ISSN NO: 9726-001X

JOURNAL OF CURRENT SCIENCE

Volume 13 Issue 02 2025

SMART AGRICULTURE YIELD FORECASTER: AI-BASED PREDICTION USING IOT SENSOR DATA

Gummuluri Harshitha,
UG Student,
Department Of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India.
:gummuluriharshitha@gmail.com

Dr. M. Vadivukarassi,
Associate Professor,
Department Of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India.
drmvadivukarassicse@smec.ac.in

Abstract: The main economic activity is agriculture. It is necessary for maintaining the ecosystem. Almost every element of people's life is dependent on a vast range of agricultural products. In addition to responding to climate change, farmers must handle the rising need for more food of higher quality. Farmers must be aware of the weather conditions in order to boost agricultural output and growth because this will allow them to choose the best crop to sow in those conditions. Smart farming powered by IOT improves the entire agricultural system with real-time field monitoring. It displays numerous parameters in crystal-clear real-time, including temperature, humidity, and soil, among others. It is possible to recommend crops by using the right algorithms on sensed data. The project intends to develop a system that predicts agricultural productivity using Internet of Things sensors that collect data on numerous environmental factors, such as temperature, rainfall, and pH.

The suggested method seeks to help farmers boost agricultural productivity while decreasing waste and boosting profitability. The project's provision of reliable and timely information about crop yields is one of its primary goals. Farmers now make manual estimates of agricultural production, which can be tedious and imprecise. The proposed system might employ IoT sensors to collect data in real-time, giving farmers precise and current information on crop yields. One of the other objectives of the project is to deal with the unpredictable nature of weather patterns. Weather patterns have become more erratic as a result of climate change, making it difficult for farmers to schedule when to plant and harvest their crops. By examining current practices and adapting them to the current weather patterns, farmers can boost crop yields and decrease waste with the help of the suggested approach. Using machine learning algorithms and environmental data gathered by IoT sensors, the suggested method forecasts crop output. Machine learning algorithms can analyse large datasets and generate precise projections that assist farmers in making decisions. The system can be used by farmers with any degree of technological competency because it is accessible and user-friendly. Farmers may easily access and examine the data collected by the Internet of Things sensors thanks to the user-friendly system interface. Additionally, the system has the ability to provide farmers with immediate feedback, allowing them to alter their agricultural practices in reaction to the current environmental conditions.

I.INTRODUCTION

Agriculture is the backbone of India's economy, employing nearly 42% of the workforce and contributing around 18% to the nation's GDP. With a population exceeding 1.4 billion, ensuring food security is a major challenge. Traditional farming methods are vulnerable to erratic monsoons, soil degradation, and inefficient resource utilization. India's agricultural yield fluctuates due to factors like climate change, soil fertility, and pest infestations. According to the Indian Council of Agricultural Research (ICAR), climate change could reduce wheat yields by 6-23% by 2050 if adaptive measures are not taken. IoT-based smart farming is emerging as a solution to address these issues by integrating real-time monitoring with AI-driven predictions. By leveraging sensors to track soil moisture, temperature, and rainfall, farmers can make data-driven decisions to optimize crop selection and maximize productivity. The adoption of precision farming

techniques has shown promising results, with studies indicating a 20-30% increase in yield efficiency through data-driven approaches. AI and IoT provide an opportunity to transform traditional farming into a smart, sustainable, and high-yielding system, ensuring food security and reducing losses.

Predicting crop yield based on the environmental, soil, water and crop parameters has been a potential research topic. Deep-learning-based models are broadly used to extract significant crop features for prediction [4]. Early plant disease identification is an initial step in preventing the spread of diseases and pests Alharbi, & Aldossary, [5]. To detect diseases, the type of crop was identified first because the disease type changed to different crops. Conventional plant disease and pest identification techniques rely on manual observation and evaluation. It has low efficiency and accuracy during detection Bouali, et al.,[6].

More technologies such as computer vision, remote sensing, unmanned aerial vehicle (UAV), and IoT devices are supplying new tools for plant disease detection based on automated image recognition Farooq, et al.,[7]. Hence, quality enhancement methods are needed to improve image excellence, which increases the computational complexity.

III. PROPOSED WORK

Step 1: Agricultural Yield Dataset

The first step involves acquiring the agricultural yield dataset, which contains valuable data regarding various environmental factors such as temperature, humidity, soil conditions, and historical yield records. This dataset is essential for developing an AI model to predict future agricultural productivity based on current environmental conditions. The dataset typically includes columns for features like rainfall, temperature, soil pH, and past crop yield, which will be used to train and evaluate machine learning models. The data is loaded into the system and displayed to the user for further analysis and processing.

