BACHELOR OF SCIENCE IN ELECTRONIC AND TELECOMMUNICATION ENGINEERING

Machine Learning-based Crop Yield Forecasting in Bangladesh: An Early Prediction Approach.

Submitted by

Md Forhad

T191047

Supervised by

Ahmad

Lecturer

Department of ETE

International Islamic University Chittagong

Department of Electronic and Telecommunication Engineering(ETE)

International Islamic University Chittagong

Kumira, Sitakunda, Chattogram-4318 , Bangladesh.

February, 2024

CERTIFICATION OF APPROVAL

The undergraduate project titled **"Early Prediction of Crop Yield Prediction in Bangladesh Using Machine Learning"** was submitted by **MD FORHAD (T191047)** and have been accepted by **International Islamic University Chittagong** as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science (B.Sc.) in **Electronic and Telecommunication Engineering (ETE).**

Ahmad

Lecturer

Department of Electronic & Telecommunication Engineering

International Islamic University Chittagong

DECLARATION

"We declare that this thesis book is the result of our possess unique inquire about which all sources utilized were appropriately acknowledged and cited within the reference segment. This work was never been submitted some time recently for any other scholastic certification or title. Any help gotten all through the investigate and planning of this proposal has been acknowledged, in spite of the fact that it is as it were counseling in nature."

MD FORHAD

T191047

ACKNOWLEDGMENT

In the name of Allah (SWT), the most forgiving and pardoning of all creatures. We thank Allah foremost for giving us the quality and knowledge all through our lives. We appreciate our family's fondness as well as their spiritual and financial back. We would like to thank Ahmed, Lecturer, Department of ETE, IIUC, for giving us the conceivable supportive proposals, recommendations and assessments all through this project. His exceptional help and valuable perceptions and concepts driven to the project's victory throughout the explore and extend work. At long last however noticeably, we would need to specific our significant appreciation to our teachers, friends, and family.

ABSTRACT

Farmers in Bangladesh are facing substantial financial losses due to challenges in selecting the optimal crops for production, influenced by factors such as pesticides, fertilizer, and annual rainfall. The agriculture supply chain can benefit from yield prediction to make crucial decisions that mitigate risks associated with changes in crop growth. This research focuses on the critical task of early crop yield prediction in Bangladesh, specifically targeting rice, sugarcane, potato, and maize. A diverse set of machine learning algorithms, including K-Nearest Neighbors, Random Forest, Gradient Boosting, AdaBoost, CatBoost, Decision Tree, and XGBoost, are employed in the initial phase to assess their individual predictive performances. After a comprehensive analysis, K-Nearest Neighbors, Random Forest, and Gradient Boosting emerge as the three most effective models. The study progresses to explore ensemble learning techniques, utilizing a Voting Regression in these top three models to combine their predictions. Furthermore, advanced ensemble methodologies, specifically stacking with varying both final and base estimators using the three best models, are implemented to harness the collective intelligence of diverse models and enhance overall predictive accuracy. The methodology includes a rigorous evaluation process, considering various metrics to assess the effectiveness of the models and ensemble techniques. The findings provide valuable insights into how different algorithms and ensemble methodologies can collaborate to optimize early crop yield prediction for specific crops in Bangladesh. Among all combinations, for maize crop yield prediction achieved R^2 0.99 and MAE 0.04, rice crop yield prediction achieved R^2 0.99 and MAE 0.06, potato crop yield prediction achieved R^2 0.99 and MAE 0.04, sugarcane crop yield prediction achieved R^2 0.99 and MAE 0.36. This research contributes not only to the field of agricultural prediction but also to the broader application of ensemble learning in optimizing machine learning models for complex tasks. The findings are significant for stakeholders in agriculture, providing them with reliable tools for early crop yield estimation, crucial for effective resource management and decision-making.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Overview

The issue of agriculture sustainability is a challenge that agricultural scientists around the globe must contend with due to a number of threats, including rising food and energy prices, shifting climates, ongoing resource depletion, scary declines in water availability, and the anticipated rise in global population in the next several centuries. Since crop yield has a significant impact on both the domestic and global economies and can help address the issue of food scarcity, it is one of the key areas of agriculture that has drawn the interest of scientists [1]. The need to boost agricultural output has grown more urgent due to the growing global population and the growing challenges caused by climate change. Accurately predicting agricultural yields well in advance is essential to this endeavor because it helps farmers, policymakers, and other stakeholders make well-informed decisions and reduce risks. Predicting crop yields is essential to enhancing food security and guaranteeing sufficient food supply [2].

The challenge of ensuring sustainability in agriculture has become increasingly complex, as agricultural scientists worldwide grapple with a myriad of threats. These challenges include the escalating prices of both food and energy, the dynamic shifts in climates, the ongoing depletion of essential resources, alarming declines in water availability, and the foreseen increase in the global population over the coming centuries. Among the numerous facets of agriculture, crop yield stands out as a focal point for scientific inquiry, given its substantial impact on both domestic and global economies, and its potential to address the pressing issue of food scarcity.

As the world faces the twin challenges of a burgeoning population and the complex repercussions of climate change, the imperative to enhance agricultural output has become more pronounced [3]. The urgency stems from the necessity to meet the escalating demands of a growing global populace while simultaneously grappling with the unpredictable and adverse effects of a changing climate. In this context, accurately predicting agricultural yields assumes paramount importance. The ability to forecast crop yields well in advance is crucial for farmers, policymakers, and other stakeholders, as it empowers them to make well-informed decisions and mitigate risks associated with agricultural production. This predictive capability is not merely a theoretical

exercise; rather, it is a linchpin in the practical efforts to bolster food security and ensure a consistent and ample food supply for the ever-expanding global population. In essence, the accurate anticipation of crop yields emerges as a linchpin in the overarching goal of creating a sustainable and resilient agricultural system that can navigate the complexities of the contemporary world [4]. At both regional and national levels, the accurate completion of the duty of crop yield prediction is of paramount importance, serving as a linchpin for facilitating swift decision-making processes. The implications of precise predictions resonate across various sectors, enabling officials to make informed choices regarding import and export strategies based on the anticipated agricultural output. This precision is equally crucial for farmers who rely on accurate yield estimates to make sound financial decisions, while seed producers benefit from insights into how new seeds perform under diverse conditions.

In recent years, the convergence of machine learning and agriculture has emerged as a revolutionary force, endowing researchers with unprecedented capabilities to decipher large datasets and extract insightful conclusions. This transformative potential has sparked a concerted effort to usher in a new era of precision agriculture by harnessing the power of machine learning algorithms for early crop yield prediction [5]. Unlike conventional techniques that often rely on historical records, climate trends, and empirical models, which may lack precision and struggle to capture the dynamic interactions among various factors influencing crop growth, machine learning presents an opportunity for a paradigm shift in crop yield forecasting.

The distinctive strength of machine learning lies in its ability to identify intricate patterns within massive datasets, providing a nuanced understanding of the complex interplay between diverse factors affecting crop development.

By leveraging machine learning algorithms, researchers aspire to transcend the limitations of traditional methods, offering a more accurate and sophisticated approach to predicting crop yields. This not only enhances the reliability of decision-making processes for agricultural stakeholders but also opens up new frontiers in understanding the intricate dynamics of crop growth in response to an ever-changing set of variables [6]. As the synergy between machine learning and agriculture continues to evolve, there is a growing optimism that these advancements will not only bolster the resilience of agricultural systems but also contribute significantly to global efforts aimed at ensuring

food security in the face of a rapidly changing world. The ambitious goal of our research endeavors is to forge models that seamlessly integrate data from an array of sources, including satellite imagery, weather patterns, soil conditions, and historical agricultural practices. By harnessing cutting-edge algorithms and leveraging robust computing capacities, our aim is to transcend the limitations of traditional forecasting methods. The objective is to produce more precise and timely predictions that can play a pivotal role in advancing the frontiers of agricultural science [7].

Recognizing the pivotal role of agriculture in the overall development of any nation, our efforts align with the imperative to foster sustainable practices within this crucial field. The ever-expanding global population and the unforeseen shifts in soil and climate conditions compel experts worldwide to seek measures that go beyond mere productivity and delve into sustainable agricultural practices. The crux of this challenge lies in enhancing crop output without causing adverse effects on our natural resources. In this dynamic context, anticipating crop yields for the current growing season emerges as an essential first step.

The complexity of modern agricultural systems demands a holistic approach that considers a multitude of variables. Our research aspires to meet this demand by amalgamating data from diverse sources, thus providing a comprehensive understanding of the factors influencing crop growth. Satellite images offer a bird's-eye view of vast agricultural landscapes, weather patterns contribute insights into atmospheric conditions, soil data delves into the vital substrate for plant growth, and historical agricultural practices provide crucial context for informed predictions [8].

The integration of these diverse datasets with advanced algorithms and computing prowess is not merely a technical endeavor; it is a strategic move towards sustainable agriculture. The ability to forecast crop yields with accuracy and timeliness is instrumental in empowering farmers, policymakers, and other stakeholders to make informed decisions. It not only aids in optimizing resource allocation but also contributes to the larger goal of ensuring food security in the face of global challenges.

In essence, our pursuit aims to be at the forefront of the transformative intersection between technology and agriculture, ushering in a new era where data-driven insights play a central role in shaping the future of sustainable and resilient food production. Through the integration of cutting-edge technologies, our vision is to contribute

significantly to the broader discourse on global food security and sustainable development [1].

In poorer nations, sampling surveys are the primary means of tracking crop productivity. Early crop output predictions are based on subjective surveys, such as field officers' visual assessments and growers' judgments. Using objective surveys, such as a whole plot harvest or crop cut measurement from sample fields, the later crop yield estimation is carried out [2]. The current methodology for collecting crop data, which primarily involves sampling easily accessible fields, poses inherent limitations. This selective approach results in data sets that may not accurately represent the overall status of seasonal croplands. The choice of sampling sites within each region follows a systematic random sampling scheme, introducing an element of randomness into the data collection process. While the labor-intensive crop monitoring system serves as a valuable tool for holistic agricultural management, its utility for estimating crop yields is constrained by the labor-intensive nature of the process and the substantial uncertainty associated with the collected data.

A significant drawback of this sampling approach is that national crop production forecasts are often finalized months after the harvest. This delayed timeline poses challenges for making timely import and export decisions critical for ensuring both food security and economic growth. The lag in obtaining accurate and up-to-date information hampers the ability of policymakers, farmers, and other stakeholders to make informed decisions, contributing to uncertainties in the agricultural market [10].

In the broader context of global food security, there have been commendable efforts over the past few decades to address the challenges posed by a rapidly expanding world population and persistent hunger. While agricultural yields have seen a substantial increase over the last fifty years, a staggering 800 million people still lack access to sufficient food. Recognizing the urgency of the situation, the United Nations' 2030 Agenda for Sustainable Development places a high priority on initiatives aimed at reducing hunger and enhancing food security. The limitations in the current crop monitoring and forecasting systems underscore the need for innovation and technological interventions to address the challenges of modern agriculture. The development of more advanced and efficient methods, such as those integrating machine learning, satellite technology, and comprehensive data sets, holds promise for

revolutionizing crop yield predictions [12]. These advancements not only align with the goals of sustainable development but also offer practical solutions to pressing global issues, contributing to the overarching aim of ensuring food security for all.

As we navigate the complexities of agricultural practices and global food production, the integration of innovative technologies becomes increasingly crucial in achieving the objectives outlined in .In the intricate landscape of agriculture, accurately estimating the potential yield of crops is a significant milestone, particularly for those involved in the production and trading stages. Providing farmers with reliable forecasts for their production plays a pivotal role in enabling effective budgetary planning and resource management. The advent of machine learning has ushered in a transformative era in the realm of early crop production prediction, fundamentally altering the dynamics of agricultural practices.

Machine learning, with its ability to meticulously analyze vast and diverse datasets, stands out as a revolutionary force in agriculture [13]. The integration of historical yield records, satellite imaging, soil characteristics, and weather patterns empowers machine learning algorithms to discern intricate patterns and interactions that might elude conventional analytical approaches. This amalgamation of diverse data sources enables the algorithms to generate remarkably accurate predictions regarding crop production.

The significance of early crop production predictions becomes apparent in the tangible benefits it offers to farmers. Armed with timely and precise information, farmers can make proactive decisions that have far-reaching implications for their agricultural operations. Optimizing resource allocation becomes more achievable as farmers gain insights into the expected yield of their crops. Additionally, the early foresight provided by machine learning algorithms allows farmers to implement targeted interventions against potential threats, ranging from pest infestations to adverse weather conditions [14]. Perhaps equally important is the positive impact on overall crop management methods. The integration of machine learning into the decision-making processes of farmers facilitates a paradigm shift towards more informed and strategic agricultural practices. By leveraging the power of data-driven insights, farmers can fine-tune their approaches, adopting more efficient and sustainable methods that align with the goal of maximizing crop yield while minimizing environmental impact. In essence, the marriage of machine learning and agriculture not only enhances the precision of early

crop production predictions but also empowers farmers with the tools and knowledge necessary to navigate the complexities of modern agriculture.

As technology continues to advance, the synergy between machine learning and agriculture holds immense potential for fostering a more resilient, sustainable, and productive agricultural sector [15]. Through the proactive utilization of these innovations, the agricultural community stands poised to meet the challenges of feeding a growing global population while ensuring the responsible stewardship of natural resources. Processing shifted from massive centralized computers to supercomputers and the cloud throughout the most defining period. Through experience and data consumption, the algorithms used to decipher computers are progressively enhanced. It is known as machine learning (ML) and is seen as a component of artificial intelligence. Machine learning (ML) is a method of data analysis that automates the creation of intelligent models. The machine learning system has the flexibility to create pronouncements with minimal human intervention by drawing on prior knowledge. Specifically, supervised learning, unsupervised learning, and semi-supervised learning are the three kinds of machine learning algorithms. Clustering is the process of identifying comparable groups of data within a dataset [16]. Moreover, machine learning contributes to sustainable agriculture by reducing the risks brought on by unfavorable weather, illnesses, and pests.

With the help of machine learning insights, precision agriculture reduces waste and its negative effects on the environment by using water, fertilizer, and pesticides sparingly. Because of this, resilient and sustainable agriculture methods are encouraged by machine learning algorithms' ability to adapt to changing environmental conditions and their data-driven precision, which is vital in tackling global food security concerns.

Crop yield prediction has seen success with the application of machine learning models, such as decision trees, stepwise multiple linear regression, random forests, neural networks, convolutional neural networks, recurrent neural networks, weighted histogram regression, interaction-based models, and association rule mining. Due to a dearth of publicly accessible genetic data at the state or national level, the majority of these research were solely reliant on management and environmental factors. From a plant breeding standpoint, some research investigated the association between genotype and grain yield from regional yield trials; nevertheless, it would be difficult to scale up

to statewide or nationwide projections. A lot of machine learning algorithms can handle big datasets and produce predictions with a respectable degree of accuracy [17].

Estimating crop production has long been a complex challenge, shaped by a multitude of intricate elements. A wide array of factors, including terrain, soil quality, pest infestations, genotype, water quality and accessibility, climate, and harvest scheduling, among others, collectively influence crop output [18]. The methods and procedures employed for predicting crop yield are inherently nonlinear and time-specific. The complexity is further heightened by the interconnectedness of various parameters, influenced by non-arbitrary and external circumstances.

Traditionally, farmers relied on personal experiences and reliable historical data to forecast crop yields, making crucial production decisions based on these predictions. However, recent advancements, notably in the realms of machine learning and crop model simulation, have showcased the potential for more accurate yield estimations. Machine learning algorithms, in particular, have emerged as powerful tools in this domain, surpassing classical statistical approaches in various studies [19].

In the field of artificial intelligence, specifically within machine learning, computers can be trained without explicit programming. This unique characteristic enables computers to develop a remarkable capacity for prediction, overcoming the challenges posed by both linear and non-linear agricultural structures. The learning methodology of machine learning agricultural systems involves training the model using specific data to perform a particular task. Successful completion of the training phase allows the model to be tested with new data, applying the learned patterns and relationships to make predictions. The potential of machine learning algorithms in agriculture lies in their ability to discern complex patterns and relationships within diverse datasets. Unlike traditional methods that may struggle with the intricacies of nonlinear systems and interconnected variables, machine learning thrives on complexity [20]. By leveraging large and diverse datasets encompassing a broad spectrum of influencing factors, these algorithms can provide more nuanced and accurate predictions for crop yield.

The shift towards machine learning in agriculture signifies a departure from reliance solely on historical and experiential data. Instead, it embraces a data-driven and dynamic approach, allowing for real-time adaptations to changing agricultural

landscapes. As technology continues to evolve, the integration of machine learning into agricultural practices holds the promise of not only enhancing prediction accuracy but also revolutionizing the way farmers approach decision-making, resource allocation, and overall crop management [4].

The Food and Agriculture Organization (FAO) reports that, in contrast to developing countries' production, grain demand and consumption have grown sharply. From 1964 to 2030, the demand for rice, wheat, and other coarse grains increased steadily [5]. Cereal imports into emerging nations increased significantly between 1970 and 1997–1999, from 39 million tonnes to 130 million tonnes annually, in response to the rising demand. It is anticipated that imports will continue to climb and may even get worse in the years to come. It is anticipated that by 2030, these developing nations will import 265 million tons of grains annually, or about 14% of their total consumption[6]. As a result, the state of the global market is extremely unstable and seeing a decline in actual prices.

These market conditions have the potential to be disastrous for the development of countries that are not considering taking action to lessen their total reliance on imports for traditional crops. Therefore, changing the current situation and helping countries become increasingly self-sufficient in meeting their food needs is a global problem, which calls for a timely and precise assessment of crop production [5]. Crop yield estimation is crucial, but accurate measurement is a very difficult and tough task due to the presence of multiple connected environmental factors. Weather differences affect plants at different phases of growth, resulting in significant variances in yield within a season. Accurately determining agricultural production is made more difficult by the regional variability of soil qualities, farmer decisions about irrigation frequency, fertilizer and pest management, crop rotation, and site preparation techniques. Therefore, conducting crop and soil management experiments is necessary to accurately assess weather and soil factors in order to design an efficient and reliable crop yield forecast system. Environmental conditions always affect crop output.

These elements have different effects on crops at different phases of growth. In order to forecast the final production or yield of a crop, these various interactions between plant physiological systems and environment can be represented mathematically [22]. These mathematical representations are known as crop growth models or crop process

models. The daily crop development simulator is used by these mathematical models to estimate the potential biomass production and to give an abstract perspective of how the physiological stages of the plant's dynamic behavior are implemented [24]. The real data and multiple hypotheses regarding different types of soil, solar radiation, and different management techniques used, rainfall, and temperature fluctuations are incorporated into models that simulate how seeds develop and grow into plants.

This paper aims to investigate the performance of different ML regression model on our dataset and then by selecting the best three models as base model, ensemble methods like Voting repressor, Stacking Repressor is applied to get the best yield prediction result for rice, maize, sugarcane and potato crop.

Chapter 2
Literature Review

2.1 Overview

The review of Chlingaryan et al. on machine learning prediction of nitrogen status emphasizes how merging sensing and machine learning technologies in agriculture can have a revolutionary effect . The findings underscore a significant shift in farming methods and highlight the rapid evolution of technologies that could soon offer affordable alternatives. The synergy of modern sensing tools and machine learning algorithms has emerged as a powerful combination, enabling precise and effective assessments of the nitrogen status of crops. This invaluable information empowers farmers with critical insights for targeted nutrient management, marking a notable advancement in precision agriculture [7].

The contemporary landscape of precision agriculture has entered a new phase characterized by the swift development of sensing technology, including drones equipped with sophisticated sensors, and the integration of machine learning algorithms in data processing. This interconnected approach facilitates real-time monitoring of crops without causing any damage, providing farmers with timely information to make informed decisions about fertilization plans. The affordability of these solutions, as proposed by Chlingaryan et al., introduces a positive trend towards technologies that are not only useful but also accessible for farmers, regardless of the scale of their operations.

Beyond its immediate implications for estimating nitrogen status, the study reflects a broader trend within the agriculture industry. The amalgamation of sensing technology and machine learning not only enhances the accuracy of crop monitoring but also aligns with sustainable agricultural practices. By minimizing environmental impact and maximizing resource utilization, this approach promotes a more environmentally conscious and resource-efficient form of agriculture. The consequences of this research extend beyond the specific context of nitrogen status estimation, pointing towards a future where technology provides farmers with accurate, data-driven insights to enhance both productivity and sustainability. The promise of scalable and reasonably priced agricultural solutions heralds a transformative era, wherein technology becomes an integral tool for farmers, ushering in a new age of precision, efficiency, and

environmental stewardship in agriculture. As these innovations continue to unfold, the vision of a technologically empowered agriculture sector that contributes to global food security while prioritizing sustainability comes into clearer focus [7].

In their effort to address challenges associated with evaluating features in nodes with limited instances, Deng et al. introduce the Regularized Random Forest (RRF) as a feature selection technique. They identify a potential issue with RRF in selecting weakly relevant characteristics in situations where nodes have few instances and propose a solution by establishing an upper bound for the number of unique Gini information gain values in a node. To further refine this approach, the authors introduce an enhanced version called the Guided RRF (GRRF). In GRRF, the feature selection process in RRF is guided by important ratings obtained from a standard Random Forest (RF).

The study employs ten gene datasets to assess the performance of RRF and GRRF, demonstrating that GRRF generally outperforms RRF in terms of accuracy, particularly when parameters are adjusted. The strengths of GRRF over other methods, such as RRF, varSelRF, and LASSO logistic regression, include competitive accuracy performance, computational efficiency, and the ability to select compact feature subsets [28].

Notably, the study reveals that, for the majority of datasets, Random Forest applied to features chosen by RRF with minimal regularization performs better than Random Forest applied to all features. This observation underscores the utility of RRF in enhancing prediction accuracy. The research sheds light on the limitations of weak classifiers in capturing information present in feature subsets and emphasizes the evaluation of feature selection techniques based on the accuracy performance of Random Forest, a robust classifier. In summary, Deng et al.'s exploration of Regularized Random Forest and its enhanced variant, Guided RRF, offers valuable insights into addressing challenges associated with feature selection in nodes with limited instances.[30] The comparative analysis with other techniques, coupled with the emphasis on accuracy performance, contributes to the understanding of the strengths and limitations of these approaches in the context of gene datasets [8]. Gopal et al. demonstrated how feature selection, which takes into account the model utilizing all the features as a baseline, can effectively be used to generate a crop yield forecast

model. They demonstrated that employing the chosen features (all the characteristics) to create the crop yield prediction model in Tamil Nadu, India, resulted in an adjusted R 2 of 85% (84%) [9]. With an emphasis on climatic characteristics, Elavarasan et al.'s assessment on machine learning models related to agricultural production prediction offers insightful information about the multidisciplinary nexus of technology and agriculture. Their recommendation to investigate a wider range of criteria highlights how intricate and diverse the elements affecting crop output are. The authors urge researchers to take into account a wide range of variables beyond climatic considerations by recommending a comprehensive approach, acknowledging the interdependence of many components in the agricultural environment [10].

Liakos et al. (2018) made a similar contribution to the subject in parallel by publishing a review paper on the wider uses of machine learning in agriculture. Their analysis covers a wide range of topics, including crop management, animal management, water management, and more, in addition to crop yield prediction. The comparison of these two research shows how scientists are working together to use machine learning to improve agriculture holistically. The focus that Elavarasan et al. placed on climatic characteristics is consistent with the understanding that weather conditions are critical to crop growth. In the meantime, the thorough assessment of Liakos et al. demonstrates the vast potential of machine learning applications in changing different facets of agricultural operations. When taken as a whole, these studies demonstrate the dynamic field of study at the nexus of technology and agriculture, highlighting the necessity of multidisciplinary methods to handle the complexity of contemporary farming [11]. A decision support system utilizing the capabilities of recurrent neural networks (RNNs), namely Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), as well as their bidirectional equivalents, Bidirectional Long Short-Term Memory (BLSTM) and Bidirectional Gated Recurrent Units (BGRU), was presented by Alibabaei et al. [2]. This system's main goal is to increase end-of-season crop production estimation accuracy, which is important for agricultural planning. The Bidirectional Long Short-Term Memory (BLSTM) model demonstrated better performance with a Mean Squared Error (MSE) ranging from 0.017 to 0.039, outperforming the other RNN architectures, according to the authors' research. The use of BLSTM demonstrates how well bidirectional models capture contextual data from previous and next time steps, improving prediction accuracy. This suggested decision assistance system is important

in ways that go beyond scholarly investigations. The method offers farmers useful information by accurately estimating crop yields at the end of the season. The features of the system are intended to help farmers make well-informed choices regarding the frequency and effectiveness of irrigation on their properties. This is in line with the more general objectives of precision agriculture, which uses cutting-edge technologies like RNNs to improve crop management techniques, maximize resource use, and eventually raise agricultural productivity [12].

Using a variety of machine learning algorithms, Sarijaloo et al [13]. conducted a study aimed at forecasting the yield performance of tested corn hybrids. Decision trees, gradient boosting machines, random forests, adaptive boosting, XGBoost, and neural networks were among the models they investigated in their study. Finding the best model to predict corn production was the aim, which was then extended to unproven inbred and tester combinations.

Based on their analysis, they found that XGBoost performed better than the other machine learning models on average, with an RMSE of 0.0524. The better results of XGBoost indicate that it is useful for identifying intricate patterns in the data and producing precise estimates of maize output. The resilience of this model is especially impressive when compared to other models.

The decision tree model, on the other hand, performed the poorest. This result is explained by the intrinsic properties of decision trees, which are thought to be weak learners and prone to overfitting. When a model over fits to training data, it captures noise or random oscillations in the data, which results in poor generalization to new, unknown data. Applying the chosen XGBoost model to untested inbred and tester combinations showed its usefulness in real-world scenarios [31].

The approach was successful in identifying hybrids with high expected yields in this way. The use of machine learning in this way has important ramifications for corn production since it makes it possible to identify viable hybrid combinations that can be carefully developed to increase total maize yield [14].

In order to provide a baseline for large-scale crop production prediction, Paddle et al. used an innovative approach by merging machine learning with agronomic concepts of

crop modeling. The goal was to create a machine learning workflow where accuracy, modularity, and reuse were given top priority.

They used crop simulation outputs and integrated data from several sources, such as weather, remote sensing, and soil information from the MARS Crop Yield Forecasting System (MCYFS) database, to develop the features for their machine learning model. The goal of this extensive feature set was to capture the various factors that affect crop productivity, giving machine learning algorithms a wealth of data. Three different machine learning techniques were included in the suggested workflow: k-Nearest Neighbors, Support Vector Regression (SVR), and Gradient Boosting [33]. These algorithms were used to forecast the yields of many crops at the regional level in three European countries: the Netherlands, Germany, and France. The crops included sunflower, sugar beet, spring barley, and potatoes.

The importance of using domain-specific knowledge to improve prediction accuracy was brought to light by the machine learning workflow's incorporation of agronomic concepts. Through the integration of diverse environmental data with crop simulation outputs, the model has the potential to capture complex interrelationships between different factors that impact crop production. The application of machine learning algorithms like SVR, k-Nearest Neighbors, and gradient boostinging highlighted the adaptability of these techniques in managing big, real-world agricultural datasets [15].

A thorough investigation was carried out by Shahhosseini et al. with the objective of determining how crop modeling and machine learning integration could improve maize production projections in the US maize Belt. Their main goals were to ascertain whether a hybrid technique that included crop modeling with machine learning might produce better forecasts and, if so, whether combinations of hybrid models and crop modeling features would produce the most accurate maize yield estimates.

The study found that a significant reduction in yield forecast Root Mean Square Error (RMSE), ranging from 7% to 20%, was achieved by using simulation crop model variables as input features into machine learning models [16].

A new method for utilizing Support Vector Regression (SVR) to forecast rice yield was presented by Jackal et al. The soil nitrogen prediction, rice stem weight prediction, and rice grain weight prediction were the three separate stages of their prediction process.

This thorough approach represents a sophisticated grasp of the several factors affecting rice yield, enabling a more precise and thorough prediction procedure.

The authors utilized a commercial program called DSSAT4, which implements the Crop Simulation Model (CSM-Rice simulation model), to compare the results of their SVR-based model with those acquired from it in order to confirm their findings. This comparative study offers a standard by which to measure the performance of their suggested method in comparison to well-known simulation models [17]. Dey et al. investigated the possibility of various data mining algorithms for rice yield prediction in a similar study [18]. Support vector regression (SVR), modified nonlinear regression (MNR), adaptive boosting (AdaBoost), and multiple linear regression (MLR) were all compared. Using training data, the algorithm parameters were adjusted. These optimized models were then used to estimate rice yield for test data, in conjunction with independent variables. Key measures, including as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-square, were used to assess the effectiveness of the model. This thorough assessment method yields a solid evaluation of each algorithm's prediction power and sheds light on its respective advantages and disadvantages. The linear discriminant analysis algorithm is used for feature extraction in the research by Gupta el al., whereas the Relief algorithm is used for feature selection. Afterwards, , K-nearest neighbor, and random forest are the machine learning predictors used for classification in this scenario [19].

To construct a predictive model for agricultural productivity, this all-inclusive methodology integrates feature selection, machine learning algorithms, and data-driven approaches. The focus on different predictors, such as random forest, K-nearest neighbor, shows a sophisticated comprehension of the range of variables affecting agricultural productivity. Their suggested framework gives farmers a useful tool to help them choose crops wisely, which will ultimately improve productivity, maximize yields, and promote sustainability in agriculture as a whole. Another study by Agarwal et al. determines the best crop option by utilizing machine learning (ML) methods such Random Forest, Decision Tree, and Artificial Neural Network (ANN), building on previous research in the field [20]. Still, by applying deep learning techniques to improve the suggested model, this work makes a major advancement. The program not only forecasts which crop will be best, but it also gives precise information on the quantities of soil materials needed and the costs involved. With greater accuracy than

its predecessors, this improved model gives farmers an effective tool for making decisions. In order to demonstrate the efficacy of Support Vector Machine (SVM) as a machine learning technique, it is used in this research to predict crops. Furthermore, the model gains sophistication from the integration of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) as deep learning techniques. By enabling the model to examine intricate relationships and patterns in the data, these deep learning strategies help the model provide predictions that are eventually more accurate. Their study intends to create a comprehensive and advanced system for crop prediction, resource optimization, and cost-effective agricultural methods by merging machine learning and deep learning methodologies. Precision agriculture has a promising future thanks to the integration of SVM, LSTM, and RNN, which represents a multifaceted strategy to solve the challenges of predicting the most productive crop under variable conditions.

Another study by Mamun et al. shows several machine learning techniques were contrasted. Machine learning algorithms were utilized to estimate the jute yield after a model was built based on input parameters [21]. With an average prediction accuracy of 96%, the decision tree Regression performed better than all the other algorithms taken into consideration. On the other hand, a number of machine learning techniques, including regression, logistic regression, k-nearest neighbor, and gradient boosting, demonstrated average prediction accuracies of roughly 42%, 40%, and 92%, respectively. With a 96% prediction accuracy, the decision tree regression proved to be the most successful algorithm in this situation. A more complex insight of each algorithm's performance can be gained by comparing it to other algorithms. The research's conclusions offer insightful information about how machine learning methods might be applied to predict jute yield. With a 96% prediction accuracy, the decision tree regression proved to be the most successful algorithm in this situation. A more complex comprehension of the algorithms' varying performance levels is provided by the comparison with other algorithms. The study's conclusion emphasizes the importance of this research for jute-related future studies and suggests that future academics working on related topics may find value in the knowledge gathered. Hasan et al. present K-nearest Neighbor Random Forest Ridge Regression (KRR), an ensemble machine learning technique that was developed by an analysis of five machine learning algorithms already in use: Support Vector Regression, Naïve Bayes,

Ridge Regression, Random Forest, and Cat Boost [22]. The four traditional metrics of mean absolute error, mean square error (MSE), root MSE, and R2 are used to assess the suggested KRR model. The outcomes show that KRR is effective in predicting crop production; low MSE values and strong R^2 values across a variety of crops indicate promising performance. When the Diebold–Mariano test is used to evaluate KRR's robustness against benchmark machine learning models, it typically demonstrates a considerable improvement at the 1% and 5% significance levels.

An ensemble deep learning system that forecasts rice crop yield based on soil nutrient levels is proposed by R.K. et al. approach leverages a comprehensive dataset comprising both crop production statistics and soil nutrient information as input data [23]. Crop production statistics provide insights into the quantity of crops produced in a specific area, while soil nutrient information encompasses details about the various nutrients present in the soil. During the pre-processing stages, Karim addresses missing values in the input dataset through normalization and mean attribute approaches. These techniques ensure the creation of a clear and complete dataset, laying the groundwork for subsequent prediction modeling.

The central component of Karim's methodology for categorization is the Model Agnostic Meta-Learning (MAML), a stacking-based ensemble deep learning algorithm. MAML combines the outputs of three different classifiers—Support Vector Machine (SVM), Deep Neural Network (DNN), and Deep Belief Network (DBN). The amalgamated output from MAML serves as the ultimate forecast, predicting the anticipated quantity of rice crop in a specific soil.

The results of this approach demonstrate its effectiveness, achieving a noteworthy accuracy of 89.5%. This high level of accuracy signifies that the model design is reliable and effective in predicting rice crop yield. Karim's work addresses the challenge of improving predictions of rice crop output based on soil nutrition levels, aiming to enhance accuracy and reliability. The utilization of diverse classifiers and ensemble deep learning contributes to the robustness of the model, offering a promising avenue for more precise and trustworthy predictions in the realm of agriculture.

The random forest algorithm is a unique method proposed by Kamath et al. for the quick and effective analysis of agricultural yield estimates [24]. The fact that they acknowledge that managing large amounts of data is necessary for crop output forecasts

emphasizes the importance of their idea. The authors suggest using data mining approaches to tackle this problem, believing that they are especially appropriate for the complexity of agricultural yield prediction. In this setting, data mining—a technique distinguished by its capacity to extract information from large databases that was previously unknown and expected—emerges as a vital tool. The authors stress its importance in revealing patterns and features linked to crop production in the future. Data mining is a crucial method for understanding agricultural systems better since it makes it easier to explore complex patterns in big datasets. This work highlights the wider importance of utilizing data mining techniques in the field of agricultural output forecasting in addition to introducing a particular algorithmic methodology. Malik et al. shows a study to apply machine learning algorithms to estimate crop output and fertility, with a focus on three important crops: tomato, chili pepper, and potato [25]. Specifically, the data gathered for these crops was utilized by the writers to carry out their research. The Decision Trees classifier, the Naive Bayes algorithm, and the K-Nearest Neighbor method were the machine learning algorithms selected for their investigation. These algorithms were chosen due to their effectiveness and applicability for agricultural yield estimation. Their work highlights the importance of machine learning in agriculture and its ability to provide insightful predictions about crop production and fertility for particular crops. A comprehensive and varied approach to tackling the intricate challenges associated with crop yield prediction is demonstrated by the Decision Trees classifier, which is renowned for its capacity to handle complex decision-making scenarios; the Naive Bayes algorithm, which is acknowledged for its simplicity and efficiency in probabilistic classification; and the proximity-based K-Nearest Neighbour approach [35]. The integration of various machine learning methods points to a comprehensive approach to address the complex issues surrounding agricultural production prediction, taking into consideration the particular traits and needs of each crop. In a study, Iniyan et al. proposed an inventive ensemble regression crop prediction model [26]. They found that this original methodology performed better than a variety of supervised machine learning and advanced ensemble learning techniques. Their research was focused on creating a crop yield prediction model, and the findings showed that their ensemble regression model outperformed other well-known methods. When it comes to agricultural forecasting, where precise crop production estimation is essential, their novel ensemble regression model performed exceptionally well [36]. The predictions from several separate models are usually

combined in ensemble regression models, and in this investigation, the ensemble approach outperformed both more sophisticated ensemble learning techniques and conventional supervised machine learning techniques. The phrase "sophisticated ensemble regression crop prediction model" refers to more complex and sophisticated ensemble methodology. An important consideration in agricultural planning and decision-making is crop production, which our model showed to be more accurate in predicting. The favorable results of their investigation highlight the advantages of using cutting-edge ensemble regression methods for predicting crop yield, demonstrating developments in the field of agricultural data analysis and prediction modeling.

Nazir et al. proposed a phenology-based approach and a linear regression model to estimate rice crop yield across various phonological periods [27]. The primary focus of their study was to assess rice crop yield using hyper-temporal vegetation indices derived from time series Sentinel-II satellite imagery. To achieve this, the study employed a range of vegetation metrics, and the accuracy of the model was evaluated using statistical techniques such as Mean Error (ME) and Root Mean Squared Error (RMSE).

The study specifically aimed to enhance the precision of rice yield prediction by utilizing sequential time-stamped vegetation indices and employing partial least squares regression (PLSR). PLSR is a statistical technique commonly used in regression analysis and chemo metrics, known for its effectiveness in handling multicollinearity in predictor variables [39]. The choice of PLSR in this investigation suggests its success in capturing the intricate relationship between vegetation indicators and rice crop yield.

It is important to note that the accuracy of the PLSR algorithm was not directly compared to other machine learning methods employed in the study, such as support vector machines, artificial neural networks, or other geostatistics often used in yield forecasting systems. Despite this, the study's emphasis on phenology-based strategies and linear regression models contributes to our understanding of how vegetation derived from satellite imagery can be leveraged for accurate yield estimation.

In summary, Nazir et al.'s work provides valuable insights into the application of phenology-based approaches and linear regression models for estimating rice crop yield using satellite-derived vegetation indices. While the study focuses on specific

methodologies, it offers a foundation for further exploration and comparison with other machine learning techniques in the realm of crop yield prediction.

A study by Ahmed et al. was dedicated to estimating annual crop yields in various regions of Bangladesh, utilizing a dataset provided by the Bangladesh Agricultural Research Institute (BARI) [28]. The research incorporated a diverse set of input variables, encompassing biotic components, environmental factors, and area-specific characteristics. Two distinct methodologies, K-means classification and linear regression, were employed in the prediction process.

K-means classification, as applied in this study, is a clustering method that categorizes data points into K clusters based on their similarities, thereby unveiling patterns within the dataset. On the other hand, linear regression, a statistical modeling technique, was used to establish a linear relationship between independent and dependent variables. The combination of these two methods aimed to provide insights into the anticipated yearly crop yields [40].

The study's outcomes played a pivotal role in identifying the top three crops recommended for Bangladesh's main agricultural districts based on the projected yields. This underscores the effectiveness of the K-means classification and linear regression methods in collaboratively offering valuable information for crop yield forecasts, facilitating the selection of suitable crops for different regions.

A notable aspect of the analysis is its comprehensive approach, considering biotic, environmental, and geographical variables as central factors. This thorough consideration of various variables reflects a holistic approach to understanding and incorporating all the elements that influence crop productivity. Ahmed et al.'s study not only contributes to the field of crop yield prediction but also demonstrates the synergy between K-means classification and linear regression in providing meaningful insights for agricultural decision-making. The incorporation of a diverse set of variables underscores the importance of a comprehensive approach in forecasting crop yields, enhancing the study's applicability and relevance in agricultural planning and resource allocation.

Using a dataset gathered from Warangal over a 20-year period, Aishwarya's work focuses on the prediction of rice productivity in Bangladesh. The study's input variables

include the production area, biotic components, and environmental conditions [29]. The seven attributes that make up each year's dataset are rainfall, maximum and minimum temperatures, sunlight hours, wind speed, humidity, and cloud cover [44]. Three rice varieties—Boro, Aman, and Aus—are also included in the data on rice yield for each year. Clustering algorithms are applied in the study's approach to split regions according to specific features. Afterwards, appropriate categorization methods are used to anticipate crop yields. ANNs and linear regressions are two examples of these methods that were employed in the study for prediction. The results show that ANN performs better in predicting some crops, especially Aus, where the dataset may have more missing values. Artificial Neural Networks are useful in situations where the data is noisy or incomplete because of their well-known capacity to identify intricate patterns and relationships within the data. However, for the Boro and Aman types, Linear Regression shows superior prediction performance.An ensemble deep learning system that forecasts rice crop yield based on soil nutrient levels is presented by Siddique et al. [30]. As input data, this system considers crop production statistics as well as soil nutrients. Crop production statistics show the amount of crop produced in a given area, whereas the soil nutrients information covers the different amounts of nutrients found in the soil. Model Agnostic Meta-Learning (MAML), a stacking-based ensemble deep learning algorithm, is the central component of the methodology used for categorization. Three different classifiers' outputs are combined by MAML: Support Vector Machine (SVM), Deep Neural Network (DNN), and Deep Belief Network (DBN). The ultimate forecast, which shows the anticipated quantity of rice crop in a specific soil, is derived from the combined output of MAML. The outcomes reveal that the suggested strategy is effective, with a noteworthy accuracy of 89.5%. This degree of accuracy indicates that the model's design predicts crop yield in a reliable and effective way. The methodology seeks to improve soil nutrition level-based predictions of rice crop output in terms of accuracy and reliability by utilizing diverse classifiers and ensemble deep learning.

Using a Recurrent Neural Network (RNN) model based on deep learning, Bali et al [31]. tried to forecast wheat crop yield in the northern area of India. Because deep learning models, and RNN in particular, are good at extracting features from big datasets, they have an inherent advantage that makes them suitable for precise prediction in complicated circumstances. [18]Additionally, by utilizing Long Short-

Term Memory (LSTM) units—which are intended to capture long-term dependencies in sequential data—the study tackles the vanishing gradient issue that is connected to RNN models. To provide a thorough historical perspective for the prediction problem, the trials are conducted using a benchmark dataset that spans 43 years. Random Forest, Multivariate Linear Regression, and Artificial Neural Network are the three machine learning models that are contrasted with the suggested RNN-LSTM model.

The RNN-LSTM model's performance is demonstrated by its lower Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) values when compared to the other models. In particular, the RNN-LSTM model performs better than the others, showing that it can predict wheat crop yield with an RMSE of 147.12 and an MAE of 60.50 [41]. The suggested RNN-LSTM approach's efficiency is further highlighted by comparison with other machine learning models, such as Multivariate Linear Regression, Random Forest, and Artificial Neural Network.

The study shows that the RNN-LSTM model can produce predictions that are almost accurate in predicting crop production, in addition to validating the model's efficacy in doing so. This highlights how deep learning methods, particularly RNN with LSTM units, can be an effective tool for improving agricultural production estimates' accuracy when it comes to growing wheat in northern India. This research study contributes the following terms:

- A comprehensive agricultural dataset of rice, maize, sugarcane and potato was collected, providing essential features such as crop type, cropping season, district, area under cultivation, production quantity, annual rainfall, fertilizer usage, pesticide application, and calculated crop yield.
- The dataset underwent meticulous preprocessing, including handling missing values, scaling numerical features, and encoding categorical variables.
- A diverse set of machine learning models, including decision trees, support vector machines, random forests, and neural networks, was considered as base models. These models were chosen for their ability to capture different aspects of the complex relationships present in agricultural data.
- Each base model was trained on the preprocessed dataset, optimizing hyperparameters to achieve the best possible individual performance.

- Ensemble learning techniques, such as voting, and stacking were employed to combine predictions from multiple models.
- The models were evaluated using key metrics such as R-squared (R2) value, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics provided a comprehensive understanding of the models' accuracy, precision, and generalization capabilities.

The results obtained from individual models, ensemble techniques, and variations in final estimator models were analyzed comprehensively. Insights were drawn regarding the strengths and weaknesses of each approach.

Chapter 3
Materials and Methodology

3.1 Overview

The methodology employed in this work revolves around the application of ensemble learning techniques to enhance the predictive performance of crop yield models. The overarching goal is to leverage the collective intelligence of multiple models, addressing the limitations of individual models and improving overall accuracy in predicting agricultural outcomes. This methodology facilitated a systematic exploration of ensemble learning techniques in the context of crop yield prediction, providing valuable insights for optimizing agricultural forecasting model.

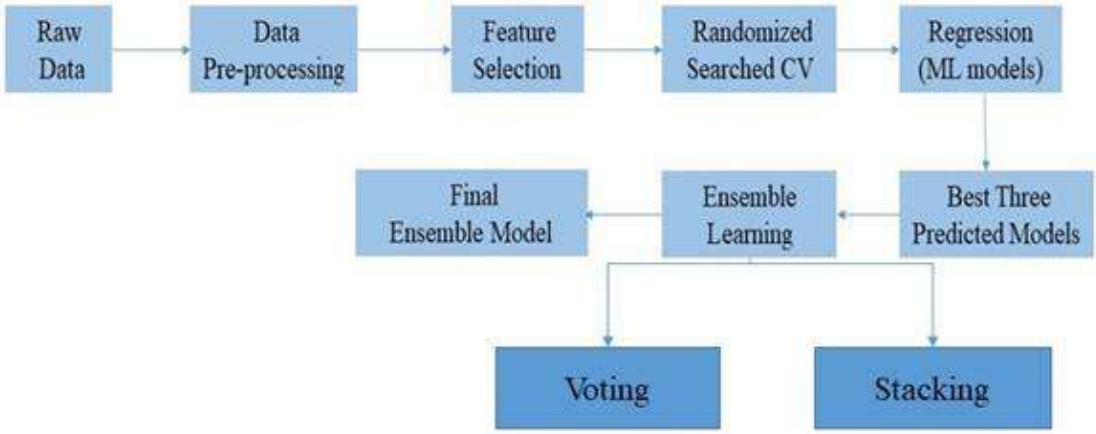


Figure 1 Proposed methodology for crop yield regression

In Figure. 1 the methodology of this work can be seen.

3.1 DATASET DESCRIPTION

This comprehensive dataset encapsulates agricultural data spanning multiple crops cultivated across diverse states in Bangladesh, encompassing the years from last two decades.

The dataset is meticulously curated to provide essential features crucial for crop yield prediction, offering valuable insights into the dynamics of agricultural production. In Fig. 2.

The count plot of the dataset of 6 season is plotted to see it visually. The crop column specific name of the crop cultivated, providing insight into the variety of crops under

consideration. This season column delineates the cropping season, such as winter, monsoon, or the entire year, allowing for a seasonal breakdown of crop cultivation.

