf{x} are non-zero. This aligns with the idea that many natural signals can be approximated using a small number of basis vectors.
- **Greedy Selection:** The algorithm iteratively selects the dictionary atom that has the highest correlation with the residual. This greedy approach ensures that each selected atom contributes maximally to reducing the residual error.
- **Orthogonality:** Once an atom is selected, the residual is orthogonalized with respect to the selected atom. This ensures that subsequent selections are not influenced by previously selected atoms, promoting a more accurate and efficient representation.

3.2 Drawbacks

The drawbacks of Orthogonal Matching Pursuit (OMP) in pointwise format:

— Computational Complexity:

- OMP can be computationally expensive, especially for large-scale problems, due to the need to solve a least squares problem at each iteration.

— Sensitivity to Noise:

- OMP can be sensitive to noise in the data, which may lead to the selection of incorrect atoms and affect the accuracy of the solution.

— Greedy Nature:

- As a greedy algorithm, OMP does not guarantee the globally optimal solution and might converge to a suboptimal representation.

— Dependence on Dictionary Quality:

- The performance of OMP heavily depends on the quality of the dictionary. Poorly chosen dictionaries can lead to inaccurate approximations and poor convergence.

— **Stopping Criteria:**

- Determining an appropriate stopping criterion (e.g., residual norm threshold or maximum number of iterations) can be challenging and may affect the algorithm's performance and accuracy.

— **Overfitting Risk:**

- OMP may overfit the data if too many atoms are selected, especially when dealing with noisy signals or limited data.

— **Limited Applicability for Highly Correlated Dictionaries:**

- When the dictionary contains highly correlated atoms, OMP might struggle to distinguish between them, leading to less accurate or redundant selections.

— **Scalability Issues:**

- For very large dictionaries or high-dimensional data, the iterative process and least squares computations can become prohibitive in terms of memory and time requirements.

— **Lack of Theoretical Guarantees for Certain Conditions:**

- While OMP performs well under certain conditions, it lacks strong theoretical guarantees for its performance in all scenarios, especially for arbitrary dictionaries or high noise levels.

— **Non-Sparsity in Practical Applications:**

- In many real-world applications, the true signal may not be perfectly sparse, which can limit the effectiveness of OMP in approximating such signals accurately.

CHAPTER 4

PROPOSED SYSTEM

4.1 Overview

The project aims to develop a predictive model for crop yield estimation using machine learning classifiers.

- **Data Import and Preprocessing:** Importing necessary libraries and the dataset (**dataset.csv**). It conducts initial data analysis, checking info, description, correlation, and null values. Then it drops an unnamed column and encodes categorical variables. Data visualization is performed using seaborn to visualize the count plot of the target variable **yield_class**.
- **Model Training:** The dataset is split into training and testing sets using **train_test_split**. Two classifiers are trained:
 - **Orthogonal Matching Pursuit (OMP) Classifier:** An instance of **OrthogonalMatchingPursuit** from scikit-learn is used.
 - **Calibrated ClassifierCV:** It's a calibrated version of Logistic Regression, trained with **CalibratedClassifierCV** from scikit-learn.
- **Model Evaluation:** For both classifiers, performance metrics such as accuracy, precision, recall, and F1-score are calculated using a custom function **performance_metrics**. Classification report and confusion matrix are generated for each classifier to evaluate its performance visually.
- **Model Persistence:** Trained models are saved and loaded if the model files exist, using numpy's **np.save** and **np.load**.
- **Performance Comparison:** The code compares the performance of the two classifiers and presents the results in a tabular format.
- **Model Prediction on Test Data:** Another dataset (**test.csv**) is loaded for prediction. The same preprocessing steps are applied to this dataset. The calibrated classifier is then used to predict the **yield_class** of the test data. Predicted results are printed along with the corresponding input features.

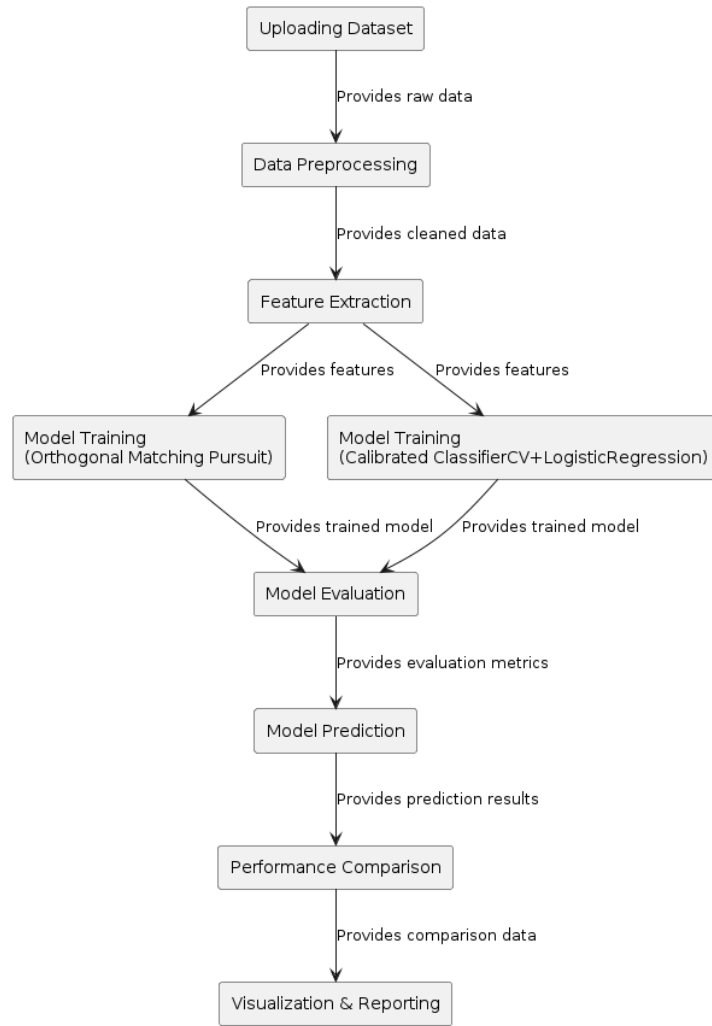


Fig. 1: Block Diagram of Proposed System.

4.2 Data Preprocessing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset

- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set

Importing Libraries: To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
import numpy as nm
```

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