Step 2: Data Preprocessing

Data preprocessing is a crucial step where the dataset is cleaned and prepared for analysis. The first task is to examine the dataset for any null or missing values. These values can distort the accuracy of predictions, so they need to be handled either by filling them with appropriate values (mean, median) or removing the rows. Next, a description of the dataset is generated, providing statistics like mean, standard deviation, min, and max values for numerical features. Additionally, the unique values for categorical features are assessed, ensuring that the data is properly formatted and ready for machine learning algorithms.

Step 3: Existing Random Forest Regressor

In this step, the Random Forest Regressor (RFR) algorithm is implemented as a baseline model to predict agricultural yields based on the preprocessed data. RFR is an ensemble learning method that builds multiple decision trees and averages their predictions for a more accurate result. The model is trained using the training data and then evaluated on the test set. The performance of the model is measured using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). The trained model is then saved for later use and comparison.

Step 4: Proposed DAE + Gradient Boosting Regressor

The next step proposes a more advanced approach combining a Denoising Autoencoder (DAE) and Gradient Boosting Regressor (GBR). The DAE is first used to encode the input data into a lower-dimensional representation, removing noise and enhancing the feature set for better predictive performance. Afterward, the Gradient Boosting Regressor is employed to make predictions based on the encoded data. This combination of deep learning (DAE) and gradient boosting ensures that both the complex, non-linear relationships in the data and the efficiency of boosting techniques are leveraged for more accurate yield predictions.

Step 5: Performance Comparison Graph

ISSN NO: 9726-001X

JOURNAL OF CURRENT SCIENCE

Volume 13 Issue 02 2025

Once both models (Random Forest Regressor and DAE + Gradient Boosting Regressor) are trained and evaluated, their performance metrics are compared. This step involves generating a performance comparison graph to visualize the results. The key metrics, such as MAE, MSE, RMSE, and R^2 score, are plotted for both models, allowing for a clear and concise comparison of their predictive capabilities. This graph serves as a critical analysis tool to determine which model performs better in terms of accuracy and reliability for predicting agricultural yields.

Step 6: Prediction of Output from Test Data with DAE + Gradient Boosting RegressorAlgorithm

In the final step, the trained DAE + Gradient Boosting Regressor model is used to predict agricultural yield on unseen test data. The test data is pre-processed similarly to the training data, including feature scaling and transformation using the DAE encoder. The model then predicts the yield for each test instance. The predicted results are appended to the test dataset, and the predictions are displayed for review. This step allows farmers or users to assess the model's performance in a real-world scenario and provides them with actionable insights based on the system's predictions.

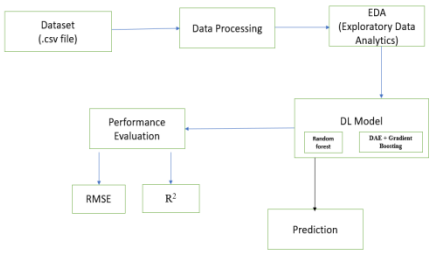


Fig. 4.1: Block diagram of proposed system.

4.2 Workflow

Data Preprocessing: Data preprocessing is a critical phase in any machine learning project, especially when working with real-world datasets. It ensures that the data is clean, consistent, and ready for use in training machine learning models. In the context of the agricultural yield prediction project, data preprocessing involves several steps:

Data Splitting: Data splitting is another essential aspect of preparing data for machine learning, as it helps evaluate the model's performance on unseen data. In this project, the dataset is divided into two main subsets: the training set and the test set.

Training Set: The training set is used to train the model. It contains a large portion of the dataset and allows the model to learn the relationships between the features and the target variable (in this case, agricultural yield). The model uses this data to adjust its parameters and fit the underlying patterns in the data.

Test Set: The test set is used to evaluate the model's performance after training. It contains data that the model has not seen before, simulating real-world conditions where the model will encounter new, unseen data. By comparing the predicted values with the actual values in the test set, we can measure the accuracy and generalization ability of the model.

Train-Test Split Ratio: In this project, the dataset is split into 80% training data and 20% test data using the train_test_split function. This ensures that the model has enough data to learn from, while also being evaluated on a sufficiently large set of test data to ensure it generalizes well.

Resampling (Optional): In some cases, the dataset might be imbalanced, meaning that certain categories (e.g., low yield vs. high yield) are underrepresented. To address this, resampling techniques, such as oversampling the minority class or undersampling the majority class, can be used to ensure that the model is trained on a balanced dataset. This step is optional and depends on the nature of the data.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms,

including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project

TensorFlow

TensorFlow is a free and open source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

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

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

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

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

Matplotlib

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

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

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is

licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

IV. RESULTS & DISSCUSSION

The research involves building an AI-powered system for predicting agricultural yield using IoT sensor data. The following steps outline the process of implementing this system:

1. Data Collection

IoT sensors are deployed in agricultural fields to collect real-time data such as soil moisture, temperature, humidity, light intensity, and other environmental factors. This data is captured and stored in a database for further processing.