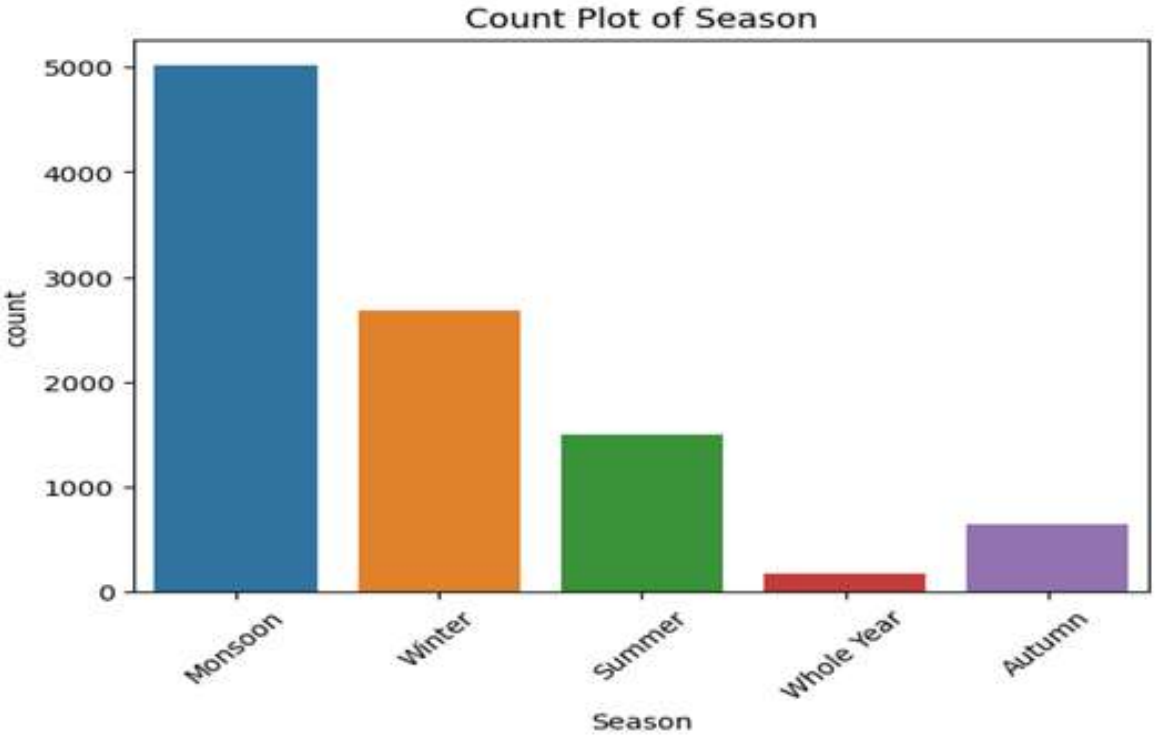


Figure 2 Maize

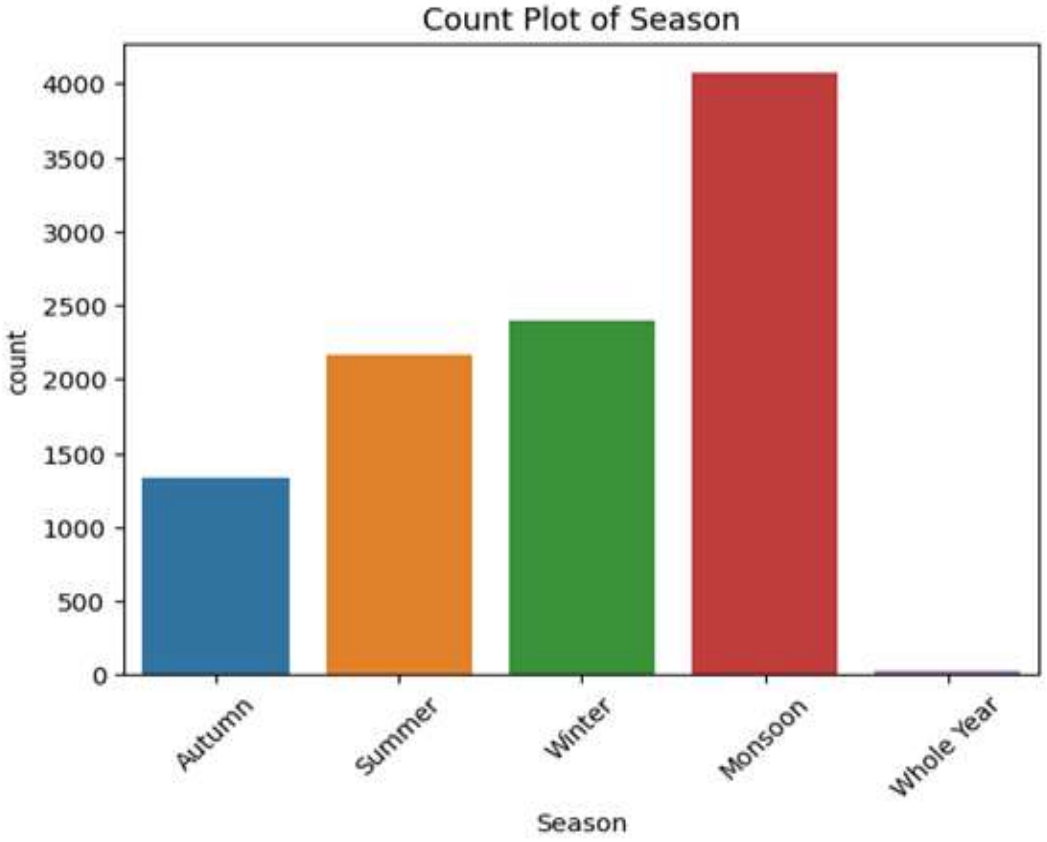


Figure 3 Rice

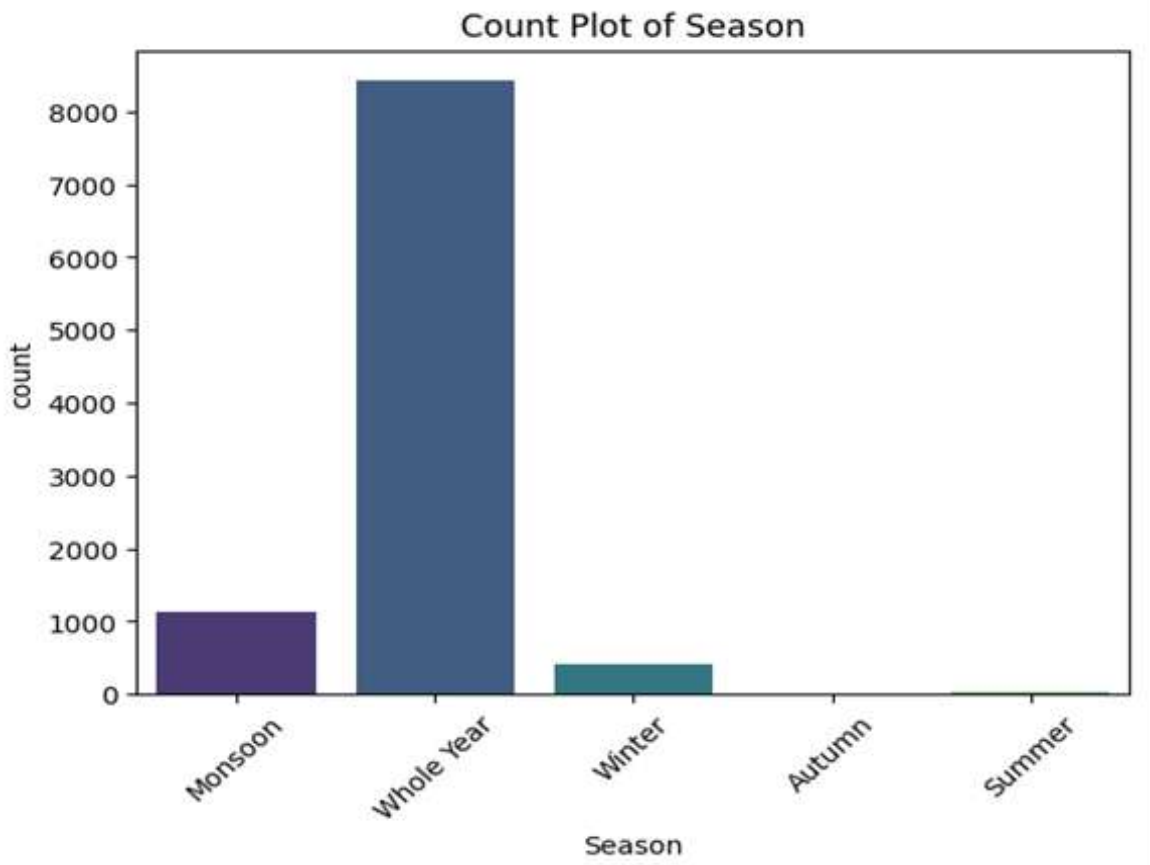


Figure 4 Sugarcane

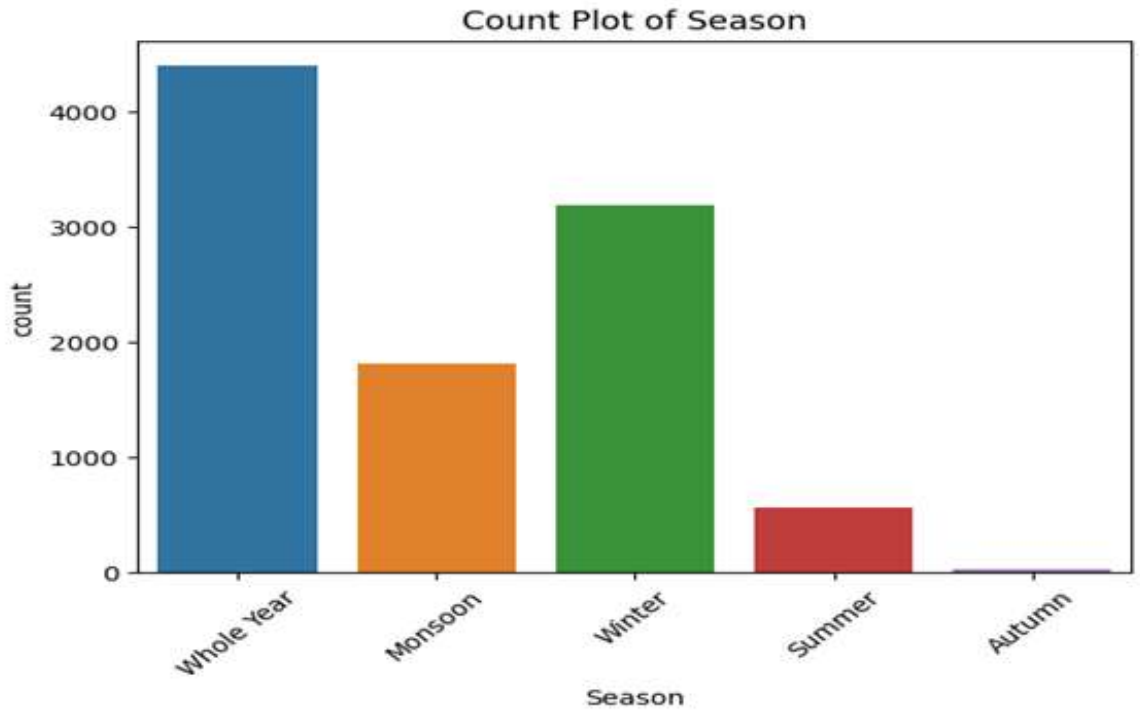


Figure 5 Potato

In Figure 2-5, Count plot of the season has given.

Indicates the Bangladeshi districts where the crop was cultivated, providing a regional perspective on agricultural practices. Area column represents the total land area under cultivation for the specific crop, measured in hectares. This feature is pivotal for understanding the scale of agricultural operations. Production column specifies the quantity of crop production, measured in metric tons, offering a quantitative measure of agricultural output. Annual rainfall column Indicates the annual rainfall received in the crop-growing region, measured in millimeters. This information is crucial for assessing the impact of climate on crop yield.

Fertilizer column represents the total amount of fertilizer used for the crop, measured in kilograms, highlighting the agricultural input employed to enhance productivity. Pesticide column denotes the total amount of pesticide used for the crop, measured in kilograms, providing insights into pest management practices. Yield column denotes The calculated crop yield, expressed as production per unit area (hectares), serving as a key metric for assessing the efficiency and success of agricultural practices.

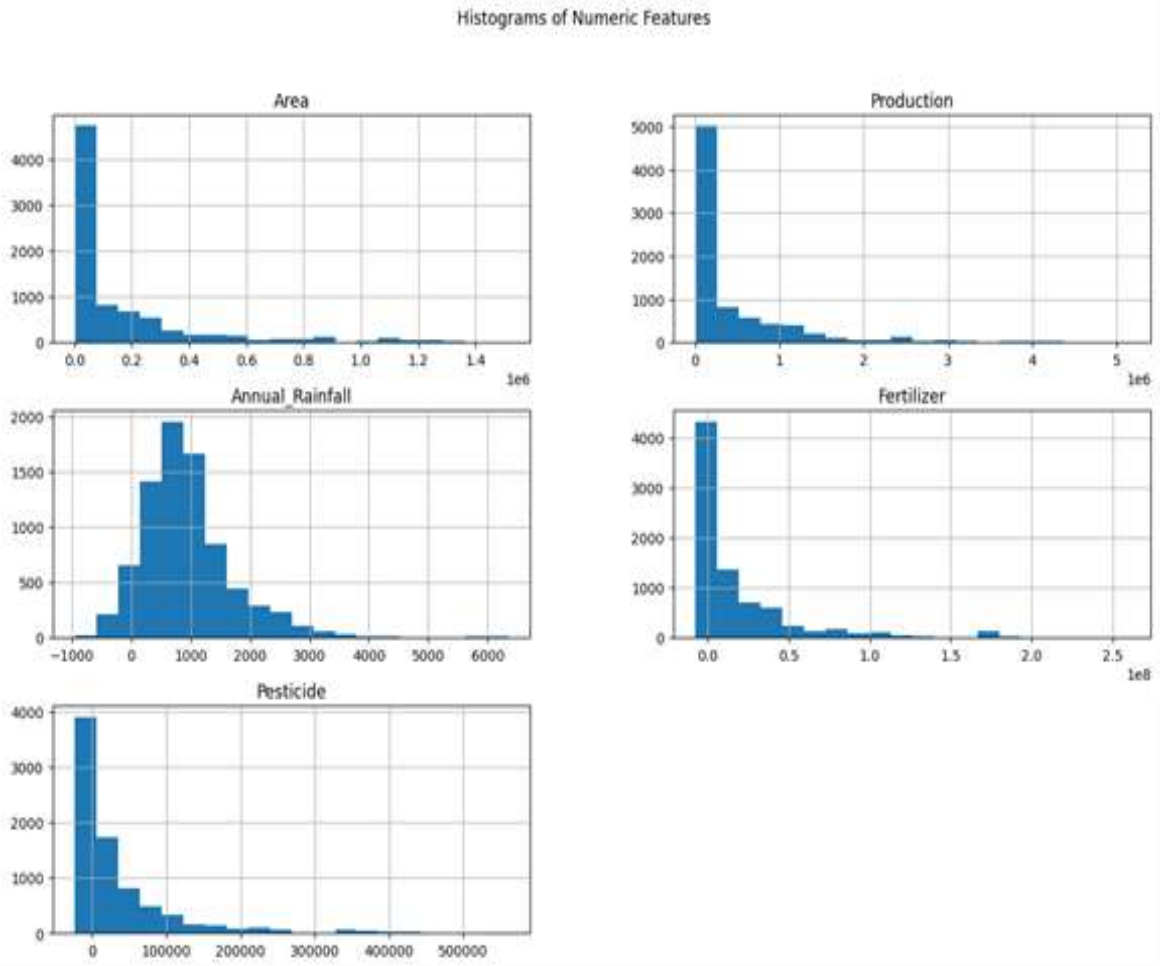


Figure 6 Histogram of the “Maize” from the dataset

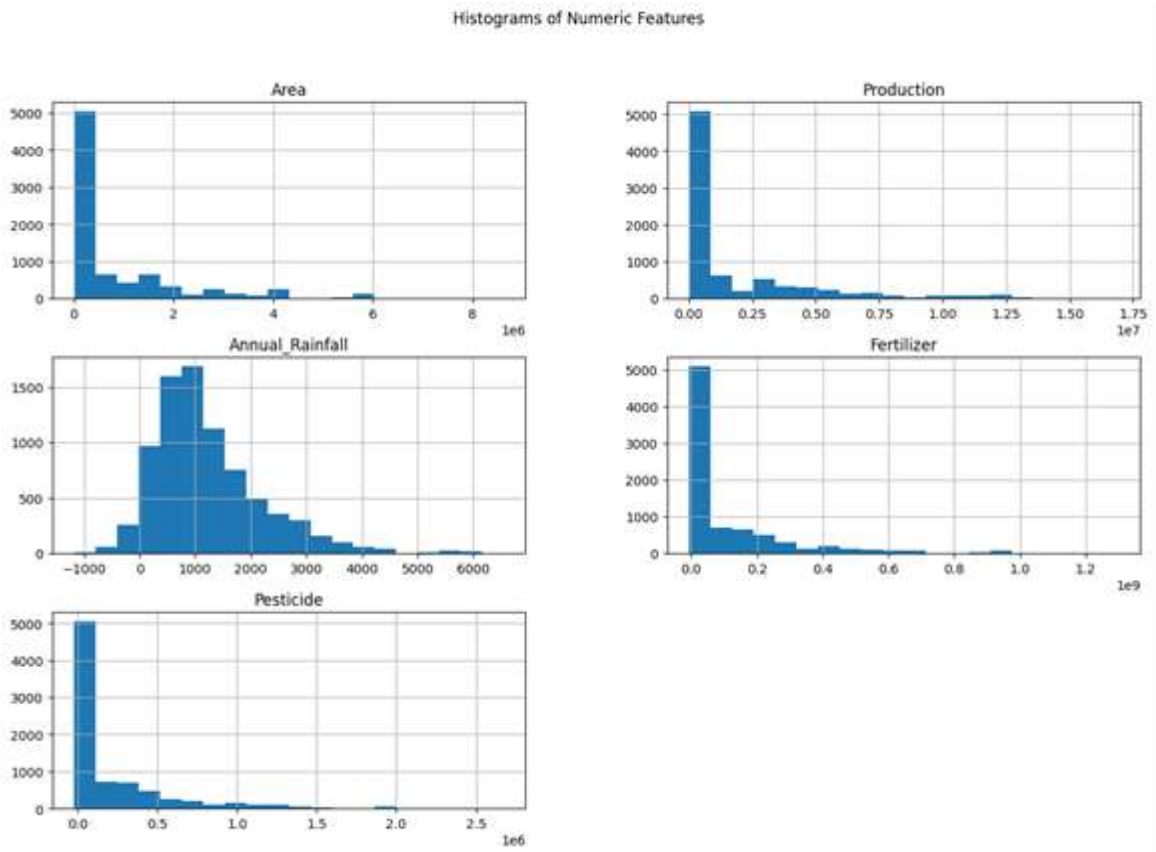


Figure 7 Histogram of the "Rice" from the dataset.

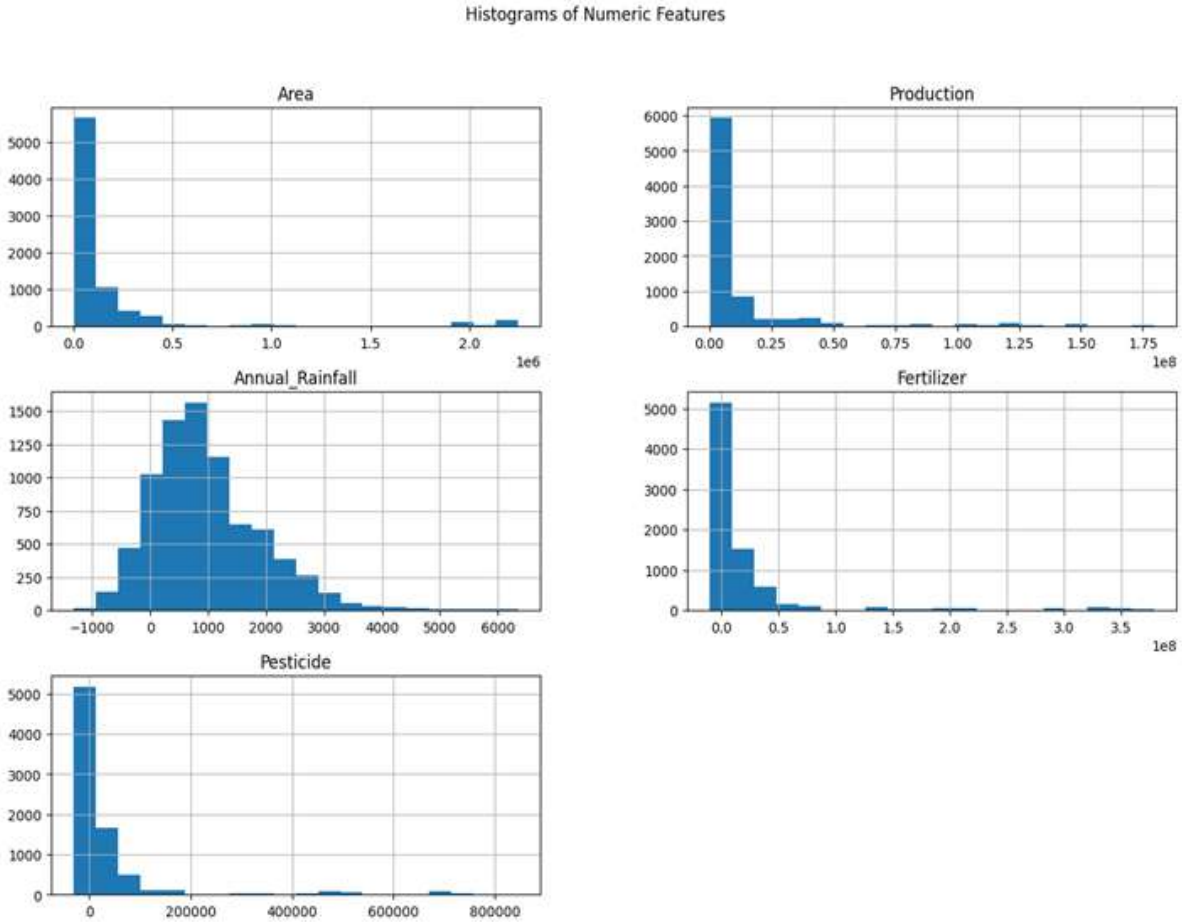


Figure 8 Histogram of the "Sugarcane" from the dataset

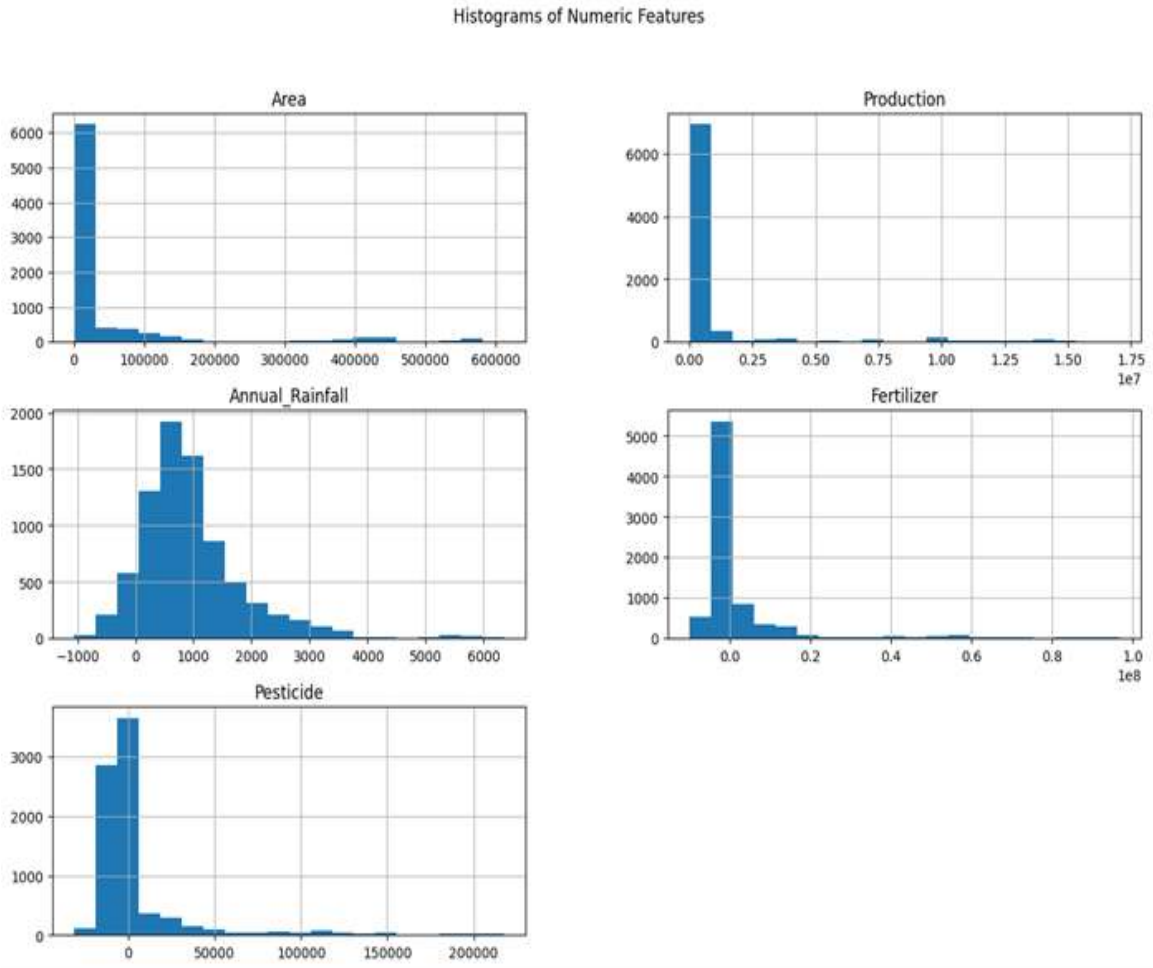
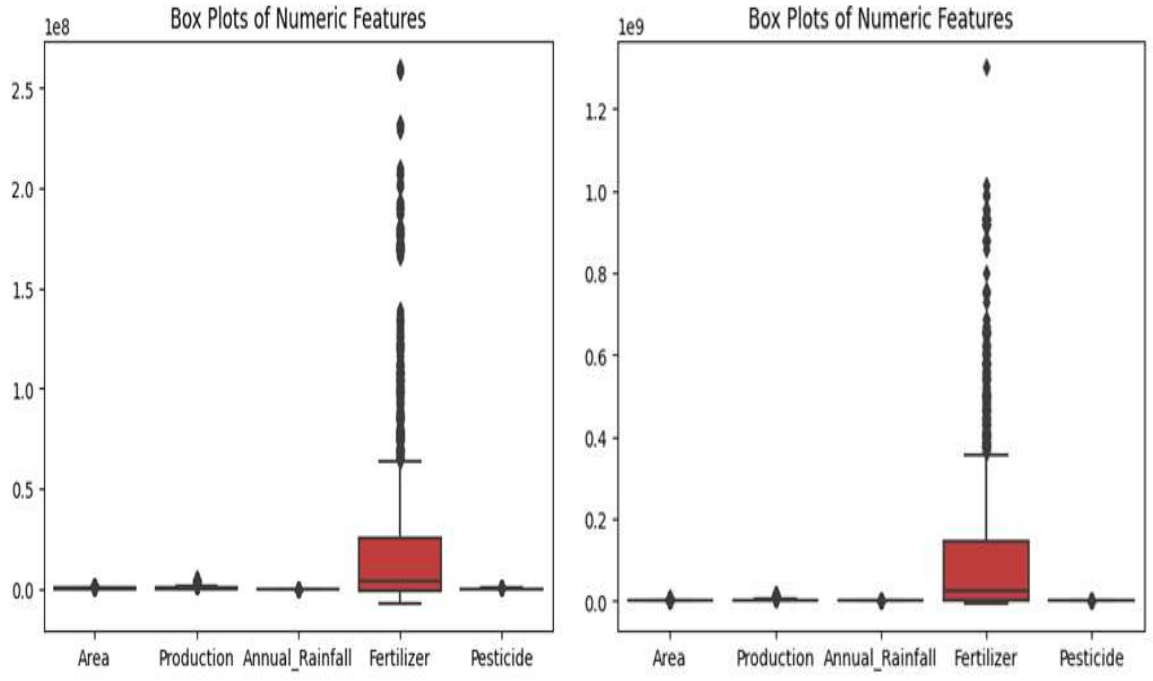
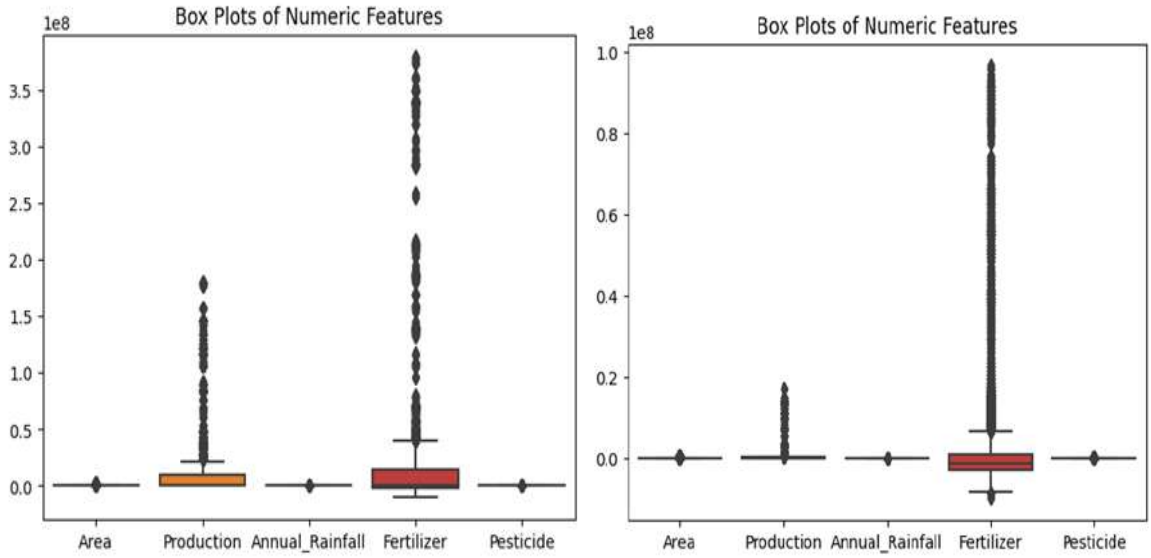


Figure 9 Histogram of the “Potato” from the dataset

In Figure 6-9. The count plot of the data of different state is plotted to see it visually.



(a) (b)



(c) (d)

Figure 10 Box plot of the numeric feature of (a)Maize, (b)Rice, (c)Sugarcane, (d)Potato

3.2 DATA PREPROCESSING

Data preprocessing is a crucial step in regression jobs since it cleans and prepares the data, resolving issues and anomalies that could impair the performance of the model. In addition to increasing the precision and dependability of regression models, a well-preprocessed dataset also helps machine learning applications perform better overall in real-world settings. Since incomplete information is frequently present in datasets, handling missing values is an essential step in data preprocessing. It is ensured that machine learning models may learn from the available data without being overly influenced by gaps in the information by properly resolving missing values. So, we handled missing values of the dataset by estimating missing values based on the values of their nearest neighbors in the feature space. Next one hot encoding is done. In machine learning, one-hot encoding is a widely used method for numerically representing categorical variables. When working with categorical data in regression or classification problems, it is especially helpful. Utilizing MinMax scaling and one-hot encoding to transform numerical and category columns. Next standard scaling is performed. Standard scaling, sometimes referred to as Z-score normalization, is a preprocessing method for data that is used to translate numerical features into a mean of 0 and a standard deviation of 1.

When working with machine learning algorithms that are sensitive to the size of input characteristics, this procedure is quite helpful. After that the data are splitted into train

and test set where 80% is taken in training set and 20% is taken in test set. Preparing the dataset to be compatible with regression algorithms is the main goal of data pre-processing for regression tasks, such as utilizing machine learning to estimate crop yield in Bangladesh. For regression-focused data pre-processing, follow these particular steps:

3.2.1 Managing Value Missing

Determine which of the dataset's missing values to address. Imputation, which replaces missing numbers with approximated values based on other data, and row/column deletion are two methods. Think about the kind of missing data and how it might affect the prediction process.

3.2.2 Handling Unusual Data

Recognize and deal with outliers, as they can have a big impact on regression models. Removing extreme data or using robust regression algorithms, which are less susceptible to outliers, are two possible options.

3.2.3 Handling Outliers

Regression models can be greatly impacted by outliers, thus it's important to identify and manage them. Removing extreme data or using robust regression algorithms, which are less susceptible to outliers, are two possible options.

3.2.4 Normalization/Standardization

To bring numerical features to a same scale, normalize or standardize them. This is especially crucial for regression methods like linear regression that depend on the size of the input features.

3.2.5 Encoding of Categorical Variables

Provide numerical representations for any categorical variables (such as soil type or crop type) that are included in the dataset. For this, one-hot encoding is a frequently used method.

3.2.6 Regression through Feature Engineering

Provide new features that could enhance the model's capacity to identify patterns in the data. You may, for instance, incorporate polynomial characteristics or construct interaction terms between features.

3.2.7 Managing Temporal Information

To capture temporal patterns that could impact crop output, think about adding time-based features or developing lag features if the dataset includes time-related data.

3.2.8 Dealing with Multicollinearity

Look for multicollinearity in the variables that predict. Regression models' interpretability and stability may be impacted by highly correlated features. Variance inflation factor (VIF) analysis is one method that can be used to detect and deal with multicollinearity.

3.2.9 Target Variable Transformation

To make the target variable more normally distributed, use transformations (such as logarithmic transformations) if the crop yield data shows skewness or a non-normal distribution. Regression models may perform better as a result. Divide the dataset into test, validation, and training sets. This is known as data splitting. This makes sure that your model is evaluated on a third set of data to assess its generalization performance after it has been trained on one set and validated on another.

3.2.10 Evaluate and Iterate

Throughout training and validation, keep a close eye on the regression model's performance. If needed, modify the pre-processing stages or try with other features.

3.2.11 Input from Domain Experts

Speak with agronomists or agricultural scientists, for example, to learn which characteristics are most important to consider when predicting crop yield in Bangladesh.

The choices made for data pre-processing should be based on the unique difficulties and features of the crop yield prediction task in Bangladesh. In order to increase predictive accuracy, evaluate your model's performance on fresh data on a regular basis and be ready to make adjustments to the pre-processing stages.

3.3 PROPOSED FRAMEWORK

After preprocessing next each regression model is optimized separately. Optimizing the hyperparameters of each model through this procedure is essential to improving its predicted accuracy. Cross-validation, a reliable method for methodically assessing

various hyperparameter combinations and measuring their performance, is employed in the iterative grid search strategy. XGBoost, CatBoost, AdaBoost, Random Forest, K-Nearest Neighbors (KNN), Decision Tree and Random Forest are some of the regression models that were been examined. A comprehensive search over predefined hyper parameter grids for every model is carried out during the grid search procedure. To ensure robustness and avoid overfitting, cross-validation is used to evaluate the model's performance on several subsets of the training data. The optimal performing models can be identified by combining grid search and cross-validation to determine the hyper-parameter values. "Best-performing models" are those combinations that produce the highest cross-validation performance measures, e.g., lowest mean squared error or highest R-squared score. Upon identifying these ideal hyper-parameters, the models undergo training utilizing the complete training dataset. From all these regression model performances best three in terms of result is used for later ensemble learning. Then ensemble learning is applied, a potent strategy that combines the capacities for prediction of several different individual models to improve overall robustness and performance. As base learners, three different ensemble techniques—voting, and stacking are used to harness the combined intelligence of previously improved regression models. These ensembles use the regression models that were previously optimized as their base learners. Performance measures such as R-squared, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are used to assess the models after they have been trained on the given training set and tested on the test set.

3.4 PROPOSED REGRESSION MODELS

There are total seven types of models we've evaluated. They are described below :

3.4.1 Decision Tree Regression

A non-parametric supervised machine learning technique called a decision tree regression model builds a hierarchical tree structure by using training data as a source of knowledge. Using decision nodes and leaf nodes, the DTR model partitions the data into more manageable groupings.

A decision node can have more sub-classes, and a leaf node can have an associated decision value. Deep tree structures are prone to overfitting and provide complex decision rules. Alternatively, many coarse decision tree structures are integrated in

Ensemble Learning Regression (ELR) for prediction in order to lessen the issue of overfitting [32].

3.4.2 K-Nearest Neighbors (KNN) Regression

K-Nearest Neighbors (KNN) Regression is a non-parametric and instance-based supervised learning algorithm used for regression tasks. Unlike traditional parametric models, KNN makes predictions by considering the 'k' nearest data points in the training set. In the context of regression, the algorithm predicts the target variable by averaging the values of its k nearest neighbors [33].

3.4.3 Random Forest

An ensemble learning technique called Random Forest Regression leverages the strength of several decision trees to produce reliable and accurate predictions for regression tasks. It is a member of the tree-based algorithm family, and its power comes from combining the predictions of a variety of trees to create one[34]. Individual decision trees have some drawbacks, such overfitting and noise sensitivity, which Random Forest helps to alleviate. A series of decision trees, each trained on a distinct subset of the data and/or characteristics, makes up Random Forest. Every tree generates a separate forecast. From the original dataset, the process generates numerous bootstrap samples, which are random samples with replacement. One of these samples is used to train each tree, adding variety to the training procedure. Only a random subset of features is taken into account at each split in the tree. This keeps a single feature from taking over the entire forest and helps the trees decorrelate. The average (or weighted average) of the predictions provided by each individual tree is frequently the final prediction for regression problems [35].

3.4.4 Gradient Boosting Regression

Gradient Boosting Regression is a potent ensemble learning technique that combines the strengths of several weak learners to produce a robust and accurate predictive model. This method produces a predictive model step-by-step. Its main goal is to reduce the errors of the preceding models in a sequential manner. It is a member of the boosting algorithm family. Decision trees are commonly used in gradient boosting as weak learners. Each of these shallow trees fixes the mistakes made by its predecessors [36]. The model's residuals, or the variations between expected and actual values, are minimized using the procedure. A new tree is fitted to the loss function's negative

gradient at each iteration, thus advancing the model in the direction of decreasing prediction errors. Iteratively, trees are added, with each new tree concentrating on the errors of the merged prior trees. The ultimate prediction is the sum of the predictions made by each tree. Regularization parameters are a part of Gradient Boosting, which regulates the model's complexity to avoid overfitting [37].

3.4.5 XgBoost Regression

Extreme Gradient Boosting, or XGBoost for short, is a sophisticated and very effective use of the gradient boosting technique. Because of its remarkable prediction performance, it is especially well-liked in data science projects and machine learning competitions. Because of its scalability, flexibility, and computational efficiency, XGBoost can be applied to a variety of regression and classification tasks. Renowned for its exceptional prediction precision, XGBoost is frequently the preferred algorithm for attaining superior results in machine learning contests. Due to its computational efficiency, the approach can be applied to time-sensitive tasks and big datasets. Regularization strategies improve the model's capacity for generalization and assist avoid overfitting [38]

3.4.6 Catboost Regression

A potent machine learning technique for regression applications is called CatBoost Regression. Working with categorical features is where it excels because it can handle them naturally and doesn't need a lot of preprocessing. Yandex is credited with creating CatBoost, which is well-known for its excellent performance and capacity for precise forecasting. The ability of CatBoost to handle categorical variables removes the need for such characteristics to be manually encoded or transformed, making it one of its primary benefits. This facilitates a more direct and effective modeling procedure by streamlining the data pretreatment phase.

To enhance overall model performance, CatBoost combines the predictions of several decision trees using an ensemble learning technique. By gradually adding new trees that correct the mistakes of the older ones, it employs a gradient boosting framework to improve the predictive power of each individual tree. In order to manage overfitting and enhance generalization to fresh, untested data, the method also integrates sophisticated regularization algorithms. CatBoost minimizes the need for significant tuning by automatically choosing appropriate values for hyper-parameters [39].

3.4.7 Adaboost Regression

AdaBoost Regression, or Adaptive Boosting Regression, is an ensemble learning technique that aims to construct a robust and accurate regression model by combining predictions from multiple weak learners, typically decision trees. The key characteristic of AdaBoost is its sequential training process, where each new model in the ensemble assigns greater weight to cases that were incorrectly predicted by the preceding models.

The foundation of AdaBoost lies in the utilization of weak learners, often shallow decision trees. Weak learners are models that perform slightly better than random chance. During the training process, these weak learners are fitted successively to the dataset, with a focus on giving higher weights to instances that were mispredicted by the prior models. This adaptive approach allows AdaBoost to continually enhance its performance.

In the prediction phase, each weak learner contributes to the final forecast based on its individual performance. The models that exhibit higher accuracy receive greater weight in the final prediction. This weighted combination ensures that the ensemble model places more emphasis on the strengths of models that perform well on the given data.

AdaBoost Regression, through its adaptive and sequential training approach, leverages the collective insights of weak learners to create a powerful and accurate regression model. The ensemble's ability to adapt and learn from prior mistakes contributes to improved generalization and predictive performance in regression tasks[40]. Adaptive Boosting, indeed focuses on addressing the mistakes made by earlier models in the ensemble. It achieves this by assigning larger weights to the instances that were misclassified by the preceding models. This strategy ensures that the subsequent models in the sequence pay more attention to the previously misclassified cases, aiming to correct those errors and improve the overall accuracy of the ensemble.

The flexibility of AdaBoost allows it to adapt to complex relationships within the data and effectively handle various regression tasks. By continually adjusting the weights of instances based on their classification accuracy in the ensemble, AdaBoost can learn from its mistakes and enhance its performance over successive iterations. This adaptability makes AdaBoost a powerful and versatile ensemble learning technique, well-suited for addressing a wide range of regression challenges.

3.5 Ensemble Method

Ensemble learning, a set of machine learning techniques, aims to enhance overall performance and robustness by combining predictions from multiple models. In contrast to single-model approaches, ensemble methods leverage the collective intelligence of diverse models to generate predictions that are not only more stable but also more accurate. This approach often involves employing a range of basic models, such as decision trees, support vector machines, and neural networks, in conjunction to achieve improved results.

The fundamental idea behind ensemble methods is grounded in the concept that different models tend to make different mistakes. By amalgamating the predictions from various models, the deficiencies of each individual model can be mitigated, leading to more reliable and accurate overall predictions. The central concept underpinning ensemble methods lies in the decomposition of bias-variance and covariance. This theoretical foundation explains the enhanced performance of ensembles over the base predictors that constitute them.

Variety serves as a cornerstone in ensemble methods, encompassing various aspects such as fuzzy methods, multi-objective optimization, data variety, parameter variety, and structural variety. This diverse integration of different elements contributes to the ensemble's ability to generalize well and perform effectively across a range of scenarios.In summary, ensemble learning represents a powerful approach in machine learning, leveraging the diversity of multiple models to achieve superior predictive performance. By addressing the limitations of individual models and combining their strengths, ensemble methods provide a robust and effective framework for tackling complex problems in diverse domains [41].

3.5.1 Stacking:

Stacking is an ensemble learning technique designed to enhance predictive accuracy by combining the outputs of multiple regression models. The process involves training distinct base regression models independently on the same dataset to capture various facets of the underlying patterns. The predictions generated by these base models then serve as input features for a meta-model, which learns how to effectively weigh and combine these diverse predictions. The entire stacking process comprises training the

base models, generating predictions, and training the meta-model. The outcome is a final regression model that generally outperforms individual base models.

The core idea behind stacking is to address issues like bias, overfitting, and limited generalization by leveraging the strengths of different algorithms or model configurations. Each base model contributes its unique perspective and strengths, and the meta-model learns to synthesize these diverse insights, resulting in a more robust and accurate final regression model. Stacking is particularly beneficial when dealing with complex datasets where a single model may struggle to capture all the intricacies [42]. It involves combining separate learners through a two-tiered approach that includes cardinal learners and meta learners. Cardinal learners operate independently, while meta learners work collaboratively. The fundamental concept behind stacking is to utilize the original information to train the cardinal learner and generate a modified dataset that will be subsequently used with the meta learner.

The output produced by the cardinal learner serves as the input features for the meta learner. The objective is to enhance prediction accuracy by applying techniques such as least squares and cross-validation with a non-negativity constraint. Stacked generalization initiates the process by creating a primary classifier based on the cross-validation practice data partition. Subsequently, each block in the training dataset is divided, and the initial training of the classifier is performed using blocks from the practice data.

In essence, stacking leverages the strengths of individual learners, combining them through a meta-learning process to achieve improved predictive performance. This two-tiered approach, involving cardinal and meta learners, allows for a more nuanced understanding of the data and enhances the overall effectiveness of the ensemble model [43]. The classifier is first trained using blocks of the training data once the entire training dataset has been divided into blocks. Subsequently, every classifier is evaluated based on the block that was not accessible during training. The training data for the Meta classifier, which ultimately produces a combination rule for the primary classifiers, is created by the classifiers' output on the training blocks[44]. For our work we selected best three regression models result to put on stacking to perform. The base regression models included in the ensemble are the pre-trained models for K-Nearest

Neighbors, Random Forest, and Gradient Boosting as they showed the best result among all other regression models.

3.5.2 Voting Regression

Voting classifiers are commonly employed in classification problems, aiming to predict the class labels of incoming cases. However, a similar concept, often referred to as a "voting Regression " or "regression ensemble," can be applied in regression problems. Regression through voting represents another popular ensemble technique in the context of regression tasks.

In voting regression, conceptually distinct regression models are combined, and the average projected value is returned as the final prediction. Unlike bagging, where multiple base models are trained on different subsets of the data, each voting regression base model fits the entire dataset. The key idea is to leverage the diversity and unique strengths of different regression models to enhance the overall predictive performance.

This ensemble technique is particularly useful in scenarios where a single regression model may not capture all the nuances in the data. By combining the predictions of multiple regression models, a voting Regression aims to provide a more robust and accurate prediction for regression problems. The ensemble approach mitigates the limitations of individual models, leading to improved generalization and predictive accuracy in regression tasks [5]. In the context of regression tasks, an ensemble approach known as a voting Regression is employed to predict continuous values by combining the results of several regression models. Unlike classification problems where the objective is to predict discrete class labels, regression deals with predicting continuous outcomes.