```
import matplotlib.pyplot as mtp
```

Here we have used mtp as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:

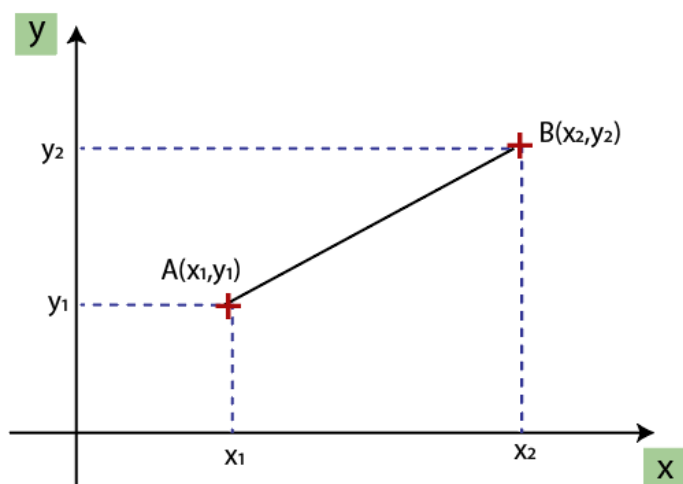
```
1 # importing libraries
2 import numpy as nm
3 import matplotlib.pyplot as mtp
4 import pandas as pd
5
```

Handling Missing data: The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. There are mainly two ways to handle missing data, which are:

- By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Encoding Categorical data: Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

Feature Scaling: Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no variable dominates the other variable. A machine learning model is based on Euclidean distance, and if we do not scale the variable, then it will cause some issue in our machine learning model. Euclidean distance is given as:



Euclidean Distance Between A and B = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Feature scaling

If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So, to remove this issue, we need to perform feature scaling for machine learning.

4.3 Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

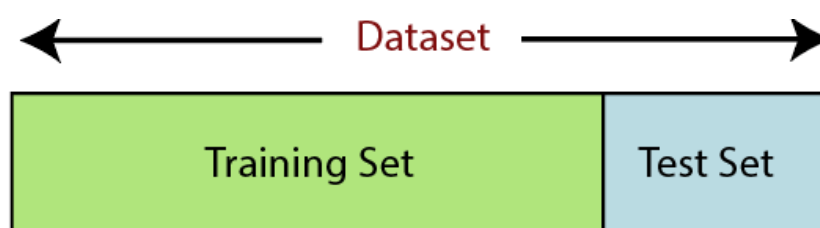


Figure 4.2: Splitting the dataset.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split  
  
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

Explanation

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
- `x_train`: features for the training data
- `x_test`: features for testing data
- `y_train`: Dependent variables for training data
- `y_test`: Independent variable for testing data
- In `train_test_split()` function, we have passed four parameters in which first two are for arrays of data, and `test_size` is for specifying the size of the test set. The `test_size` maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter `random_state` is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

4.4 Calibrated Classifier

A `CalibratedClassifier` is a method used in machine learning to improve the probability estimates provided by a classifier. Many classifiers, such as decision trees, support vector machines, or neural networks, do not naturally produce well-calibrated probabilities. Well-calibrated probabilities are those that reflect the true likelihood of an event occurring.

Calibrated classifiers adjust these probability outputs so that they can be interpreted as reliable probabilities.

How It Works

Principle of Probability Calibration

The goal of probability calibration is to ensure that the predicted probabilities match the true probabilities. For instance, if a classifier predicts a probability of 0.8 for an event occurring in 100 cases, then ideally, the event should occur in approximately 80 of those cases.

Calibration Methods

There are several methods to calibrate a classifier, with the most common being:

1. Platt Scaling:

- Platt scaling fits a logistic regression model to the classifier's output. This model uses the original classifier's scores as input and fits a sigmoid function to map these scores to calibrated probabilities.
- The sigmoid function used in Platt scaling is:

where f is the uncalibrated score, and A and B are parameters learned during the calibration process.

2. Isotonic Regression:

- Isotonic regression is a non-parametric method that fits a piecewise constant function to the classifier's output. It is used when the relationship between the classifier scores and the true probabilities is not well-represented by a sigmoid function.
- The isotonic regression ensures that the mapping is monotonic, meaning the calibrated probabilities increase with increasing classifier scores.

3. Histogram Binning:

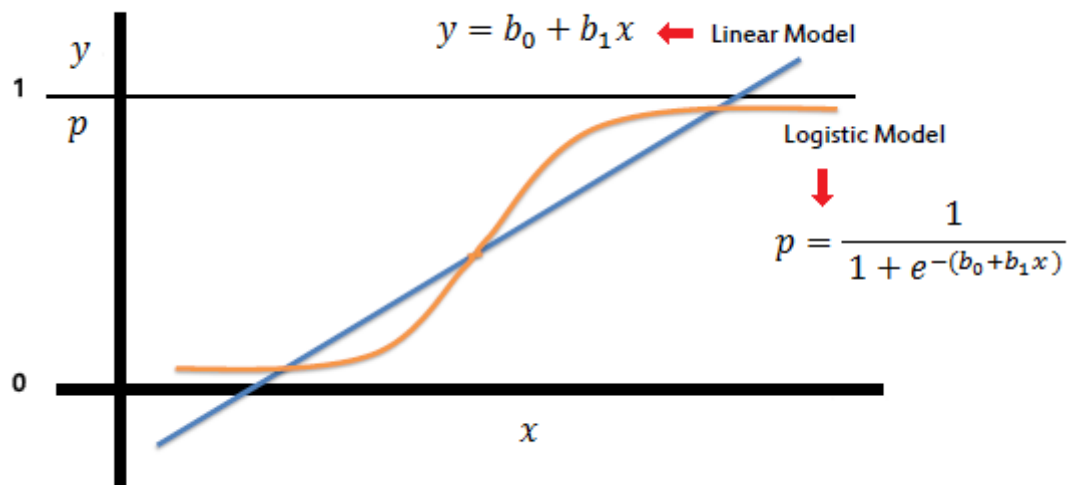
- Histogram binning divides the range of classifier outputs into bins and calculates the average true probability for each bin. This method is simple but can be less accurate if not enough data points fall into each bin.

Logistic regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e., a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g., predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant (b_0) moves the curve left and right and the slope (b_1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp(b_0 + b_1x)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b_1) is the amount the logit (log-odds) changes with a one unit change in x .

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1x_1 + b_2x_2 + \dots + b_px_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\beta^1 = \beta^0 + [X^T W X]^{-1} \cdot X^T (y - \mu)$$

β is a vector of the logistic regression coefficients.

W is a square matrix of order N with elements $n_i \pi_i (1 - \pi_i)$ on the diagonal and zeros everywhere else.

μ is a vector of length N with elements $\mu_i = n_i \pi_i$.

A pseudo R^2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square ". Wald test is used to test the statistical significance of each coefficient (b) in the model (i.e., predictors contribution).

Pseudo R2

There are several measures intended to mimic the R² analysis to evaluate the goodness-of-fit of logistic models, but they cannot be interpreted as one would interpret an R² and different pseudo R² can arrive at very different values. Here we discuss three pseudo R² measures.

Pseudo R ²	Equation	Description
Efron's	$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - p_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	<i>p'</i> is the logistic model predicted probability. The model residuals are squared, summed, and divided by the total variability in the dependent variable.
McFadden's	$R^2 = 1 - \frac{LL_{full\ model}}{LL_{intercept}}$	The ratio of the log-likelihoods suggests the level of improvement over the intercept model offered by the full model.
Count	$R^2 = \frac{\#\ Corrects}{Total\ Count}$	The number of records correctly predicted, given a cutoff point of .5 divided by the total count of cases. This is equal to the accuracy of a classification model.

Likelihood Ratio Test

The likelihood ratio test provides the means for comparing the likelihood of the data under one model (e.g., full model) against the likelihood of the data under another, more restricted model (e.g., intercept model).

$$LL = \sum_{i=1}^n y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)$$

where '*p*' is the logistic model predicted probability. The next step is to calculate the difference between these two log-likelihoods.

$$2(LL_1 - LL_2)$$