A voting Regression comprises multiple basic regression models, similar to a voting classifier in classification tasks. These models can either be a combination of distinct regression methods, all trained on the same dataset, or they may be of the same type. The fundamental idea is to aggregate the predictions of these individual regression models to obtain the final regression output.

The ensemble technique of a voting Regression leverages the diversity and complementary strengths of different regression models, aiming to provide a more accurate and robust prediction for continuous outcomes. This approach helps overcome

the limitations of individual models and contributes to improved generalization and predictive accuracy in regression scenarios [45]. Regression voting is a process that involves combining predictions from base regression models using techniques such as weighted averaging or simple averaging. In this approach, the output of each individual model is considered, and the aggregate prediction of the models influences the final outcome. To fully capitalize on ensemble learning, it is crucial to employ a diverse set of base regression models.

The diversity in base regression models can be achieved through various means, including using different feature subsets for training, varying hyper parameters, or employing alternative algorithms. This diversity is instrumental in capturing a broader range of patterns and relationships within the data, ultimately enhancing the overall predictive performance of the ensemble. By aggregating predictions from multiple base regression models, regression voting aims to leverage the strengths of each model and mitigate the impact of individual model weaknesses. Weighted averaging or simple averaging allows the ensemble to make predictions that are more robust and accurate than those of individual models alone [46].

Chapter 4
Result Analysis

4.1 Overview

For predicting crop yield in Bangladesh, this study focused on the problem based on different machine learning regression models and by seeing the best result among these repressors next step was chosen. While best regression model was chose based on the , , MSE and RMSE. measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating better model fit.

$$R^2 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{(\sum_{i=1}^{n}(x_i-\bar{x})^2)(\sum_{i=1}^{n}(y_i-\bar{y})^2)}} \quad (1)$$

MAE is the average absolute difference between the predicted and true values. It is less sensitive to outliers compared to MSE.

$$MAE = \frac{\sum_{i=1}^{n}|x_i-y_i|}{n} \quad (2)$$

MSE measures the average squared difference between the predicted values and the actual values.

It provides a comprehensive assessment of overall model performance. RMSE is the square root of the MSE and is in the same unit as the dependent variable. It provides a more interpretable measure of the average prediction error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i-y_i)^2}{n}} \quad (3)$$

In the comprehensive evaluation of performance parameters, nine distinct machine learning repressors were utilized, and their effectiveness was assessed using key metrics such as R-squared (R^2), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The summary of each model's performance is presented in Table 1, highlighting their respective R^2, MSE, and RMSE values.

Notably, among the evaluated regression models, K-Nearest Neighbors (KNN), Random Forest, and Gradient Boosting emerged as the top performers, achieving the highest R-squared (R^2) values. Specifically, they obtained scores of 0.99, 0.98, and 0.99, respectively. A high R-squared value is indicative of a substantial proportion of

the variability in the dependent variable being explained by the independent variable(s) in the model.

Values close to 1 suggest a strong fit, implying that the chosen independent variable(s) contribute significantly to the variation in the dependent variable. In practical terms, a high R-squared signifies that the model is effective in predicting or explaining the observed outcomes.

These results underscore the effectiveness of KNN, Random Forest, and Gradient Boosting in capturing the underlying patterns and relationships within the data. The high R-squared values affirm their ability to provide accurate and reliable predictions, reinforcing their suitability for the task of predicting or explaining crop yield outcomes. Furthermore, these three repressors—KNN, Random Forest, and Gradient Boosting—also exhibit the lowest Mean Absolute Error (MAE).A low MAE indicates that the model's predictions are, on average, very close to the true values, reflecting high accuracy in the predictions. The average absolute difference between predicted and actual values is small, suggesting that, on average, the model's predictions closely align with the true values.

In summary, the evaluation of these performance metrics underscores the effectiveness of KNN, Random Forest, and Gradient Boosting as regression models. Their high R-squared values and low MAE indicate that these models provide strong fits to the data, with accurate predictions that closely align with the observed outcomes. Both MSE and RMSE are measures of how well a regression model is able to approximate the true underlying relationship between input features and the target variable.

Lower values of MSE and RMSE indicate better model performance. When the values are close to zero, it means the model's predictions are, on average, very close to the true values. MSE and RMSE can be used to compare different models or variations of a model.

A model with a lower MSE or RMSE is generally considered better in terms of prediction accuracy. Both metrics are sensitive to outliers, as the squared differences magnify the impact of larger errors. The RMSE is often preferred over MSE because it provides an interpretable measure in the same units as the target variable.

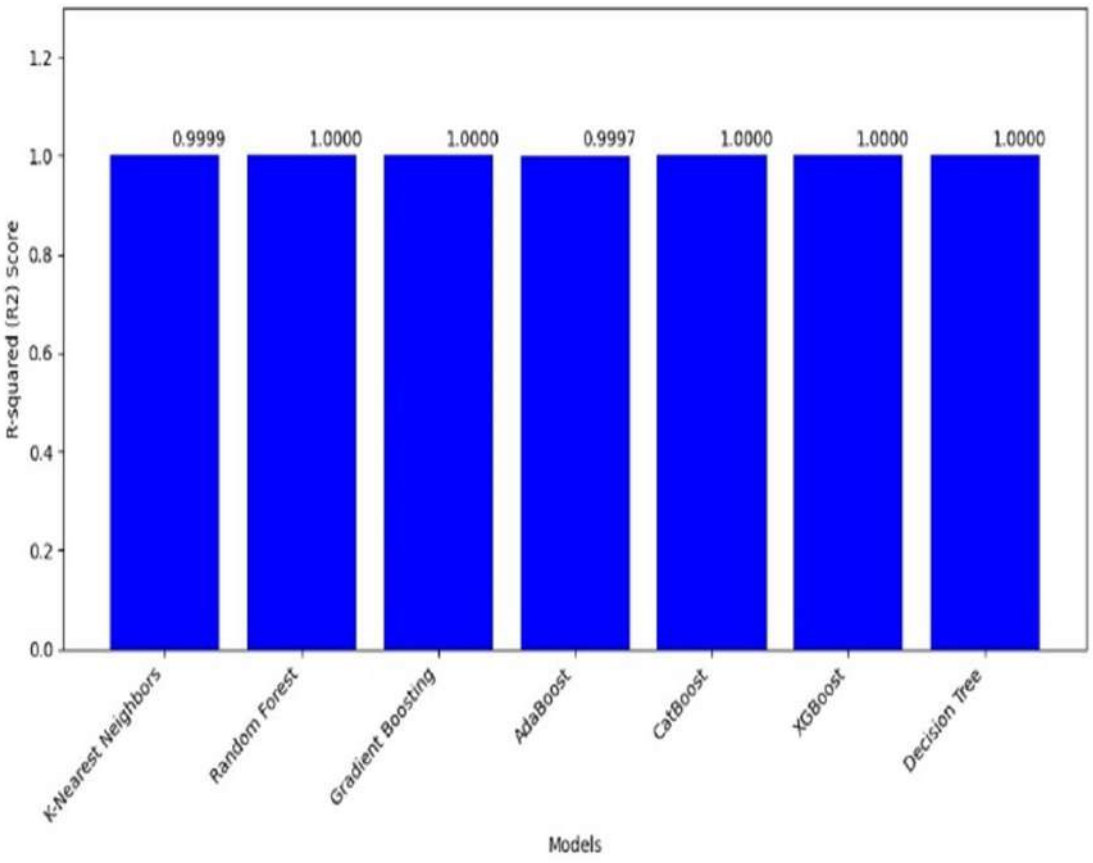
4.2 MAIZE RESULT ANALYSIS

Here the performance evaluation will be carried on using different machine learning. Results are given below :

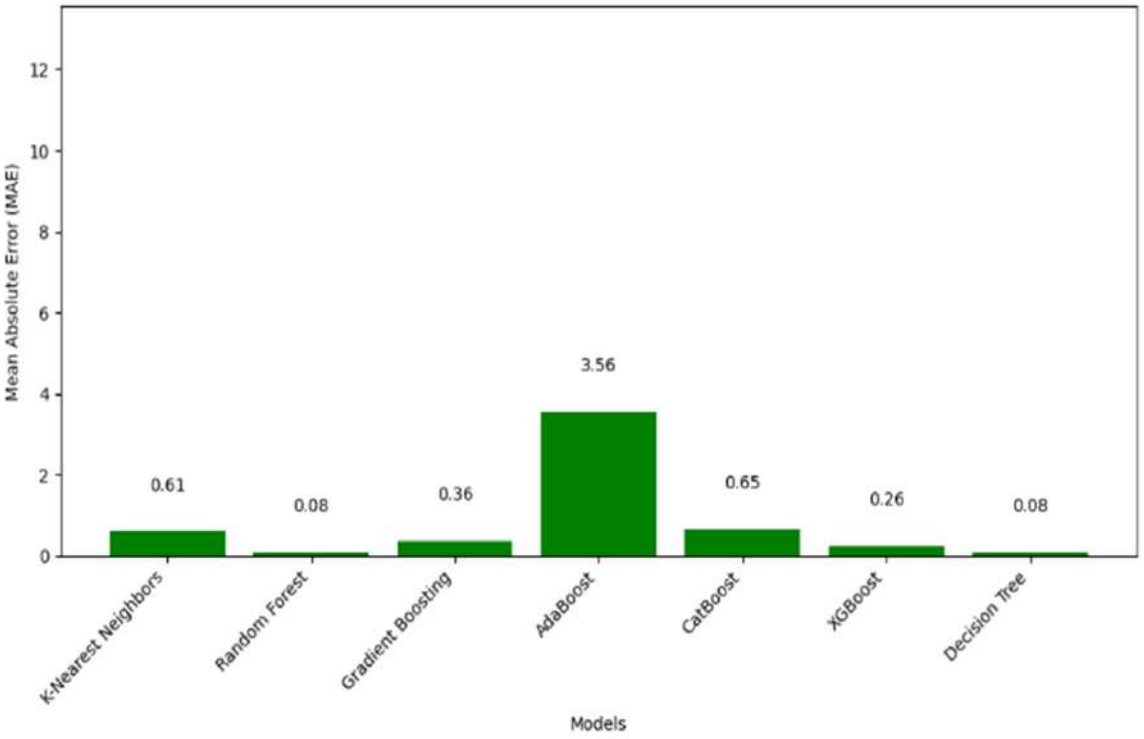
TABLE I. PERFORMANCE EVALUATION USING DIFFERENT MACHINE LEARNING REPRESSORS FOR MAIZE CROP PREDICTION.

ML Repressor	R^2	MAE	MSE	$RMSE$
Decision Tree	0.99	0.42	0.45	0.21
KNN	0.99	0.11	0.67	0.82
Random Forest	0.99	0.04	0.03	0.15
Gradient Boosting	0.99	0.05	0.01	0.13
AdaBoost	0.99	0.96	1.53	1.23
CatBoost	0.99	0.32	0.19	0.43
XgBoost	0.99	0.11	0.04	0.20

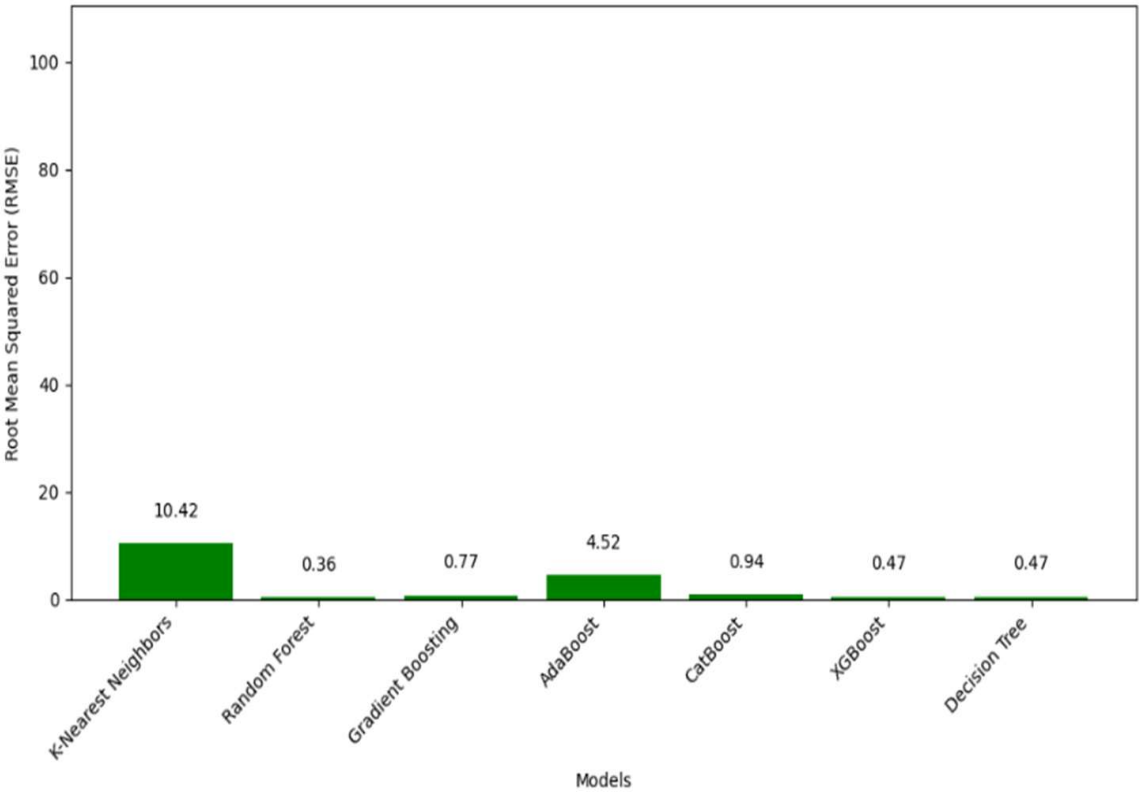
In Fig. 11 the bar chart is showing comparison between all these 7 repressors. It is giving visual representation of how well the model has performed on the dataset.



(a)



(b)



(c)

Figure 11 Bar chart (a),(b),(c) showing performance of different models on the dataset

It works by aggregating the predictions of its constituent repressors, which can be of different types or even the same type with different hyper parameters. The final

prediction is usually the average (or weighted average) of the individual predictions. Common repressors used in the ensemble might include linear regression, decision trees, support vector machines, or any other regression model supported by scikit-learn. By combining multiple models, the ensemble approach can help mitigate overfitting, especially if individual models over fit the training data.

Evaluate the performance of the Voting Repressor using standard regression metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), or R-squared (R^2). Lower values of MSE, RMSE, and MAE indicate better performance, while higher R^2 values suggest a better fit to the data. In Fig 7. The performance of voting regression is showed by visual representation using bar chart where R-squared (R^2) is achieved as 1,00, Mean Absolute Error (MAE) as 0.06 and Root Mean Squared Error (RMSE) as 0.14.

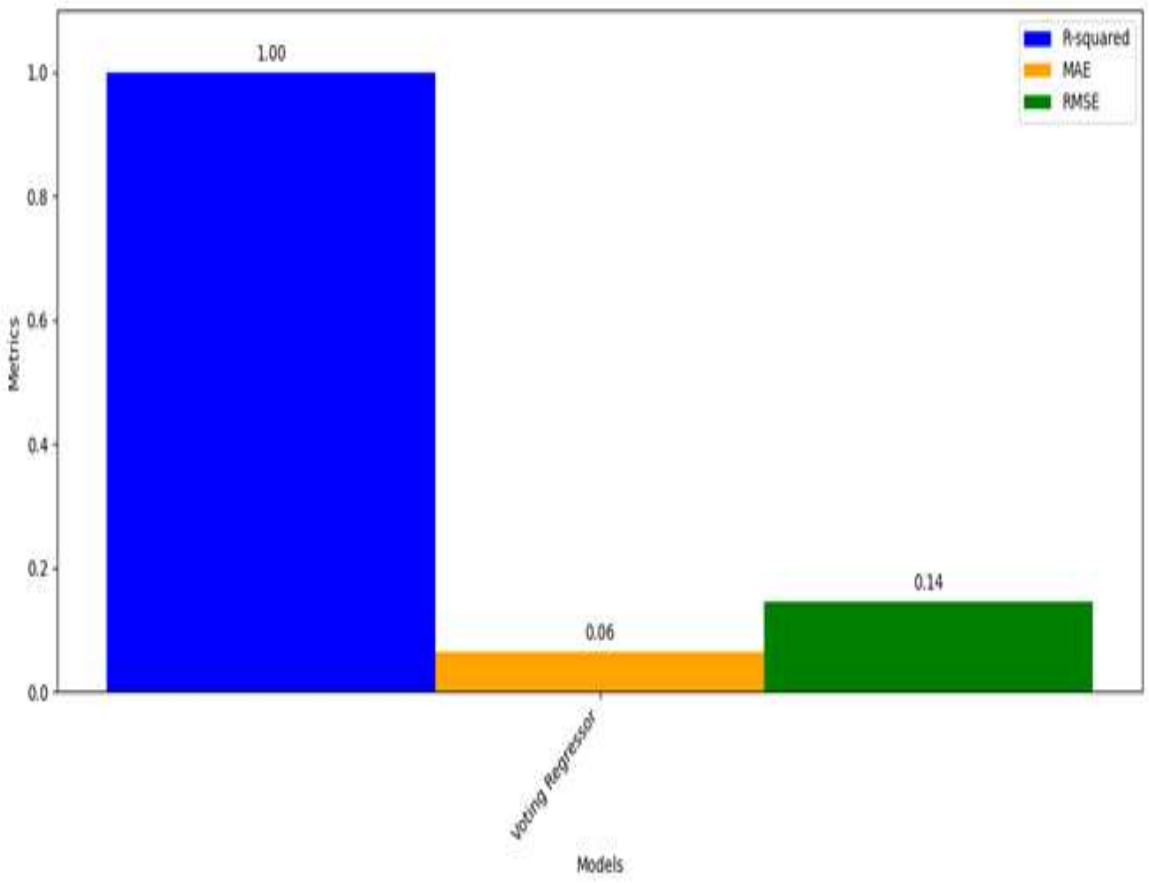


Figure 12 Bar chart showing performance of voting regression

Stacking, also known as stacked generalization, stands as an ensemble learning technique that combines predictions from multiple base models to construct a more robust and accurate model. In the stacking process, a meta-model is trained to make predictions based on the predictions generated by several base models. This

methodology capitalizes on the strengths of different models, often surpassing the performance of individual models.

In Table 2, the stacking model utilizing RF as the final estimator exhibited exceptional performance among all combinations.

This is evident in its high R-squared (R^2) value of 0.99 signifying the model's ability to effectively explain the variance in the data. The MAE of 0.04 further substantiates the accuracy of the predictions, indicating how well the model captured the average size of mistakes. Moreover, the efficacy of the Stacking with SVR model in reducing both systematic and non-systematic errors is demonstrated by the MSE of 0.02 and RMSE of 0.15.

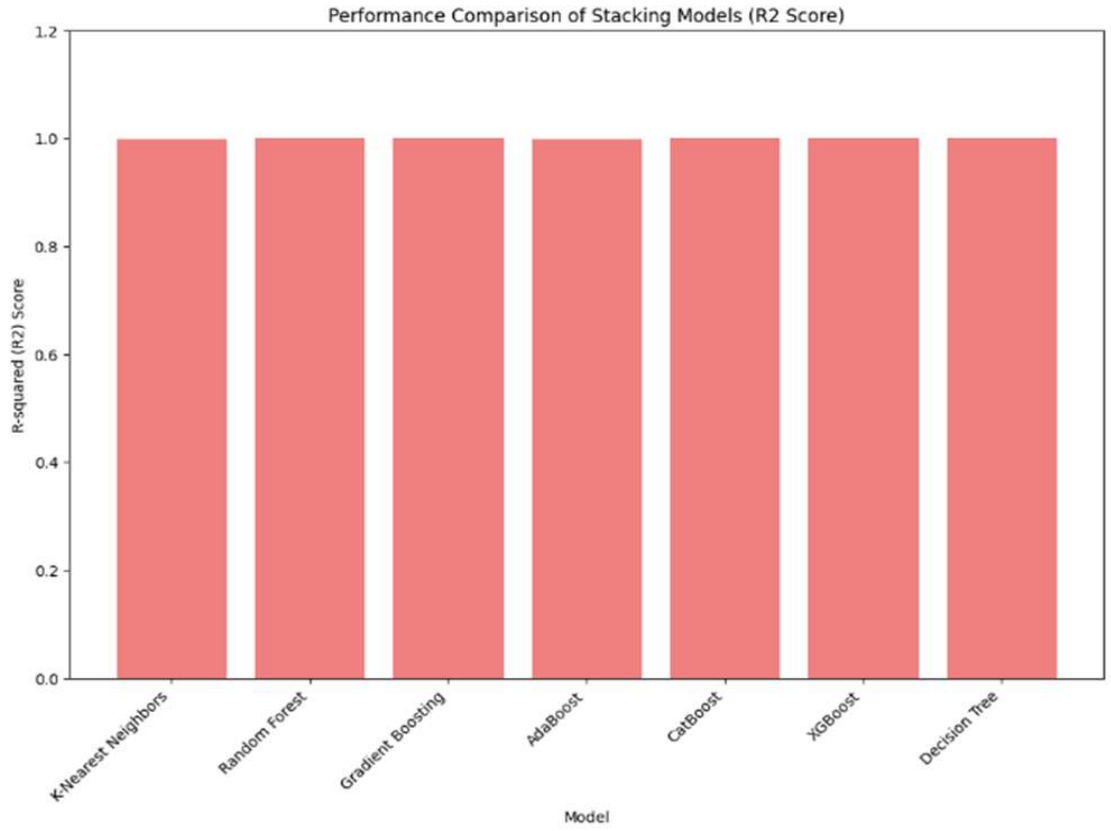
The stacking model, particularly when employing SVR as the final estimator, showcases noteworthy performance metrics.

The high R-squared value and low error measures such as MAE, MSE, and RMSE collectively highlight the accuracy and effectiveness of the stacking ensemble in predicting crop yield, underlining its potential as a robust approach for yield forecasting.

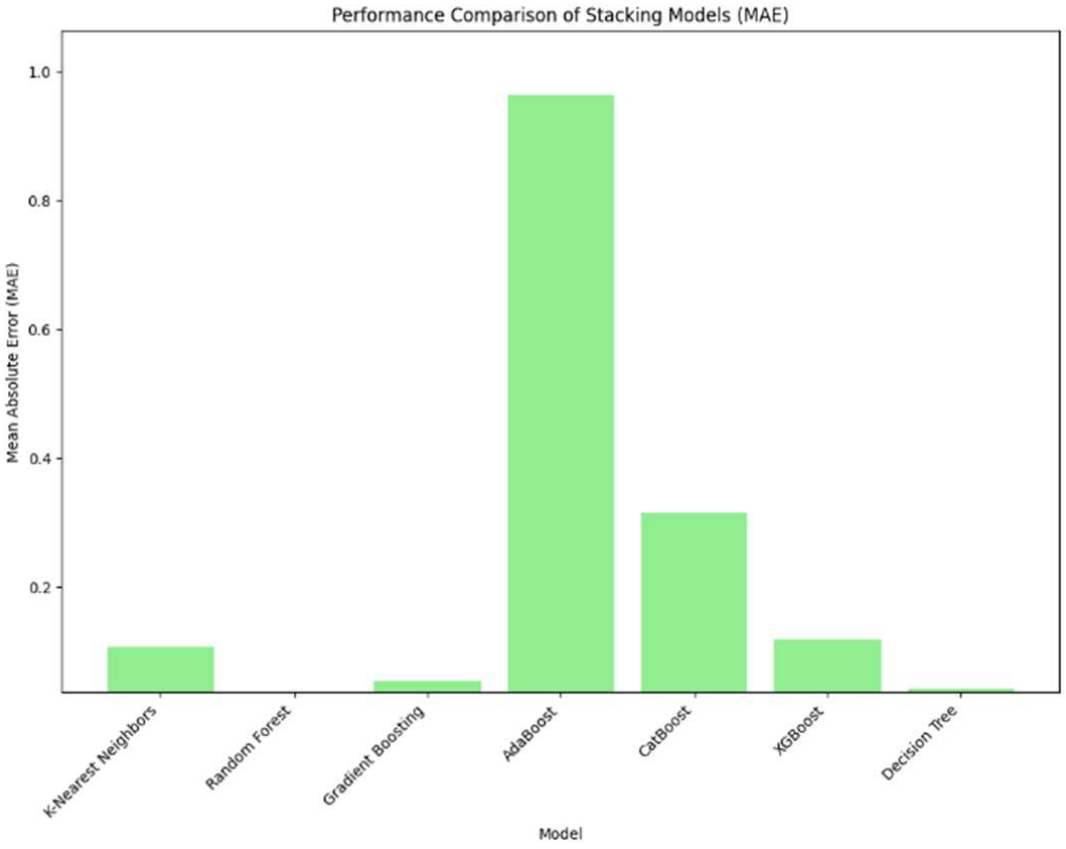
TABLE II. EVALUATING STACKING PERFORMANCE BY CHANGING FINAL ESTIMATORS FOR MAIZE PREDICTION

ML Repressor	R^2	MAE	MSE	$RMSE$
Decision Tree.	0.99	0.05	0.03	017
KNN.	0.99	0.04	0.02	0.13
Random Forest	0.99	0.04	0.02	0.15
Gradient Boosting	0.99	0.06	0.02	0.13
AdaBoost.	0.99	0.15	0.73	0.85
CatBoost.	0.99	0.11	0.05	0.23
XgBoost.	0.99	0.05	0.03	0.19

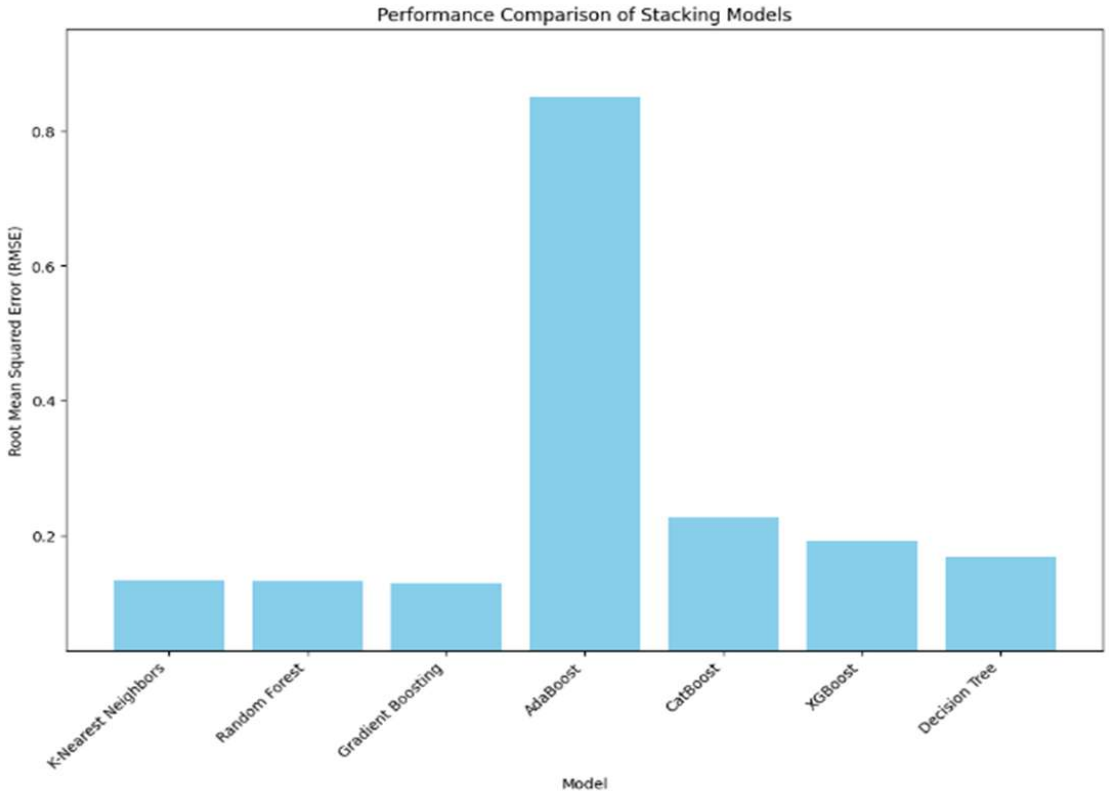
In Fig 13. the bar chart is showing comparison between all stacking performances. It is giving visual representation of how well stacking has performed on the dataset.


Performance Comparison of Stacking Models (R2 Score)
R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(a)


Performance Comparison of Stacking Models (MAE)
Mean Absolute Error (MAE)
1.0
0.8
0.6
0.4
0.2
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(b)



(c)

Figure 13 Bar chart (a),(b),(c) showing performance of stacking by varying final estimators

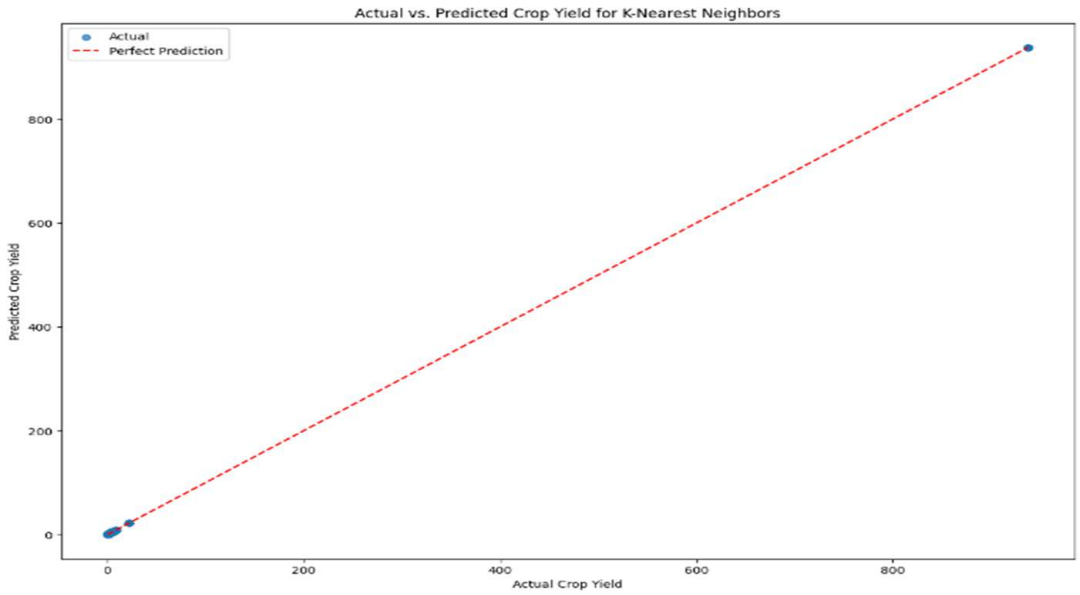
A scatter plot depicting the results of a stacking ensemble model provides a visual representation of the predicted values versus the actual values for a given dataset. In this context, stacking involves combining the predictions of multiple machine learning models to create a meta-model that potentially outperforms its individual components. Each point on the scatter plot represents an observation, with its coordinates corresponding to the actual and predicted values.

In an ideal scenario, where the stacking model performs exceptionally well, the points on the scatter plot would align closely to a diagonal line, indicating a perfect match between the predicted and actual values. However, real-world data often introduces variability, leading to deviations from this ideal alignment. The spread, pattern, and concentration of points on the scatter plot provide valuable insights into the performance of the stacking model.

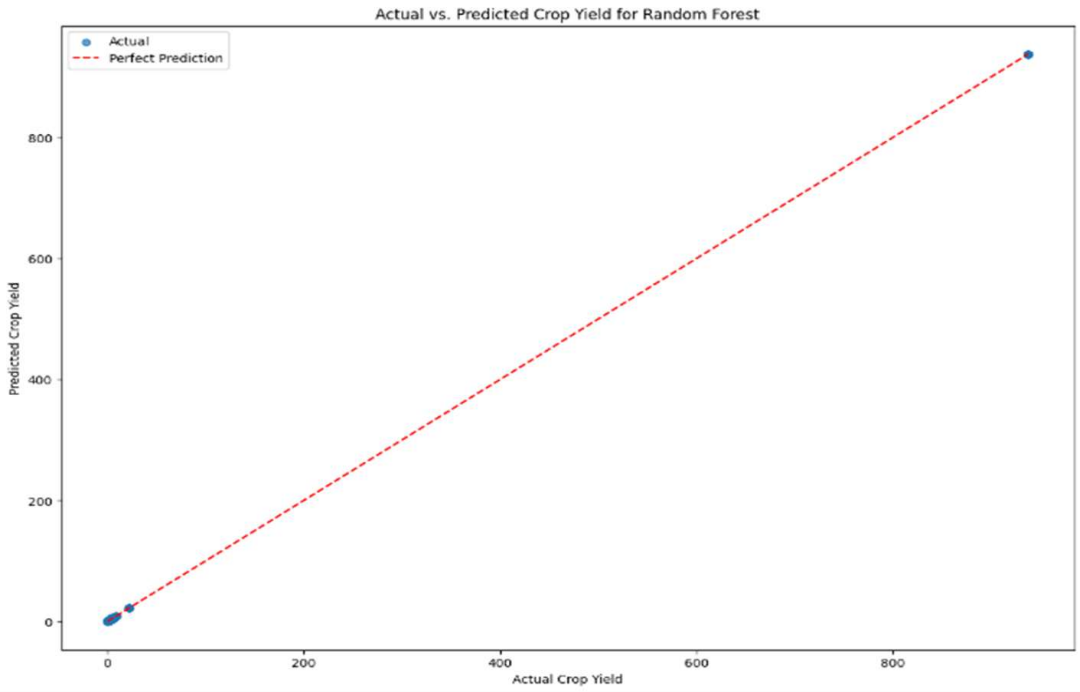
A tightly clustered grouping of points around the diagonal line suggests that the stacking model has accurately predicted the target variable across a range of observations. On the other hand, points scattered further from the diagonal may indicate instances where the model's predictions diverge from the actual values.

Outliers in the scatter plot, points that significantly deviate from the overall pattern, can offer specific insights. These outliers may represent challenging instances for the model, where the stacking approach might struggle to capture the underlying patterns in the data.

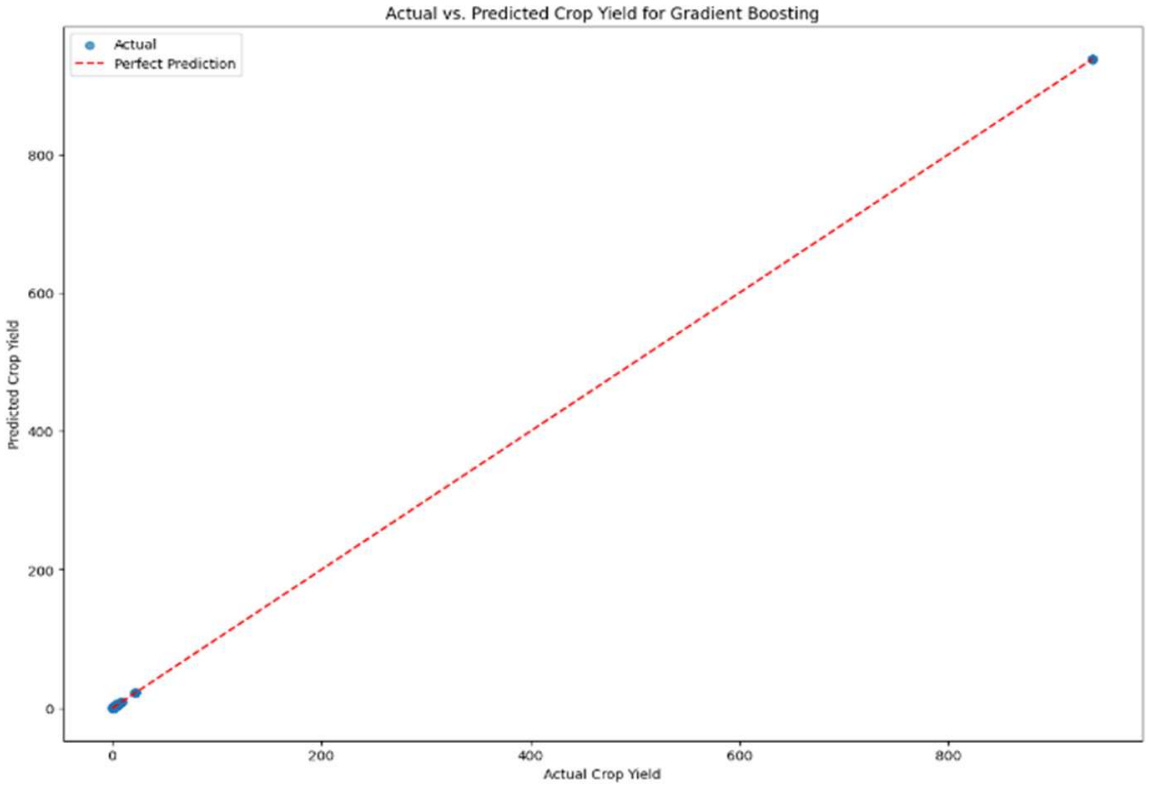
Overall, a scatter plot of stacking results serves as a diagnostic tool for assessing the efficacy of the ensemble model. It aids in understanding the model's strengths, weaknesses, and areas of improvement by visually comparing predicted and actual values, providing a nuanced perspective beyond numerical evaluation metrics.



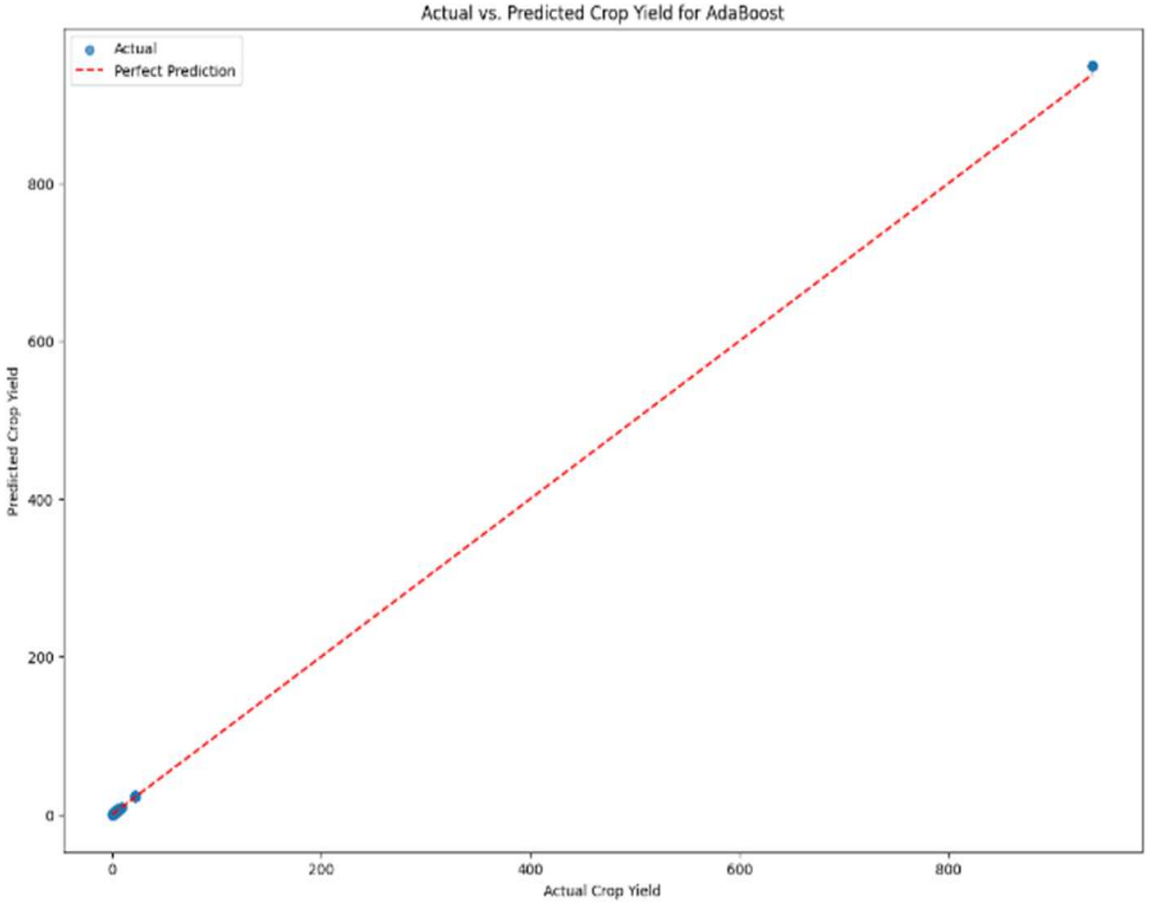
(a)



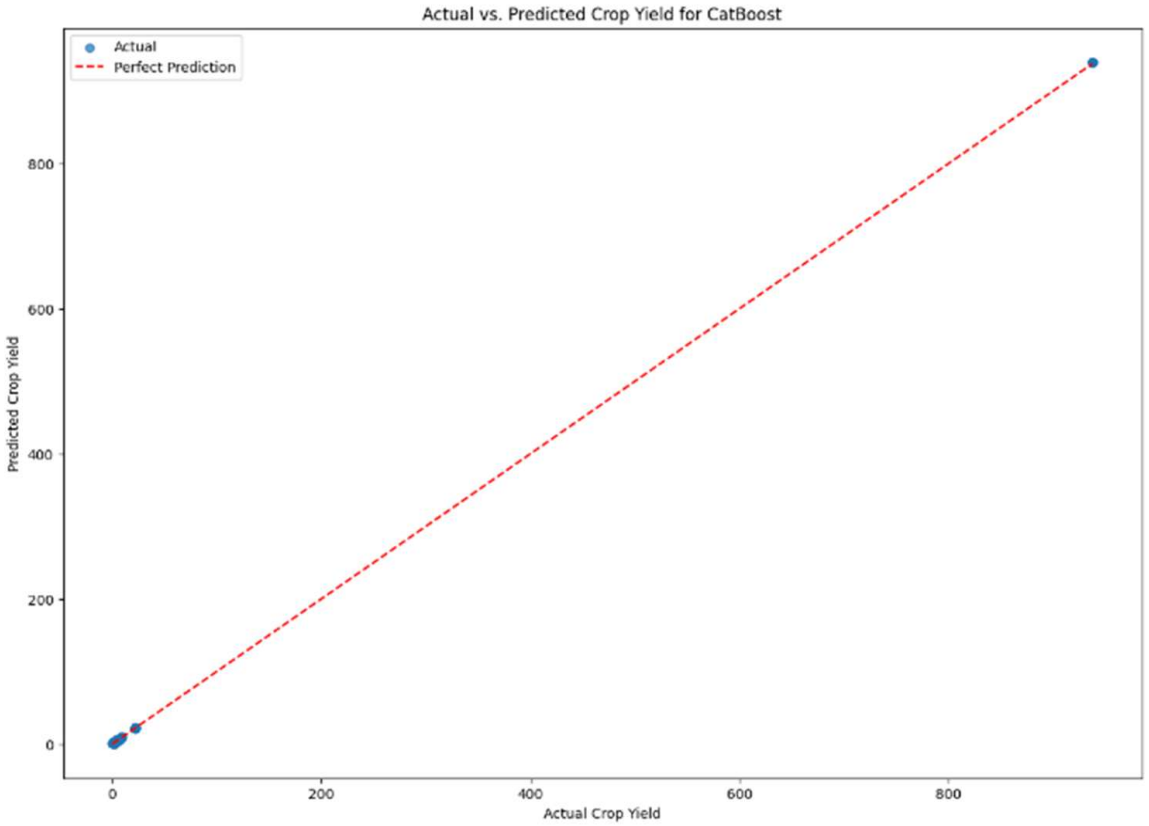
(b)


Actual vs. Predicted Crop Yield for Gradient Boosting
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
200
400
600
800

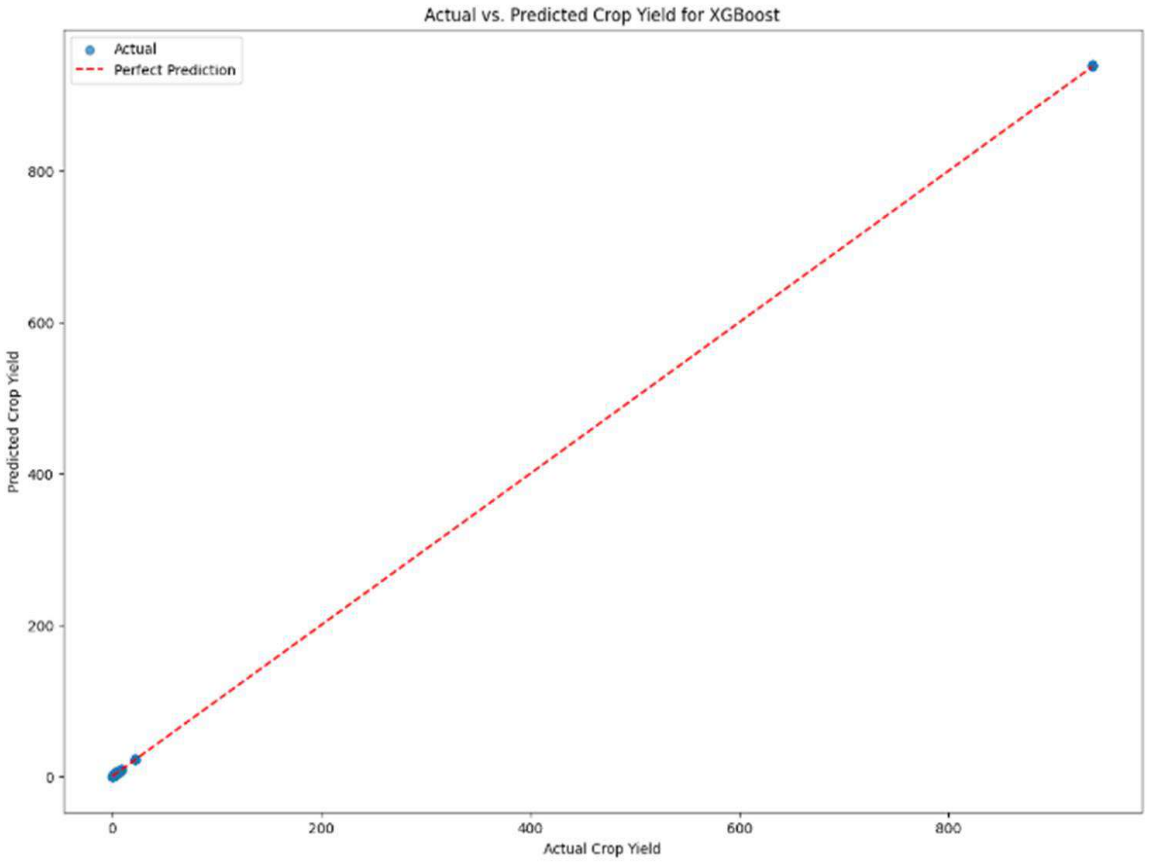
(c)


Actual vs. Predicted Crop Yield for AdaBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
200
400
600
800

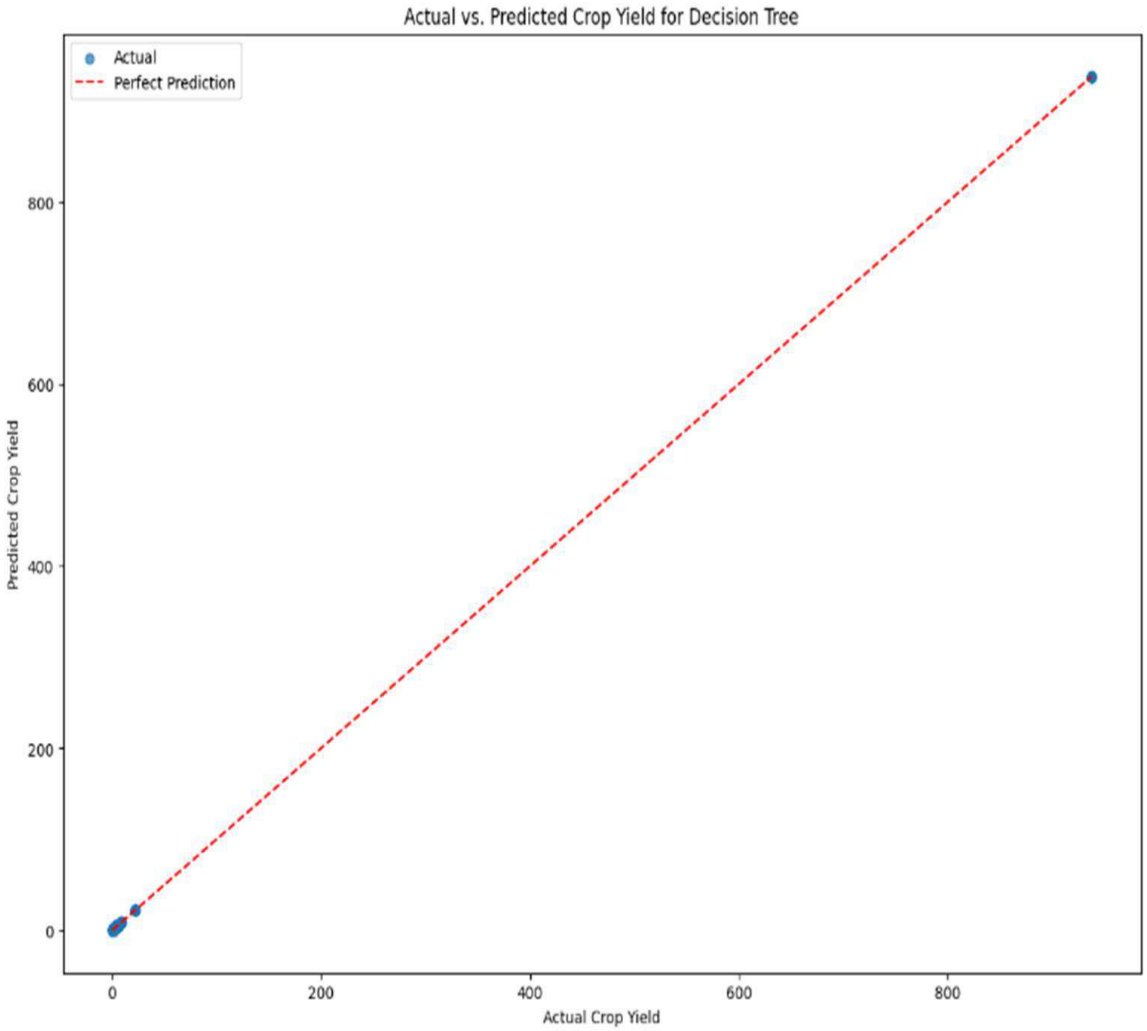
(d)


Actual vs. Predicted Crop Yield for CatBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
200
400
600
800

(e)


Actual vs. Predicted Crop Yield for XGBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
200
400
600
800

(f)



(g)

Figure 14 Scattering plot of different stacking performance(a),(b),(c),(d),(e),(f),(*g*)

In this work, the base three models remain constant but the final estimator varies, the objective is to observe how different final estimator models influence the stacking ensemble. The final estimator, responsible for combining the predictions of the base models, can be diverse, ranging from decision trees to more complex models.