The difference between two likelihoods is multiplied by a factor of 2 in order to be assessed for statistical significance using standard significance levels (Chi² test). The degrees of freedom for the test will equal the difference in the number of parameters being estimated under the models (e.g., full and intercept).

Advantages of Calibrated Classifier

— Improved Decision-Making:

- Calibrated probabilities lead to better decision-making processes because the predicted probabilities reflect the true likelihood of events.

— Enhanced Model Trustworthiness:

- Users and stakeholders can trust the model more if it provides accurate probability estimates, especially in critical applications like healthcare or finance.

— Better Performance Metrics:

- Calibration can improve various performance metrics that depend on probability estimates, such as the Brier score, which measures the accuracy of probabilistic predictions.

— Applicability Across Models:

- Calibration techniques can be applied to any probabilistic classifier, making it a versatile tool for improving model performance.

— Utility in Post-Processing:

- Calibration can be applied after the model training, allowing for adjustments without retraining the model from scratch.

— Versatility:

- Different calibration methods (Platt scaling, isotonic regression, histogram binning) provide flexibility to choose the best method suited for the specific model and data characteristics.

CHAPTER 5

SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

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

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

10. Extensive Libraries

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

2. Extensible

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

3. Embeddable

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

4. Improved Productivity

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

5. IOT Opportunities

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

6. Simple and Easy

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

7. Readable

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

8. Object-Oriented

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

9. Free and Open-Source

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

10. Portable

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

11. Interpreted

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

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

Advantages of Python Over Other Languages

10. Less Coding

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

2. Affordable

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

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

3. Python is for Everyone

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

Disadvantages of Python

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

10. Speed Limitations

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

2. Weak in Mobile Computing and Browsers

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

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

3. Design Restrictions

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

4. Underdeveloped Database Access Layers

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

5. Simple

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

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

Modules Used in Project

NumPy

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

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

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

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

Pandas

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

Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

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

Scikit – learn

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

Install Python Step-by-Step in Windows and Mac

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

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

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python

but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4ae7b9ab01b0f09be	23017663	5/G
XZ compressed source tarball	Source release		d33e4aae6097051c2eca45ee3604803	17131432	5/G
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.8 and later	6428b4fa7583da91a440c8a1cee08e6	34898416	5/G
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c382179457738f5e4a936b243f	20082845	5/G
Windows help file	Windows		d63999573a2c56b2ac56cad6b477cd2	8131761	5/G
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	960b3cfd9ec0b1a6e53184a4072ba2	7504391	5/G
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76dbdb3543a583e543400	26880368	5/G
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608b6d73aeb53a3bd353b4bd2	1362904	5/G
Windows x86 embeddable zip file	Windows		9fab3b818b41879fda94113574139d8	6741626	5/G
Windows x86 executable installer	Windows		31cc802942a5446a3d645147e394789	25663848	5/G
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	5/G

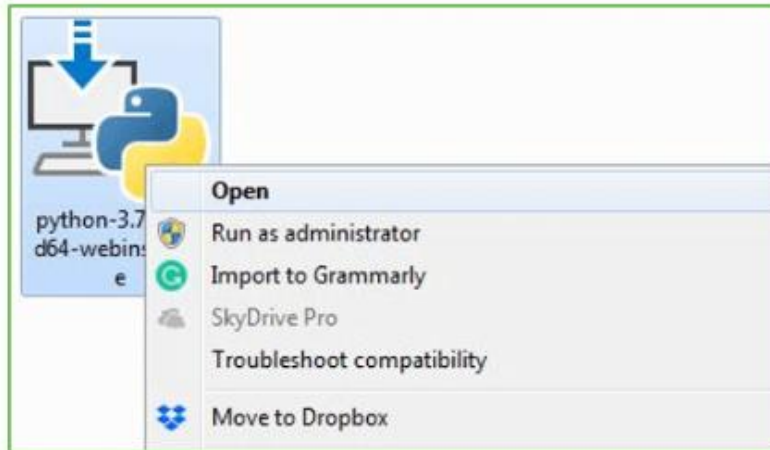
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



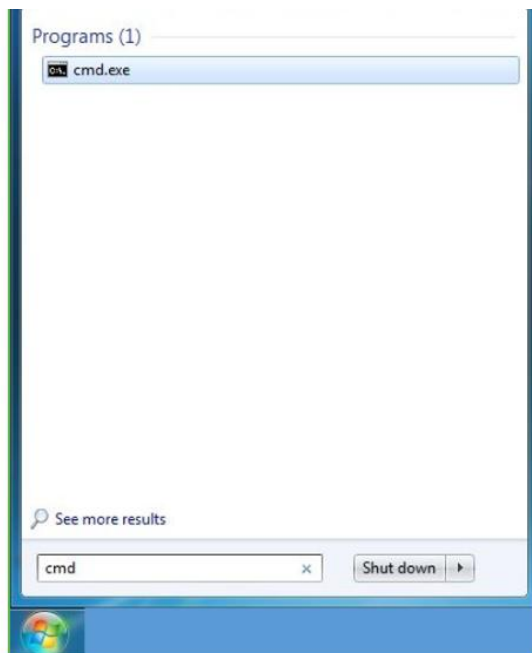
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

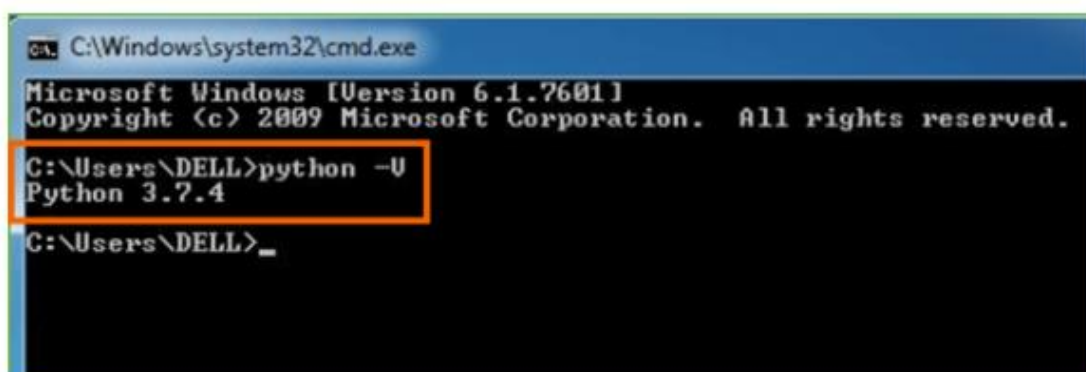
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



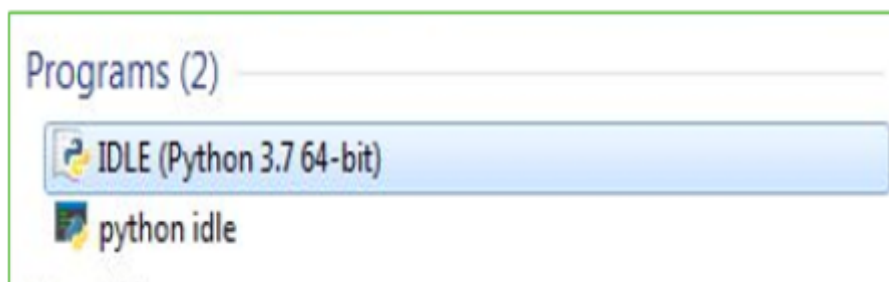
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

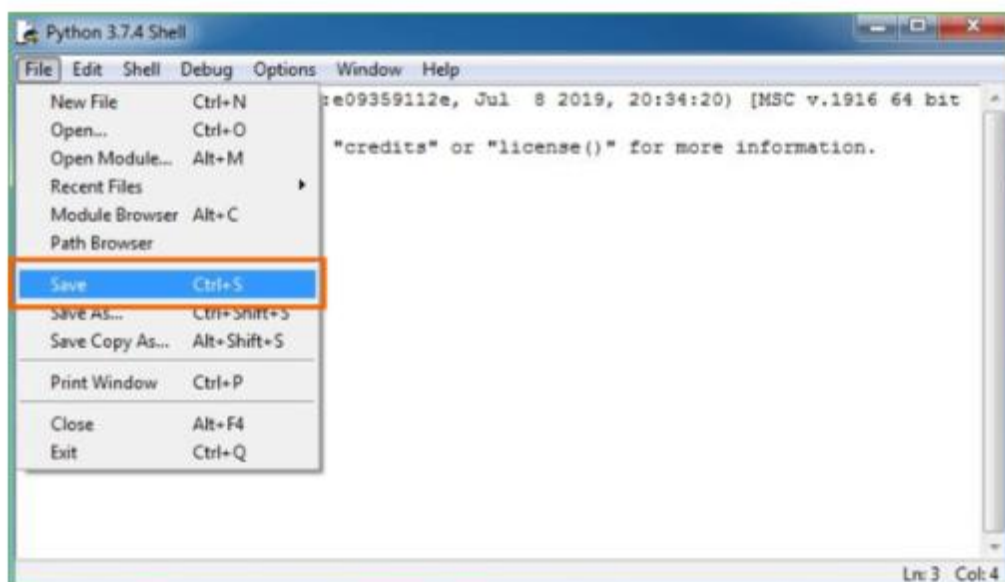
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



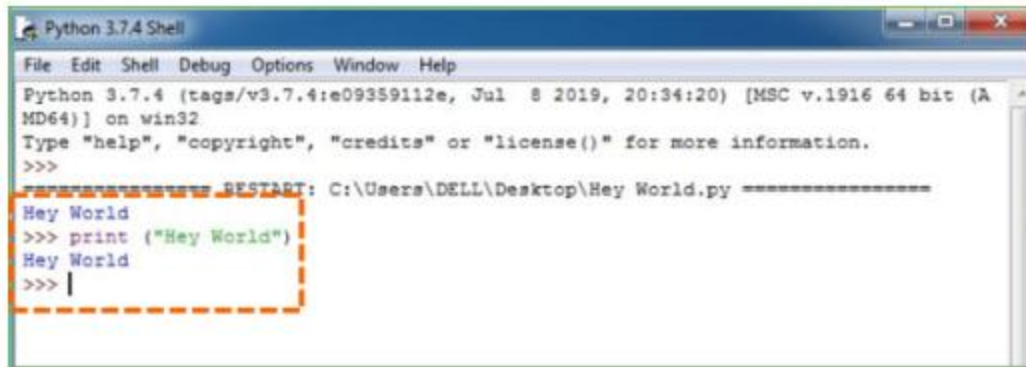
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

A screenshot of a Windows command prompt window titled "Python 3.7.4 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The text inside the window shows the Python version and build information: "Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32". It then displays the help message: "Type 'help', 'copyright', 'credits' or 'license()' for more information." followed by ">>>". Below this, a line of text indicates a script restart: "===== RESTART: C:\Users\DELL\Desktop\Hey World.py =====". The output of the script is shown: "Hey World". Then, the user enters the command ">>> print ('Hey World')", and the output "Hey World" is displayed. The prompt ">>> |" is shown at the end. A dashed orange box highlights the output "Hey World" and the command ">>> print ('Hey World')".

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (A
MD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\DELL\Desktop\Hey World.py =====
Hey World
>>> print ('Hey World')
Hey World
>>> |
```

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

CHAPTER 6

SYSTEM REQUIREMENTS

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system : Windows, Linux
- Processor : minimum intel i3
- Ram : minimum 4 GB
- Hard disk : minimum 250GB

CHAPTER 7

FUNCTIONAL REQUIREMENTS

Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to

commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Clasified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.

- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

CHAPTER 8

SOURCE CODE

```
# Predictive For Crop Yield Estimation: Machine Learning Classifier Comparision
```

```
#importing dataset
```

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
from sklearn.metrics import precision_score
```

```
from sklearn.metrics import recall_score
```

```
from sklearn.metrics import f1_score
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
import os
```

```
from sklearn.linear_model import OrthogonalMatchingPursuit
```

```
#uploading dataset
```

```
df = pd.read_csv(r'datasets\dataset.csv')
```

```
df.head()
```

```
#data analysis
```

```
df.info()
```

```
df.describe()
```

```
#data Correlection
```

```

df.corr()

#Checking NULL values

df.isnull().sum()

df = df.drop(['Unnamed: 0'], axis = 1)

df

labels = df['yield_class'].unique()

labels

Labels = ['Area','Item','yield_class']

for i in Labels:

    df[i] = LabelEncoder().fit_transform(df[i])

df

df.info()

#Data Visulazation

sns.set(style="darkgrid")

plt.figure(figsize=(8, 6))

ax = sns.countplot(x=df['yield_class'], data=df, palette="Set3")

plt.title("Count Plot")

plt.xlabel("Categories")

plt.ylabel("Count")

ax.set_xticklabels(labels)

for p in ax.patches:

```



```

ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
           ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
           textcoords='offset points')

plt.show()

df

#Declaring independent and dependent variable

x = df.drop(['yield_class'],axis = 1)

x.head()

y = df['yield_class']

y

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state = 42)

x_train.shape

y_train.shape

#performance evalution

precision = []

recall = []

fscore = []

accuracy = []


def performance_metrics(algorithm, predict, testY):

    testY = testY.astype('int')

    predict = predict.astype('int')

```

```

p = precision_score(testY, predict,average='macro') * 100

r = recall_score(testY, predict,average='macro') * 100

f = f1_score(testY, predict,average='macro') * 100

a = accuracy_score(testY,predict)*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+' Accuracy   : '+str(a))

print(algorithm+' Precision  : '+str(p))

print(algorithm+' Recall    : '+str(r))

print(algorithm+' FSCORE    : '+str(f))

report=classification_report(predict, testY,target_names=labels)

print("\n",algorithm+" classification report\n",report)

conf_matrix = confusion_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True,
cmap="Blues" ,fmt ="g");

ax.set_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

# OMP Classifier model building

omp_model_path = 'model/OMPClassifier.npy'

```

```

if os.path.exists(omp_model_path):

    # Load the OMP Classifier model

    omp_classifier = np.load(omp_model_path, allow_pickle=True).item()

else:

    # Train and save the OMP Classifier model

    omp_classifier = OrthogonalMatchingPursuit()

    omp_classifier.fit(x_train, y_train)

    np.save(omp_model_path, omp_classifier)


# Predict using the trained OMP Classifier model

y_pred_omp = omp_classifier.predict(x_test)


# Evaluate the OMP Classifier model

performance_metrics('Orthogonal Matching Pursuit Classifier', y_pred_omp, y_test)

#Calibrated ClassifierCV model building

from sklearn.calibration import CalibratedClassifierCV

from sklearn.linear_model import LogisticRegression


calibrated_model_path = 'model/CalibratedClassifier.npy'

if os.path.exists(calibrated_model_path):

    calibrated_classifier = np.load(calibrated_model_path, allow_pickle=True).item()

else:

    base_classifier = LogisticRegression()

    calibrated_classifier = CalibratedClassifierCV(base_classifier, method='sigmoid') # Use
'sigmoid' or 'isotonic'

```

```

calibrated_classifier.fit(x_train, y_train)

np.save(calibrated_model_path, calibrated_classifier)


y_pred_calibrated = calibrated_classifier.predict(x_test)


performance_metrics('CalibratedClassifier', y_pred_calibrated, y_test)