When employing decision trees as the final estimator in stacking, the ensemble benefits from their ability to capture intricate patterns in the data. Decision trees are known for their flexibility and can adapt well to various types of relationships within the dataset. However, this may lead to overfitting on the training data. On the other hand, using a linear model, such as linear regression, as the final estimator can introduce a different perspective. Linear models are less complex and provide a smoother fit to the data. This may mitigate overfitting but could potentially overlook intricate patterns captured by decision trees.

4.3 SUGARCANE RESULT ANALYSIS

Here the performance evaluation will be carried on using different machine learning. Results are given below :

TABLE III. PERFORMANCE EVALUATION USING DIFFERENT MACHINE LEARNING REPRESSORS FOR SUGARCANE CROP PREDICTION.

ML Repressor	R^2	MAE	MSE	$RMSE$
Decision Tree	0.99	0.37	5.32	2.30
KNN	0.98	0.88	15.13	3.89
Random Forest	0.99	0.37	3.83	1.95
Gradient Boosting	0.99	0.42	3.09	1.76
AdaBoost	0.81	11.04	180.17	13.42
CatBoost	0.99	1.12	4.22	2.05
XgBoost	0.99	0.53	3.62	1.90

In table 3 The Decision Tree, Random Forest, Gradient Boosting, CatBoost, and XgBoost models exhibit high R-squared values, indicating strong predictive capabilities.

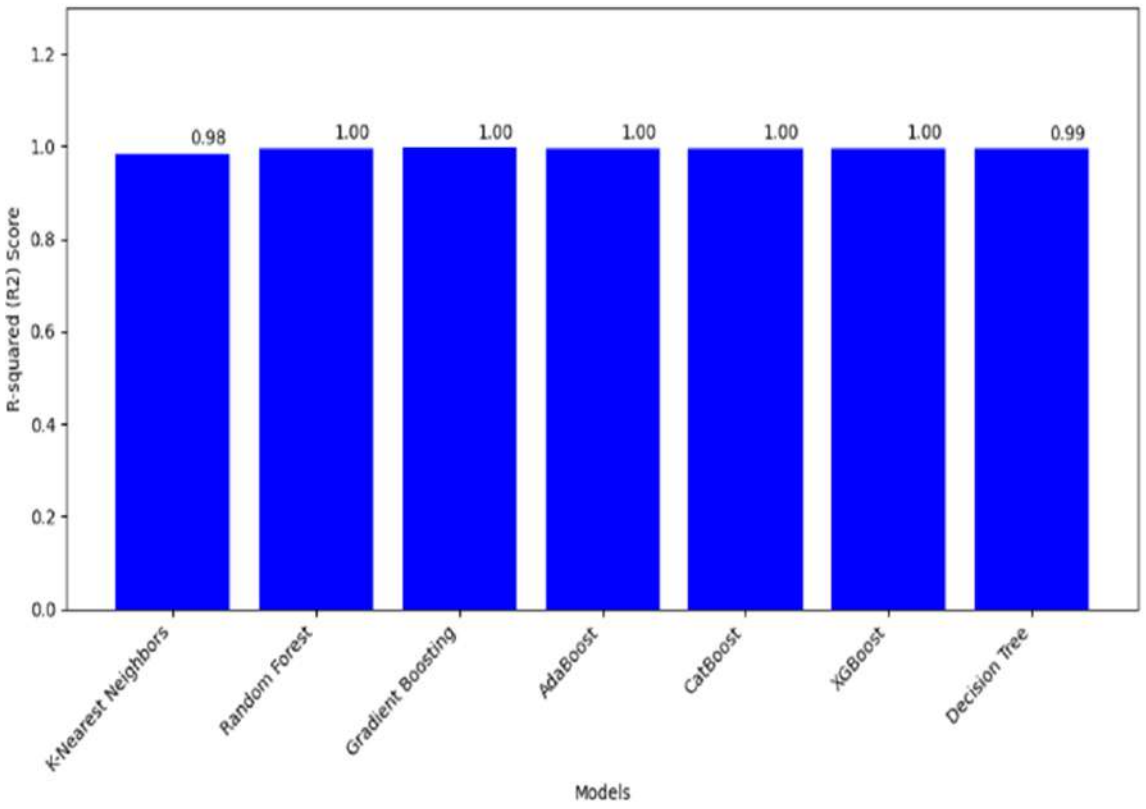
Among these, Random Forest and XgBoost stand out with the lowest RMSE values, suggesting superior accuracy in predicting target values.

KNN performs well but with slightly higher errors. AdaBoost, while achieving a reasonable R-squared value, exhibits significantly higher errors, potentially indicating challenges in capturing the underlying patterns of the data.

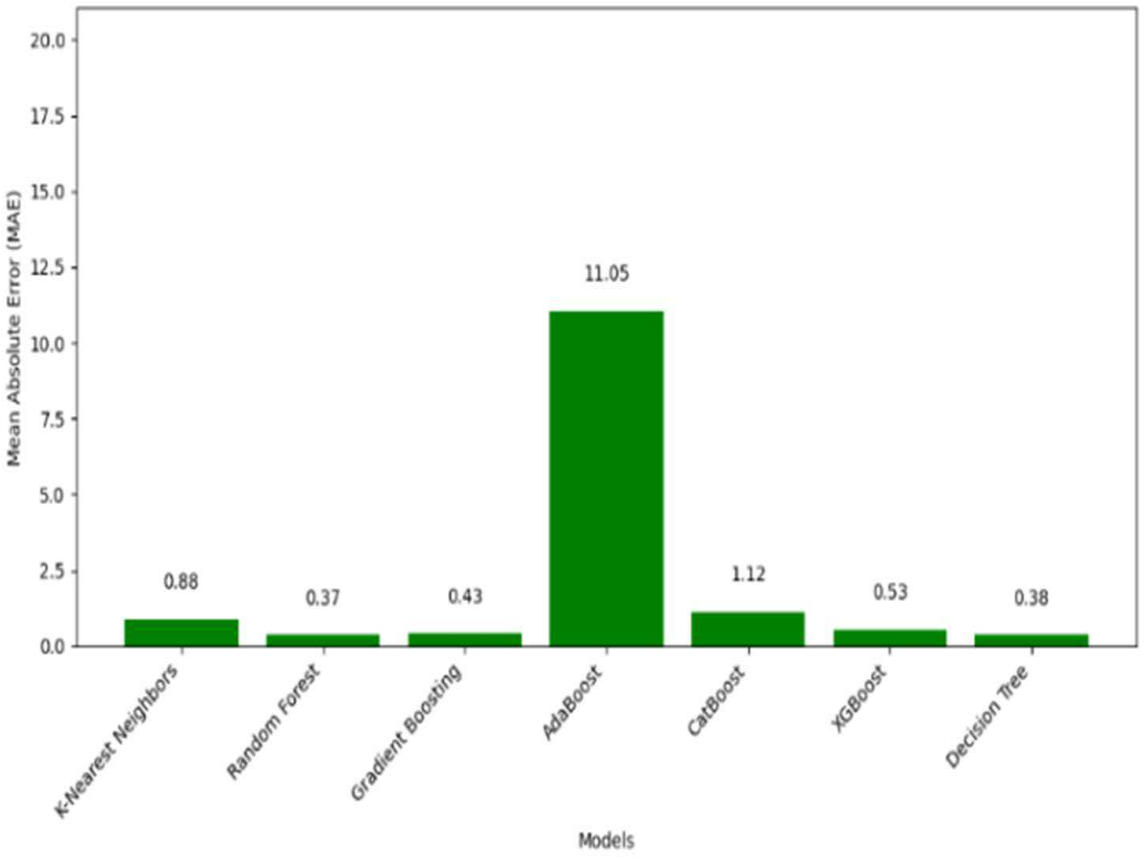
In summary, Random Forest and XgBoost demonstrate excellent overall performance, providing accurate and reliable regression predictions.

In Fig. 10 the bar chart is showing comparison between all these 7 Regression s to predict crop yield of sugarcane.

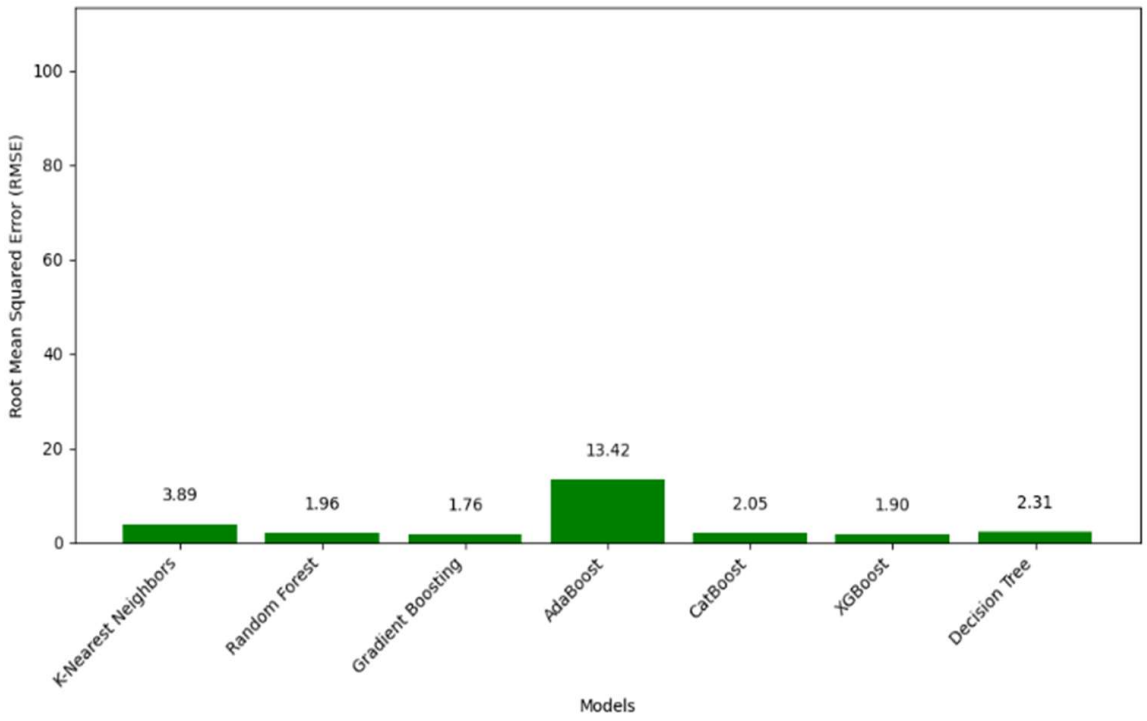
It is giving visual representation of how well the model has performed on the dataset. of sugarcane.


R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.98
1.00
1.00
1.00
1.00
1.00
0.99
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(a)


Mean Absolute Error (MAE)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.88
0.37
0.43
11.05
1.12
0.53
0.38
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(b)



(c)

Figure 15 Bar chart (a),(b),(c) showing performance of different models on the dataset.

In Fig 15. The performance of voting Regression is showed by visual representation using bar chart where R-squared (R^2) is achieved as 1.00, Mean Absolute Error (MAE) as 0.63 and Root Mean Squared Error (RMSE) as 1.66.

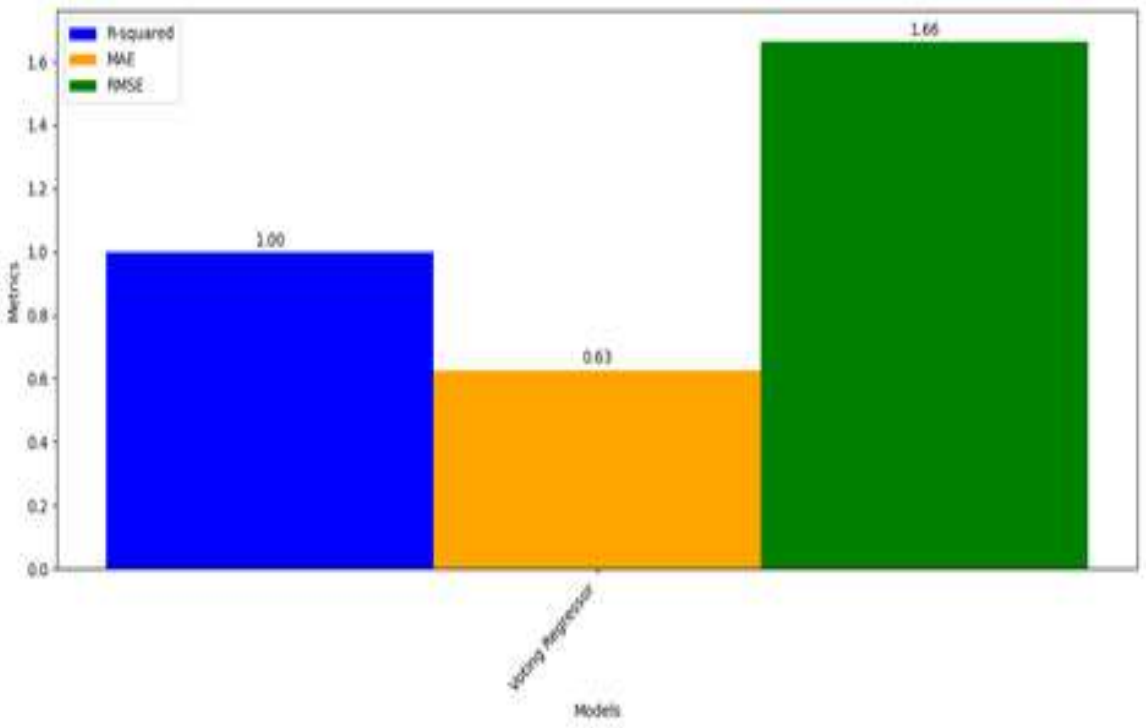


Figure 16 Bar chart showing performance of voting regression

In Table IV, the stacking model utilizing KNN as the final estimator exhibited exceptional performance among all combinations.

This is evident in its high R-squared (R^2) value of 0.99, signifying the model's ability to effectively explain the variance in the data. The low Mean Absolute Error (MAE) of 0.36 further substantiates the accuracy of the predictions, indicating how well the model captured the average size of mistakes.

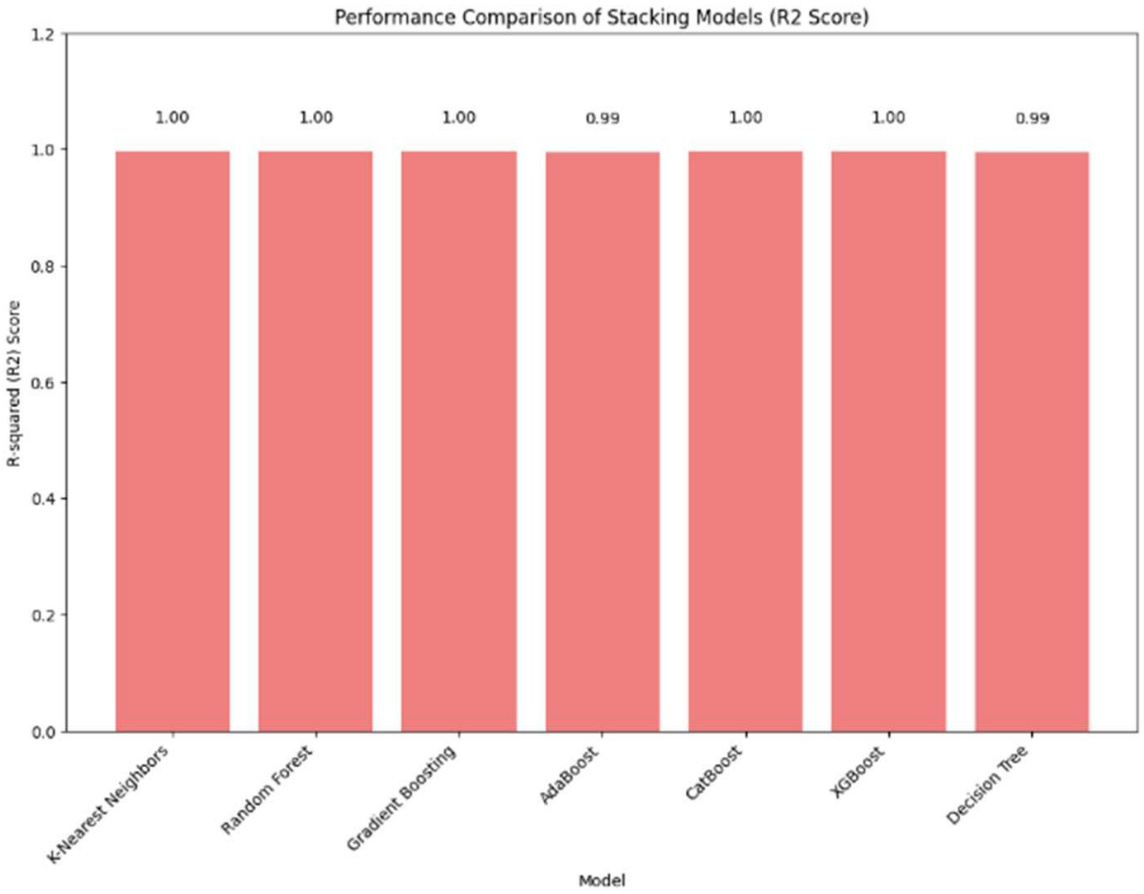
Moreover, the efficacy of the Stacking with SVR model in reducing both systematic and non-systematic errors is demonstrated by the Mean Squared Error (MSE) of 2.62 and the Root Mean Squared Error (RMSE) of 1.62.

The stacking model, particularly when employing KNN as the final estimator, showcases noteworthy performance metrics. The high R-squared value and low error measures such as MAE, MSE, and RMSE collectively highlight the accuracy and effectiveness of the stacking ensemble in predicting crop yield of sugarcane, underlining its potential as a robust approach for yield forecasting.

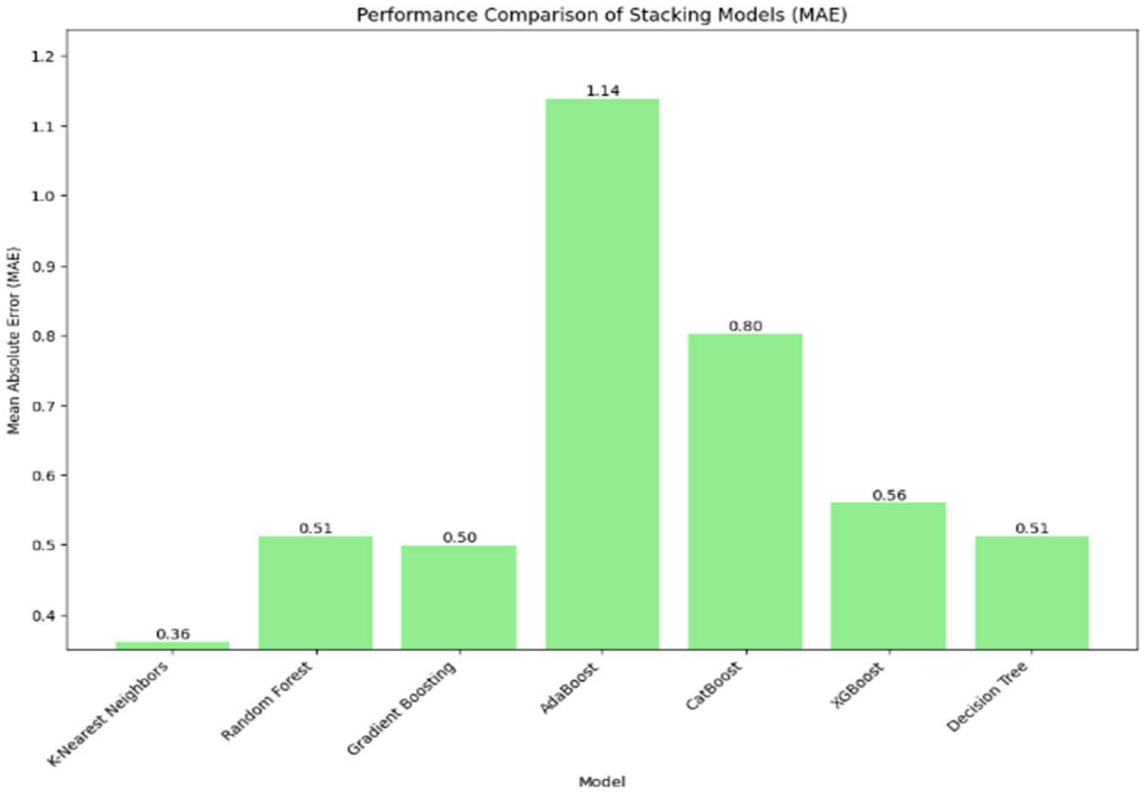
TABLE IV. PERFORMANCE EVALUATION USING DIFFERENT MACHINE LEARNING REPRESSORS FOR SUGARCANE CROP PREDICTION.

ML Regression	R^2	MAE	MSE	$RMSE$
Decision Tree.	0.99	0.51	5.01	2.23
KNN.	0.99	0.36	2.62	1.62
Random Forest	0.99	0.51	3.14	1.77
Gradient, Boosting	0.99	0.49	3.36	1.83
AdaBoost.	0.99	1.13	5.10	2.26
CatBoost.	0.99	0.80	3.02	1.73
XgBoost.	0.99	0.56	3.80	1.95

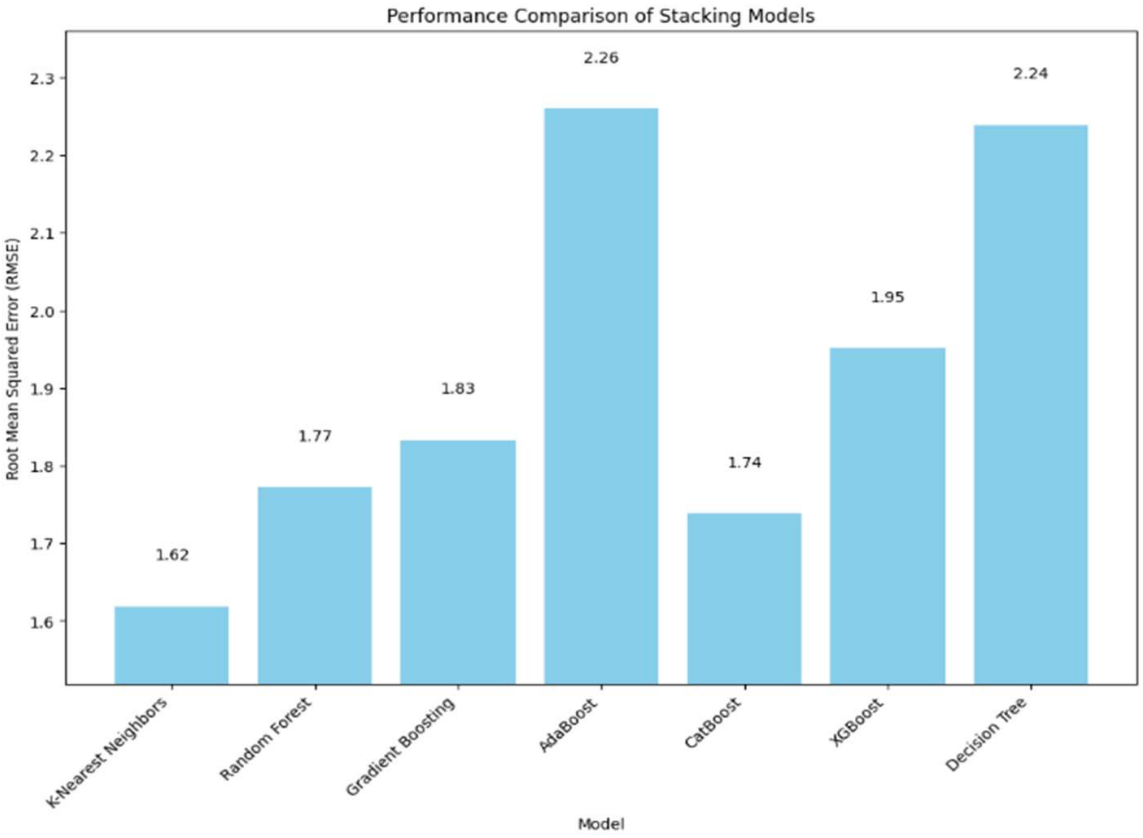
In Fig 16. the bar chart is showing comparison between all stacking performances. It is giving visual representation of how well stacking has performed on the dataset. Fig. 13 is indicating the scattering plot of the all stacking models.


Performance Comparison of Stacking Models (R2 Score)
R-squared (R2) Score
1.00
1.00
1.00
0.99
1.00
1.00
0.99
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(a)

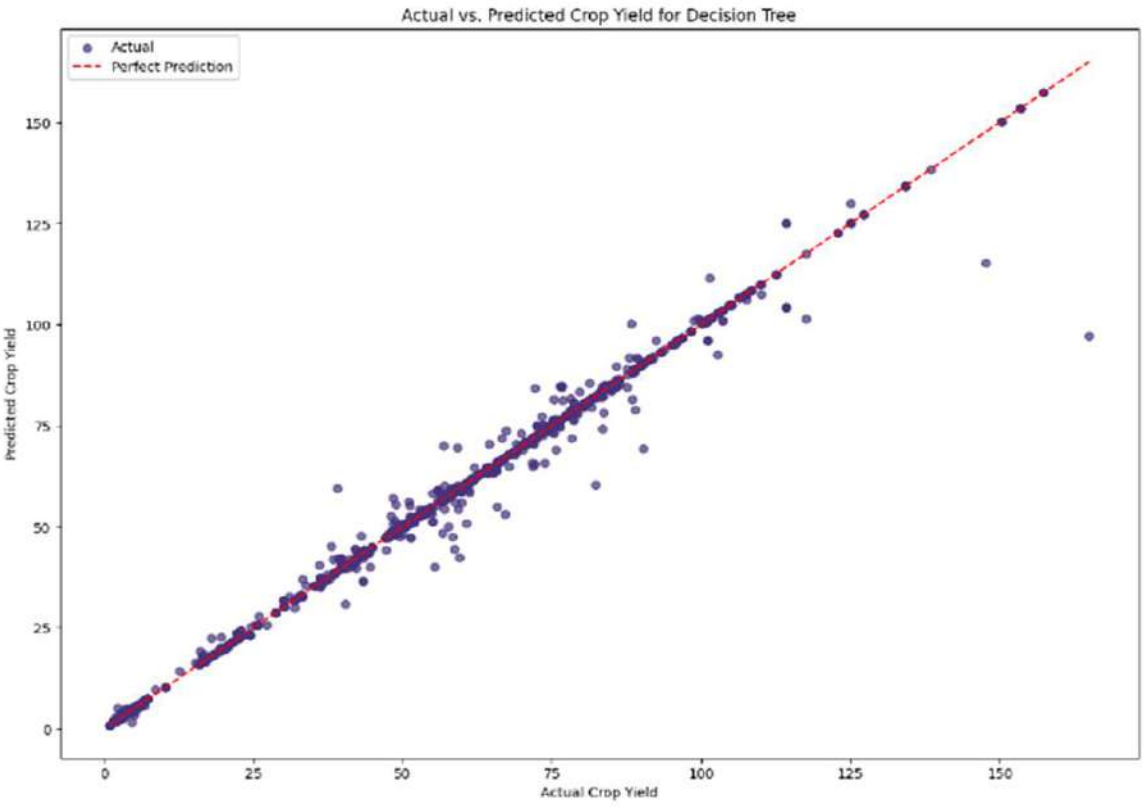

Performance Comparison of Stacking Models (MAE)
Mean Absolute Error (MAE)
0.36
0.51
0.50
1.14
0.80
0.56
0.51
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(b)

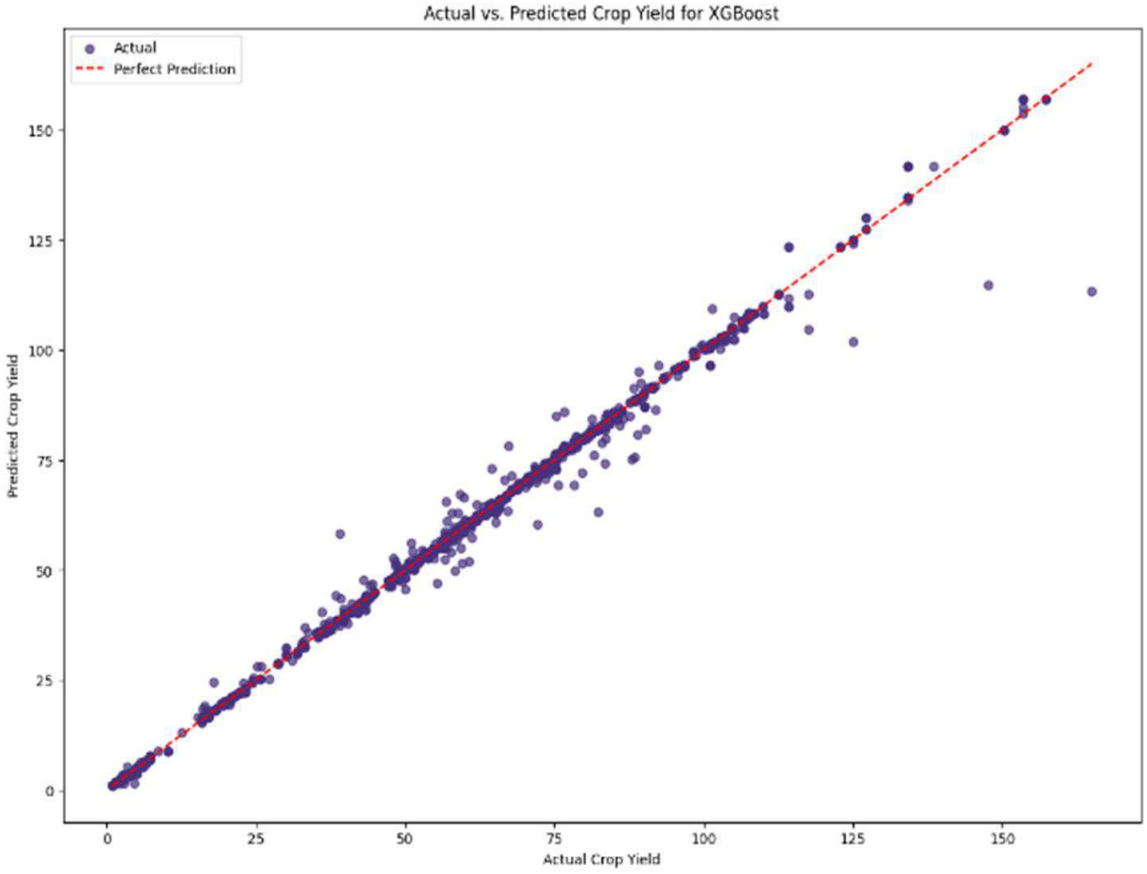


(c)

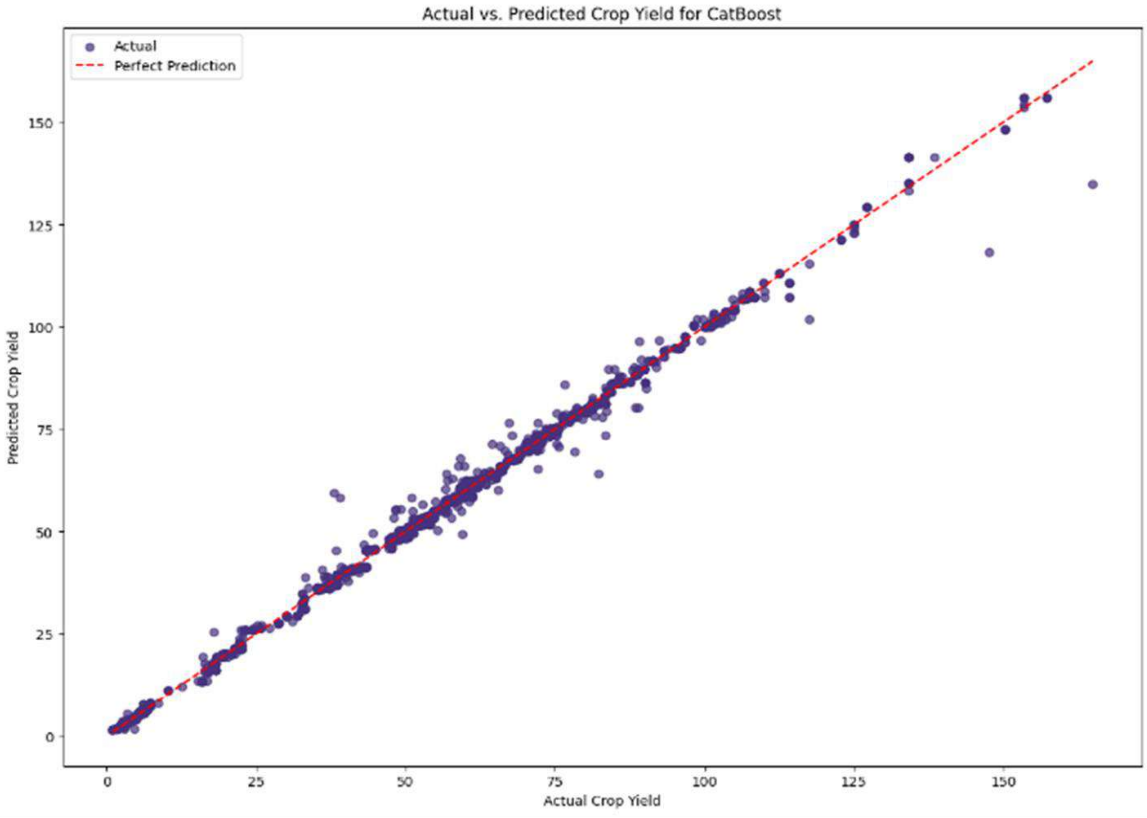
Figure 17 Bar chart (a),(b),(c) showing performance of stacking by varying final estimators.



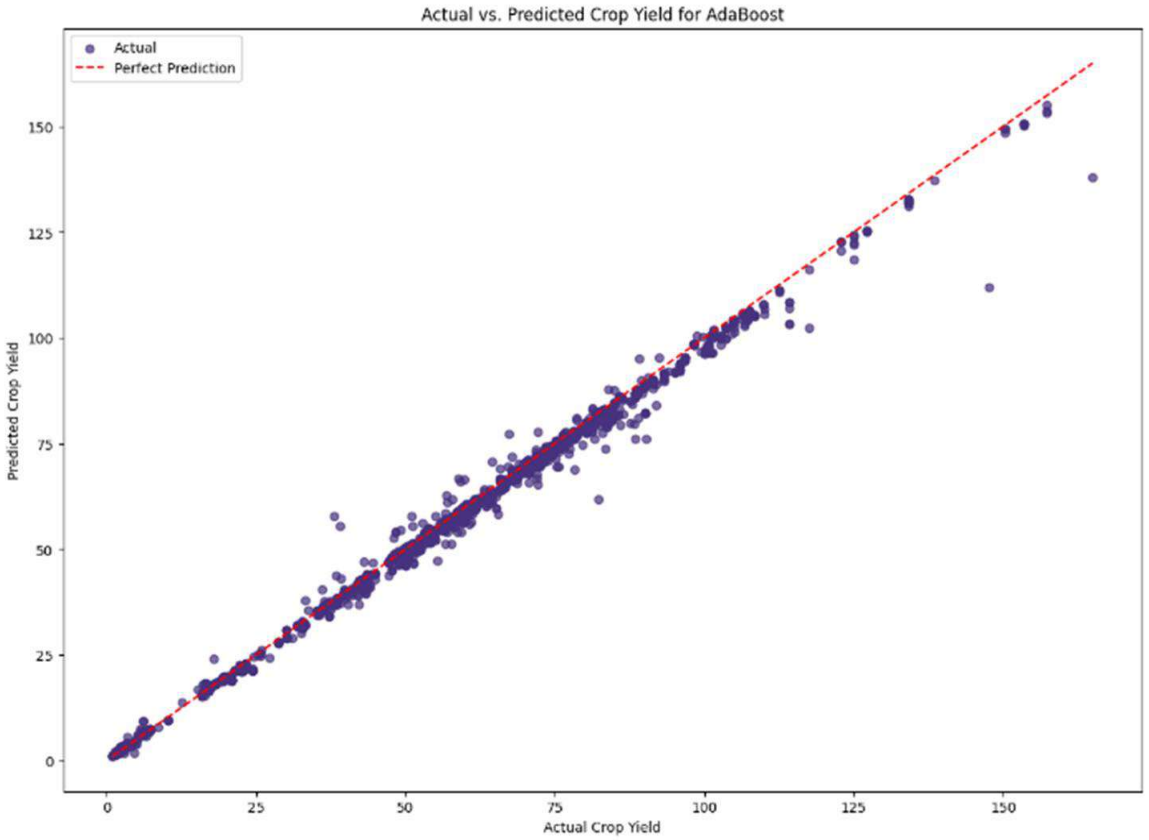
(a)


Actual vs. Predicted Crop Yield for XGBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
25
50
75
100
125
150

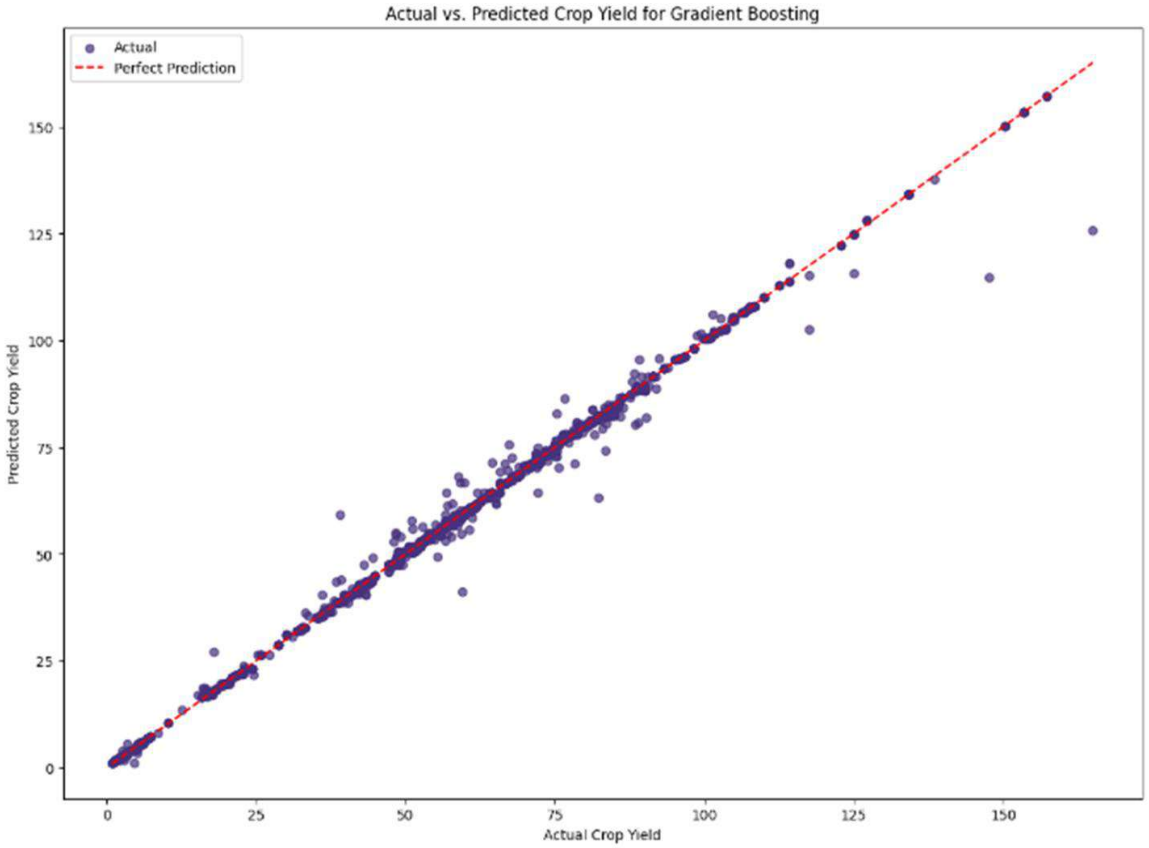
(b)


Actual vs. Predicted Crop Yield for CatBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
25
50
75
100
125
150

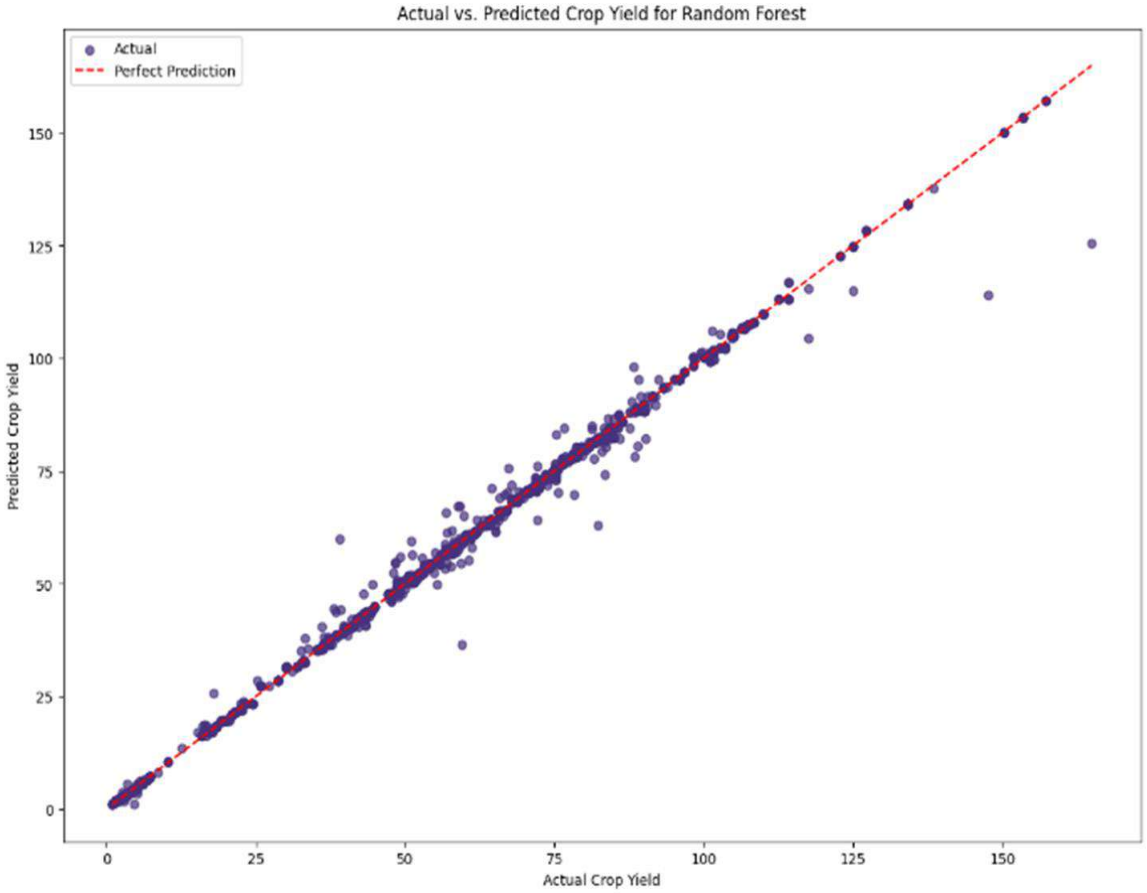
(c)


Actual vs. Predicted Crop Yield for AdaBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
25
50
75
100
125
150

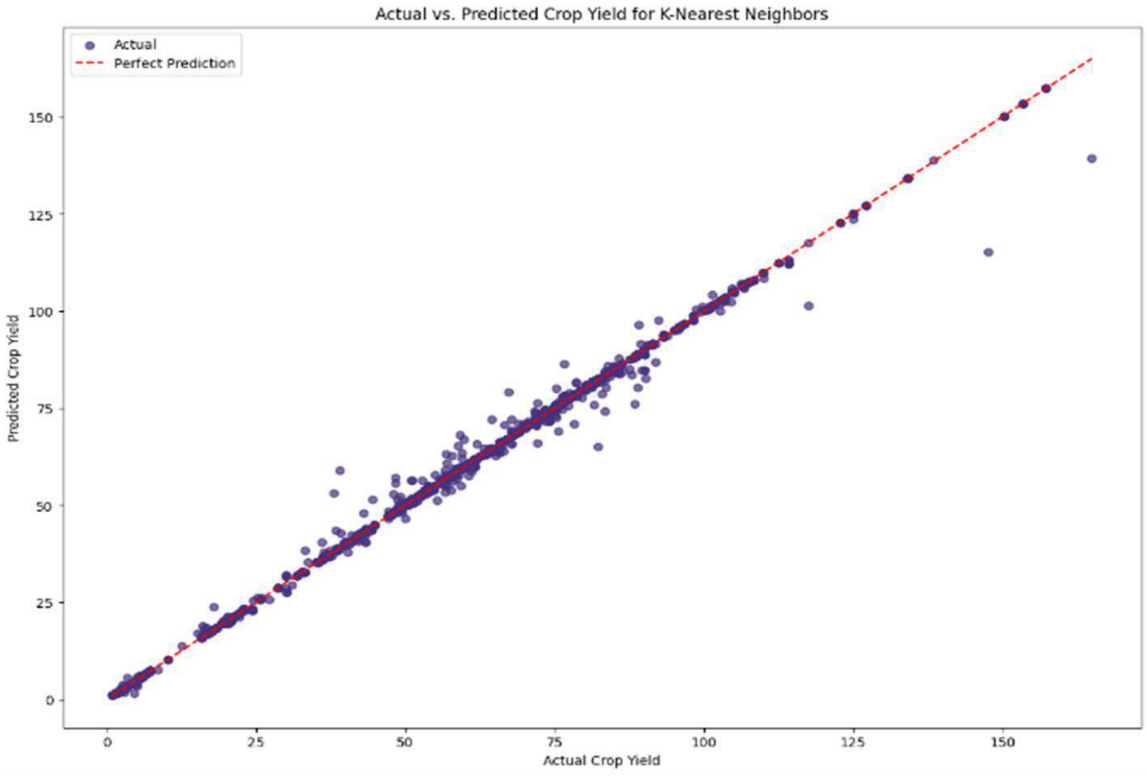
(d)


Actual vs. Predicted Crop Yield for Gradient Boosting
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
25
50
75
100
125
150

(e)



(f)



(g)

Figure 18 Scattering plot of different stacking performance (a),(b),(c),(d),(e),(f),(g).