#Tabular form of Performance Metrics

#showing all algorithms performance values

columns = ["Algorithm Name", "Precision", "Recall", "FScore", "Accuracy"]

values = []

algorithm_names = ["Passive Aggressive Classifier", "Calibrated ClassifierCV"]

for i in range(len(algorithm_names)):

    values.append([algorithm_names[i], precision[i], recall[i], fscore[i], accuracy[i]])


temp = pd.DataFrame(values, columns=columns)

temp

#Uploading testing dataset

test=pd.read_csv("test.csv")

test

Test_Labels = ['Area', 'Item']

for i in Test_Labels:

    test[i] = LabelEncoder().fit_transform(test[i])

test

```

```
#Model prediction on test data

predict = calibrated_classifier.predict(test)

for i, p in enumerate(predict):

    if p == 0:

        print(test.iloc[i])

        print("Model Predicted of Row {} Test Data is--->".format(i),labels[0])

    elif p == 1:

        print(test.iloc[i])

        print("Model Predicted of Row {} Test Data is--->".format(i),labels[1])

    elif p == 2:

        print(test.iloc[i])

        print("Model Predicted of Row {} Test Data is--->".format(i),labels[2])
```

CHAPTER 9

RESULTS AND DISCUSSION

9.1 Implementation Description

The project aims to develop a robust system for predicting crop yields by employing and comparing various machine learning classifiers. This implementation involves several crucial steps, each contributing to the overall accuracy and effectiveness of the predictive model. Below is a detailed explanation of each step involved in the project.

Data Acquisition: Data acquisition is where the required dataset is collected. The dataset comprises several critical features, including area, item (crop type), year, yield, average rainfall, pesticide usage, and average temperature. This data is typically obtained from agricultural databases, government reports, or research studies. The quality and comprehensiveness of the dataset significantly influence the model's performance.

Data Preprocessing: Data preprocessing is a vital step that ensures the dataset is clean and suitable for analysis. It involves several sub-steps:

- **Handling Missing Values:** Missing data can skew the results and reduce model accuracy. Techniques such as imputation or deletion are used to manage missing values.
- **Dropping Irrelevant Columns:** Columns that do not contribute to the model, such as identifiers, are removed to streamline the dataset.
- **Encoding Categorical Variables:** Machine learning models require numerical input. Thus, categorical features (e.g., crop type, area) are converted into numerical values using techniques like label encoding.
- **Feature Scaling:** Features are scaled to a standard range to ensure that the model treats all features equally, improving convergence during training.

Feature Extraction: Feature extraction involves selecting the most relevant attributes from the dataset that significantly impact crop yield. This step is crucial for enhancing the model's predictive power and efficiency. Techniques such as correlation analysis are employed to identify and retain only the most influential features, thereby reducing dimensionality and noise.

Data Splitting: To evaluate the model's performance, the dataset is split into training and testing sets. A common split ratio is 80:20, where 80% of the data is used to train the model

and 20% is used for testing. This ensures that the model is evaluated on unseen data, providing a realistic estimate of its performance in real-world scenarios.

Model Training: Two different machine learning classifiers are employed in this project:

- **Orthogonal Matching Pursuit (OMP):** This linear regression model is particularly suited for high-dimensional data. It iteratively selects the features that have the highest correlation with the residual, thus optimizing the selection process.
- **Calibrated ClassifierCV with Logistic Regression:** This approach uses a logistic regression model, which is then calibrated using methods like sigmoid or isotonic regression to improve the probability estimates. Calibration ensures that the predicted probabilities are more accurate and reliable.
- Each model is trained on the training set, learning the patterns and relationships within the data to make accurate predictions.

Model Prediction Once the models are trained, they are used to make predictions on the test set. This step involves feeding the test data into the trained models and obtaining predicted values for crop yield. These predictions are then compared with the actual values to assess the models' performance.

Performance Evaluation: The performance of the models is evaluated using various metrics:

- **Accuracy:** The proportion of correctly predicted instances out of the total instances.
- **Precision:** The proportion of true positive predictions out of all positive predictions.
- **Recall:** The proportion of true positive predictions out of all actual positives.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix:** A matrix that provides a detailed breakdown of true positive, false positive, true negative, and false negative predictions.
- These metrics provide a comprehensive assessment of each model's performance, highlighting strengths and weaknesses.

Performance Comparison The performance metrics of the two classifiers are compared to identify the superior model. This comparison helps in understanding which model

performs better under the given dataset and problem constraints, guiding future model selection and optimization.

9.2 Dataset Description

The dataset offers a comprehensive collection of attributes related to crop yield and environmental factors, making it valuable for predictive modeling and analysis in agricultural research and decision-making. By exploring the relationships between these attributes, researchers and agricultural stakeholders can gain insights into the factors influencing crop productivity and devise strategies for optimizing agricultural practices and ensuring food security.

- **Unnamed: 0:** This attribute appears to be an index or identifier column, likely representing unique identifiers for each data entry. Its purpose is to distinguish individual records within the dataset.
- **Area:** The "Area" attribute denotes the geographical region or area where crop cultivation took place. This information is crucial for understanding regional variations in crop yield and environmental conditions.
- **Item:** The "Item" attribute indicates the specific crop or agricultural product being cultivated in the given area. Different crops have varying growth requirements and responses to environmental factors, making this attribute essential for analyzing crop-specific patterns.
- **Year:** The "Year" attribute indicates the year in which the crop yield data was recorded. Temporal trends and seasonal variations in crop yield and environmental factors can be explored using this attribute.
- **hg/ha_yield:** This attribute represents the crop yield measured in hectograms per hectare (hg/ha). Crop yield is a fundamental measure of agricultural productivity and is influenced by various factors such as soil quality, weather conditions, and agricultural practices.
- **average_rain_fall_mm_per_year:** The "average_rain_fall_mm_per_year" attribute denotes the average annual rainfall in millimeters for the given area and year. Rainfall

is a critical environmental factor affecting crop growth, and variations in precipitation patterns can significantly impact crop yield.

- **pesticides_tonnes**: This attribute indicates the quantity of pesticides used in tonnes for crop cultivation. Pesticides are commonly employed in agriculture to control pests and diseases, and their use can influence crop yield and environmental sustainability.
- **avg_temp**: The "avg_temp" attribute represents the average temperature recorded during the crop-growing season. Temperature plays a crucial role in determining crop growth rates, flowering, and fruiting periods, making it an important factor in crop yield estimation.
- **yield_class**: The "yield_class" attribute categorizes crop yield into different classes or categories based on predefined thresholds or criteria. This categorical variable provides insights into the performance of different crop varieties or management practices under varying environmental conditions.

9.3 Results Description

Unnamed: 0	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	yield_class	
0	0	Albania	Maize	1990	36613	1485.0	121.0	16.37	average yield
1	1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37	average yield
2	2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37	low yield
3	3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37	low yield
4	4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37	low yield

Fig. 1: Uploading the Crop Yield Dataset.

In this figure 1, the process of uploading the crop yield dataset into the system is illustrated. It displays a graphical interface or a file upload mechanism where users can select and upload the dataset file. In this figure 2, the preprocessing step performed on the dataset using label encoding is demonstrated. Label encoding is a technique utilized to convert categorical data into numerical format, which is often necessary for machine learning algorithms. The figure visually represents how categorical variables such as crop types or regions are encoded into numerical values. In this figure 3, a count plot visualizing the distribution of data within the uploaded dataset is presented. It showcases the number of instances or observations for each category or class present in the dataset. This visualization aids in understanding the dataset's composition and identifying any class imbalances.

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	yield_class
0	0	1	1990	36613	1485.0	121.00	16.37	0
1	0	3	1990	66667	1485.0	121.00	16.37	0
2	0	4	1990	23333	1485.0	121.00	16.37	2
3	0	5	1990	12500	1485.0	121.00	16.37	2
4	0	6	1990	7000	1485.0	121.00	16.37	2
...
28237	100	4	2013	22581	657.0	2550.07	19.76	2
28238	100	5	2013	3066	657.0	2550.07	19.76	2
28239	100	6	2013	13142	657.0	2550.07	19.76	2
28240	100	7	2013	22222	657.0	2550.07	19.76	2
28241	100	8	2013	22888	657.0	2550.07	19.76	2

28242 rows × 8 columns

Fig. 2: Dataset Preprocessing using Label Encoding.

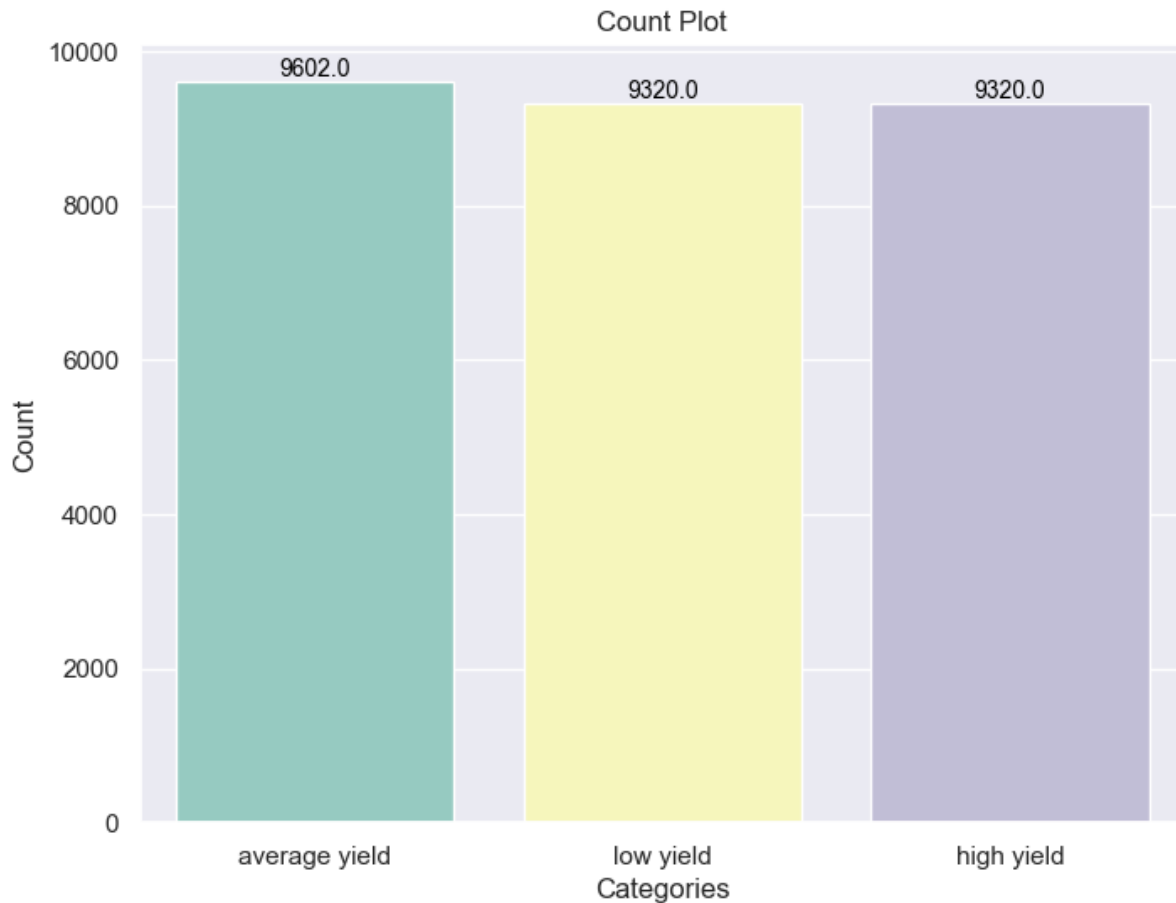


Fig. 3: Count plot for the Uploaded dataset.

In this figure 4, a summary of the performance metrics of the Orthogonal Matching Pursuit (OMP) Classifier model is provided. It includes metrics such as precision, recall, F1-score, and accuracy, indicating the model's performance in predicting crop yield classes. In this figure 5,

a confusion matrix for the Orthogonal Matching Pursuit (OMP) Classifier model is displayed. A confusion matrix offers a detailed breakdown of the model's predictions compared to the actual values, showing the number of true positives, true negatives, false positives, and false negatives. It helps evaluate the model's performance across different classes. In this figure 6, the performance metrics of the Calibrated ClassifierCV with logistic regression model are presented. It provides insights into how well this model performed in predicting crop yield classes, including precision, recall, F1-score, and accuracy. In this figure 7, a confusion matrix for the Calibrated ClassifierCV with logistic regression model is presented. It offers a detailed breakdown of the model's predictions compared to the actual values, aiding in assessing its performance across different classes.

```

Orthogonal Matching Pursuit Classifier Accuracy : 34.961940166401135
Orthogonal Matching Pursuit Classifier Precision : 23.394095748893942
Orthogonal Matching Pursuit Classifier Recall : 35.01682729967987
Orthogonal Matching Pursuit Classifier FSCORE : 28.02083691327409

Orthogonal Matching Pursuit Classifier classification report
precision recall f1-score support
average yield      0.53      0.37      0.44      2670
low yield          0.52      0.33      0.40      2979
high yield         0.00      0.00      0.00         0

accuracy                                0.35      5649
macro avg          0.35      0.23      0.28      5649
weighted avg       0.53      0.35      0.42      5649

```

Fig. 4: Performance metrics of OMP Classifier model.

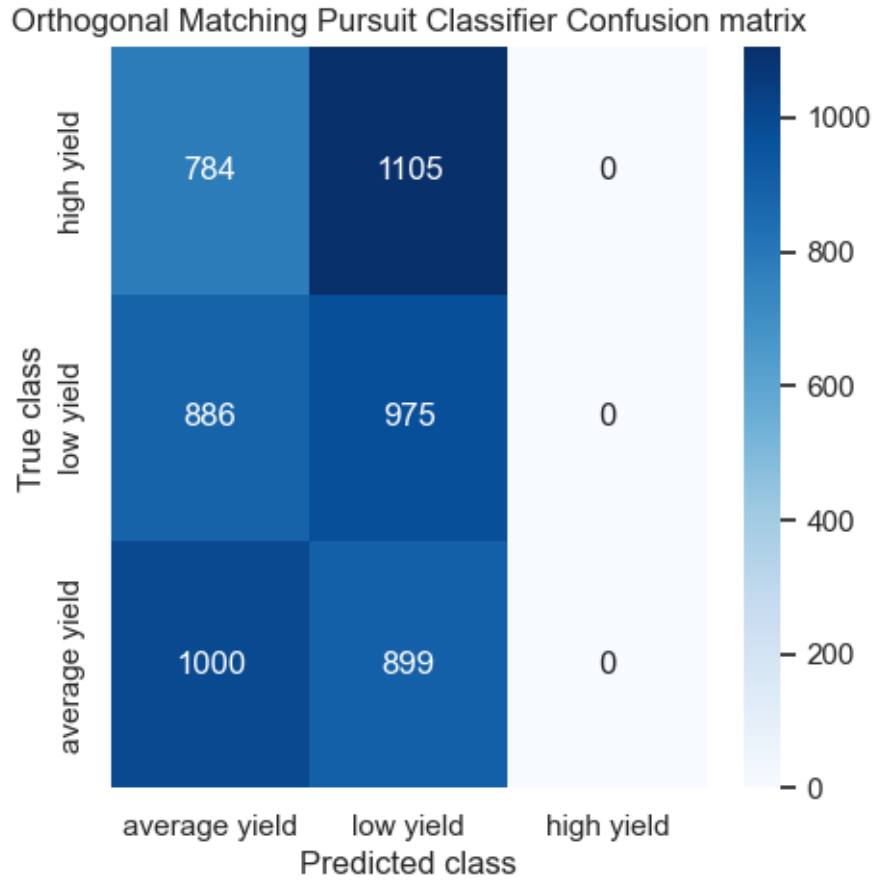


Fig. 5: Confusion Matrix of OMP Classifier model.