4.4 POTATO RESULT ANALYSIS

Here the performance evaluation will be carried on using different machine learning. Results are given below :

TABLE V. PERFORMANCE EVALUATION USING DIFFERENT MACHINE LEARNING REPRESSORS FOR POTATO CROP PREDICTION.

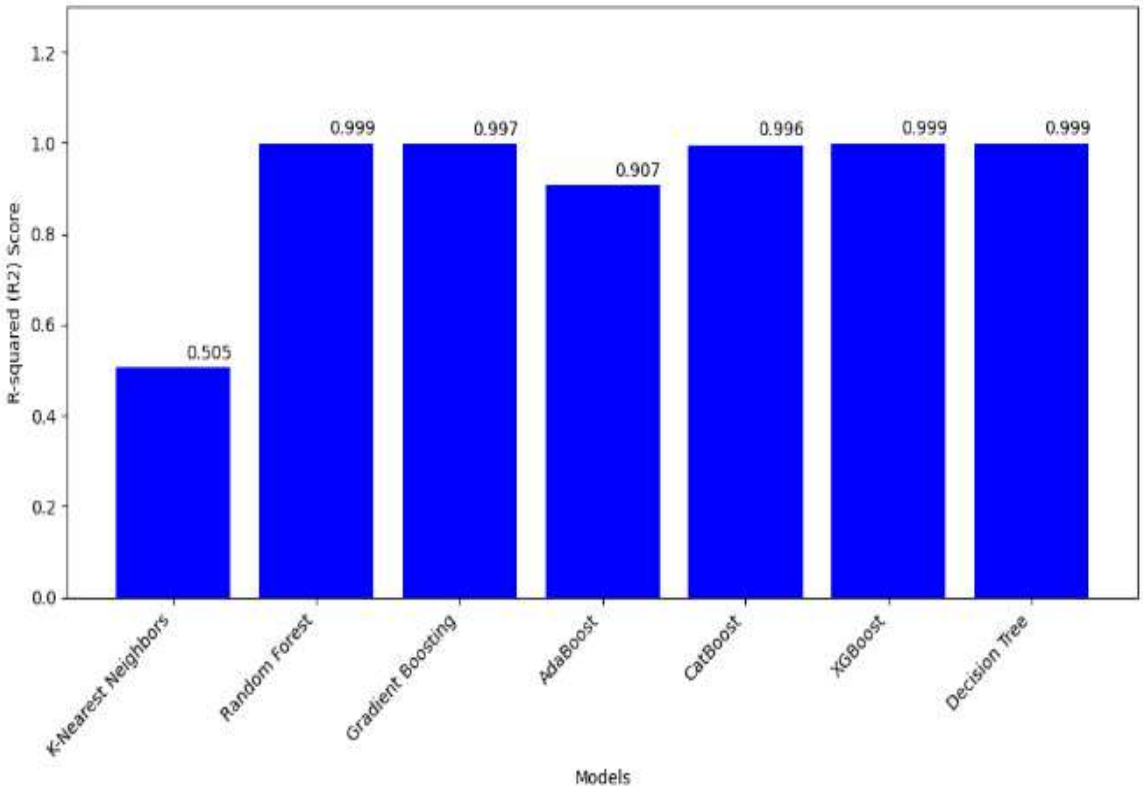
ML Regression	R^2	MAE	MSE	$RMSE$
Decision Tree	0.99	0.08	0.22	0.47
KNN	0.50	0.60	108.64	10.42
Random Forest	0.99	0.08	0.12	0.36
Gradient Boosting	0.99	0.36	0.59	0.77
AdaBoost	0.90	3.55	20.40	4.51
CatBoost	0.99	0.65	0.88	0.94
XgBoost	0.99	0.26	0.22	0.47

Table 5 presents the performance evaluation of various machine learning regression models for maize crop prediction. The Decision Tree, Random Forest, Gradient Boosting, CatBoost, and XgBoost models demonstrate high R-squared values, indicating robust predictive capabilities.

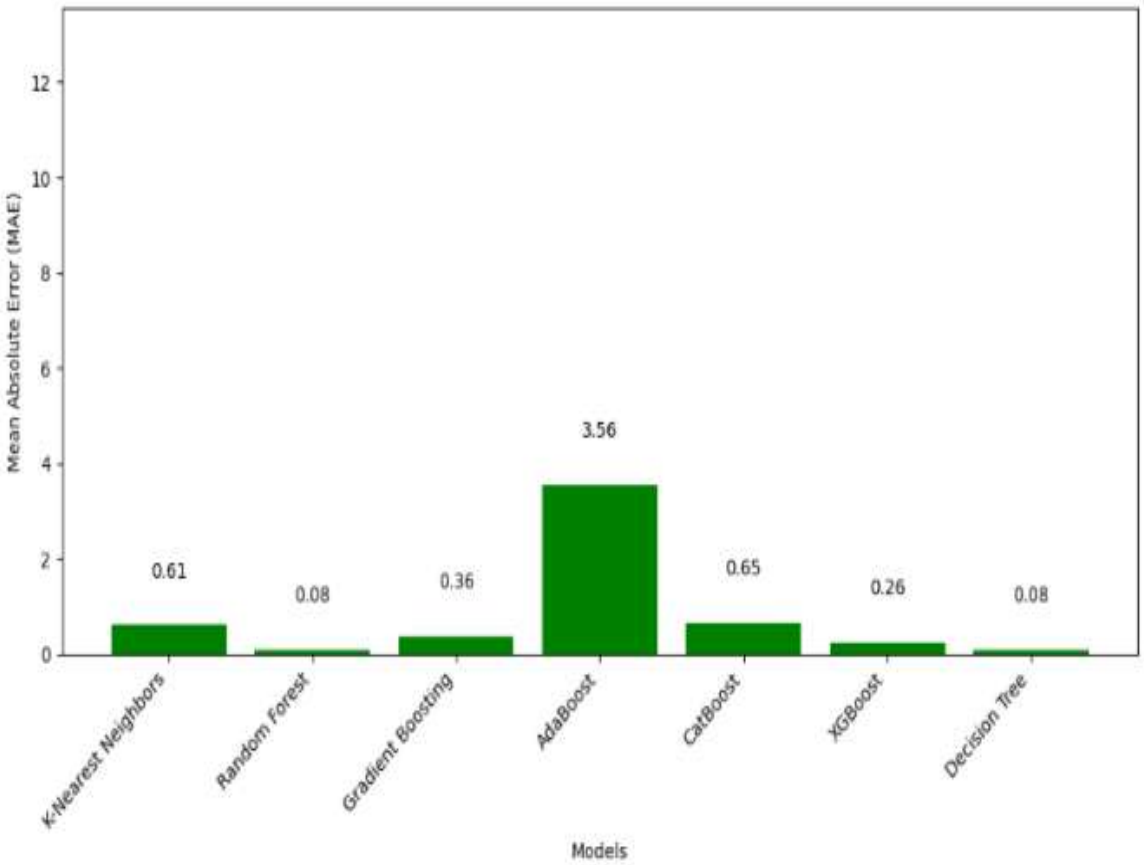
Notably, Random Forest achieves the best performance overall, with the lowest MAE, MSE, and RMSE values, showcasing high accuracy and precision in predicting maize crop yields.

On the other hand, KNN exhibits the lowest R-squared value and significantly higher errors, suggesting limitations in capturing the underlying patterns of the data. AdaBoost, while achieving a respectable R-squared value, has the highest errors, particularly in MSE and RMSE, indicating potential challenges in accurately predicting maize crop outcomes.

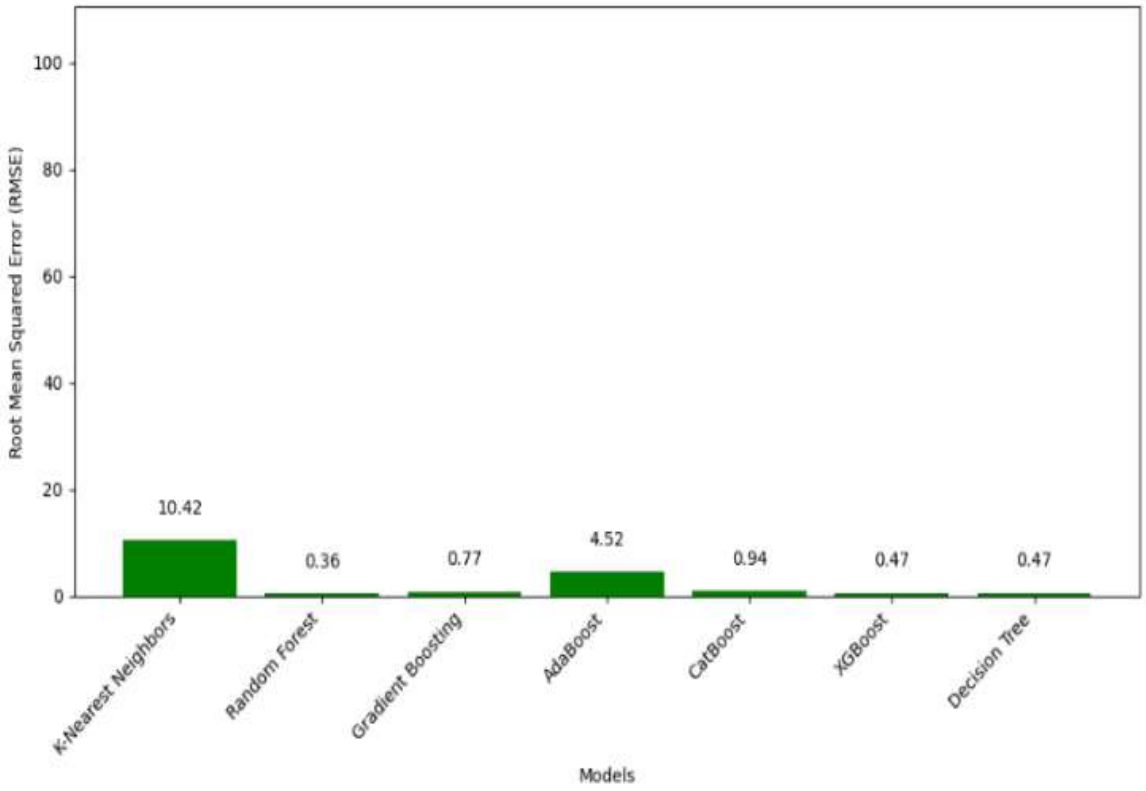
In summary, Random Forest stands out as the top performer, offering the most accurate predictions, while KNN and AdaBoost show comparatively weaker performance in this context.


R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.505
0.999
0.997
0.907
0.996
0.999
0.999
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(a)


Mean Absolute Error (MAE)
12
10
8
6
4
2
0
0.61
0.08
0.36
3.56
0.65
0.26
0.08
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(b)



(c)

Figure 19 Bar chart (a),(b),(c) showing performance of different models on the dataset.

In Fig 20. The performance of voting Regression is showed by visual representation using bar chart where R-squared (R^2) is achieved as 1.00, Mean Absolute Error (MAE) as 0.22 and Root Mean Squared Error (RMSE) as 0.41.

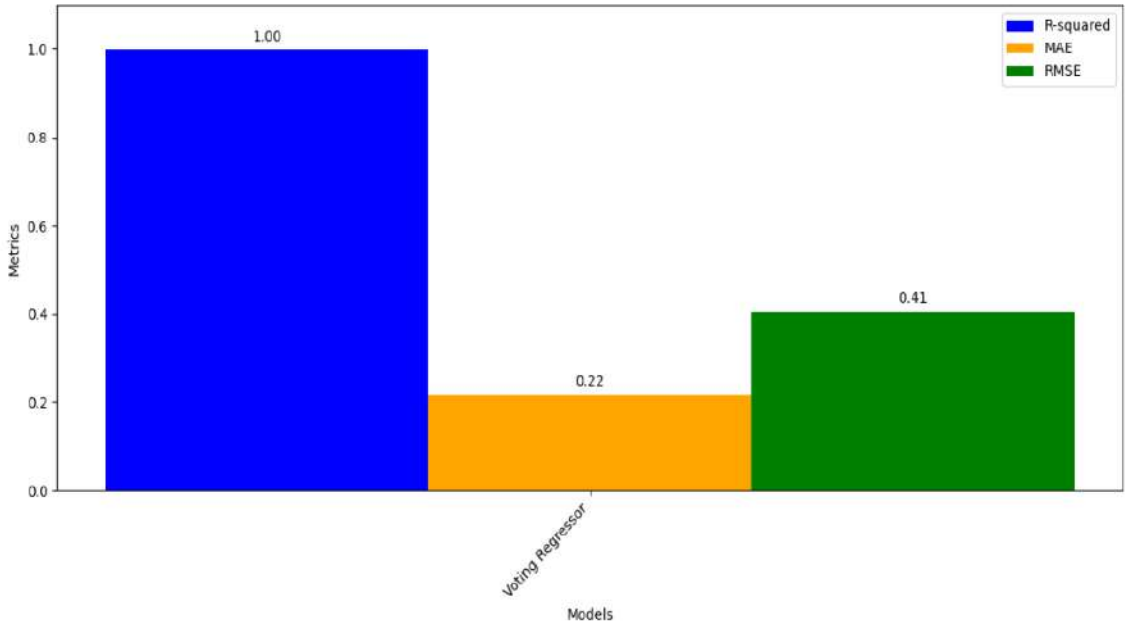


Figure 20 Bar chart showing performance of voting regression

In Table VI, the stacking model utilizing KNN as the final estimator exhibited exceptional performance among all combinations.

This is evident in its high R-squared (R^2) value of 0.99, signifying the model's ability to effectively explain the variance in the data. The low Mean Absolute Error (MAE) of 0.11 further substantiates the accuracy of the predictions, indicating how well the model captured the average size of mistakes.

Moreover, the efficacy of the Stacking with SVR model in reducing both systematic and non-systematic errors is demonstrated by the Mean Squared Error (MSE) of 0.12 and the Root Mean Squared Error (RMSE) of 0.35.

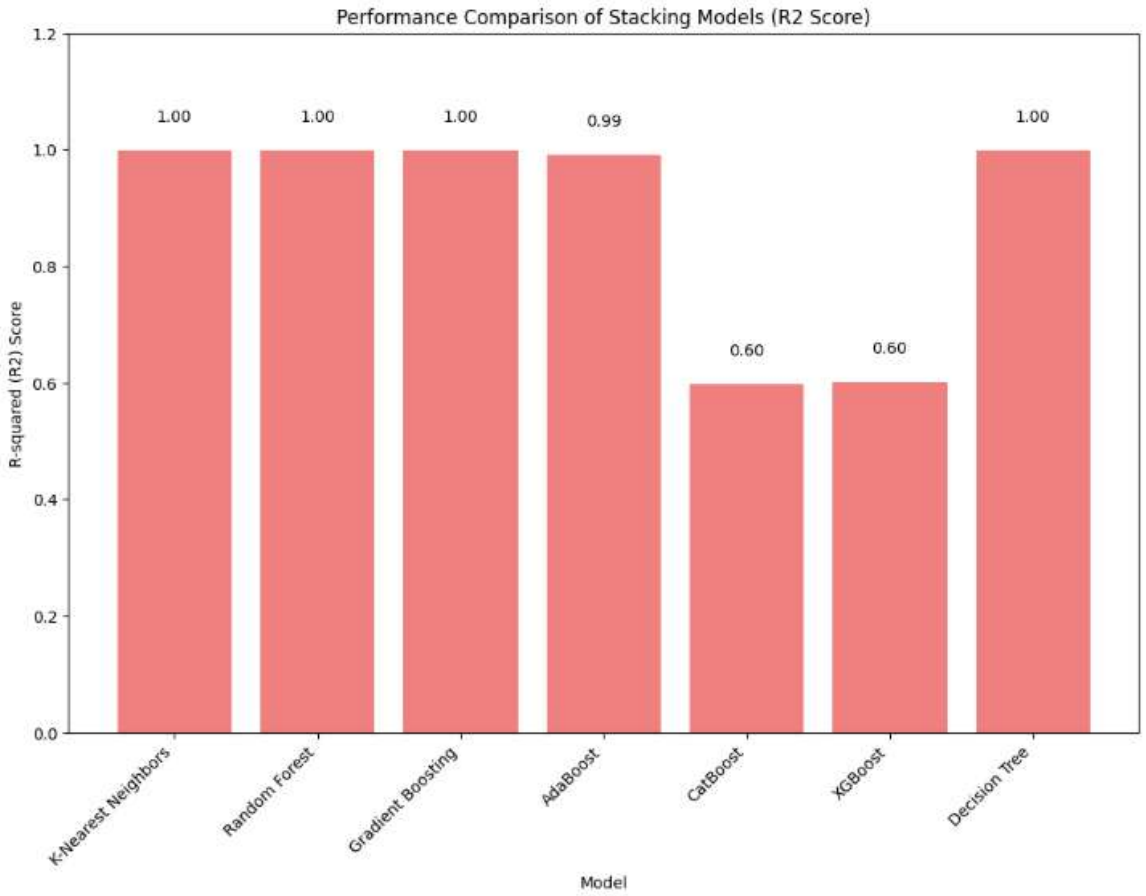
The stacking model, particularly when employing KNN as the final estimator, showcases noteworthy performance metrics.

The high R-squared value and low error measures such as MAE, MSE, and RMSE collectively highlight the accuracy and effectiveness of the stacking ensemble in predicting crop yield of potato, underlining its potential as a robust approach for yield forecasting.

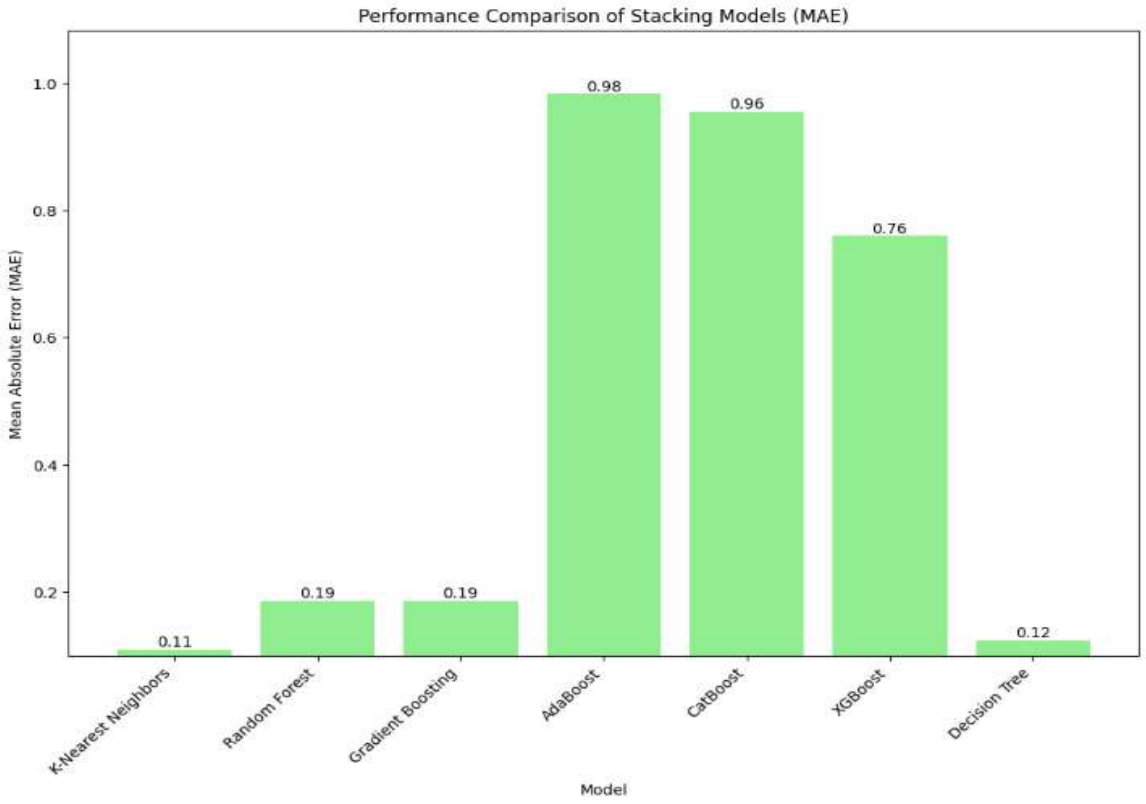
TABLE VI. EVALUATING STACKING PERFORMANCE BY CHANGING FINAL ESTIMATORS FOR POTATO PREDICTION.

ML Regression	R^2	MAE	MSE	$RMSE$
Decision Tree	0.99	0.13	0.26	0.50
KNN	0.99	0.11	0.12	0.35
Random Forest	0.99	0.17	0.14	0.38
Gradient.Boosting	0.99	0.17	0.14	0.38
AdaBoost	0.96	1.56	8.69	2.94
CatBoost	0.59	0.94	88.58	9.41
XgBoost	0.60	0.76	87.43	9.35

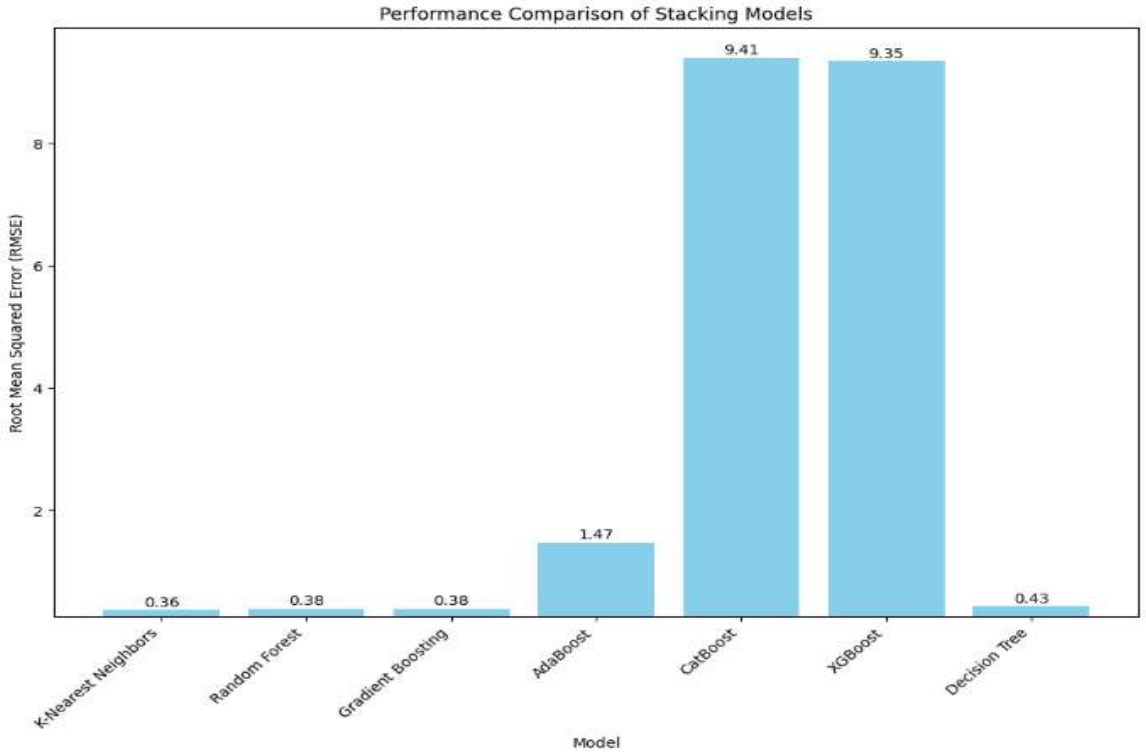
In Fig 21. the bar chart is showing comparison between all stacking performances. It is giving visual representation of how well stacking has performed on the dataset.


Performance Comparison of Stacking Models (R2 Score)
R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.00
1.00
1.00
0.99
0.60
0.60
1.00
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(a)

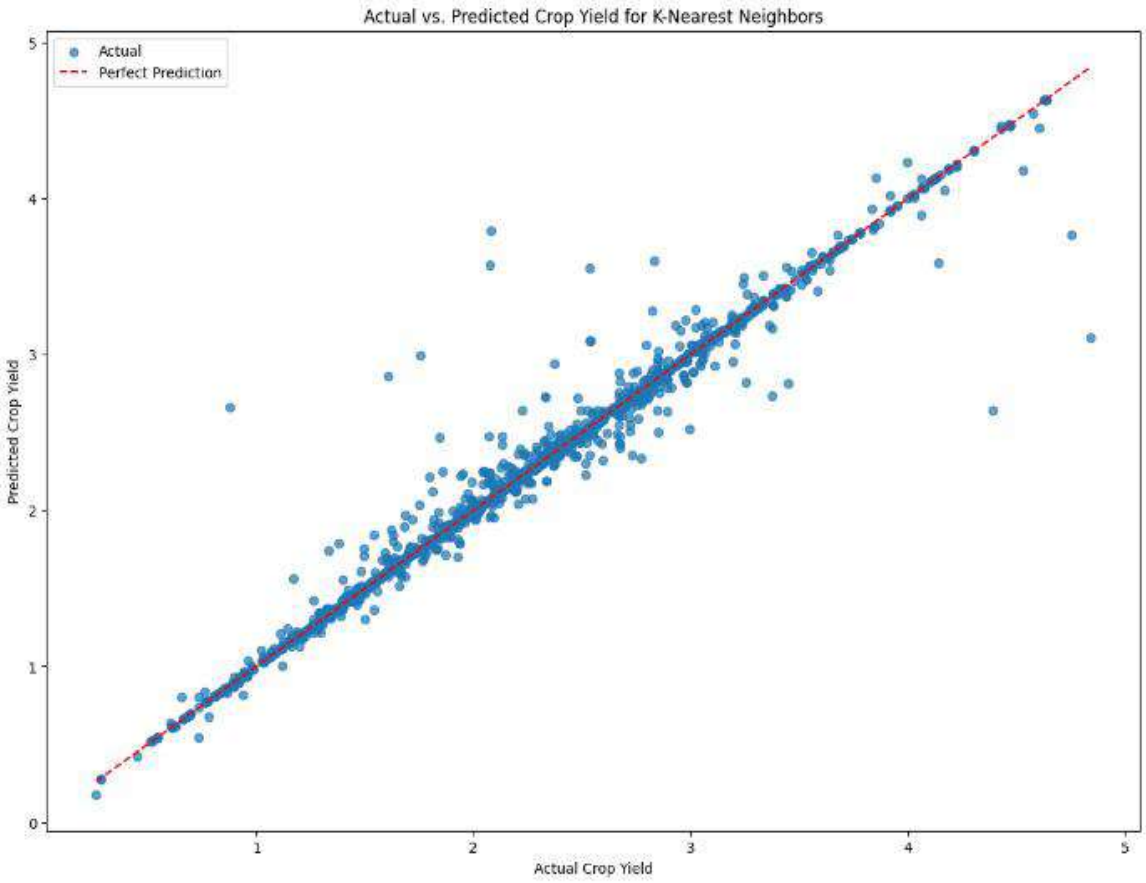

Performance Comparison of Stacking Models (MAE)
Mean Absolute Error (MAE)
1.0
0.8
0.6
0.4
0.2
0.11
0.19
0.19
0.98
0.96
0.76
0.12
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(b)

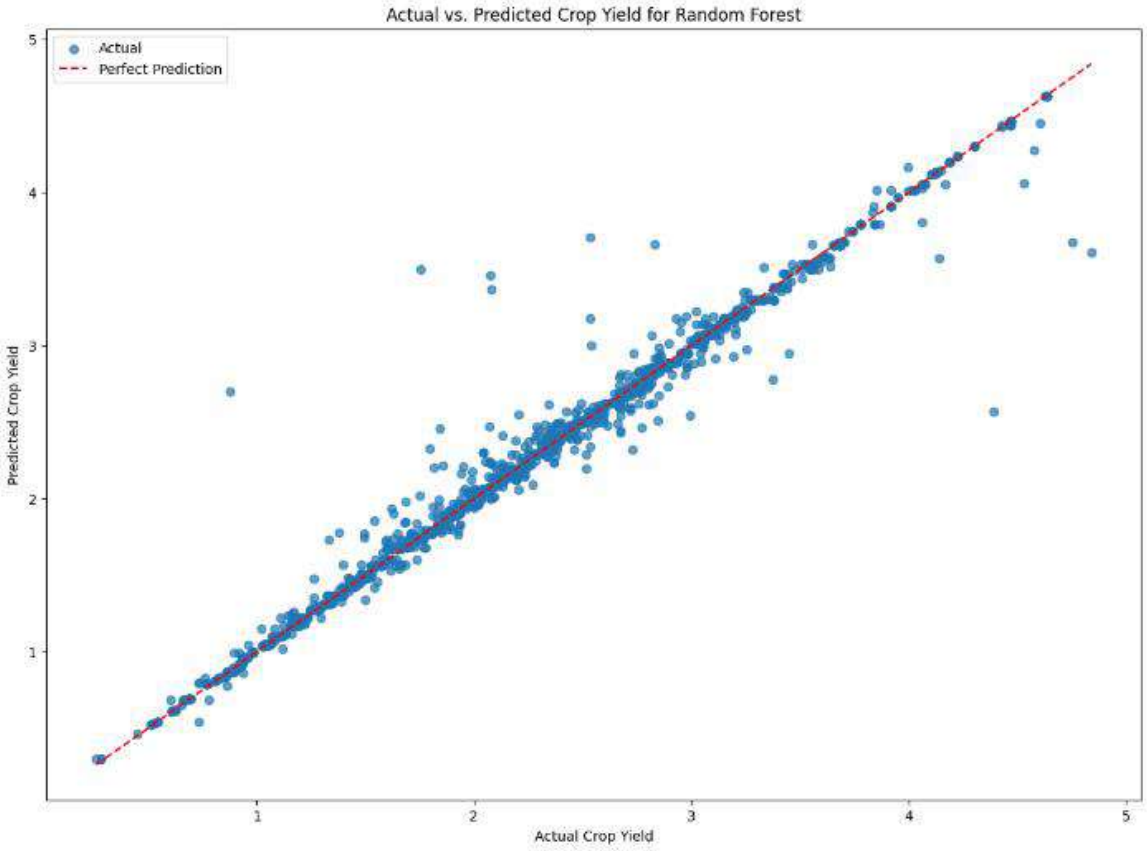


(c)

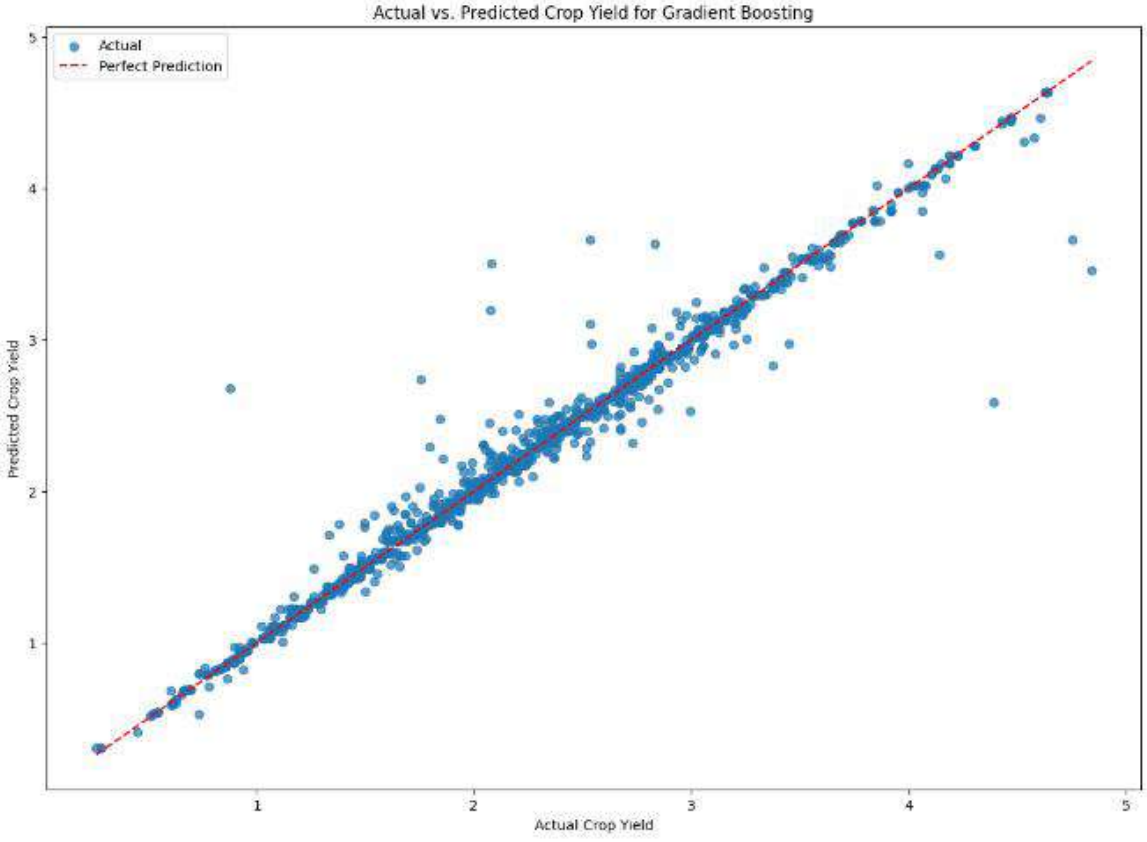
Figure 21 Bar chart (a),(b),(c) showing performance of stacking by varying final estimators.



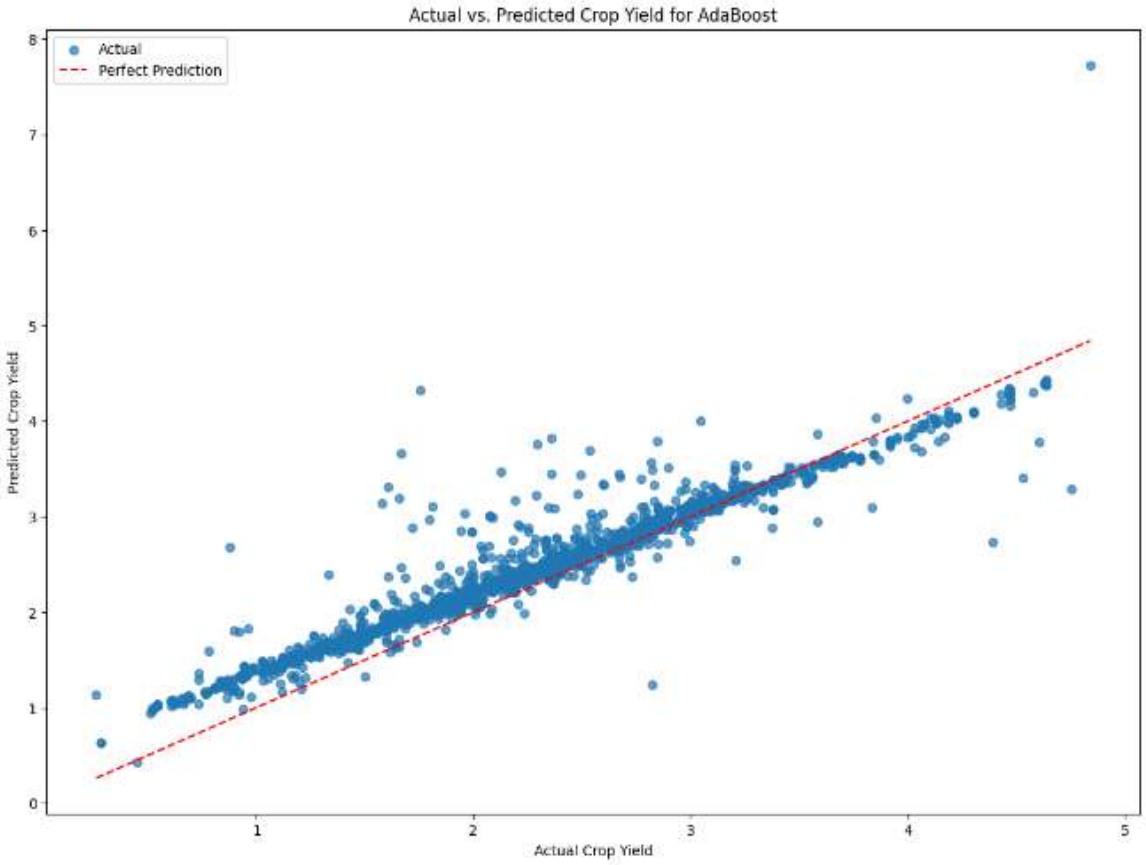
(a)


Actual vs. Predicted Crop Yield for Random Forest
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
1
2
3
4
5

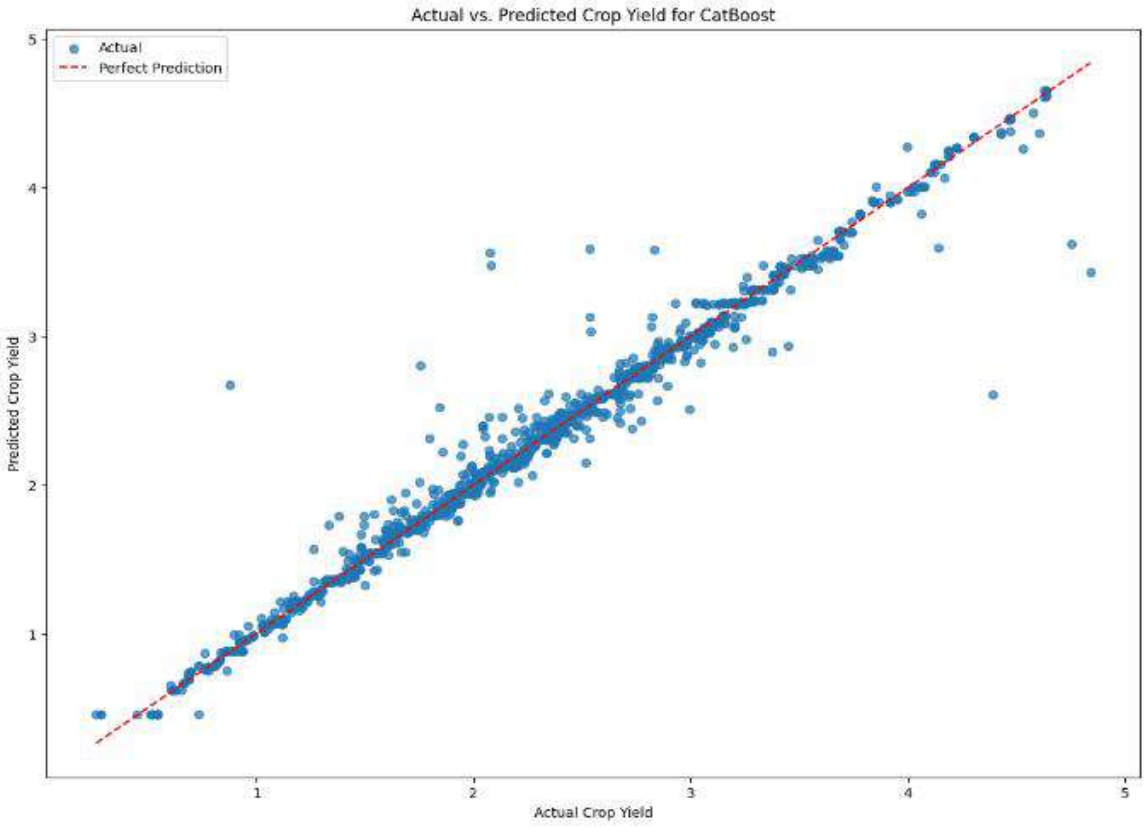
(b)


Actual vs. Predicted Crop Yield for Gradient Boosting
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
1
2
3
4
5

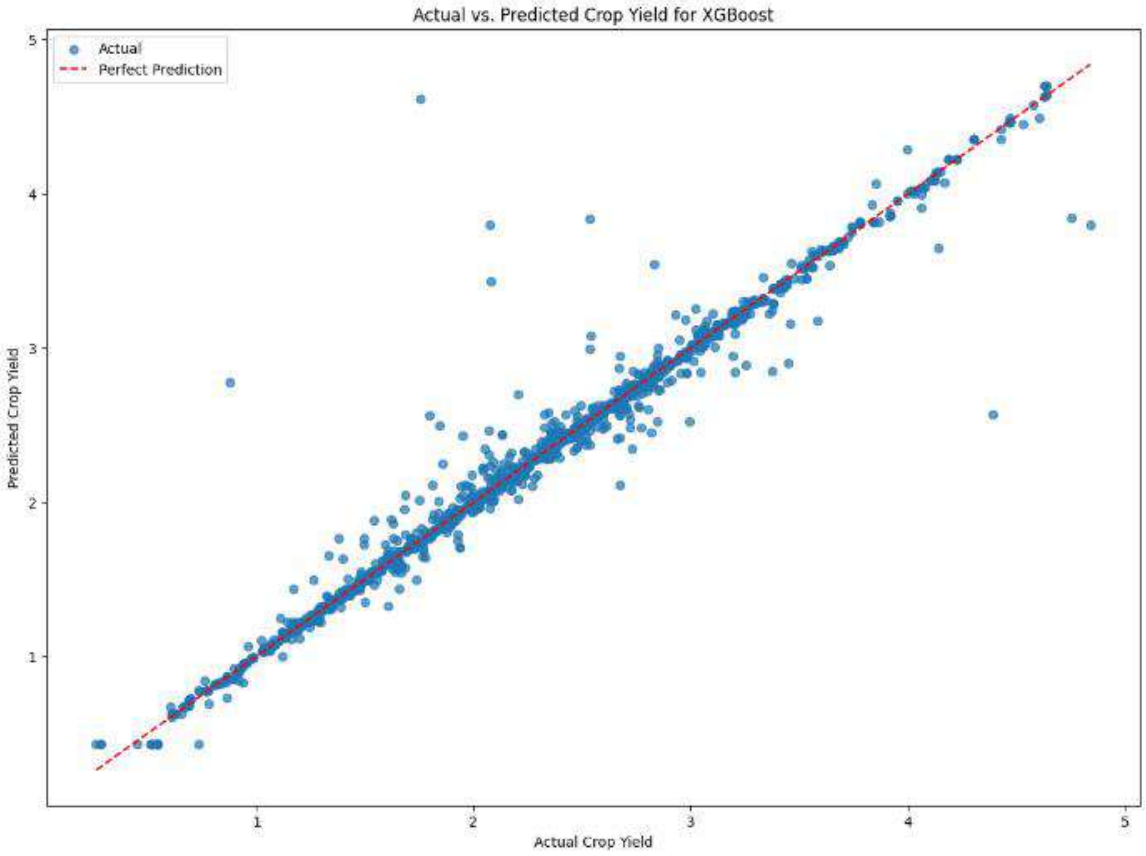
(c)


Actual vs. Predicted Crop Yield for AdaBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
1
2
3
4
5
6
7
8

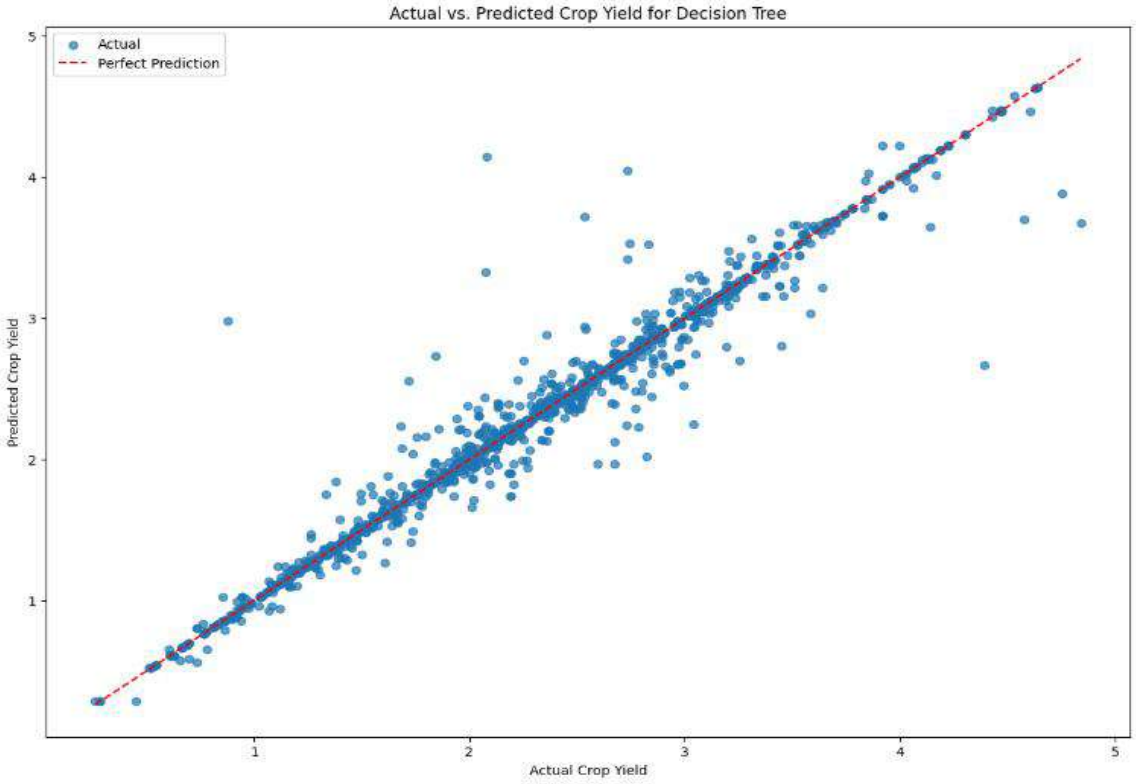
(d)


Actual vs. Predicted Crop Yield for CatBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
1
2
3
4
5

(e)



(f)



(g)

Figure 22 Scattering plot of different stacking performance (a),(b),(c),(d),(e),(f),(g).