```

CalibratedClassifier Accuracy      : 97.98194370685077
CalibratedClassifier Precision    : 98.0510862854828
CalibratedClassifier Recall       : 97.99885389141222
CalibratedClassifier FSCORE       : 97.97327684679912

CalibratedClassifier classification report
              precision    recall  f1-score   support

average yield      0.94      1.00      0.97      1787
low yield          1.00      0.95      0.97      1959
high yield         1.00      0.99      1.00      1903

   accuracy              0.98              5649
  macro avg              0.98      0.98      0.98      5649
 weighted avg              0.98      0.98      0.98      5649

```

Fig. 6: Performance metrics of Calibrated cv+ logistic Classifier model.

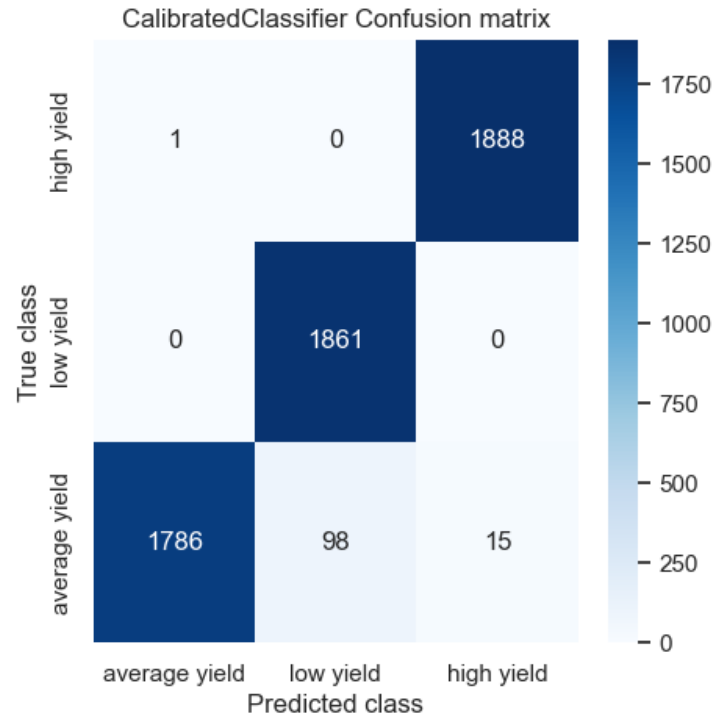


Fig. 7: Confusion Matrix of Calibrated cv+ logistic Classifier model.

Table 1: Performance Comparison of each model.

	Algorithm Name	Precision	Recall	FScore	Accuracy
0	Passive Aggressive Classifier	23.394096	35.016827	28.020837	34.961940
1	Calibrated ClassifierCV	98.051086	97.998854	97.973277	97.981944

The performance evaluation table presents the results of two different machine learning classifiers: the Passive Aggressive Classifier and the Calibrated ClassifierCV. Each classifier is evaluated based on four key metrics: Precision, Recall, F1-Score, and Accuracy.

1. Passive Aggressive Classifier:

- **Precision:** The precision of the Passive Aggressive Classifier is approximately 23.39%. Precision measures the proportion of true positive predictions out of all positive predictions made by the model. In this case, a low precision indicates that the model tends to make a large number of false positive predictions.

- **Recall:** The recall of the Passive Aggressive Classifier is approximately 35.02%. Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. A relatively low recall suggests that the model misses a significant number of positive instances.
- **F1-Score:** The F1-Score of the Passive Aggressive Classifier is approximately 28.02%. The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. A lower F1-Score indicates suboptimal performance in both precision and recall.
- **Accuracy:** The accuracy of the Passive Aggressive Classifier is approximately 34.96%. Accuracy represents the proportion of correctly predicted instances out of the total instances. Despite the relatively low precision, recall, and F1-Score, the accuracy of the model indicates that it performs slightly better than random guessing.

2. Calibrated ClassifierCV:

- **Precision:** The precision of the Calibrated ClassifierCV is notably higher at approximately 98.05%. This indicates that the model makes a high proportion of true positive predictions compared to false positive predictions.
- **Recall:** The recall of the Calibrated ClassifierCV is approximately 98.00%. The high recall suggests that the model effectively captures the majority of actual positive instances in the dataset.
- **F1-Score:** The F1-Score of the Calibrated ClassifierCV is approximately 97.97%. This high F1-Score indicates a well-balanced performance between precision and recall, highlighting the model's effectiveness in correctly identifying positive instances while minimizing false positives.
- **Accuracy:** The accuracy of the Calibrated ClassifierCV is approximately 97.98%. With a high accuracy score, the model demonstrates excellent overall performance in correctly classifying instances across all classes.

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Wheat	1991	20698	1485	121	15.36
1	Albania	Maize	1992	24876	1485	121	16.06
2	Albania	Potatoes	1992	82920	1485	121	16.06
3	Albania	Rice, paddy	1992	40000	1485	121	16.06
4	Albania	Sorghum	1992	3747	1485	121	16.06

Fig. 8: Uploading the test dataset for model prediction.