Fig. 22 is indicating the scattering plot of the all stacking models or potato yield.

4.5 RICE RESULT ANALYSIS

Here the performance evaluation will be carried on using different machine learning. Results are given below :

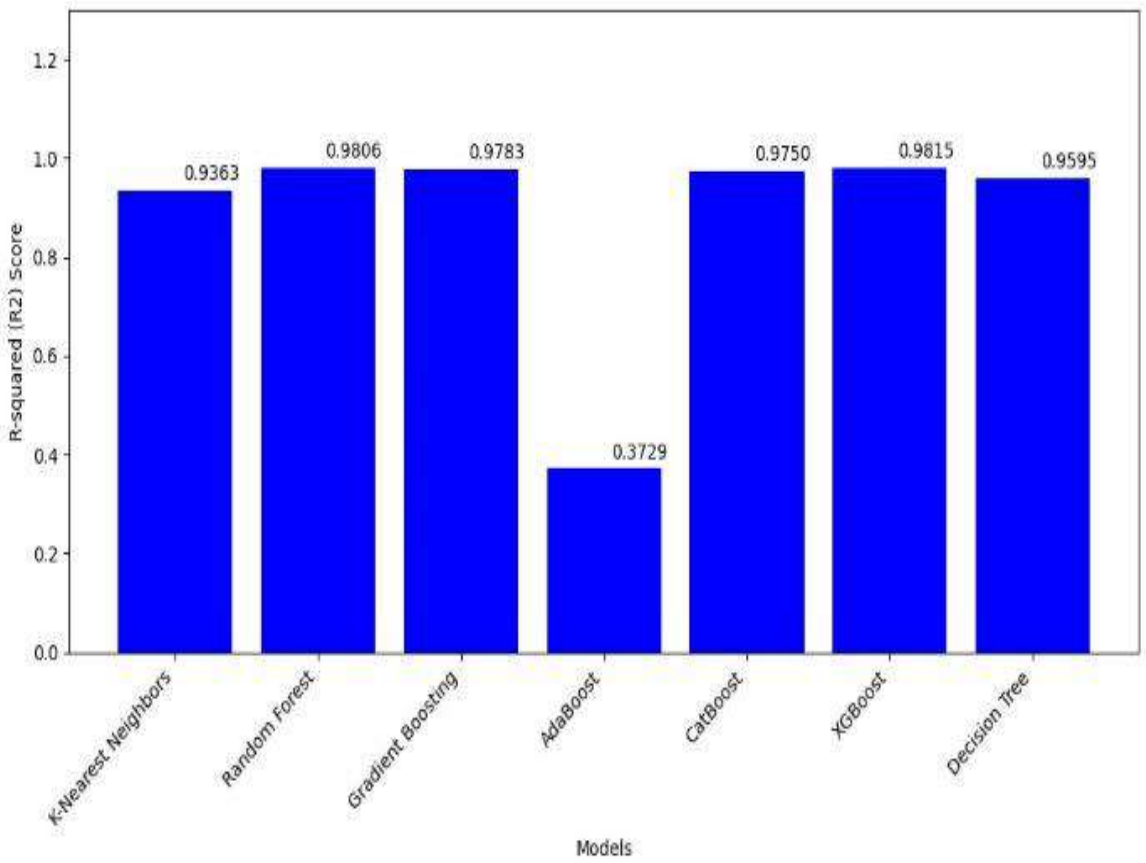
TABLE VII. PERFORMANCE EVALUATION USING DIFFERENT MACHINE LEARNING REPRESSORS FOR RICE CROP PREDICTION.

ML Regression	R^2	MAE	MSE	$RMSE$
Decision Tree	0.95	0.04	0.03	0.16
KNN	0.93	0.06	0.04	0.20
Random Forest	0.98	0.03	0.01	0.11
Gradient Boosting	0.97	0.03	0.01	0.12
AdaBoost	0.37	0.515	0.41	0.64
CatBoost	0.97	0.70	0.02	0.13
XgBoost	0.98	0.04	0.01	0.11

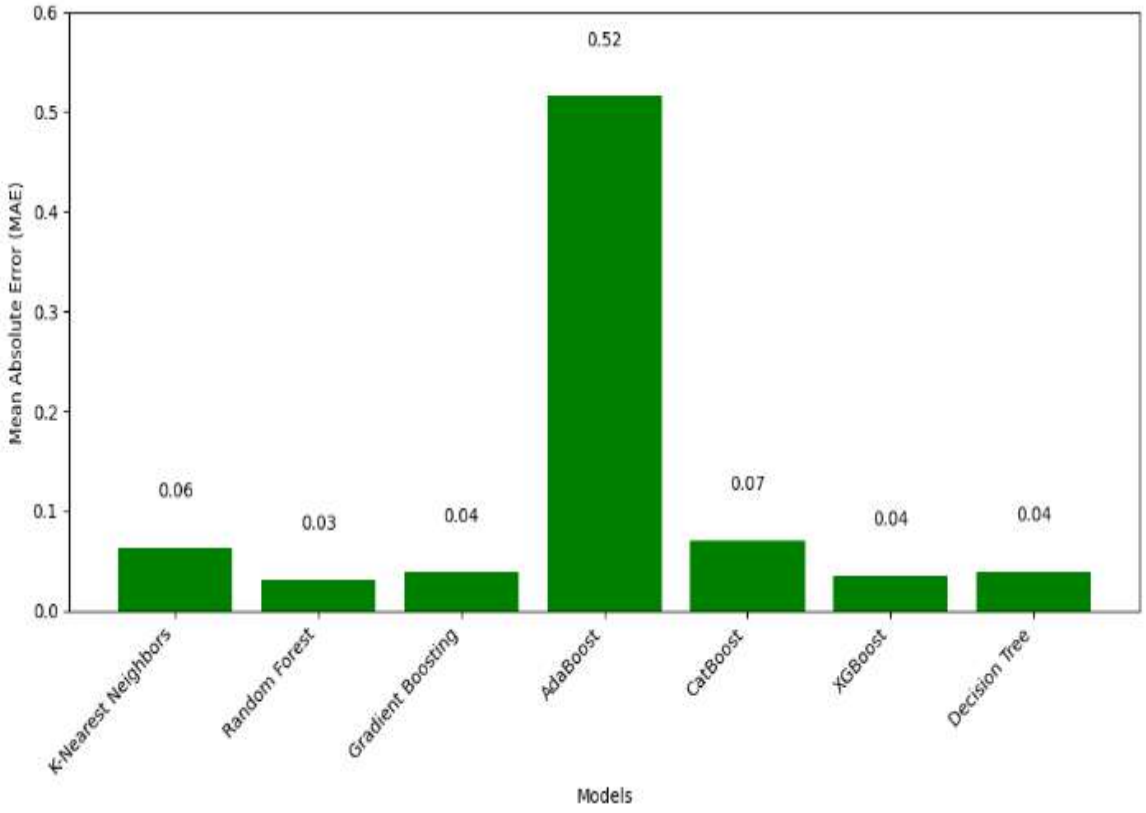
Table VII outlines the performance assessment of diverse machine learning regression models for rice crop prediction. Random Forest and XgBoost emerge as the top performers, achieving the highest R-squared values and the lowest MAE, MSE, and RMSE metrics. These models exhibit exceptional accuracy and precision in predicting rice crop outcomes.

Decision Tree and Gradient Boosting also show strong performance, with high R-squared values and relatively low error metrics. CatBoost performs well overall, but it demonstrates a slightly higher MAE and RMSE compared to Random Forest and XgBoost. KNN exhibits a good R-squared value but with slightly higher errors, indicating a moderate level of accuracy.

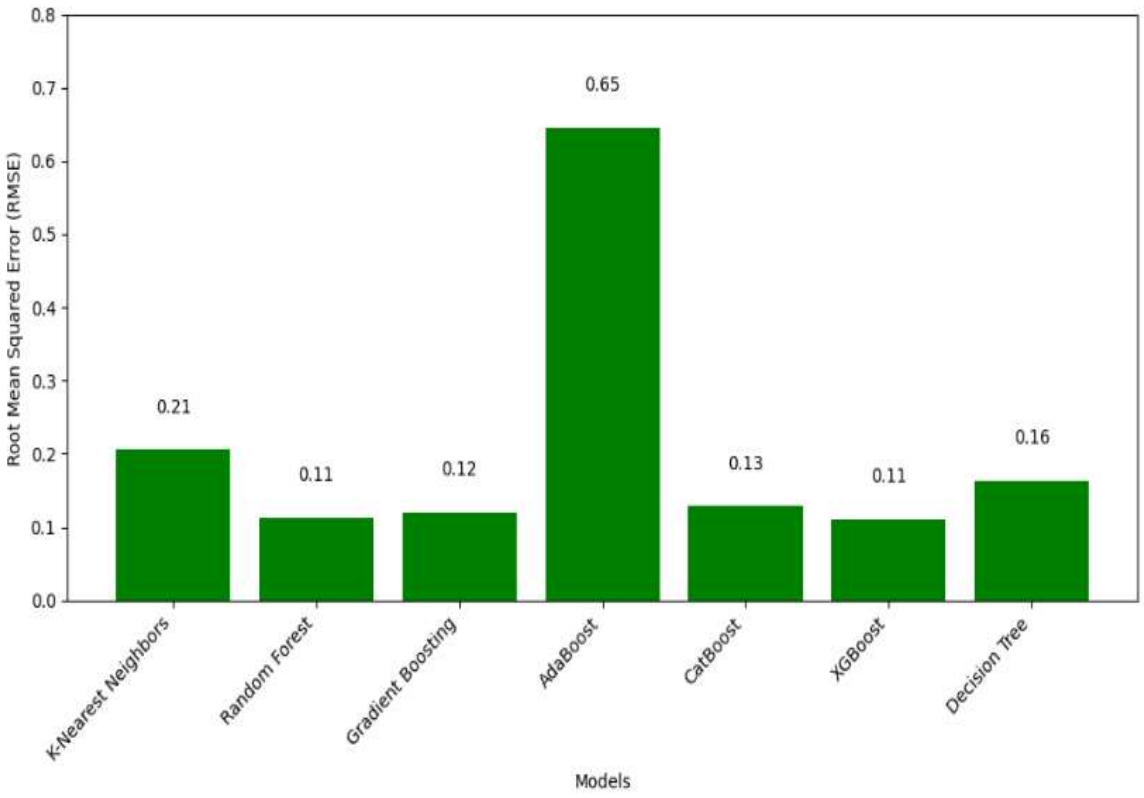
However, AdaBoost performs less effectively with the lowest R-squared value and higher errors, suggesting challenges in capturing the complexities of rice crop data. In summary, Random Forest and XgBoost stand out as the best performers, showcasing superior predictive capabilities, while AdaBoost shows the least effective performance among the models assessed for rice crop prediction.


R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.9363
0.9806
0.9783
0.3729
0.9750
0.9815
0.9595
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(a)


Mean Absolute Error (MAE)
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.06
0.03
0.04
0.52
0.07
0.04
0.04
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Models

(b)



(c)

Figure 23 Bar chart (a),(b),(c) showing performance of different models on the dataset.

In Fig 24. The performance of voting Regression is showed by visual representation using bar chart where R-squared (R^2) is achieved as 1.00, Mean Absolute Error (MAE) as 0.06 and Root Mean Squared Error (RMSE) as 0.14.

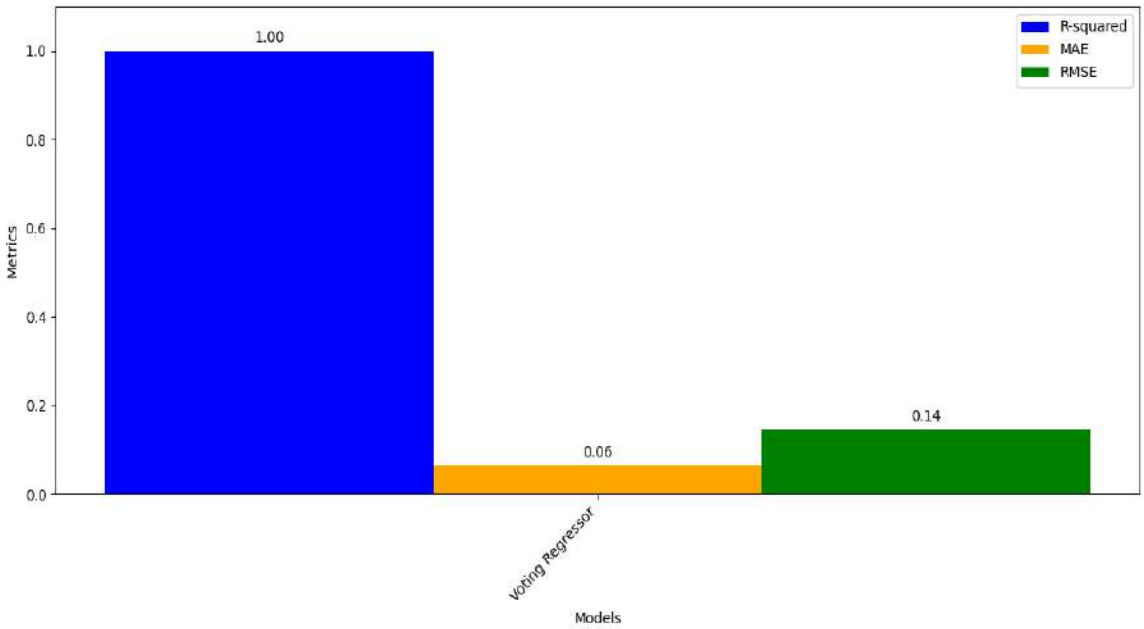


Figure 24 Bar chart showing performance of voting regression

TABLE VIII. EVALUATING STACKING PERFORMANCE BY CHANGING FINAL ESTIMATORS FOR RICE PREDICTION.

ML Regression	R^2	MAE	MSE	$RMSE$
Decision Tree.	0.97	0.04	0.01	0.13
KNN.	0.97	0.03	0.01	0.11
Random Forest	0.98	0.04	0.01	0.12
Gradient .Boosting	0.98	0.04	0.01	0.11
AdaBoost.	0.86	0.22	0.09	0.29
CatBoost.	0.98	0.04	0.01	0.12
XgBoost.	0.97	0.04	0.02	0.13

Table 8 presents an evaluation of stacking performance by varying final estimators for rice crop prediction.

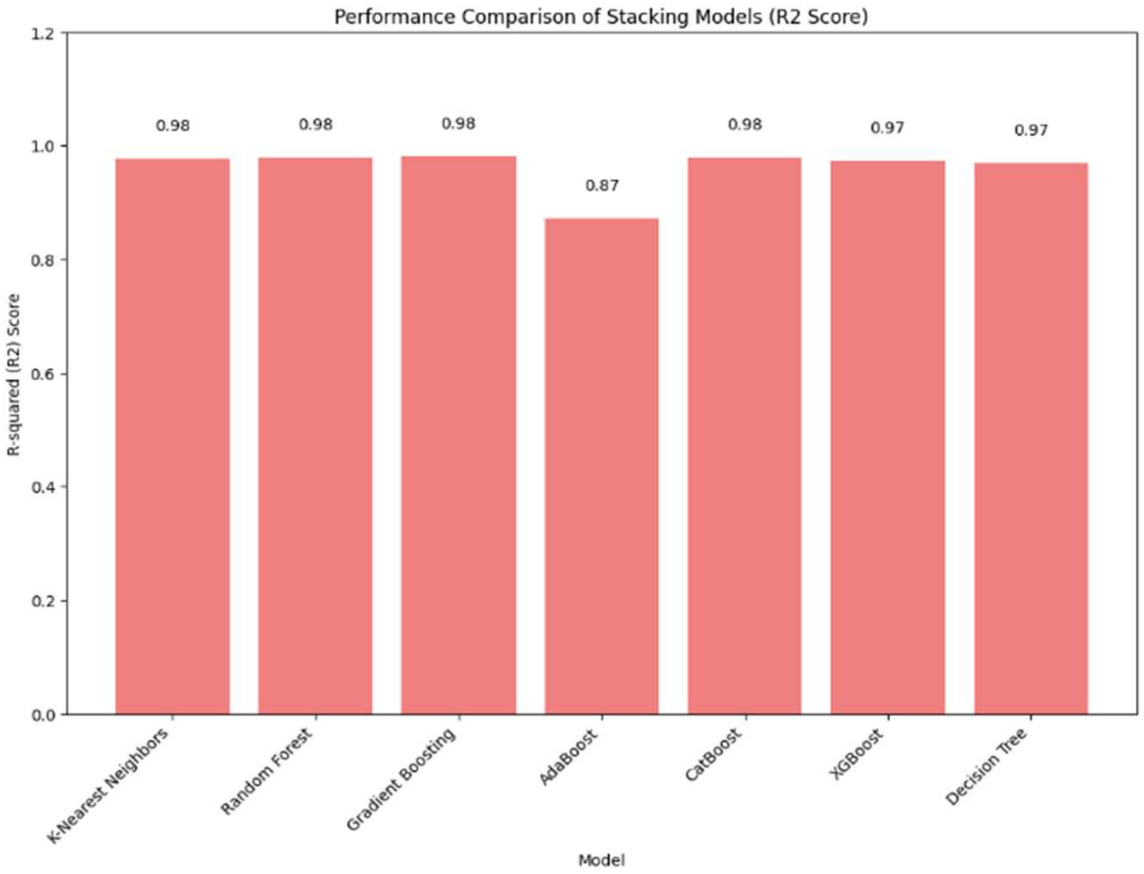
The stacked models, comprising Decision Tree, KNN, Random Forest, Gradient Boosting, AdaBoost, CatBoost, and XgBoost, demonstrate consistently high R-squared values, indicative of strong predictive capabilities.

Notably, Random Forest, Gradient Boosting, and CatBoost as final estimators consistently yield the highest performance, with the lowest MAE, MSE, and RMSE values, showcasing their effectiveness in enhancing the overall predictive accuracy of the stacking ensemble. Decision Tree, KNN, and XgBoost also contribute positively to the stacking ensemble, providing accurate predictions.

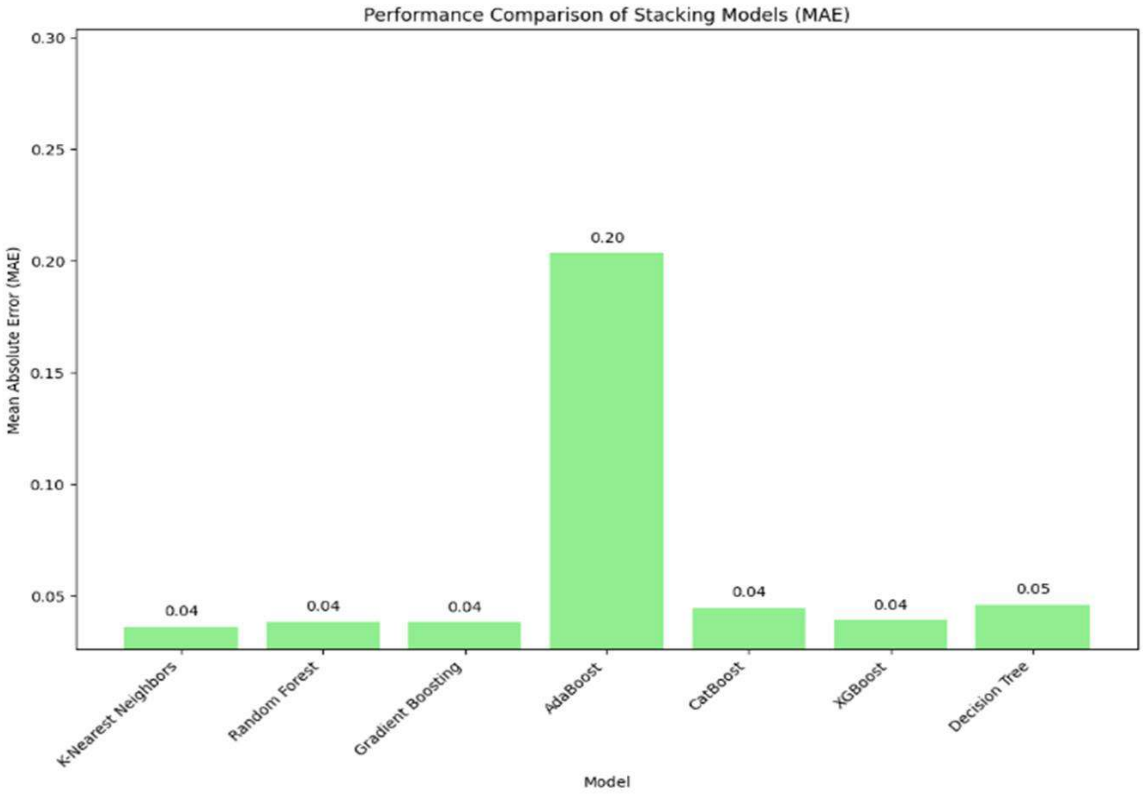
However, AdaBoost, while achieving a respectable R-squared value, exhibits higher errors, suggesting a comparatively weaker contribution to the stacking ensemble.

Overall, the stacking approach with Random Forest, Gradient Boosting, and CatBoost as final estimators appears to be particularly effective in improving the predictive performance for rice crop prediction.

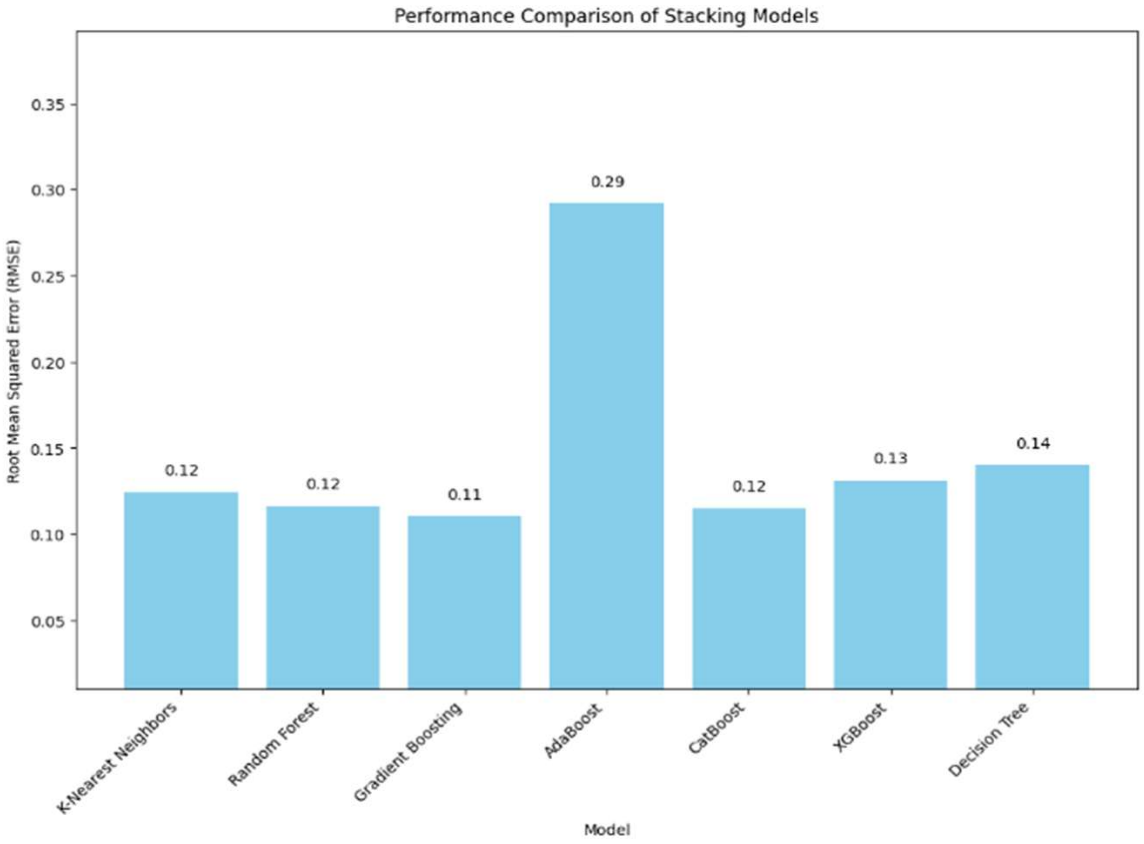
In Fig 20. the bar chart is showing comparison between all stacking performances. It is giving visual representation of how well stacking has performed on the dataset.


Performance Comparison of Stacking Models (R2 Score)
R-squared (R2) Score
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.98
0.98
0.98
0.87
0.98
0.97
0.97
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(a)

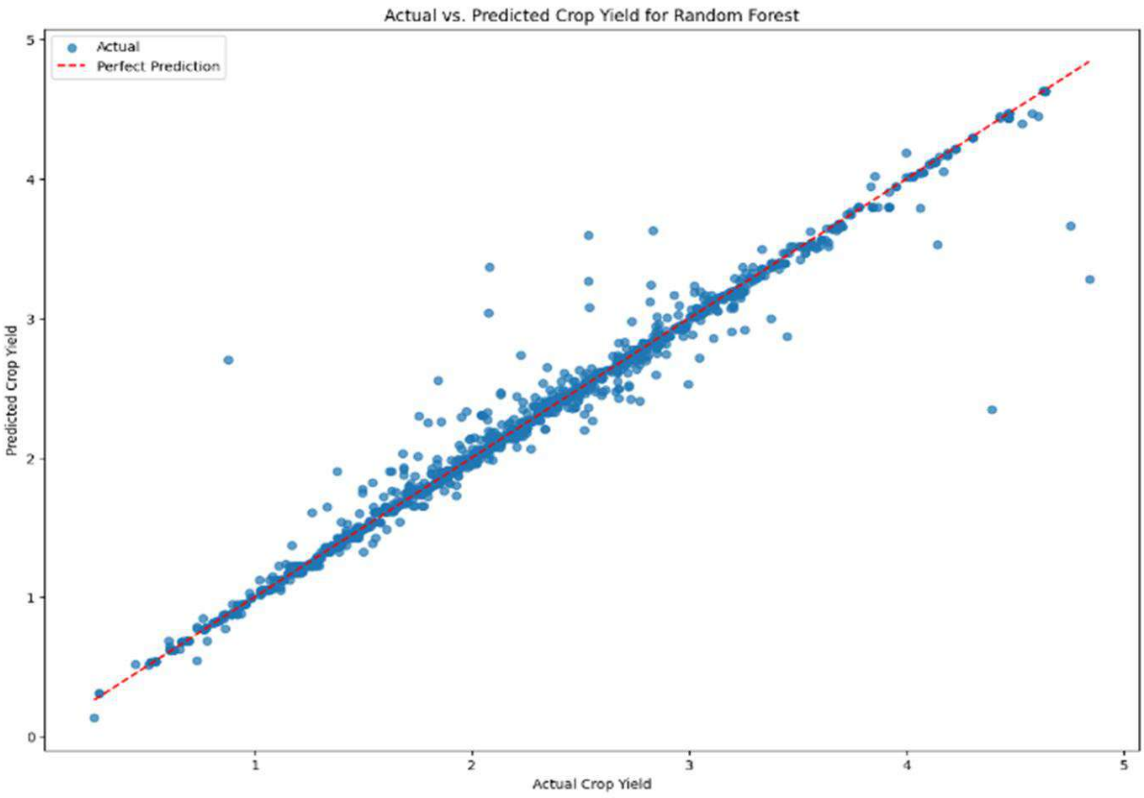

Performance Comparison of Stacking Models (MAE)
Mean Absolute Error (MAE)
0.30
0.25
0.20
0.15
0.10
0.05
0.04
0.04
0.04
0.20
0.04
0.04
0.05
K-Nearest Neighbors
Random Forest
Gradient Boosting
AdaBoost
CatBoost
XGBoost
Decision Tree
Model

(b)

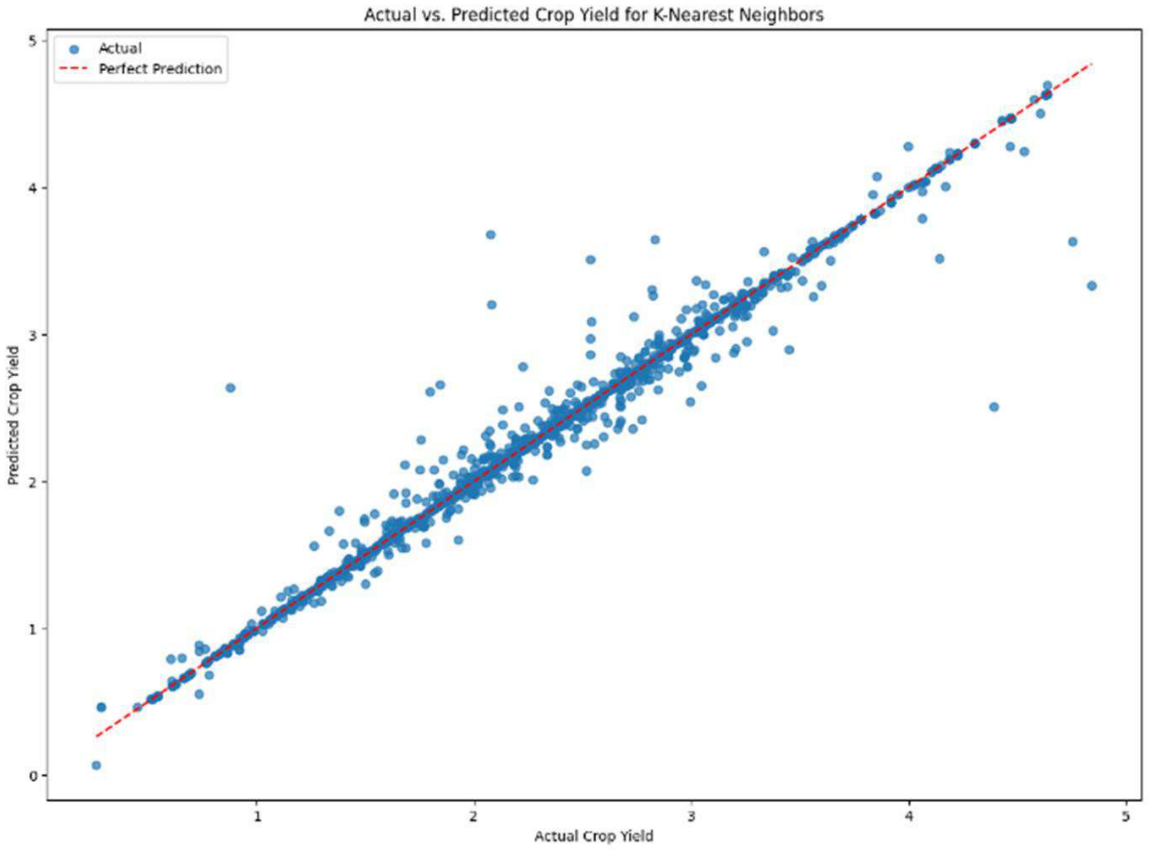


(c)

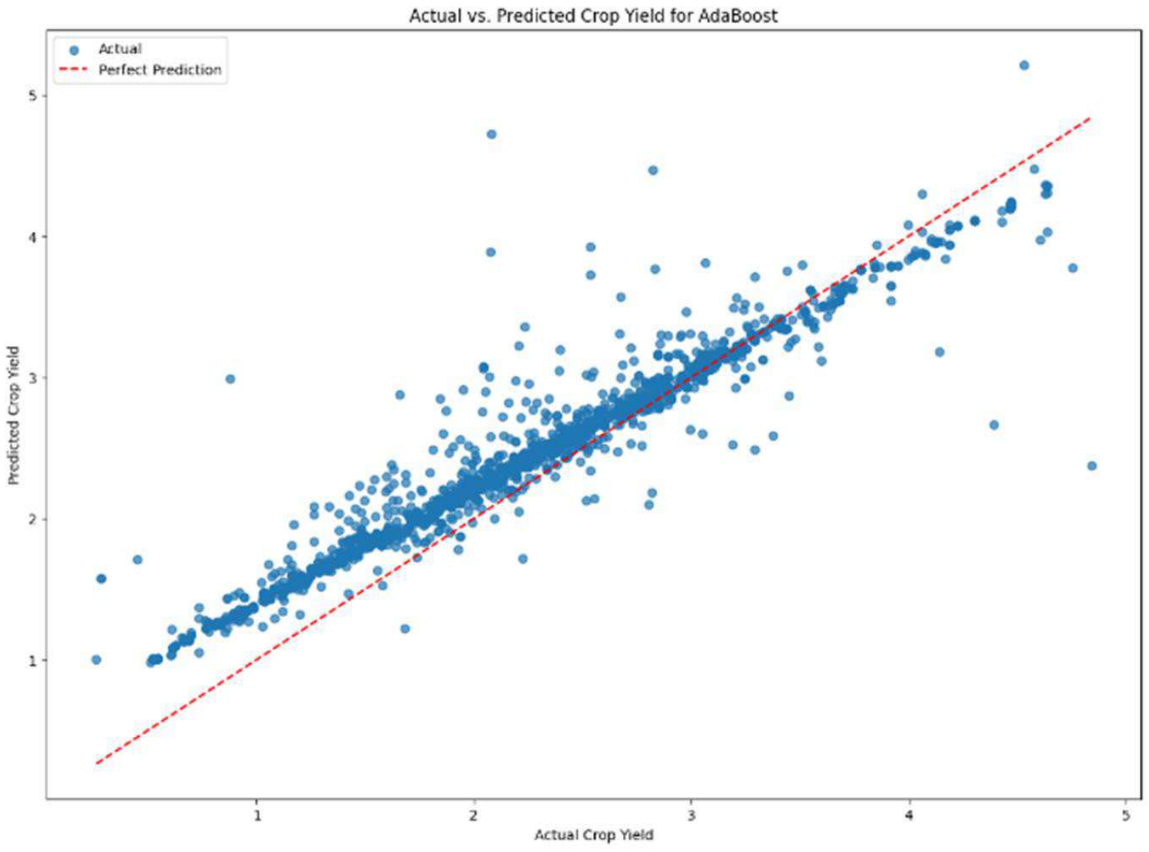
Figure 25 Bar chart (a),(b),(c) showing performance of stacking by varying final estimators.



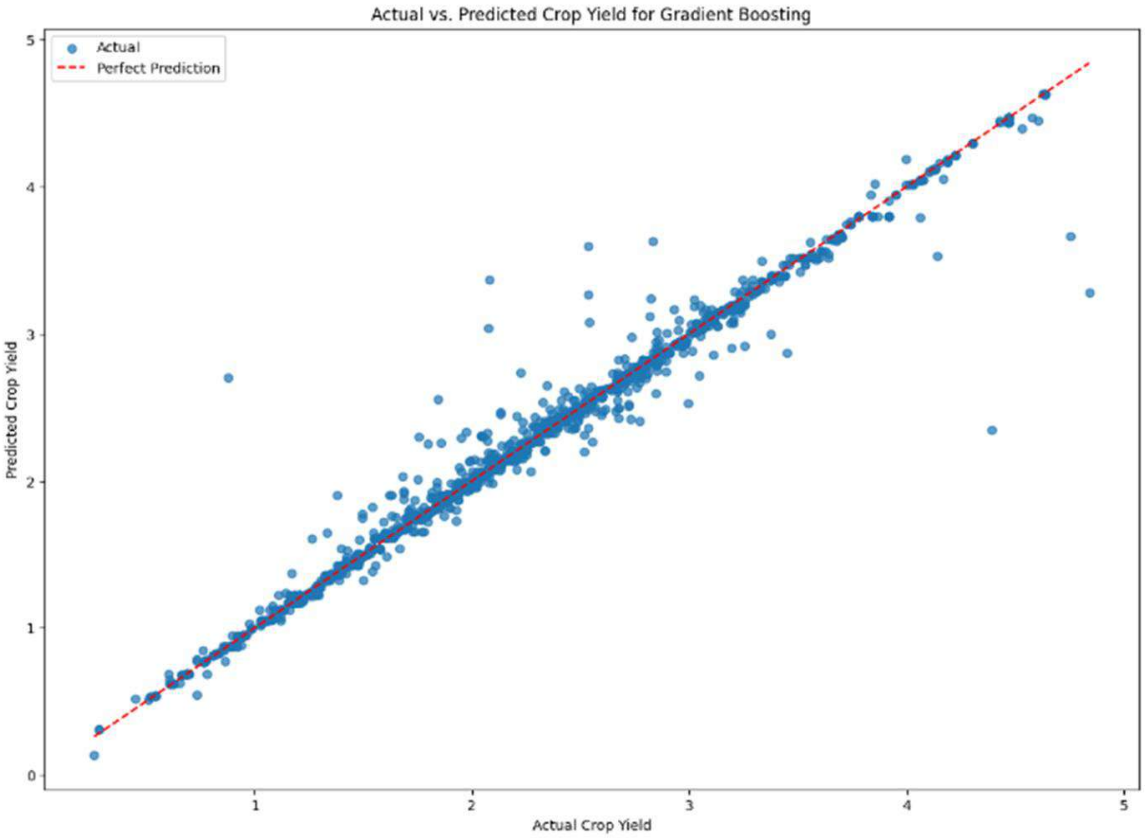
(a)


Actual vs. Predicted Crop Yield for K-Nearest Neighbors
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
1
2
3
4
5

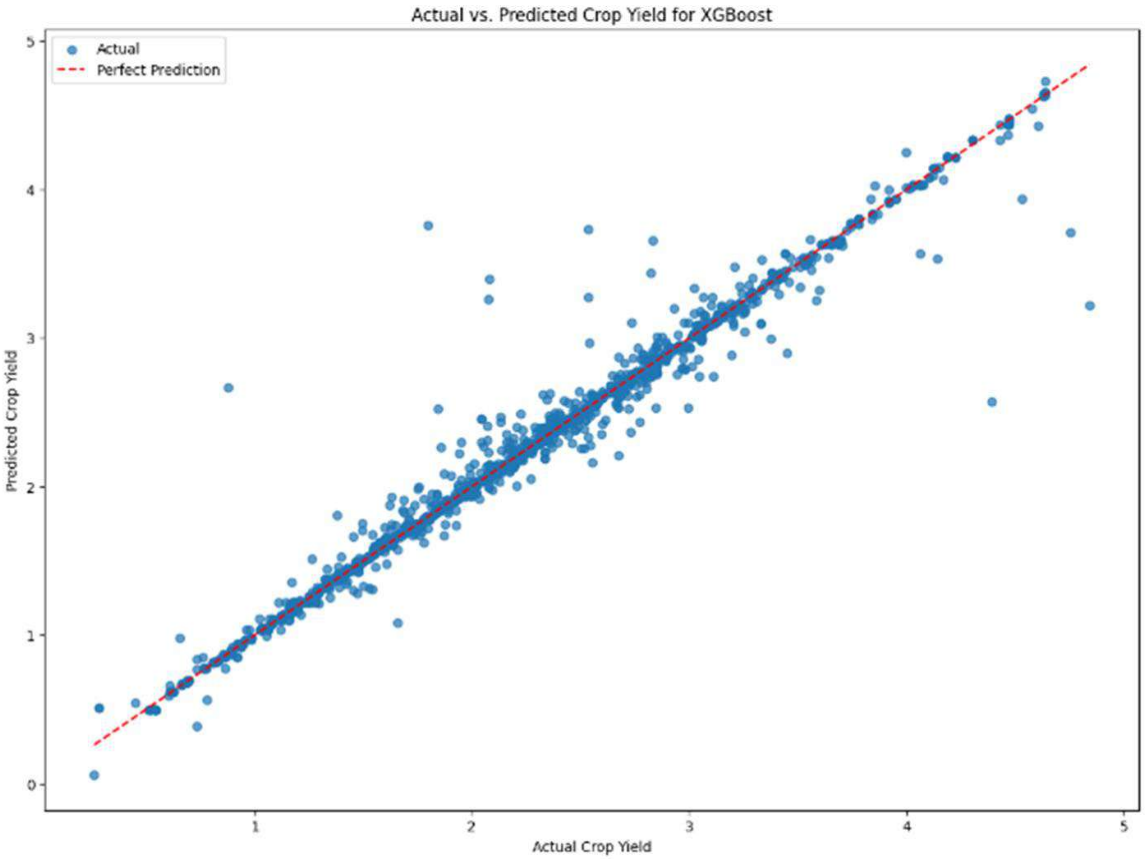
(b)


Actual vs. Predicted Crop Yield for AdaBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
1
2
3
4
5

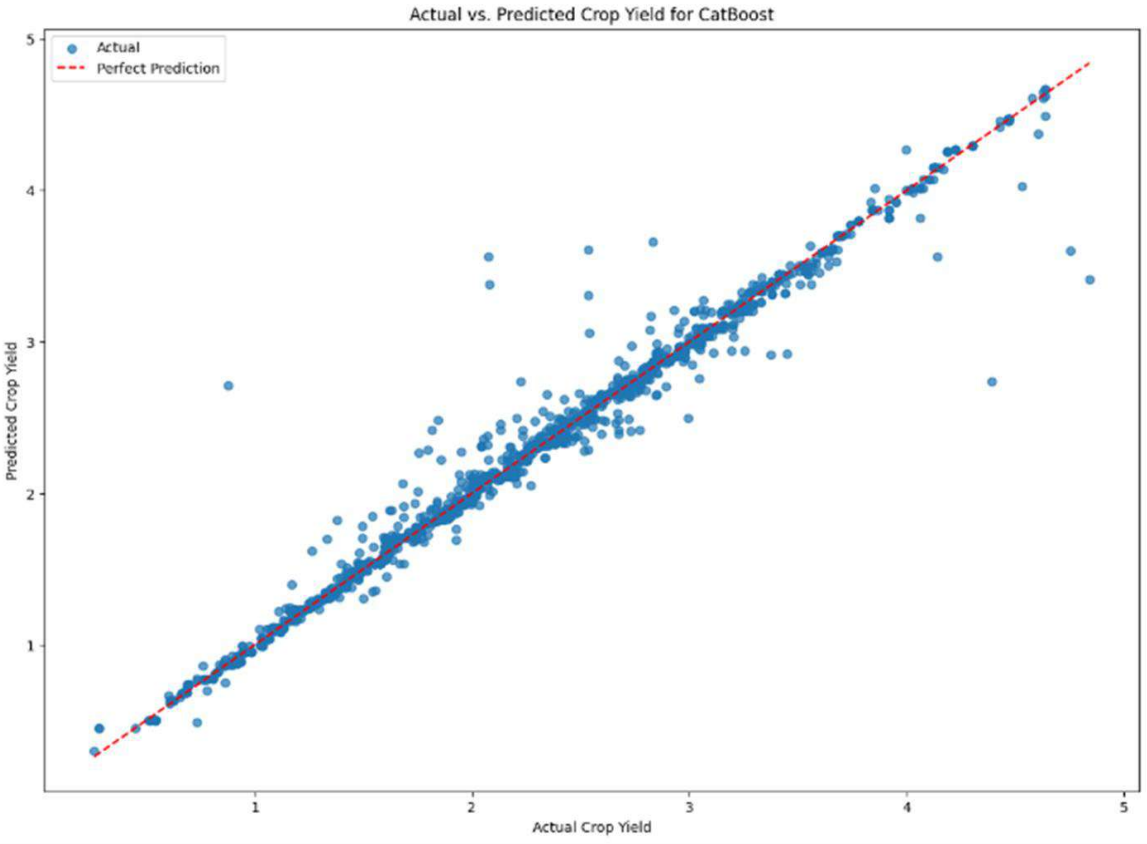
(c)


Actual vs. Predicted Crop Yield for Gradient Boosting
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
1
2
3
4
5

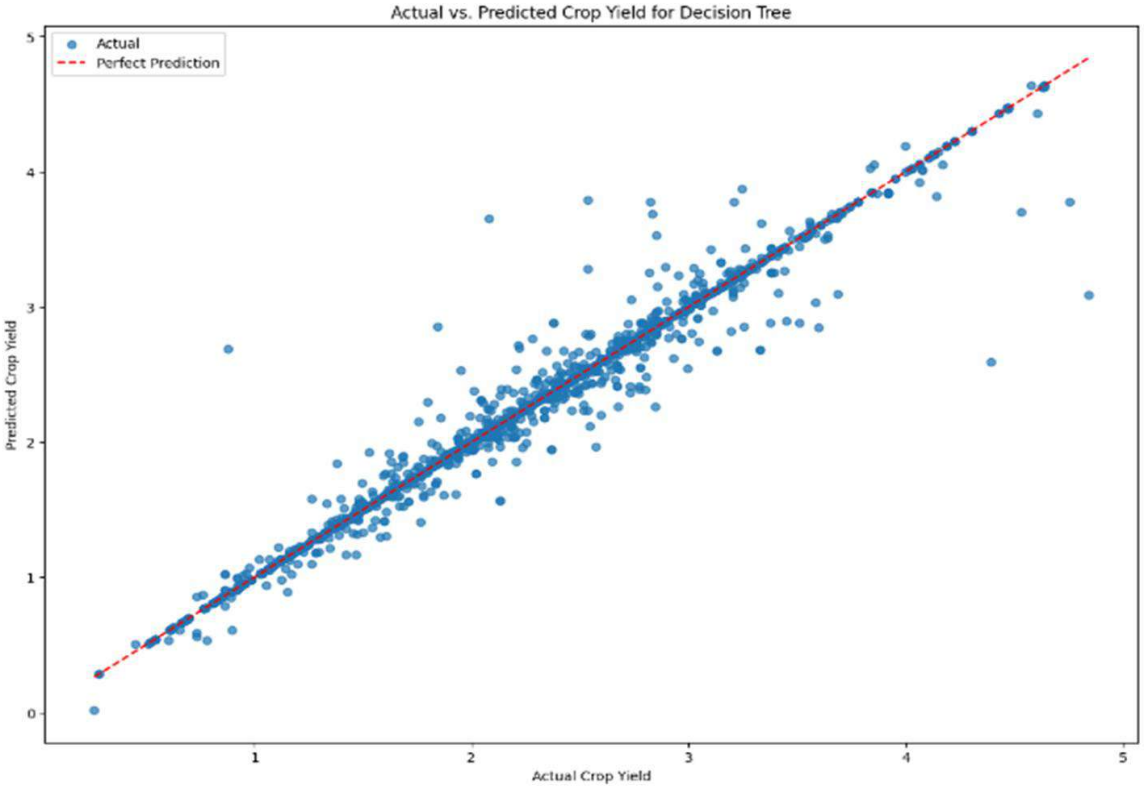
(d)


Actual vs. Predicted Crop Yield for XGBoost
Actual
Perfect Prediction
Predicted Crop Yield
Actual Crop Yield
0
1
2
3
4
5

(e)



(f)



(g)

Figure 26 Scattering plot of different stacking performance (a),(b),(c),(d),(e),(f),(g).