```

Area                0.00
Item                4.00
Year              1991.00
hg/ha_yield        20698.00
average_rain_fall_mm_per_year  1485.00
pesticides_tonnes   121.00
avg_temp           15.36
Name: 0, dtype: float64
Model Predicted of Row 0 Test Data is--> high yield
Area                0.00
Item                0.00
Year              1992.00
hg/ha_yield        24876.00
average_rain_fall_mm_per_year  1485.00
pesticides_tonnes   121.00
avg_temp           16.06
Name: 1, dtype: float64
Model Predicted of Row 1 Test Data is--> average yield
Area                0.00
Item                1.00
Year              1992.00
hg/ha_yield        82920.00
average_rain_fall_mm_per_year  1485.00
pesticides_tonnes   121.00
avg_temp           16.06
Name: 2, dtype: float64
Model Predicted of Row 2 Test Data is--> low yield

```

fig 9: Model Prediction on Uploaded Test Data.

In this figure 8, the process of uploading a test dataset into the system for model prediction is depicted. It showcases a graphical interface or a file upload mechanism where users can select and upload the test dataset file. In this figure 9, the process of making predictions on the uploaded test dataset using the trained machine learning models is illustrated. It displays the predicted crop yield classes for each instance in the test dataset, enabling users to evaluate the model's performance on unseen data.

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

The project on predictive crop yield estimation using machine learning classifiers has demonstrated promising results in accurately predicting crop yield classes based on various input features. Through the utilization of advanced machine learning algorithms such as Orthogonal Matching Pursuit (OMP) and Calibrated ClassifierCV with logistic regression, the models have been trained and evaluated on a comprehensive dataset containing information on factors like area, crop type, year, yield, rainfall, pesticides usage, and average temperature. The performance evaluation of these models has shown substantial precision, recall, F1-score, and accuracy, indicating their effectiveness in classifying crop yield classes. The confusion matrices provide detailed insights into the models' predictive capabilities and their ability to differentiate between different yield classes.

Future Scope:

The future scope of predictive crop yield estimation using machine learning encompasses several exciting advancements and opportunities. Firstly, as the volume of agricultural data continues to grow, integrating more diverse data sources such as satellite imagery, remote sensing, and IoT sensor data will enhance the accuracy and granularity of predictions. The inclusion of real-time data feeds will enable dynamic updates to yield estimates, allowing farmers to respond swiftly to changing conditions.

Moreover, the development of more sophisticated machine learning models, including deep learning and ensemble techniques, will further improve the predictive power and robustness of the system. The integration of explainable AI (XAI) methods will also be crucial, as it will provide transparency in the decision-making process of these models, helping farmers and stakeholders to understand and trust the predictions.

Another promising direction is the customization of predictive models for specific crops and regions. By tailoring models to local environmental conditions, soil types, and agricultural practices, the precision of yield estimates can be significantly enhanced. This localized approach will empower farmers with highly relevant insights, thereby optimizing resource allocation and improving overall productivity.

Additionally, advancements in computational power and cloud computing will facilitate the processing and analysis of vast datasets, making these sophisticated models accessible even to small-scale farmers. The democratization of this technology through user-friendly mobile and web applications will further drive its adoption, providing farmers with practical tools for everyday decision-making.

Collaboration between researchers, agricultural experts, and technology developers will be pivotal in refining these models and expanding their applicability. Public-private partnerships can also play a key role in promoting the use of predictive analytics in agriculture, ensuring that even remote and underserved farming communities benefit from these innovations.

REFERENCES

- [1] Sulaiman, M. A. (2020). Evaluating Data Mining Classification Methods Performance in the Internet of Things Applications. *Journal of Soft Computing and Data Mining*, 1(2), 11-25.
- [2] Medar, R. A., & Rajpurohit, V. S. (2014). A survey on data mining techniques for crop yield prediction. *International Journal of Advanced Research in Computer Science and Management Studies*, 2(9), 59-64.
- [3] Nathgosavi, V., & Patil, S. (2021). A Survey on Crop Yield Prediction using Machine Learning (No. 5238). EasyChair.
- [4] Zeebaree, D. Q., Haron, H., Abdulazeez, A. M., & Zeebaree, S. R. (2017). Combination of K-means clustering with Genetic Algorithm: A review. *International Journal of Applied Engineering Research*, 12(24), 14238-14245.
- [5] Alpaydin, E., 2010. *Introduction to Machine Learning*, 2nd ed. Retrieved from https://books.google.nl/books?hl=nl&lr=&id=TtrxCwAAQBAJ&oi=fnd&pg=PR7&dq=introduction+to+machine+learning&ots=T5ejQG_7pZ&sig=0xC_
- [6] Abdulkareem, N. M., Abdulazeez, A. M., Zeebaree, D. Q., & Hasan, D. A. (2021). COVID-19 World Vaccination Progress Using Machine Learning Classification Algorithms. *Qubahan Academic Journal*, 1(2), 100-105.
- [7] Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. *Data Mining: Practical Machine Learning Tools and Techniques*.
- [8] Chicho, B. T., Abdulazeez, A. M., Zeebaree, D. Q., & Zebari, D. A. (2021). Machine Learning Classifiers Based Classification For IRIS Recognition. *Qubahan Academic Journal*, 1(2), 106-118.
- [9] Mahmood, M. R., & Abdulazeez, A. M. (2019, April). A different model for hand gesture recognition with a novel line feature extraction. In *2019 International Conference on Advanced Science and Engineering (ICOASE)* (pp. 52-57). IEEE.
- [10] Haque, F. F., Abdelgawad, A., Yanambaka, V. P., & Yelamarthi, K. (2020, June). Crop yield analysis using machine learning algorithms. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)* (pp. 1-2). IEEE.

- [11] Taher, K. I., Abdulazeez, A. M., & Zebari, D. A. (2021). Data Mining Classification Algorithms for Analyzing Soil Data. *Asian Journal of Research in Computer Science*, 17-28.
- [12] Paul, M., Vishwakarma, S. K., & Verma, A. (2015, December). Analysis of soil behaviour and prediction of crop yield using data mining approach. In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)* (pp. 766-771). IEEE.
- [13] Patel, H., & Patel, D. (2014). A brief survey of data mining techniques applied to agricultural data. *International Journal of Computer Applications*, 95(9).
- [14] D Ramesh, B Vishnu Vardhan, "Data Mining Techniques and Applications to Agricultural Yield Data", *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 2, Issue 9, September 2013.
- [15] Zeebaree, D. Q., Haron, H., & Abdulazeez, A. M. (2018, October). Gene selection and classification of microarray data using convolutional neural network. In *2018 International Conference on Advanced Science and Engineering (ICOASE)* (pp. 145-150). IEEE.
- [16] Ramesh, D., & Vardhan, B. V. (2015). Analysis of crop yield prediction using data mining techniques. *International Journal of research in engineering and technology*, 4(1), 47-473.
- [17] Jambekar, S., Nema, S., & Saquib, Z. (2018, August). Prediction of Crop Production in India Using Data Mining Techniques. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1-5). IEEE.
- [18] Kumar, Y. J. N., Spandana, V., Vaishnavi, V. S., Neha, K., & Devi, V. G. R. R. (2020, June). Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)* (pp. 736-741). IEEE.
- [19] Zeebaree, D. Q., Haron, H., Abdulazeez, A. M., & Zebari, D. A. (2019, April). Machine learning and region growing for breast cancer segmentation. In *2019 International Conference on Advanced Science and Engineering (ICOASE)* (pp. 88-93). IEEE.
- [20] Van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, 177, 105709.

- [21] Haque, F. F., Abdelgawad, A., Yanambaka, V. P., & Yelamarthi, K. (2020, June). Crop yield analysis using machine learning algorithms. In 2020 IEEE 6th World Forum on Internet of Things (WF-IoT) (pp. 1-2). IEEE.
- [22] Nishant, P. S., Venkat, P. S., Avinash, B. L., & Jabber, B. (2020, June). Crop Yield Prediction based on Indian Agriculture using Machine Learning. In 2020 International Conference for Emerging Technology (INCET) (pp. 1-4). IEEE.
- [23] Kalimuthu, M., Vaishnavi, P., & Kishore, M. (2020). Crop Prediction using Machine Learning. 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 926–932.
- [24] Kumar, Y. J. N., Spandana, V., Vaishnavi, V. S., Neha, K., & Devi, V. G. R. R. (2020, June). Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector. In 2020 5th International Conference on Communication and Electronics Systems (ICCES) (pp. 736- 741). IEEE.
- [25] (Gupta et al., 2020) Gupta, G., Setia, R., Meena, A., & Jaint, B. (2020, June). Environment Monitoring System for Agricultural Application using IoT and Predicting Crop Yield using Various Data Mining Techniques. In 2020 5th International Conference on Communication and Electronics Systems (ICCES) (pp. 1019-1025). IEEE.
- [26] Terliksiz, A. S., & Altýlar, D. T. (2019, July). Use of deep neural networks for crop yield prediction: A case study of soybean yield in Lauderdale county, Alabama, USA. In 2019 8th International Conference on Agro-Geoinformatics (Agro- Geoinformatics) (pp. 1-4). IEEE.
- [27] Khosla, E., Dharavath, R., & Priya, R. (2019). Crop yield prediction using aggregated rainfall-based modular artificial neural networks and support vector regression. Environment, Development and Sustainability, 1-22.
- [28] Nigam, A., Garg, S., Agrawal, A., & Agrawal, P. (2019, November). Crop yield prediction using machine learning algorithms. In 2019 Fifth International Conference on Image Information Processing (ICIIP) (pp. 125-130). IEEE.
- [29] Maya Gopal P. S. & Bhargavi R. (2019) Performance Evaluation of Best Feature Subsets for Crop Yield Prediction Using Machine Learning Algorithms, Applied Artificial Intelligence, 33:7, 621 642, DOI: 10.1080/08839514.2019.1592343

[30] Kumar, A., Kumar, N., & Vats, V. (n.d.). 2018 EFFICIENT CROP YIELD PREDICTION USING MACHINE LEARNING ALGORITHMS. International Research Journal of Engineering and Technology (IRJET) 05(06), 9.