Fig. 26 is indicating the scattering plot of the all stacking models or rice yield.

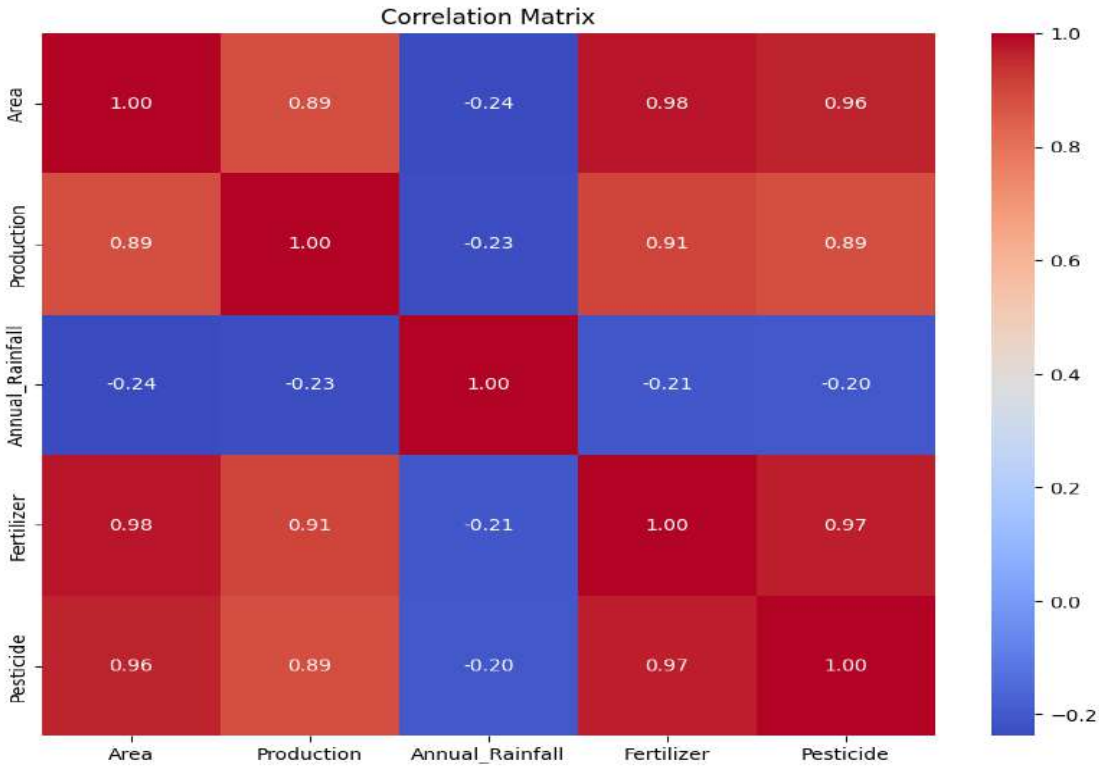
The criteria for selecting the best model is maximum R^2 and minimum MAE. The best result for maize crop yield prediction, achieved R^2 0.99 and MAE 0.04 stacking model using RF as final estimator, rice achieved R^2 0.99 and MAE 0.06 voting model, potato crop yield achieved R^2 0.99 and MAE 0.04 by only simple ML model, sugarcane crop yield prediction achieved R^2 0.99 and MAE 0.36 by stacking model using KNN as final estimator. In regression analysis, the correlation matrix plays a pivotal role in elucidating the interrelationships between various variables involved in the analysis. This matrix comprises correlation coefficients, which quantify the strength and direction of linear associations between pairs of variables. Ranging from -1 to 1, these coefficients provide valuable insights into the nature of relationships within the dataset. The correlation coefficient's interpretation involves understanding the direction of the relationship: a positive coefficient indicates a positive linear association, while a negative coefficient signifies a negative linear relationship. A coefficient near 0 suggests a weak or negligible linear connection between variables. Within the correlation matrix, each row and column correspond to a variable, forming a square matrix. Diagonal elements possess a correlation coefficient of 1, denoting a perfect correlation with itself. Off-diagonal elements contain the coefficients representing associations between pairs of variables.

One critical application of the correlation matrix is in detecting multi collinearity a condition where independent variables are highly correlated. Identifying multi collinearity is vital for addressing issues such as inflated standard errors and challenges in interpreting individual variable contributions within the regression model.

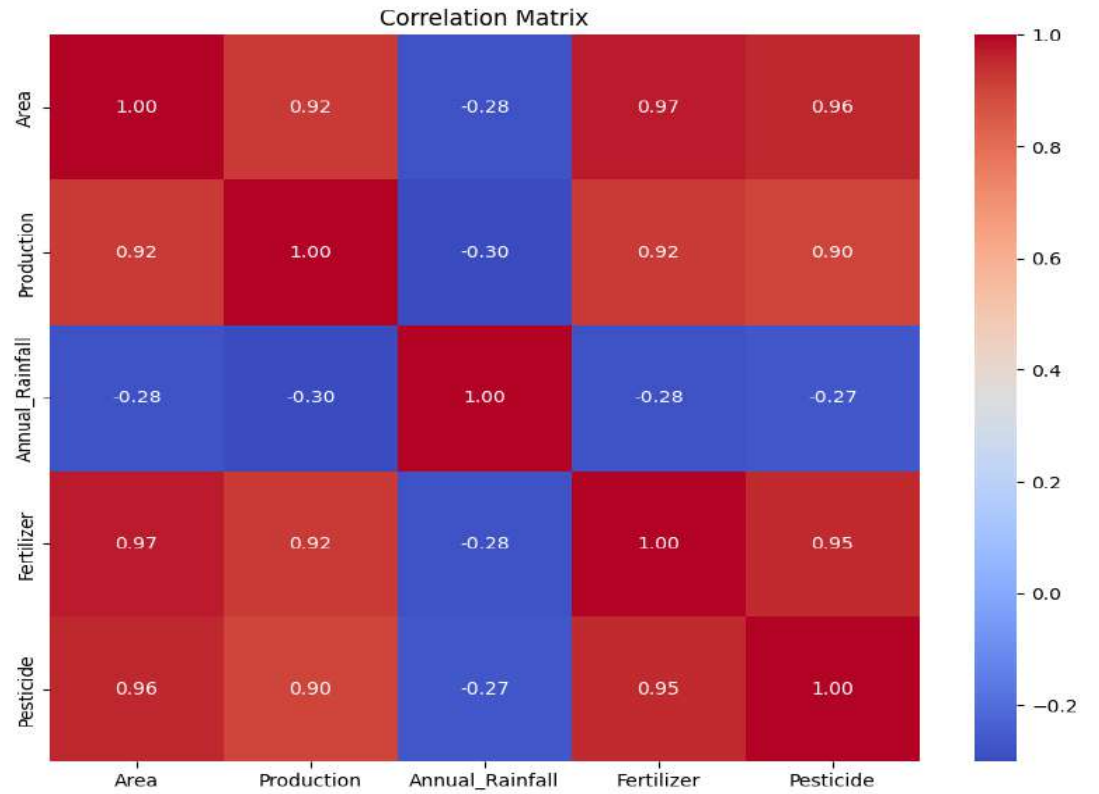
The correlation matrix aids in the interpretation of the regression model by providing insights into the relationships between predictors and the response variable. High correlations may indicate redundancy among predictors, while low correlations suggest that variables contribute unique information. Moreover, adherence to the assumption of independence among predictor variables is essential in regression analysis. The correlation matrix assists in verifying this assumption, ensuring the validity of the regression model.

Visualization of the correlation matrix through techniques like heatmaps facilitates a rapid assessment of the strength and direction of relationships between variables. It

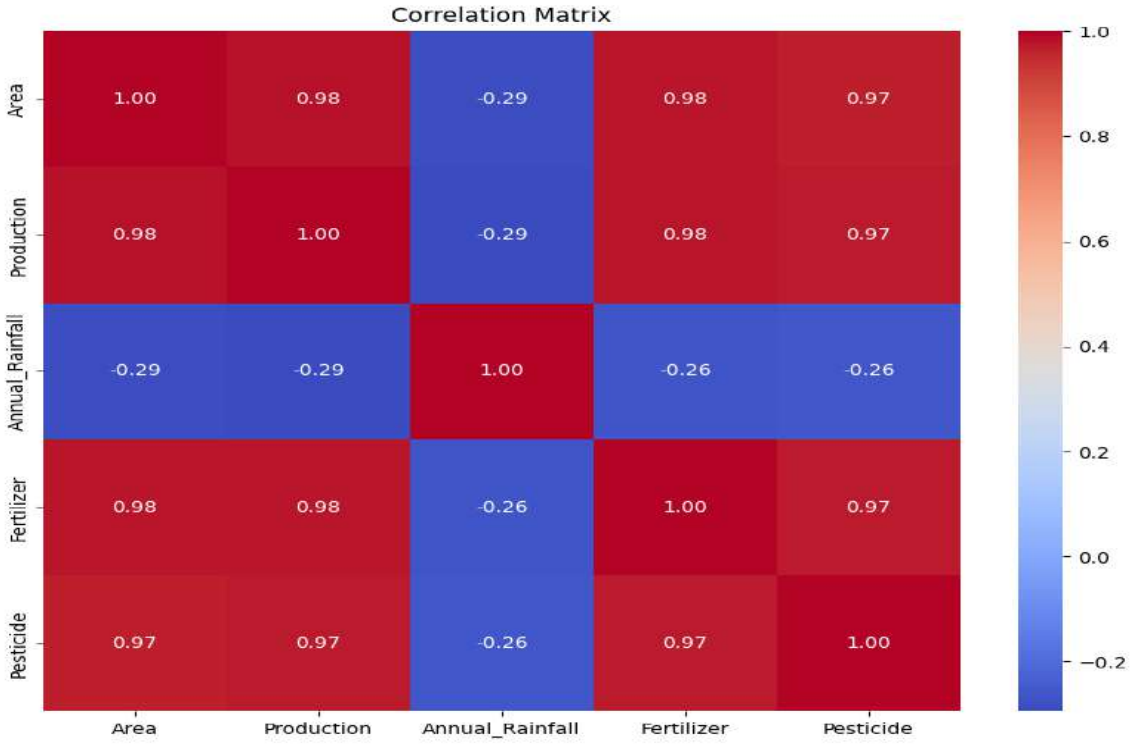
serves as a visual guide for identifying patterns and making informed decisions during the model-building process.



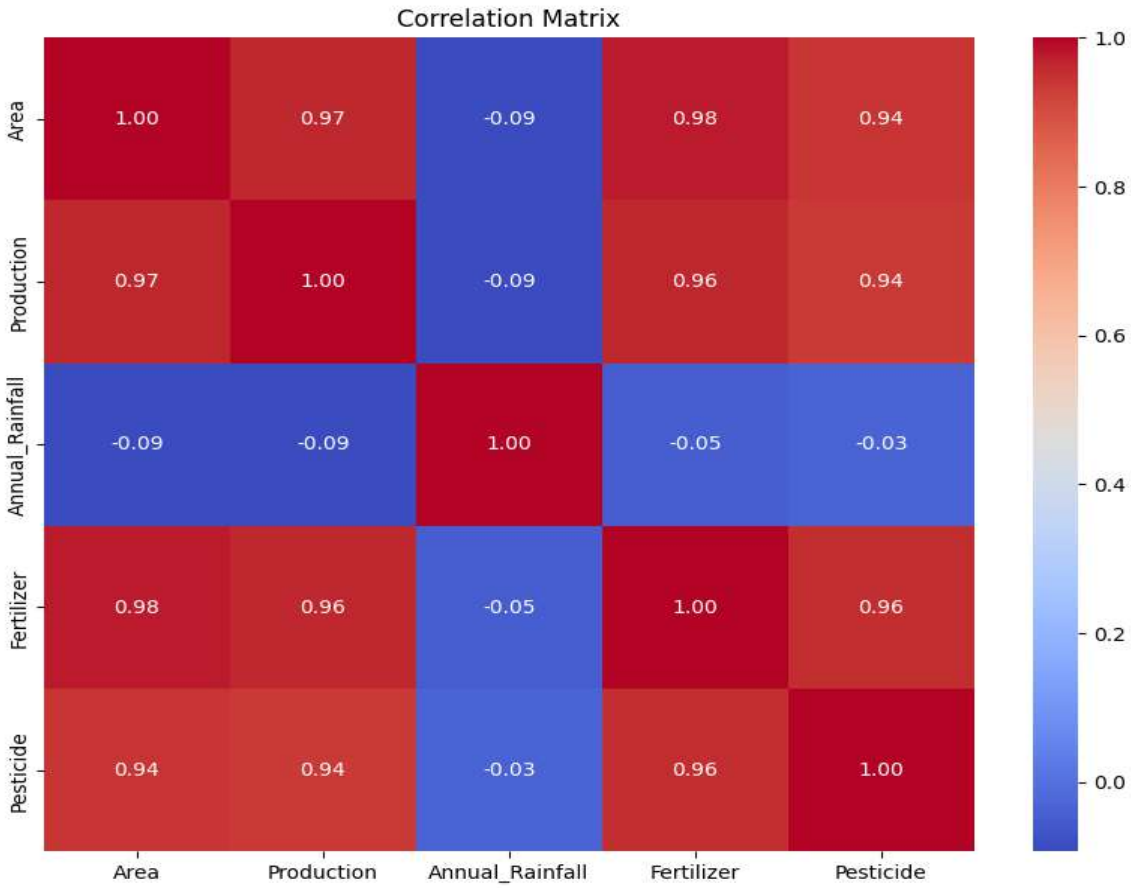
(a) Maize



(b) Rice



(c) Sugarcane



(d) Rice

Figure 27 Correlation matrix of proposed model (a)Maize ,(b)Rice ,(c)Sugarcane, (d)Potato

Chapter 5
Conclusion

5.1 Discussion

In the realm of agriculture, the timely prediction of crop yields holds a pivotal role in determining the success and sustainability of farming practices. This article delves into a groundbreaking research initiative that aims to address the urgent issue of early crop yield prediction in Bangladesh. By leveraging a diverse range of machine learning algorithms, this study presents a solution that has the potential to significantly mitigate financial losses for farmers, revolutionizing traditional farming practices.

5.2 Empowering Farmers through Machine Learning

The primary objective of this research is to empower farmers with accurate early crop yield predictions, enabling them to make informed decisions regarding resource allocation for pesticides, fertilizer, and irrigation. The significance lies in the strategic approach adopted, which minimizes the risk of financial losses resulting from unforeseen changes in crop growth. This not only enhances the sustainability of agricultural activities but also improves the overall profitability for farmers.

5.3 Diverse Machine Learning Algorithms

The research incorporates a myriad of machine learning algorithms, including K-Nearest Neighbors, Random Forest, Gradient Boosting, AdaBoost, Decision Tree, and XGBoost. This diverse range allows for a comprehensive evaluation of predictive models, ensuring the identification of the most effective algorithms for crop yield prediction. The study recognizes K-Nearest Neighbors, Random Forest, and Gradient Boosting as the most promising models in this context.

5.4 Ensemble Learning Techniques

Building on the success of individual machine learning models, the research delves into ensemble learning techniques to enhance predictive accuracy further. This involves the application of Voting Regression and advanced methodologies like stacking. The collaborative intelligence demonstrated through these techniques showcases the potential for improving predictive accuracy and reliability.

5.5 Exceptional Results and Model Performance

The research highlights exceptional results achieved for different crops using various machine learning models. For maize crop yield prediction, the stacking model using Random Forest as the final estimator achieved an impressive R2 of 0.99 and MAE of 0.04. Similarly, rice crop yield prediction using the voting model attained an R2 of 0.99 and MAE of 0.06. Potato crop yield prediction achieved remarkable results with an R2 of 0.99 and MAE of 0.04 using a simple machine learning model. Furthermore, sugarcane crop yield prediction reached an R2 of 0.99 and MAE of 0.36 through a stacking model using K-Nearest Neighbors as the final estimator.

5.6 Future Directions for Improvement

To enhance the accuracy of early crop yield prediction further, the study suggests avenues for future research. These include refining feature engineering, exploring additional relevant variables, and incorporating more advanced machine learning techniques. Additionally, expanding the scope to consider the impact of climate change and its dynamic effects on crop yield could provide a more comprehensive understanding of the factors influencing agricultural outcomes.

5.7 Transformative Impact on Agriculture

In conclusion, this research not only addresses a critical problem in agriculture but also provides a tangible solution that empowers farmers with the knowledge needed for effective decision-making. The success of the methodology, particularly with the Stacking model employing as the Final Estimator, offers a promising avenue for transforming early crop yield prediction. This has the potential to positively impact the livelihoods of farmers not only in Bangladesh but also in other regions facing similar challenges in agriculture. The integration of machine learning into crop yield prediction emerges as a revolutionary step towards sustainable and profitable farming practices.

References

1. Bali, N. and A. Singla, *Emerging Trends in Machine Learning to Predict Crop Yield and Study Its Influential Factors: A Survey.* Archives of Computational Methods in Engineering, 2021. **29**(1): p. 95-112.
2. McGuire, S.J.A.i.N., *FAO, IFAD, and WFP. The state of food insecurity in the world 2015: meeting the 2015 international hunger targets: taking stock of uneven progress. Rome: FAO, 2015.* 2015. **6**(5): p. 623-624.
3. Lal, R.J.F. and E. Security, *Feeding 11 billion on 0.5 billion hectare of area under cereal crops.* 2016. **5**(4): p. 239-251.
4. Saw, T. and P.H.J.I.J.C. Myint, *Swarm intelligence based feature selection for high dimensional classification: a literature survey.* 2019. **33**(1): p. 69-83.
5. Chen, S. and N.M.J.a.p.a. Luc, *RRMSE Voting* Regression*: A weighting function based improvement to ensemble regression.* 2022.
6. Desa, U., *Transforming our world: The 2030 agenda for sustainable development.* 2016.
7. Chlingaryan, A., et al., *Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review.* 2018. **151**: p. 61-69.
8. Deng, H. and G.J.P.r. Runger, *Gene selection with guided regularized random forest.* 2013. **46**(12): p. 3483-3489.
9. Gopal, P.M. and R.J.A.E.i.A. Bhargavi, *Optimum feature subset for optimizing crop yield prediction using filter and wrapper approaches.* 2019. **35**(1): p. 9-14.
10. Elavarasan, D. and P.D.J.I.a. Vincent, *Crop yield prediction using deep reinforcement learning model for sustainable agrarian applications.* 2020. **8**: p. 86886-86901.
11. Li, B., J. Lecourt, and G.J.P. Bishop, *Advances in non-destructive early assessment of fruit ripeness towards defining optimal time of harvest and yield prediction—A review.* 2018. **7**(1): p. 3.
12. Alibabaei, K., P.D. Gaspar, and T.M.J.E. Lima, *Crop yield estimation using deep learning based on climate big data and irrigation scheduling.* 2021. **14**(11): p. 3004.
13. Shanmuganathan, S., P. Sallis, and A. Narayanan. *Data Mining Techniques for Modelling the Influence of Daily Extreme Weather Conditions on Grapevine, Wine*

Quality and Perennial Crop Yield. in *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*. 2010. IEEE.

14. Sarijaloo, F.B., et al., *Yield performance estimation of corn hybrids using machine learning algorithms.* 2021. **5**: p. 82-89.
15. You, J., et al. *Deep gaussian process for crop yield prediction based on remote sensing data*. in *Proceedings of the AAAI conference on artificial intelligence*. 2017.
16. Shahhosseini, M., et al., *Coupling machine learning and crop modeling improves crop yield prediction in the US Corn Belt.* 2021. **11**(1): p. 1606.
17. Jaikla, R., S. Auephanwiriyakul, and A. Jintrawet. *Rice yield prediction using a support vector regression method*. in *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. 2008. IEEE.
18. Dey, U.K., A.H. Masud, and M.N. Uddin. *Rice yield prediction model using data mining*. in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. 2017. IEEE.
19. Gupta, S., et al., *Machine Learning- and Feature Selection-Enabled Framework for Accurate Crop Yield Prediction.* Journal of Food Quality, 2022. **2022**: p. 1-7.
20. Agarwal, S. and S. Tarar, *A Hybrid Approach for Crop Yield Prediction Using Machine Learning and Deep Learning Algorithms.* Journal of Physics: Conference Series, 2021. **1714**(1).
21. Mamun, S., et al., *JuteBangla: A Comparative Study on Jute Yield Prediction using Supervised Machine Learning Approach based on Bangladesh Perspective*, in *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2022. p. 1-5.
22. Hasan, M., et al., *Ensemble machine learning-based recommendation system for effective prediction of suitable agricultural crop cultivation.* Front Plant Sci, 2023. **14**: p. 1234555.
23. M, C. and R.K. Dhanraj, *Ensemble Deep Learning Algorithm for Forecasting of Rice Crop Yield based on Soil Nutrition Levels.* ICST Transactions on Scalable Information Systems, 2023.
24. Kamath, P., et al., *Crop yield forecasting using data mining.* 2021. **2**(2): p. 402-407.
25. Malik, P., S. Sengupta, and J.S. Jadon. *Comparative analysis of soil properties to predict fertility and crop yield using machine learning algorithms*. in *2021 11th*

International Conference on Cloud Computing, Data Science & Engineering (Confluence). 2021. IEEE.

26. Iniyan, S. and R.J.W.P.C. Jebakumar, *Mutual information feature selection (MIFS) based crop yield prediction on corn and soybean crops using multilayer stacked ensemble regression (MSER).* 2022. **126**(3): p. 1935-1964.
27. Nazir, A., et al., *Estimation and forecasting of rice yield using phenology-based algorithm and linear regression model on Sentinel-II satellite data.* 2021. **11**(10): p. 1026.
28. Ahamed, A.M.S., et al. *Applying data mining techniques to predict annual yield of major crops and recommend planting different crops in different districts in Bangladesh*. in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2015. IEEE.
29. BR, A., *A Literature Study on Application of Data Mining Tools for Rice Yield Prediction.*
30. Siddique, T., et al. *Automated farming prediction*. in *2017 Intelligent systems conference (IntelliSys)*. 2017. IEEE.
31. Bali, N. and A.J.A.A.I. Singla, *Deep learning based wheat crop yield prediction model in Punjab region of North India.* 2021. **35**(15): p. 1304-1328.
32. Bali, N. and A. Singla, *Deep Learning Based Wheat Crop Yield Prediction Model in Punjab Region of North India.* Applied Artificial Intelligence, 2021. **35**(15): p. 1304-1328.
33. Gong, L., et al., *A Novel Model Fusion Approach for Greenhouse Crop Yield Prediction.* Horticulturae, 2022. **9**(1).
34. Oikonomidis, A., C. Catal, and A. Kassahun, *Hybrid Deep Learning-based Models for Crop Yield Prediction.* Applied Artificial Intelligence, 2022. **36**(1).
35. Zhan, C., et al., *A hybrid machine learning framework for forecasting house price.* 2023. **233**: p. 120981.
36. Hara, P., M. Piekutowska, and G. Niedbała, *Selection of Independent Variables for Crop Yield Prediction Using Artificial Neural Network Models with Remote Sensing Data.* Land, 2021. **10**(6).
37. Batool, D., et al., *A Hybrid Approach to Tea Crop Yield Prediction Using Simulation Models and Machine Learning.* Plants (Basel), 2022. **11**(15).

38. Liu, J., et al., *An Investigation of a Multidimensional CNN Combined with an Attention Mechanism Model to Resolve Small-Sample Problems in Hyperspectral Image Classification.* Remote Sensing, 2022. **14**(3).
39. Yli-Heikkilä, M., et al., *Scalable Crop Yield Prediction with Sentinel-2 Time Series and Temporal Convolutional Network.* Remote Sensing, 2022. **14**(17).
40. Alam, K.F. and T. Ahamed, *Climate-Adaptive Potential Crops Selection in Vulnerable Agricultural Lands Adjacent to the Jamuna River Basin of Bangladesh Using Remote Sensing and a Fuzzy Expert System.* Remote Sensing, 2023. **15**(8).
41. Ren, Y., L. Zhang, and P.N. Suganthan, *Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article].* IEEE Computational Intelligence Magazine, 2016. **11**(1): p. 41-53.*<10.1109@IDEA49133.2020.9170675.pdf>*.
42. Jacobs, R.J.N.C., *Adaptive mixture of local experts.* 1993. **3**: p. 337-345.
43. Smyth, P. and D.J.A.i.n.i.p.s. Wolpert, *Stacked density estimation.* 1997. **10**.
44. Yulisa, A., et al., *Enhancement of voting* Regression *algorithm on predicting total ammonia nitrogen concentration in fish waste anaerobiosis.* 2023. **14**(2): p. 461-478.
45. Rizwan, A., et al., *Enhanced optimization-based voting classifier and chained multi-objective* Regression *for effective groundwater resource management.* 2021. **9**: p. 168329-168341.

APPENDIX

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.feature_selection import
RFE
from sklearn.ensemble import
RandomForestRegressor
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.linear_model import
LinearRegression, Ridge, Lasso
from sklearn.tree import
DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import
KNeighborsRegressor
from sklearn.ensemble import
RandomForestRegressor,
GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import
GridSearchCV
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
import numpy as np
data =
pd.read_csv('/content/rice_updated10k.c
sv')
data.head()
```

```
    print(f"Mean Absolute Error (MAE)
for {ensemble_name}:
{ensemble_mae}")
    print(f"Mean Squared Error (MSE)
for {ensemble_name}:
{ensemble_mse}")
    print(f"Root Mean Squared Error
(RMSE) for {ensemble_name}:
{ensemble_rmse}")
    # Store metrics in lists

ensemble_names.append(ensemble_nam
e)

ensemble_r2_scores.append(ensemble_r
2)

ensemble_mae_scores.append(ensemble
_mae)

ensemble_mse_scores.append(ensemble
_mse)

ensemble_rmse_scores.append(ensembl
e_rmse)
    # Save the fitted ensemble model
    model_filename =
f"{ensemble_name.lower().replace(' ',
'_')}_model.pkl"
    joblib.dump(ensemble_model,
model_filename)
```

```
data.to_csv('/content/rice.csv',
index=False)
data.shape
(10000, 9)
data.isnull().sum()
Crop              0
Season            0
State             0
Area              0
Production        0
Annual_Rainfall   0
Fertilizer        0
Pesticide         0
Yield             0
dtype: int64
data.select_dtypes('number').columns
Index(['Area', 'Production',
'Annual_Rainfall', 'Fertilizer', 'Pesticide',
       'Yield'],
      dtype='object')
categorical =
data.select_dtypes(exclude='number').co
lumns
numerical =
data.select_dtypes('number').columns
from sklearn.model_selection import
train_test_split
X = data.drop(['Yield'], axis=1)
y = data.Yield
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
from sklearn.preprocessing import
OneHotEncoder, MinMaxScaler
ensemble_models_saved.append(model_
filename)
# Display only ensemble model names
and their results
for name, r2, mae, mse, rmse in
zip(ensemble_names,
ensemble_r2_scores,
ensemble_mae_scores,
ensemble_mse_scores,
ensemble_rmse_scores):
    print(f"\nEnsemble Model: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")
print("\nEnsemble Models Saved:")
for filename in ensemble_models_saved:
    print(filename)
param_grids = {
    'Linear Regression': {},
    'Decision Tree': {'max_depth': [None,
10, 20], 'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]},
    'Support Vector Machine': {'C': [0.1,
1.0, 10.0], 'kernel': ['linear', 'rbf']},
    'K-Nearest Neighbors':
{'n_neighbors': [3, 5, 7], 'weights':
['uniform', 'distance']},
    'Random Forest': {'n_estimators': [50,
100, 200], 'max_depth': [None, 10, 20],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]},
    'Gradient Boosting': {'n_estimators':
[50, 100, 200], 'learning_rate': [0.01, 0.1,
0.2], 'max_depth': [3, 5, 7]},
```

```
from sklearn.compose import
ColumnTransformer
transform = [('OHE', OneHotEncoder(),
['Crop', 'Season', 'State']),
 ('MMS', MinMaxScaler(), ['Area',
'Production', 'Annual_Rainfall',
'Fertilizer',
      'Pesticide'])]
ct = ColumnTransformer(transform)
pro_X_train = ct.fit_transform(X_train)
pro_X_test = ct.transform(X_test)
!pip install xgboost
!pip install catboost
Requirement already satisfied: xgboost
in /usr/local/lib/python3.10/dist-
packages (2.0.2)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages
(from xgboost) (1.23.5)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages
(from xgboost) (1.11.4)
Collecting catboost
  Downloading catboost-1.2.2-cp310-
cp310-manylinux2014_x86_64.whl
(98.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━ 98.7/98.7 MB 8.1
MB/s eta 0:00:00
Requirement already satisfied: graphviz
in /usr/local/lib/python3.10/dist-
packages (from catboost) (0.20.1)
```

```
   'XGBoost': {'n_estimators': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2],
'max_depth': [3, 5, 7]},
   'CatBoost': {'iterations': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2],
'depth': [3, 5, 7]},
   'AdaBoost': {'n_estimators': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2]}
}
# Create lists to store the metrics and
models
model_names = []
r2_scores = []
mae_scores = []
mse_scores = []
rmse_scores = []
trained_models = {}

# Create a dictionary to store best
hyperparameters
best_params_dict = {}

# Iterate over models and perform grid
search
for model_name, model in models:
   print(f"\nPerforming Grid Search for
{model_name}...")

   # Create the GridSearchCV object
   grid_search =
GridSearchCV(estimator=model,
param_grid=param_grids[model_name],
scoring='neg_mean_squared_error',
cv=5)

   # Fit the model to the training data
   grid_search.fit(pro_X_train, y_train)

   # Print the best parameters and
corresponding score
   print(f"Best Parameters:
{grid_search.best_params_}")
   print(f"Best Score:
{grid_search.best_score_}")

   # Get the best model
   best_model =
grid_search.best_estimator_

   # Save the best model
```

```
Requirement already satisfied:
matplotlib in
/usr/local/lib/python3.10/dist-packages
(from catboost) (3.7.1)
Requirement already satisfied:
numpy>=1.16.0 in
/usr/local/lib/python3.10/dist-packages
(from catboost) (1.23.5)
Requirement already satisfied:
pandas>=0.24 in
/usr/local/lib/python3.10/dist-packages
(from catboost) (1.5.3)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages
(from catboost) (1.11.4)
Requirement already satisfied: plotly in
/usr/local/lib/python3.10/dist-packages
(from catboost) (5.15.0)
Requirement already satisfied: six in
/usr/local/lib/python3.10/dist-packages
(from catboost) (1.16.0)
Requirement already satisfied: python-
dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages
(from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied:
pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages
(from pandas>=0.24->catboost)
(2023.3.post1)
Requirement already satisfied:
contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages
(from matplotlib->catboost) (1.2.0)
```

```
    model_filename =
f"{model_name}_model.joblib"
    joblib.dump(best_model,
model_filename)
    trained_models[model_name] =
model_filename

    # Save the best hyperparameters
    best_params_dict[model_name] =
grid_search.best_params_

    # Evaluate the model on the test set
    test_predictions =
best_model.predict(pro_X_test)

    # Calculate metrics
    r2 = r2_score(y_test, test_predictions)
    mae = mean_absolute_error(y_test,
test_predictions)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)

    # Print metrics
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")

    # Store metrics in lists
    model_names.append(model_name)
    r2_scores.append(r2)
    mae_scores.append(mae)
    mse_scores.append(mse)
    rmse_scores.append(rmse)

# Print the best hyperparameters
print("\nBest hyperparameters:")
for model_name, params in
best_params_dict.items():
    print(f"{model_name}: {params}")
Streaming output truncated to the last
5000 lines.
23:     learn: 0.4658162     total:
90.5ms  remaining: 98.1ms
24:     learn: 0.4604736     total:
93.8ms  remaining: 93.8ms
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.2
import joblib
from sklearn.model_selection import GridSearchCV
```

```
25:     learn: 0.4531494        total: 97ms     remaining: 89.5ms
26:     learn: 0.4478457        total: 100ms    remaining: 85.3ms
27:     learn: 0.4414165        total: 103ms    remaining: 81.2ms
28:     learn: 0.4368089        total: 106ms    remaining: 77.1ms
29:     learn: 0.4301741        total: 110ms    remaining: 73.3ms
30:     learn: 0.4250481        total: 113ms    remaining: 69.4ms
31:     learn: 0.4203137        total: 116ms    remaining: 65.4ms
32:     learn: 0.4158133        total: 119ms    remaining: 61.5ms
33:     learn: 0.4115894        total: 123ms    remaining: 57.7ms
34:     learn: 0.4041844        total: 126ms    remaining: 53.9ms
35:     learn: 0.4006531        total: 129ms    remaining: 50.2ms
36:     learn: 0.3965249        total: 132ms    remaining: 46.5ms
37:     learn: 0.3928295        total: 136ms    remaining: 42.9ms
38:     learn: 0.3893034        total: 139ms    remaining: 39.2ms
39:     learn: 0.3849685        total: 142ms    remaining: 35.6ms
40:     learn: 0.3804285        total: 146ms    remaining: 32ms
41:     learn: 0.3760346        total: 149ms    remaining: 28.4ms
42:     learn: 0.3711667        total: 152ms    remaining: 24.8ms
43:     learn: 0.3682509        total: 156ms    remaining: 21.2ms
44:     learn: 0.3647088        total: 159ms    remaining: 17.7ms
45:     learn: 0.3608741        total: 162ms    remaining: 14.1ms
46:     learn: 0.3553525        total: 166ms    remaining: 10.6ms
...
Gradient Boosting: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200}
XGBoost: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 200}
```

```
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
import numpy as np
from sklearn.linear_model import
LinearRegression
from sklearn.tree import
DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import
KNeighborsRegressor
from sklearn.ensemble import
RandomForestRegressor,
GradientBoostingRegressor,
AdaBoostRegressor
from xgboost import XGBRegressor
from catboost import
CatBoostRegressor
# Your data preparation steps here
(pro_X_train, y_train, pro_X_test,
y_test)
# Define a list of models
models = [
    #('Linear Regression',
LinearRegression()),
    ('Decision Tree',
DecisionTreeRegressor()),
    #('Support Vector Machine', SVR()),
    ('K-Nearest Neighbors',
KNeighborsRegressor()),
    ('Random Forest',
RandomForestRegressor()),
```

```
CatBoost: {'depth': 7, 'iterations': 200,
'learning_rate': 0.2}
AdaBoost: {'learning_rate': 0.2,
'n_estimators': 100}
Output is truncated. View as a scrollable
element or open in a text editor. Adjust
cell output settings...
# Display only model names and their
results
for name, r2, mae, mse, rmse in
zip(model_names, r2_scores,
mae_scores, mse_scores, rmse_scores):
    print(f"\nModel: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")
Model: Decision Tree
R-squared (R²): 0.9594702295237152
Mean Absolute Error (MAE):
0.039642904797847975
Mean Squared Error (MSE):
0.02689080900802059
Root Mean Squared Error (RMSE):
0.1639841730412438

Model: K-Nearest Neighbors
R-squared (R²): 0.9362769314331094
Mean Absolute Error (MAE):
0.06346111058290689
Mean Squared Error (MSE):
0.04227916531725513
Root Mean Squared Error (RMSE):
0.2056189809265067

Model: Random Forest
R-squared (R²): 0.9806466220980419
Mean Absolute Error (MAE):
0.03110652093265329
Mean Squared Error (MSE):
0.012840634987709106
Root Mean Squared Error (RMSE):
0.11331652566024562

Model: Gradient Boosting
R-squared (R²): 0.9782830865252999
```

```
    ('Gradient Boosting',
GradientBoostingRegressor()),
    ('XGBoost', XGBRegressor()),
    ('CatBoost', CatBoostRegressor()),
    ('AdaBoost', AdaBoostRegressor())

# Define hyperparameters and their
possible values for each model
param_grids = {
    'Linear Regression': {},
    'Decision Tree': {'max_depth': [None,
10, 20], 'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]},
    'Support Vector Machine': {'C': [0.1,
1.0, 10.0], 'kernel': ['linear', 'rbf']},
    'K-Nearest Neighbors':
{'n_neighbors': [3, 5, 7], 'weights':
['uniform', 'distance']},
    'Random Forest': {'n_estimators': [50,
100, 200], 'max_depth': [None, 10, 20],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]},
    'Gradient Boosting': {'n_estimators':
[50, 100, 200], 'learning_rate': [0.01,
0.1, 0.2], 'max_depth': [3, 5, 7]},
    'XGBoost': {'n_estimators': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2],
'max_depth': [3, 5, 7]},
    'CatBoost': {'iterations': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2],
'depth': [3, 5, 7]},
    'AdaBoost': {'n_estimators': [50, 100,
200], 'learning_rate': [0.01, 0.1, 0.2]}
```

```
Mean Absolute Error (MAE):
0.03886245984947554
Mean Squared Error (MSE):
0.014408800386214268
Root Mean Squared Error (RMSE):
0.12003666267526046
...
R-squared (R²): 0.3728827748783846
Mean Absolute Error (MAE):
0.5158600469363658
Mean Squared Error (MSE):
0.4160815452002779
Root Mean Squared Error (RMSE):
0.645043831999251
Output is truncated. View as a scrollable
element or open in a text editor. Adjust
cell output settings...
from sklearn.ensemble import
VotingRegressor, StackingRegressor,
BaggingRegressor
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
from sklearn.linear_model import
LinearRegression
from sklearn.ensemble import
RandomForestRegressor,
GradientBoostingRegressor,
BaggingRegressor
import joblib
import numpy as np

# Load individual models
xg_model =
joblib.load("/content/XGBoost_model.jo
blib")
rf_model =
joblib.load("/content/Random
Forest_model.joblib")
gb_model =
joblib.load("/content/Gradient
Boosting_model.joblib")

# Load your test data (replace this line
with your actual test data loading)
# pro_X_test = ...

# Fit individual models if not already
fitted
xg_model.fit(pro_X_train, y_train)
```

```
}
# Create lists to store the metrics and
models
model_names = []
r2_scores = []
mae_scores = []
mse_scores = []
rmse_scores = []
trained_models = {}
# Create a dictionary to store best
hyperparameters
best_params_dict = {}
# Iterate over models and perform grid
search
for model_name, model in models:
    print(f"\nPerforming Grid Search for
{model_name}...")
    # Create the GridSearchCV object
    grid_search =
GridSearchCV(estimator=model,
param_grid=param_grids[model_name],
scoring='neg_mean_squared_error',
cv=5)
    # Fit the model to the training data
    grid_search.fit(pro_X_train, y_train)
    # Print the best parameters and
corresponding score
    print(f"Best Parameters:
{grid_search.best_params_}")
    print(f"Best Score:
{grid_search.best_score_}")
    # Get the best model
rf_model.fit(pro_X_train, y_train)
gb_model.fit(pro_X_train, y_train)

models = {'Xg': xg_model, 'Random
Forest': rf_model, 'Gradient Boosting':
gb_model}

ensemble_models = [
    ('Voting',
VotingRegressor(estimators=[('XGB',
xg_model), ('rf', rf_model), ('gb',
gb_model)])),
    ('Stacking',
StackingRegressor(estimators=[('XGB',
xg_model), ('rf', rf_model), ('gb',
gb_model)],

final_estimator=LinearRegression())),
    #('Bagging',
BaggingRegressor(base_estimator=Line
arRegression(), n_estimators=10))
]

# Create lists to store the metrics
ensemble_names = []
ensemble_r2_scores = []
ensemble_mae_scores = []
ensemble_mse_scores = []
ensemble_rmse_scores = []
ensemble_models_saved = []

# Evaluate ensemble models
for ensemble_name, ensemble_model in
ensemble_models:
    print(f"\nEvaluating
{ensemble_name}...")

    # Fit the ensemble model
    ensemble_model.fit(pro_X_train,
y_train)

    # Predict on the test set using the
ensemble model
    ensemble_predictions =
ensemble_model.predict(pro_X_test)

    # Calculate metrics
    ensemble_r2 = r2_score(y_test,
ensemble_predictions)
```

```
    best_model =
grid_search.best_estimator_
    # Save the best model
    model_filename =
f"{model_name}_model.joblib"
    joblib.dump(best_model,
model_filename)
    trained_models[model_name] =
model_filename
    # Save the best hyperparameters
    best_params_dict[model_name] =
grid_search.best_params_
    # Evaluate the model on the test set
    test_predictions =
best_model.predict(pro_X_test)
    # Calculate metrics
    r2 = r2_score(y_test, test_predictions)
    mae = mean_absolute_error(y_test,
test_predictions)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    # Print metrics
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")
    # Store metrics in lists
    model_names.append(model_name)
    r2_scores.append(r2)
    ensemble_mae =
mean_absolute_error(y_test,
ensemble_predictions)
    ensemble_mse =
mean_squared_error(y_test,
ensemble_predictions)
    ensemble_rmse =
np.sqrt(ensemble_mse)

    # Print metrics
    print(f"R-squared (R²) for
{ensemble_name}: {ensemble_r2}")
    print(f"Mean Absolute Error (MAE)
for {ensemble_name}:
{ensemble_mae}")
    print(f"Mean Squared Error (MSE)
for {ensemble_name}:
{ensemble_mse}")
    print(f"Root Mean Squared Error
(RMSE) for {ensemble_name}:
{ensemble_rmse}")

    # Store metrics in lists

ensemble_names.append(ensemble_nam
e)

ensemble_r2_scores.append(ensemble_r
2)

ensemble_mae_scores.append(ensemble
_mae)

ensemble_mse_scores.append(ensemble
_mse)

ensemble_rmse_scores.append(ensembl
e_rmse)

    # Save the fitted ensemble model
    model_filename =
f"{ensemble_name.lower().replace(' ',
'_')}_model.pkl"
    joblib.dump(ensemble_model,
model_filename)

ensemble_models_saved.append(model_
filename)
```

```
    mae_scores.append(mae)
    mse_scores.append(mse)
    rmse_scores.append(rmse)
# Print the best hyperparameters
print("\nBest hyperparameters:")
for model_name, params in
best_params_dict.items():
    print(f"{model_name}: {params}")
```

```
Streaming output truncated to the last
5000 lines.
23:     learn: 0.4658162        total:
90.5ms  remaining: 98.1ms
24:     learn: 0.4604736        total:
93.8ms  remaining: 93.8ms
25:     learn: 0.4531494        total:
97ms    remaining: 89.5ms
26:     learn: 0.4478457        total:
100ms   remaining: 85.3ms
27:     learn: 0.4414165        total:
103ms   remaining: 81.2ms
28:     learn: 0.4368089        total:
106ms   remaining: 77.1ms
29:     learn: 0.4301741        total:
110ms   remaining: 73.3ms
30:     learn: 0.4250481        total:
113ms   remaining: 69.4ms
31:     learn: 0.4203137        total:
116ms   remaining: 65.4ms
32:     learn: 0.4158133        total:
119ms   remaining: 61.5ms
33:     learn: 0.4115894        total:
123ms   remaining: 57.7ms
```

```
# Display only ensemble model names
and their results
for name, r2, mae, mse, rmse in
zip(ensemble_names,
ensemble_r2_scores,
ensemble_mae_scores,
ensemble_mse_scores,
ensemble_rmse_scores):
    print(f"\nEnsemble Model: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")

print("\nEnsemble Models Saved:")
for filename in ensemble_models_saved:
    print(filename)
Evaluating Voting...
R-squared (R²) for Voting:
0.9822404025329537
Mean Absolute Error (MAE) for Voting:
0.032505941360592414
Mean Squared Error (MSE) for Voting:
0.011783188948111766
Root Mean Squared Error (RMSE) for
Voting: 0.10855039819416494

Evaluating Stacking...
R-squared (R²) for Stacking:
0.9824048478790252
Mean Absolute Error (MAE) for
Stacking: 0.03341296440128406
Mean Squared Error (MSE) for
Stacking: 0.011674082275621457
Root Mean Squared Error (RMSE) for
Stacking: 0.10804666711945102

Ensemble Model: Voting
R-squared (R²): 0.9822404025329537
Mean Absolute Error (MAE):
0.032505941360592414
Mean Squared Error (MSE):
0.011783188948111766
Root Mean Squared Error (RMSE):
0.10855039819416494

Ensemble Model: Stacking
```

```
34:	learn: 0.4041844	total:
126ms	remaining: 53.9ms
35:	learn: 0.4006531	total:
129ms	remaining: 50.2ms
36:	learn: 0.3965249	total:
132ms	remaining: 46.5ms
37:	learn: 0.3928295	total:
136ms	remaining: 42.9ms
38:	learn: 0.3893034	total:
139ms	remaining: 39.2ms
39:	learn: 0.3849685	total:
142ms	remaining: 35.6ms
40:	learn: 0.3804285	total:
146ms	remaining: 32ms
41:	learn: 0.3760346	total:
149ms	remaining: 28.4ms
42:	learn: 0.3711667	total:
152ms	remaining: 24.8ms
43:	learn: 0.3682509	total:
156ms	remaining: 21.2ms
44:	learn: 0.3647088	total:
159ms	remaining: 17.7ms
45:	learn: 0.3608741	total:
162ms	remaining: 14.1ms
46:	learn: 0.3553525	total:
166ms	remaining: 10.6ms
...
Gradient Boosting: {'learning_rate': 0.1,
'max_depth': 7, 'n_estimators': 200}
XGBoost: {'learning_rate': 0.2,
'max_depth': 7, 'n_estimators': 200}
CatBoost: {'depth': 7, 'iterations': 200,
'learning_rate': 0.2}
```

```
R-squared (R²): 0.9824048478790252
Mean Absolute Error (MAE):
0.03341296440128406
Mean Squared Error (MSE):
0.011674082275621457
Root Mean Squared Error (RMSE):
0.10804666711945102

Ensemble Models Saved:
voting_model.pkl
stacking_model.pkl
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import r2_score

# Load the saved models

tree_model =
joblib.load("/content/voting_model.pkl")
# knn_model =
joblib.load("/content/stacking_model.pkl
")
# rf_model =
joblib.load("/content/Random
Forest_model.joblib")
# gb_model =
joblib.load("/content/Gradient
Boosting_model.joblib")
# ada_model =
joblib.load("/content/AdaBoost_model.j
oblib")
# cat_model =
joblib.load("/content/CatBoost_model.jo
blib")
# linear_model =
joblib.load("/content/Linear
Regression_model.joblib")
# svm_model =
joblib.load("/content/Support Vector
Machine_model.joblib")
# xgb_model =
joblib.load("/content/XGBoost_model.jo
blib")

# Create a dictionary for loaded models
loaded_models = {
    # 'K-Nearest Neighbors': knn_model,
    # 'Random Forest': rf_model,
    # 'Gradient Boosting': gb_model,
```

AdaBoost: {'learning_rate': 0.2, 'n_estimators': 100}

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Display only model names and their results
for name, r2, mae, mse, rmse in zip(model_names, r2_scores, mae_scores, mse_scores, rmse_scores):
    print(f"\nModel: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"Root Mean Squared Error (RMSE): {rmse}")
```

Model: Decision Tree
R-squared (R²): 0.9594702295237152
Mean Absolute Error (MAE): 0.039642904797847975
Mean Squared Error (MSE): 0.02689080900802059
Root Mean Squared Error (RMSE): 0.1639841730412438
Model: K-Nearest Neighbors
R-squared (R²): 0.9362769314331094
Mean Absolute Error (MAE): 0.06346111058290689
Mean Squared Error (MSE): 0.04227916531725513

```
    # 'AdaBoost': ada_model,
    # 'CatBoost': cat_model,
    # 'Linear Regression': linear_model,
    # 'Support Vector Machine': svm_model,
    # 'XGBoost': xgb_model,
    'Voting Regressor': tree_model,
}

# Evaluate and store R-squared scores for each model
r2_scores = []
model_names = list(loaded_models.keys())

for model_name, loaded_model in loaded_models.items():
    test_predictions = loaded_model.predict(pro_X_test)
    r2 = r2_score(y_test, test_predictions)
    r2_scores.append(r2)

# Create a bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, r2_scores, color='blue')
plt.xlabel('Models')
plt.ylabel('R-squared (R²) Score')
# plt.title('R-squared Scores for Different Models')
plt.ylim(0, 1.1)  # Set the y-axis limit based on your R-squared range (0 to 1)
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility

# Add R-squared values on top of the bars
for bar, r2_score in zip(bars, r2_scores):
    plt.text(bar.get_x() + bar.get_width() / 2 - 0.1, bar.get_height() + 0.02, f'{r2_score:.2f}', ha='center', color='black', fontsize=10)

# Adjust layout to prevent overlapping
plt.tight_layout()

plt.show()
import joblib
import matplotlib.pyplot as plt
import numpy as np
```

Root Mean Squared Error (RMSE):
0.2056189809265067
Model: Random Forest
R-squared (R^2): 0.9806466220980419
Mean Absolute Error (MAE):
0.03110652093265329
Mean Squared Error (MSE):
0.012840634987709106
Root Mean Squared Error (RMSE):
0.11331652566024562
Model: Gradient Boosting
R-squared (R^2): 0.9782830865252999
Mean Absolute Error (MAE):
0.03886245984947554
Mean Squared Error (MSE):
0.014408800386214268
Root Mean Squared Error (RMSE):
0.12003666267526046
R-squared (R^2): 0.3728827748783846
Mean Absolute Error (MAE):
0.5158600469363658
Mean Squared Error (MSE):
0.4160815452002779
Root Mean Squared Error (RMSE):
0.645043831999251
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
from sklearn.ensemble import VotingRegressor, StackingRegressor, BaggingRegressor
from sklearn.metrics import mean_absolute_error

# Load the saved models

tree_model = joblib.load("/content/Decision Tree_model.joblib")
knn_model = joblib.load("/content/K-Nearest Neighbors_model.joblib")
rf_model = joblib.load("/content/Random Forest_model.joblib")
gb_model = joblib.load("/content/Gradient Boosting_model.joblib")
ada_model = joblib.load("/content/AdaBoost_model.joblib")
cat_model = joblib.load("/content/CatBoost_model.joblib")

#linear_model = joblib.load("/content/Linear Regression_model.joblib")
#svm_model = joblib.load("/content/Support Vector Machine_model.joblib")
xgb_model = joblib.load("/content/XGBoost_model.joblib")

# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine': svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}

# Evaluate and store MAE scores for each model
mae_scores = []
```

```
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
from sklearn.linear_model import
LinearRegression
from sklearn.ensemble import
RandomForestRegressor,
GradientBoostingRegressor,
BaggingRegressor
import joblib
import numpy as np
# Load individual models
xg_model =
joblib.load("/content/XGBoost_model.jo
blib")
rf_model =
joblib.load("/content/Random
Forest_model.joblib")
gb_model =
joblib.load("/content/Gradient
Boosting_model.joblib")
# Load your test data (replace this line
with your actual test data loading)
# pro_X_test = ...
# Fit individual models if not already
fitted
xg_model.fit(pro_X_train, y_train)
rf_model.fit(pro_X_train, y_train)
gb_model.fit(pro_X_train, y_train)
models = {'Xg': xg_model, 'Random
Forest': rf_model, 'Gradient Boosting':
gb_model}
ensemble_models = [
model_names =
list(loaded_models.keys())

for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mae = mean_absolute_error(y_test,
test_predictions)
    mae_scores.append(mae)

# Create a bar chart for MAE
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names,
mae_scores, color='green')
plt.xlabel('Models')
plt.ylabel('Mean Absolute Error (MAE)')
plt.ylim(0, max(mae_scores) + 10)  #
Set the y-axis limit
plt.xticks(rotation=45, ha='right')
# plt.title('Mean Absolute Error (MAE)
for Different Models')

# Add MAE values on top of the bars
with improved placement
for bar, mae_score in zip(bars,
mae_scores):
    plt.text(bar.get_x() + bar.get_width() /
2, bar.get_height() + 1,
f'{mae_score:.2f}', ha='center',
color='black', fontsize=10)

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error

# Load the saved models

tree_model =
joblib.load("/content/Decision
Tree_model.joblib")
knn_model = joblib.load("/content/K-
Nearest Neighbors_model.joblib")
```

```
    ('Voting',
VotingRegressor(estimators=[('XGB',
xg_model), ('rf', rf_model), ('gb',
gb_model)])),
    ('Stacking',
StackingRegressor(estimators=[('XGB',
xg_model), ('rf', rf_model), ('gb',
gb_model)],
final_estimator=LinearRegression())),
    #('Bagging',
BaggingRegressor(base_estimator=Line
arRegression(), n_estimators=10))
]
# Create lists to store the metrics
ensemble_names = []
ensemble_r2_scores = []
ensemble_mae_scores = []
ensemble_mse_scores = []
ensemble_rmse_scores = []
ensemble_models_saved = []
# Evaluate ensemble models
for ensemble_name, ensemble_model in
ensemble_models:
    print(f"\nEvaluating
{ensemble_name}...")
    # Fit the ensemble model
    ensemble_model.fit(pro_X_train,
y_train)
    # Predict on the test set using the
ensemble model
    ensemble_predictions =
ensemble_model.predict(pro_X_test)
    # Calculate metrics
```

```
rf_model =
joblib.load("/content/Random
Forest_model.joblib")
gb_model =
joblib.load("/content/Gradient
Boosting_model.joblib")
ada_model =
joblib.load("/content/AdaBoost_model.j
oblib")
cat_model =
joblib.load("/content/CatBoost_model.jo
blib")
#linear_model =
joblib.load("/content/Linear
Regression_model.joblib")
#svm_model =
joblib.load("/content/Support Vector
Machine_model.joblib")
xgb_model =
joblib.load("/content/XGBoost_model.jo
blib")

# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}

# Evaluate and store RMSE scores for
each model
rmse_scores = []
model_names =
list(loaded_models.keys())

for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    rmse_scores.append(rmse)
```

```
    ensemble_r2 = r2_score(y_test,
ensemble_predictions)
    ensemble_mae =
mean_absolute_error(y_test,
ensemble_predictions)
    ensemble_mse =
mean_squared_error(y_test,
ensemble_predictions)
    ensemble_rmse =
np.sqrt(ensemble_mse)
    # Print metrics
    print(f"R-squared (R²) for
{ensemble_name}: {ensemble_r2}")
from sklearn.ensemble import
VotingRegressor, StackingRegressor,
BaggingRegressor,
RandomForestRegressor
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
from sklearn.linear_model import
LinearRegression
from sklearn.ensemble import
GradientBoostingRegressor,
AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import
KNeighborsRegressor
from catboost import
CatBoostRegressor
from xgboost import XGBRegressor
import joblib
import numpy as np
```

```
# Create a bar chart for RMSE
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names,
rmse_scores, color='green')
plt.xlabel('Models')
plt.ylabel('Root Mean Squared Error
(RMSE)')
plt.ylim(0, max(rmse_scores) + 100)  #
Set the y-axis limit
plt.xticks(rotation=45, ha='right')

# Add RMSE values on top of the bars
with improved placement
for bar, rmse_score in zip(bars,
rmse_scores):
    plt.text(bar.get_x() + bar.get_width() /
2, bar.get_height() + 5,
f'{rmse_score:.2f}', ha='center',
color='black', fontsize=10)

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()
from sklearn.ensemble import
VotingRegressor, StackingRegressor,
BaggingRegressor,
RandomForestRegressor
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
from sklearn.linear_model import
LinearRegression
from sklearn.ensemble import
GradientBoostingRegressor,
AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import
KNeighborsRegressor
from catboost import
CatBoostRegressor
from xgboost import XGBRegressor
import joblib
import numpy as np

# Load individual models
knn_model = joblib.load("/content/K-
Nearest Neighbors_model.joblib")
```

```
# Load individual models
knn_model = joblib.load("/content/K-Nearest Neighbors_model.joblib")
rf_model = joblib.load("/content/Random Forest_model.joblib")
gb_model = joblib.load("/content/Gradient Boosting_model.joblib")
# Load your test data (replace this line with your actual test data loading)
# pro_X_test = ...
# Fit individual models if not already fitted
knn_model.fit(pro_X_train, y_train)
rf_model.fit(pro_X_train, y_train)
gb_model.fit(pro_X_train, y_train)
models = {'KNN': knn_model, 'Random Forest': rf_model, 'Gradient Boosting': gb_model}
final_estimators = [
    ('Decision Tree', DecisionTreeRegressor()),
    ('AdaBoost', AdaBoostRegressor(base_estimator=LinearRegression(), n_estimators=50, learning_rate=1.0)),
    ('Gradient Boosting', GradientBoostingRegressor()),
   # ('SVR', SVR(kernel='linear')),
    ('KNN', KNeighborsRegressor()),
    ('CatBoost', CatBoostRegressor(iterations=100,
```

```
rf_model = joblib.load("/content/Random Forest_model.joblib")
gb_model = joblib.load("/content/Gradient Boosting_model.joblib")

# Load your test data (replace this line with your actual test data loading)
# pro_X_test = ...

# Fit individual models if not already fitted
knn_model.fit(pro_X_train, y_train)
rf_model.fit(pro_X_train, y_train)
gb_model.fit(pro_X_train, y_train)

models = {'KNN': knn_model, 'Random Forest': rf_model, 'Gradient Boosting': gb_model}

final_estimators = [
    ('Decision Tree', DecisionTreeRegressor()),
    ('AdaBoost', AdaBoostRegressor(base_estimator=LinearRegression(), n_estimators=50, learning_rate=1.0)),
    ('Gradient Boosting', GradientBoostingRegressor()),
   # ('SVR', SVR(kernel='linear')),
    ('KNN', KNeighborsRegressor()),
    ('CatBoost', CatBoostRegressor(iterations=100, depth=5, learning_rate=0.1, loss_function='RMSE')),
    #('Linear Regression', LinearRegression()),
    ('Gradient Boosting Base', GradientBoostingRegressor()),
    ('XGBoost', XGBRegressor())
]

# Create lists to store the metrics
ensemble_names = []
ensemble_r2_scores = []
ensemble_mae_scores = []
ensemble_mse_scores = []
ensemble_rmse_scores = []
ensemble_models_saved = []
```

```
depth=5, learning_rate=0.1,
loss_function='RMSE')),
    #('Linear Regression',
LinearRegression()),
    ('Gradient Boosting Base',
GradientBoostingRegressor()),
    ('XGBoost', XGBRegressor())
]

# Create lists to store the metrics
ensemble_names = []
ensemble_r2_scores = []
ensemble_mae_scores = []
ensemble_mse_scores = []
ensemble_rmse_scores = []
ensemble_models_saved = []

# Evaluate ensemble models with
different final estimators
for final_name, final_estimator in
final_estimators:
    stacking_model = StackingRegressor(
        estimators=[('knn', knn_model),
('rf', rf_model), ('gb', gb_model)],
        final_estimator=final_estimator
    )

    ensemble_name = f'Stacking with
{final_name} as Final Estimator'
    print(f"\nEvaluating
{ensemble_name}...")
    # Fit the ensemble model
```

```
# Evaluate ensemble models with
different final estimators
for final_name, final_estimator in
final_estimators:
    stacking_model = StackingRegressor(
        estimators=[('knn', knn_model),
('rf', rf_model), ('gb', gb_model)],
        final_estimator=final_estimator
    )

    ensemble_name = f'Stacking with
{final_name} as Final Estimator'

    print(f"\nEvaluating
{ensemble_name}...")

    # Fit the ensemble model
    stacking_model.fit(pro_X_train,
y_train)

    # Predict on the test set using the
ensemble model
    ensemble_predictions =
stacking_model.predict(pro_X_test)

    # Calculate metrics
    ensemble_r2 = r2_score(y_test,
ensemble_predictions)
    ensemble_mae =
mean_absolute_error(y_test,
ensemble_predictions)
    ensemble_mse =
mean_squared_error(y_test,
ensemble_predictions)
    ensemble_rmse =
np.sqrt(ensemble_mse)

    # Print metrics
    print(f"R-squared (R²) for
{ensemble_name}: {ensemble_r2}")
    print(f"Mean Absolute Error (MAE)
for {ensemble_name}:
{ensemble_mae}")
    print(f"Mean Squared Error (MSE)
for {ensemble_name}:
{ensemble_mse}")
    print(f"Root Mean Squared Error
(RMSE) for {ensemble_name}:
{ensemble_rmse}")
```

```
    stacking_model.fit(pro_X_train,
y_train)
    # Predict on the test set using the
ensemble model
    ensemble_predictions =
stacking_model.predict(pro_X_test)
    # Calculate metrics
    ensemble_r2 = r2_score(y_test,
ensemble_predictions)
    ensemble_mae =
mean_absolute_error(y_test,
ensemble_predictions)
    ensemble_mse =
mean_squared_error(y_test,
ensemble_predictions)
    ensemble_rmse =
np.sqrt(ensemble_mse)
    # Print metrics
    print(f"R-squared (R²) for
{ensemble_name}: {ensemble_r2}")
    print(f"Mean Absolute Error (MAE)
for {ensemble_name}:
{ensemble_mae}")
    print(f"Mean Squared Error (MSE)
for {ensemble_name}:
{ensemble_mse}")
    print(f"Root Mean Squared Error
(RMSE) for {ensemble_name}:
{ensemble_rmse}")
    # Store metrics in lists
ensemble_names.append(ensemble_nam
e)
```

```
Evaluating Stacking with Decision Tree
as Final Estimator...
R-squared (R²) for Stacking with
Decision Tree as Final Estimator:
0.9716235203548694
Mean Absolute Error (MAE) for
Stacking with Decision Tree as Final
Estimator: 0.044416923602740674
Mean Squared Error (MSE) for Stacking
with Decision Tree as Final Estimator:
0.01882730855590912
Root Mean Squared Error (RMSE) for
Stacking with Decision Tree as Final
Estimator: 0.13721263992762883

Evaluating Stacking with AdaBoost as
Final Estimator...
/usr/local/lib/python3.10/dist-
packages/sklearn/ensemble/_base.py:16
6: FutureWarning: `base_estimator` was
renamed to `estimator` in version 1.2
and will be removed in 1.4.
  warnings.warn(
R-squared (R²) for Stacking with
AdaBoost as Final Estimator:
0.8653209238352156
Mean Absolute Error (MAE) for
Stacking with AdaBoost as Final
Estimator: 0.21916615416044463
Mean Squared Error (MSE) for Stacking
with AdaBoost as Final Estimator:
0.08935726188340973
Root Mean Squared Error (RMSE) for
Stacking with AdaBoost as Final
Estimator: 0.2989268503888698

Evaluating Stacking with Gradient
Boosting as Final Estimator...
R-squared (R²) for Stacking with
Gradient Boosting as Final Estimator:
0.980503409841794
Mean Absolute Error (MAE) for
Stacking with Gradient Boosting as
Final Estimator:
0.039022369681632495
Mean Squared Error (MSE) for Stacking
with Gradient Boosting as Final
Estimator: 0.012935653868524685
```

```
ensemble_r2_scores.append(ensemble_r2)
ensemble_mae_scores.append(ensemble_mae)
ensemble_mse_scores.append(ensemble_mse)
ensemble_rmse_scores.append(ensemble_rmse)
    # Save the fitted ensemble model
    model_filename = f"{ensemble_name.lower().replace(' ', '_')}_model.pkl"
    joblib.dump(stacking_model, model_filename)
ensemble_models_saved.append(model_filename)
# Display only ensemble model names and their results
for name, r2, mae, mse, rmse in zip(ensemble_names, ensemble_r2_scores, ensemble_mae_scores, ensemble_mse_scores, ensemble_rmse_scores):
    print(f"\nEnsemble Model: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"Root Mean Squared Error (RMSE): {rmse}")
print("\nEnsemble Models Saved:")
```

```
Root Mean Squared Error (RMSE) for Stacking with Gradient Boosting as Final Estimator: 0.1137350160176042

Evaluating Stacking with KNN as Final Estimator...
R-squared (R²) for Stacking with KNN as Final Estimator: 0.9796092810015844
Mean Absolute Error (MAE) for Stacking with KNN as Final Estimator: 0.03333676571542351
Mean Squared Error (MSE) for Stacking with KNN as Final Estimator: 0.01352889305019503
Root Mean Squared Error (RMSE) for Stacking with KNN as Final Estimator: 0.11631376982195629

Evaluating Stacking with CatBoost as Final Estimator...
0:      learn: 0.7536836        total: 1.66ms   remaining: 164ms
1:      learn: 0.6865123        total: 3.16ms   remaining: 155ms
2:      learn: 0.6264848        total: 4.9ms    remaining: 158ms
3:      learn: 0.5712594        total: 6.52ms   remaining: 156ms
4:      learn: 0.5220927        total: 8.18ms   remaining: 155ms
5:      learn: 0.4790531        total: 9.8ms    remaining: 154ms
6:      learn: 0.4393823        total: 11.6ms   remaining: 154ms
...
stacking_with_knn_as_final_estimator_model.pkl
stacking_with_catboost_as_final_estimator_model.pkl
stacking_with_gradient_boosting_base_as_final_estimator_model.pkl
stacking_with_xgboost_as_final_estimator_model.pkl
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
    # Store metrics in lists
```

```
for filename in
ensemble_models_saved:
    print(filename)
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error
# Load the saved models
tree_model =
joblib.load("/content/stacking_with_deci
sion_tree_as_final_estimator_model.pkl
")
knn_model =
joblib.load("/content/stacking_with_knn
_as_final_estimator_model.pkl")
rf_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
gb_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
ada_model =
joblib.load("/content/stacking_with_ada
boost_as_final_estimator_model.pkl")
cat_model =
joblib.load("/content/stacking_with_catb
oost_as_final_estimator_model.pkl")
#linear_model =
joblib.load("/content/svr_bagging_mode
l.pkl")
```

```
ensemble_names.append(ensemble_nam
e)

ensemble_r2_scores.append(ensemble_r
2)
ensemble_mae_scores.append(ensemble
_mae)
ensemble_mse_scores.append(ensemble
_mse)
ensemble_rmse_scores.append(ensembl
e_rmse)
    # Save the fitted ensemble model
    model_filename =
f"{ensemble_name.lower().replace(' ',
'_')}_model.pkl"
    joblib.dump(stacking_model,
model_filename)
ensemble_models_saved.append(model_
filename)
# Display only ensemble model names
and their results
for name, r2, mae, mse, rmse in
zip(ensemble_names,
ensemble_r2_scores,
ensemble_mae_scores,
ensemble_mse_scores,
ensemble_rmse_scores):
    print(f"\nEnsemble Model: {name}")
    print(f"R-squared (R²): {r2}")
    print(f"Mean Absolute Error (MAE):
{mae}")
    print(f"Mean Squared Error (MSE):
{mse}")
    print(f"Root Mean Squared Error
(RMSE): {rmse}")
print("\nEnsemble Models Saved:")
for filename in ensemble_models_saved:
    print(filename)
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error

# Load the saved models

tree_model =
joblib.load("/content/stacking_with_deci
```

```
#svm_model =
joblib.load("/content/stacking_with_svr_
as_final_estimator_model.pkl")
xgb_model =
joblib.load("/content/stacking_with_xgb
oost_as_final_estimator_model.pkl")

# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}
# Evaluate and store RMSE scores for
each model
rmse_scores = []
model_names =
list(loaded_models.keys())
predictions_dict = {}
for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
```

```
sion_tree_as_final_estimator_model.pkl
")
knn_model =
joblib.load("/content/stacking_with_knn
_as_final_estimator_model.pkl")
rf_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
gb_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
ada_model =
joblib.load("/content/stacking_with_ada
boost_as_final_estimator_model.pkl")
cat_model =
joblib.load("/content/stacking_with_catb
oost_as_final_estimator_model.pkl")
#linear_model =
joblib.load("/content/svr_bagging_mode
l.pkl")
#svm_model =
joblib.load("/content/stacking_with_svr_
as_final_estimator_model.pkl")
xgb_model =
joblib.load("/content/stacking_with_xgb
oost_as_final_estimator_model.pkl")
# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}
# Evaluate and store RMSE scores for
each model
rmse_scores = []
model_names =
list(loaded_models.keys())
predictions_dict = {}
for model_name, loaded_model in
loaded_models.items():
```

```
    rmse_scores.append(rmse)
    predictions_dict[model_name] =
{'actual': y_test, 'predicted':
test_predictions}
# Scatter plot for actual vs. predicted for
each model
for model_name, predictions in
predictions_dict.items():
    plt.figure(figsize=(14, 10))
    plt.scatter(predictions['actual'],
predictions['predicted'], label='Actual',
alpha=0.7)
    plt.plot([min(predictions['actual']),
max(predictions['actual'])],
[min(predictions['actual']),
max(predictions['actual'])], '--',
color='red', label='Perfect Prediction')
    plt.xlabel("Actual Crop Yield")
    plt.ylabel("Predicted Crop Yield")
    plt.title(f"Actual vs. Predicted Crop
Yield for {model_name}")
    plt.legend()
    plt.show()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming X_train is a DataFrame
correlation_matrix = X_train.corr()

# Plotting the correlation matrix as a
heatmap
plt.figure(figsize=(10, 8))
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    rmse_scores.append(rmse)
    predictions_dict[model_name] =
{'actual': y_test, 'predicted':
test_predictions}
# Scatter plot for actual vs. predicted for
each model
for model_name, predictions in
predictions_dict.items():
    plt.figure(figsize=(14, 10))
    plt.scatter(predictions['actual'],
predictions['predicted'], label='Actual',
alpha=0.7)
    plt.plot([min(predictions['actual']),
max(predictions['actual'])],
[min(predictions['actual']),
max(predictions['actual'])], '--',
color='red', label='Perfect Prediction')
    plt.xlabel("Actual Crop Yield")
    plt.ylabel("Predicted Crop Yield")
    plt.title(f"Actual vs. Predicted Crop
Yield for {model_name}")
    plt.legend()
    plt.show()
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import r2_score,
mean_absolute_error,
mean_squared_error
# Load the saved models
tree_model =
joblib.load("/content/voting_model.pkl")
# Add other models as needed
# Create a dictionary for loaded models
loaded_models = {
    'Voting Regressor': tree_model,
    # Add other models as needed
}
# Evaluate and store metrics for each
model
metrics = {'R-squared': [], 'MAE': [],
'RMSE': []}
model_names =
list(loaded_models.keys())
```

```
sns.heatmap(correlation_matrix,
annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
import matplotlib.pyplot as plt
X_train.hist(bins=20, figsize=(15, 10))
plt.suptitle('Histograms of Numeric
Features')
plt.show()
import seaborn as sns

sns.boxplot(data=X_train[['Area',
'Production', 'Annual_Rainfall',
'Fertilizer', 'Pesticide']])
plt.title('Box Plots of Numeric Features')
plt.show()
for column in ['Crop', 'Season', 'State']:
    sns.countplot(x=column, data=data)
    plt.title(f'Count Plot of {column}')
    plt.xticks(rotation=45)  # Adjust the
rotation angle as needed
    plt.show()
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error
# Load the saved models
tree_model =
joblib.load("/content/stacking_with_deci
sion_tree_as_final_estimator_model.pkl
")
for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    r2 = r2_score(y_test, test_predictions)
    mae = mean_absolute_error(y_test,
test_predictions)
    rmse =
np.sqrt(mean_squared_error(y_test,
test_predictions))
    metrics['R-squared'].append(r2)
    metrics['MAE'].append(mae)
    metrics['RMSE'].append(rmse)
# Create a bar chart
plt.figure(figsize=(12, 6))
# Plot R-squared
plt.bar(np.arange(len(model_names)) -
0.2, metrics['R-squared'], width=0.2,
label='R-squared', color='blue')
# Plot MAE
plt.bar(np.arange(len(model_names)),
metrics['MAE'], width=0.2,
label='MAE', color='orange')
# Plot RMSE
plt.bar(np.arange(len(model_names)) +
0.2, metrics['RMSE'], width=0.2,
label='RMSE', color='green')
plt.xlabel('Models')
plt.ylabel('Metrics')
plt.ylim(0, max(max(metrics['R-
squared']), max(metrics['MAE']),
max(metrics['RMSE'])) + 0.1)
plt.xticks(np.arange(len(model_names)),
model_names, rotation=45, ha='right')
# Add metric values on top of the bars
for i, model_name in
enumerate(model_names):
    plt.text(i - 0.2, metrics['R-squared'][i]
+ 0.02, f'{metrics["R-squared"][i]:.2f}',
ha='center', color='black', fontsize=10)
    plt.text(i, metrics['MAE'][i] + 0.02,
f'{metrics["MAE"][i]:.2f}', ha='center',
color='black', fontsize=10)
    plt.text(i + 0.2, metrics['RMSE'][i] +
0.02, f'{metrics["RMSE"][i]:.2f}',
ha='center', color='black', fontsize=10)
plt.legend()
plt.tight_layout()
plt.show()
import joblib
```

```
knn_model =
joblib.load("/content/stacking_with_knn
_as_final_estimator_model.pkl")
rf_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
gb_model =
joblib.load("/content/stacking_with_gra
dient_boosting_as_final_estimator_mod
el.pkl")
ada_model =
joblib.load("/content/stacking_with_ada
boost_as_final_estimator_model.pkl")
cat_model =
joblib.load("/content/stacking_with_catb
oost_as_final_estimator_model.pkl")
#linear_model =
joblib.load("/content/svr_bagging_mode
l.pkl")
#svm_model =
joblib.load("/content/stacking_with_svr_
as_final_estimator_model.pkl")
xgb_model =
joblib.load("/content/stacking_with_xgb
oost_as_final_estimator_model.pkl")

# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error
# Load the saved models
tree_model =
joblib.load("/content/stacking_with_deci
sion_tree_as_final_estimator_model.pkl
")
knn_model =
joblib.load("/content/stacking_with_knn
_as_final_estimator_model.pkl")
rf_model =
joblib.load("/content/stacking_with_gra
dient_boosting_base_as_final_estimator
_model.pkl")
gb_model =
joblib.load("/content/stacking_with_gra
dient_boosting_as_final_estimator_mod
el.pkl")
ada_model =
joblib.load("/content/stacking_with_ada
boost_as_final_estimator_model.pkl")
cat_model =
joblib.load("/content/stacking_with_catb
oost_as_final_estimator_model.pkl")
#linear_model =
joblib.load("/content/svr_bagging_mode
l.pkl")
#svm_model =
joblib.load("/content/stacking_with_svr_
as_final_estimator_model.pkl")
xgb_model =
joblib.load("/content/stacking_with_xgb
oost_as_final_estimator_model.pkl")
# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}
# Evaluate and store RMSE scores for
each model
```

```
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}
# Evaluate and store RMSE scores for
each model
rmse_scores = []
model_names =
list(loaded_models.keys())
predictions_dict = {}

for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    rmse_scores.append(rmse)
    predictions_dict[model_name] =
{'actual': y_test, 'predicted':
test_predictions}

# Scatter plot for actual vs. predicted for
each model
for model_name, predictions in
predictions_dict.items():
    plt.figure(figsize=(14, 10))
```

```
rmse_scores = []
model_names =
list(loaded_models.keys())
predictions_dict = {}
for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    rmse_scores.append(rmse)
    predictions_dict[model_name] =
{'actual': y_test, 'predicted':
test_predictions}
# Scatter plot for actual vs. predicted for
each model
for model_name, predictions in
predictions_dict.items():
    plt.figure(figsize=(14, 10))
    plt.scatter(predictions['actual'],
predictions['predicted'], label='Actual',
alpha=0.7)
    plt.plot([min(predictions['actual']),
max(predictions['actual'])],
[min(predictions['actual']),
max(predictions['actual'])], '--',
color='red', label='Perfect Prediction')
    plt.xlabel("Actual Crop Yield")
    plt.ylabel("Predicted Crop Yield")
    plt.title(f"Actual vs. Predicted Crop
Yield for {model_name}")
    plt.legend()
    plt.show()
import joblib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import
mean_squared_error
# Load the saved models
tree_model =
joblib.load("/content/stacking_with_deci
sion_tree_as_final_estimator_model.pkl
")
knn_model =
joblib.load("/content/stacking_with_knn
_as_final_estimator_model.pkl")
rf_model =
joblib.load("/content/stacking_with_gra
```

```
    plt.scatter(predictions['actual'],
predictions['predicted'], label='Actual',
alpha=0.7)
    plt.plot([min(predictions['actual']),
max(predictions['actual'])],
[min(predictions['actual']),
max(predictions['actual'])], '--',
color='red', label='Perfect Prediction')
    plt.xlabel("Actual Crop Yield")
    plt.ylabel("Predicted Crop Yield")
    plt.title(f"Actual vs. Predicted Crop
Yield for {model_name}")
    plt.legend()
    plt.show()
# Bar chart for RMSE scores
plt.figure(figsize=(12, 8))
plt.bar(model_names, rmse_scores,
color='skyblue')
plt.xlabel('Model')
plt.ylabel('Root Mean Squared Error
(RMSE)')
plt.title('Performance Comparison of
Stacking Models')
plt.ylim([min(rmse_scores) - 0.1,
max(rmse_scores) + 0.1])  # Adjust the
y-axis limits if needed
plt.xticks(rotation=45, ha='right')
plt.show()
from sklearn.metrics import
mean_absolute_error


# Evaluate and store MAE scores for
each model
dient_boosting_base_as_final_estimator
_model.pkl")
gb_model =
joblib.load("/content/stacking_with_gra
dient_boosting_as_final_estimator_mod
el.pkl")
ada_model =
joblib.load("/content/stacking_with_ada
boost_as_final_estimator_model.pkl")
cat_model =
joblib.load("/content/stacking_with_catb
oost_as_final_estimator_model.pkl")
#linear_model =
joblib.load("/content/svr_bagging_mode
l.pkl")
#svm_model =
joblib.load("/content/stacking_with_svr_
as_final_estimator_model.pkl")
xgb_model =
joblib.load("/content/stacking_with_xgb
oost_as_final_estimator_model.pkl")
# Create a dictionary for loaded models
loaded_models = {
    'K-Nearest Neighbors': knn_model,
    'Random Forest': rf_model,
    'Gradient Boosting': gb_model,
    'AdaBoost': ada_model,
    'CatBoost': cat_model,
    #'Linear Regression': linear_model,
    #'Support Vector Machine':
svm_model,
    'XGBoost': xgb_model,
    'Decision Tree': tree_model,
}
# Evaluate and store RMSE scores for
each model
rmse_scores = []
model_names =
list(loaded_models.keys())
predictions_dict = {}
for model_name, loaded_model in
loaded_models.items():
    test_predictions =
loaded_model.predict(pro_X_test)
    mse = mean_squared_error(y_test,
test_predictions)
    rmse = np.sqrt(mse)
    rmse_scores.append(rmse)
```

```
mae_scores = []

for model_name, loaded_model in loaded_models.items():
    test_predictions = loaded_model.predict(pro_X_test)
    mae = mean_absolute_error(y_test, test_predictions)
    mae_scores.append(mae)

# Bar chart for MAE scores
plt.figure(figsize=(12, 8))
plt.bar(model_names, mae_scores, color='lightgreen')
plt.xlabel('Model')
plt.ylabel('Mean Absolute Error (MAE)')
plt.title('Performance Comparison of Stacking Models (MAE)')
plt.ylim([min(mae_scores) - 0.1, max(mae_scores) + 0.1])  # Adjust the y-axis limits if needed
plt.xticks(rotation=45, ha='right')
plt.show()
import pandas as pd
import numpy as np
# Specify the path to your original CSV file
csv_file_path = '/content/rice.csv'
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file_path)
# Drop the 'Crop_Year' column
    predictions_dict[model_name] = {'actual': y_test, 'predicted': test_predictions}
# Scatter plot for actual vs. predicted for each model
for model_name, predictions in predictions_dict.items():
    plt.figure(figsize=(14, 10))
    plt.scatter(predictions['actual'], predictions['predicted'], label='Actual', alpha=0.7)
    plt.plot([min(predictions['actual']), max(predictions['actual'])], [min(predictions['actual']), max(predictions['actual'])], '--', color='red', label='Perfect Prediction')
    plt.xlabel("Actual Crop Yield")
    plt.ylabel("Predicted Crop Yield")
    plt.title(f"Actual vs. Predicted Crop Yield for {model_name}")
    plt.legend()
    plt.show()
from sklearn.metrics import r2_score
# Evaluate and store R2 scores for each model
r2_scores = []
for model_name, loaded_model in loaded_models.items():
    test_predictions = loaded_model.predict(pro_X_test)
    r2 = r2_score(y_test, test_predictions)
    r2_scores.append(r2)
# Bar chart for R2 scores
plt.figure(figsize=(12, 8))
plt.bar(model_names, r2_scores, color='lightcoral')
plt.xlabel('Model')
plt.ylabel('R-squared (R2) Score')
plt.title('Performance Comparison of Stacking Models (R2 Score)')
plt.ylim([min(r2_scores) - 0.1, max(r2_scores) + 0.1])  # Adjust the y-axis limits if needed
plt.xticks(rotation=45, ha='right')
plt.show()
import pandas as pd
import numpy as np
# Specify the path to your original CSV file
csv_file_path = '/content/rice.csv'
```

```
df = df.drop(columns=['Crop_Year'])
# Subtract specific values from columns before creating new rows
# Get the number of rows you want to create
target_rows = 10000
# Duplicate rows randomly to reach the target number of rows
while len(df) < target_rows:
    # Randomly choose an index from the existing rows
    random_index = np.random.choice(len(df))
    # Select the row at the random index
    random_row = df.iloc[random_index].copy()
    # Modify the 'Yield' column by dividing the value by areas
    random_row['Annual_Rainfall'] -= 200
    random_row['Fertilizer'] -= 1000000 # Subtract 10 lakhs
    random_row['Pesticide'] -= 3000
    random_row['Yield'] = random_row['Production'] / random_row['Area']
    # Append the modified row to the DataFrame
    df = df.append(random_row, ignore_index=True)
# Save the updated DataFrame to a new CSV file
```

```
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file_path)
# Drop the 'Crop_Year' column
df = df.drop(columns=['Crop_Year'])
# Subtract specific values from columns before creating new rows
# Get the number of rows you want to create
target_rows = 10000
# Duplicate rows randomly to reach the target number of rows
while len(df) < target_rows:
    # Randomly choose an index from the existing rows
    random_index = np.random.choice(len(df))
    # Select the row at the random index
    random_row = df.iloc[random_index].copy()
    # Modify the 'Yield' column by dividing the value by areas
    random_row['Annual_Rainfall'] -= 200
    random_row['Fertilizer'] -= 1000000 # Subtract 10 lakhs
    random_row['Pesticide'] -= 3000
    random_row['Yield'] = random_row['Production'] / random_row['Area']
    # Append the modified row to the DataFrame
    df = df.append(random_row, ignore_index=True)
# Save the updated DataFrame to a new CSV file
output_csv_path = '/content/rice_updated10k.csv'
df.to_csv(output_csv_path, index=False)
# Display the updated DataFrame
print(df)
<ipython-input-3-fb8839960384>:34: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df = df.append(random_row, ignore_index=True)
     Crop  Season       State       Area  Production  Annual_Rainfall  \
```

```
output_csv_path =
'/content/rice_updated10k.csv'
df.to_csv(output_csv_path, index=False)
# Display the updated DataFrame
print(df)
```

```
0    Rice   Autumn       Dhaka
607358.0     398311          2051.4
1    Rice   Summer       Dhaka
174974.0     209623          2051.4
2    Rice   Winter       Dhaka  1743321.0
1647296          2051.4
3    Rice  Monsoon  Chittagong
1031530.0     2340493          1266.7
4    Rice   Winter Chittagong    53889.0
109350          1266.7
...   ...      ...         ...        ...         ...
...
9995  Rice   Winter      Khulna
100824.0     285614          2655.4
9996  Rice  Monsoon     Jessore
2812400.0     7005800          248.3
9997  Rice   Winter       Dhaka
1880737.0     3709081          1397.7
9998  Rice  Monsoon     Comilla
52985.0     128945          2528.8
9999  Rice  Monsoon   Sirajganj
3828992.0     5567644          843.7
      Fertilizer  Pesticide     Yield
0     5.780226e+07  188280.98
0.780870
1     1.665228e+07   54241.94  1.060435
2     1.659119e+08  540429.51
0.941304
3     9.817071e+07  319774.30
2.233500
4     5.128616e+06   16705.59  2.073846
...            ...       ...       ...
9995  1.156806e+07   18222.48
2.832798
9996  2.738651e+08  803596.00
2.491040
9997  2.697477e+08  501798.99
1.972142
9998  4.766714e+06    5656.70
2.433613
9999  3.749936e+08  906958.08
1.454076
[10000 rows x 9 columns]
```

Detailed code is given here: https://github.com/Forhad876/Forhad876

MD Forhad, Arman Bin Joynal

ORIGINALITY REPORT

18%	12%	11%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	publications.eai.eu Internet Source	1%
3	Qiming Zhou, Ali Ismaeel. "Integration of maximum crop response with machine learning regression model to timely estimate crop yield", Geo-spatial Information Science, 2021 Publication	1%
4	www.researchgate.net Internet Source	1%
5	doctorpenguin.com Internet Source	<1%
6	Richa Kumari Karn, A. Suresh. "Prediction of Crops Based on a Machine Learning Algorithm", 2023 International Conference on Computer Communication and Informatics (ICCCI), 2023 Publication	<1%

7	link.springer.com Internet Source	<1 %
8	Manasa Chitradurga Manjunath, Blessed Prince Palayyan. "An Efficient Crop Yield Prediction Framework Using Hybrid Machine Learning Model", Revue d'Intelligence Artificielle, 2023 Publication	<1 %
9	Rubby Aworka, Lontsi Saadio Cedric, Wilfried Yves Hamilton Adoni, Jérémie Thouakesseh Zoueu et al. "Agricultural Decision System based on Advanced Machine Learning Models for Yield Prediction: Case of East African Countries", Smart Agricultural Technology, 2022 Publication	<1 %
10	assets.researchsquare.com Internet Source	<1 %
11	ebin.pub Internet Source	<1 %
12	Thomas. Rincy. N, Roopam Gupta. "Ensemble Learning Techniques and its Efficiency in Machine Learning: A Survey", 2nd International Conference on Data, Engineering and Applications (IDEA), 2020 Publication	<1 %
	Submitted to University of Surrey	

13	Student Paper	<1%
14	Submitted to University of Sunderland Student Paper	<1%
15	Submitted to University of Hertfordshire Student Paper	<1%
16	ijritcc.org Internet Source	<1%
17	Naveed Anwer Butt, Zafar Mahmood, Khawar Shakeel, Sultan Alfarhood, Mejdl Safran, Imran Ashraf. "Performance Prediction of Students in Higher Education Using Multi-Model Ensemble Approach", IEEE Access, 2023 Publication	<1%
18	www.appinio.com Internet Source	<1%
19	www.frontiersin.org Internet Source	<1%
20	Deng, Houtao, and George Runger. "Gene selection with guided regularized random forest", Pattern Recognition, 2013. Publication	<1%
21	Submitted to Colorado Technical University Student Paper	<1%

22	G Pradeep, T Dureen V Rayen, A. Pushpalatha, P. Kavitha Rani. "Effective Crop Yield Prediction Using Gradient Boosting To Improve Agricultural Outcomes", 2023 International Conference on Networking and Communications (ICNWC), 2023 Publication	<1 %
23	Lontsi Saadio Cedric, Wilfried Yves Hamilton Adoni, Rubby Aworka, Jérémie Thouakesseh Zoueu et al. "Crops Yield Prediction Based on Machine Learning Models: Case of West African Countries", Smart Agricultural Technology, 2022 Publication	<1 %
24	udspace.udel.edu Internet Source	<1 %
25	Submitted to Manchester Metropolitan University Student Paper	<1 %
26	Pabitha C, Benila S, Suresh A. "A digital footprint in enhancing agricultural practices with improved production using machine learning", Research Square Platform LLC, 2023 Publication	<1 %
27	engrxiv.org Internet Source	<1 %