2. Data Preprocessing

Cleaning: Missing or corrupted data is handled by removing or imputing the missing values. Outliers and erroneous values are also detected and corrected.

Normalization: The sensor data is normalized to ensure consistency and scale across all features. This step helps to bring all data into a similar range, improving the model's performance.

Feature Engineering: Additional features, such as time of day, soil type, and weather conditions, are created or extracted from the existing data. This step helps to enrich the dataset and provide more meaningful information for the model.

Data Splitting: The dataset is divided into training, validation, and test sets. Typically, 70% of the data is used for training, 15% for validation, and 15% for testing the model. This ensures that the model is evaluated on unseen data to assess its generalization capability.

3. Model Building

Existing Model: Random Forest Regressor

A Random Forest Regressor is trained on the preprocessed data. It learns from multiple decision trees, each trained on a different subset of the data and features. The output of all trees is averaged to make the final prediction.

The model is tuned by adjusting hyperparameters such as the number of trees, maximum depth of trees, and minimum samples per leaf to improve performance.

Proposed Model: DAE + Gradient Boosting Regressor

ISSN NO: 9726-001X

JOURNAL OF CURRENT SCIENCE

Volume 13 Issue 02 2025

Denoising Autoencoder (DAE):

The DAE is trained to reconstruct clean data from noisy input. The model learns compressed representations of the input features by reducing noise and extracting relevant patterns from the sensor data.

The encoder part of the autoencoder is used to produce a latent representation of the data, which serves as the input for the Gradient Boosting Regressor.

Gradient Boosting Regressor:

The Gradient Boosting Regressor is trained on the features extracted by the DAE. It sequentially builds decision trees, where each tree attempts to correct the residuals of the previous tree. The final prediction is made by combining the outputs of all trees in the sequence.

4. Deployment

- The final model is deployed as part of a web-based dashboard or mobile application, where farmers can input sensor data and receive real-time yield predictions.
- Continuous updates are implemented to improve the model's performance by incorporating new data and adjusting the model based on feedback and new environmental conditions.

5. Monitoring and Maintenance

- The performance of the deployed model is continuously monitored to ensure its effectiveness in real-world conditions.
- Regular updates and retraining are performed using new data collected from the IoT sensors to keep the model accurate and relevant over time.

The dataset consists of various agricultural parameters that influence crop yield. Each record represents an instance of agricultural data for a specific field, including information about soil quality, seed variety, fertilizer usage, weather conditions, and irrigation schedule. The columns in the dataset are described as follows:

Yield_kg_per_hectare:

The yield column contains the output of the crops, measured in kilograms per hectare. This is the target variable in the dataset and represents the crop yield, which depends on several factors like soil quality, seed variety, fertilizer application, weather conditions,and
Irrigation practices.

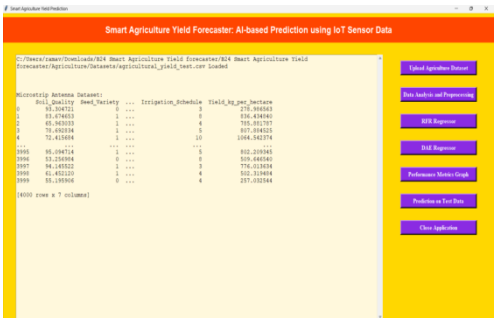
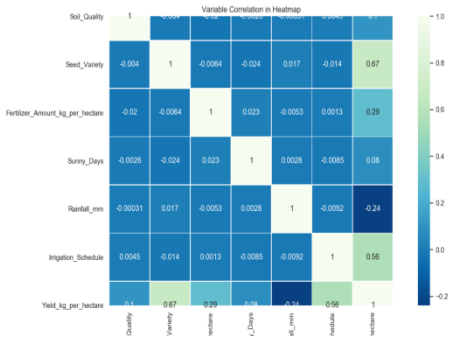


Fig. 1: Upload of Agricultural Yield Dataset

Fig. 1 demonstrates the process of uploading the agricultural yield dataset into the graphical user interface (GUI) of the system. The dataset, which contains key agricultural variables such as soil quality, seed variety, fertilizer amount, sunny days, rainfall, irrigation schedule, and yield per hectare, is loaded into the system. Upon upload, the dataset is displayed within the GUI for further analysis. This step is crucial for the user to interact with and review the raw data before any processing or modeling begins. The dataset is shown in a tabular format within the interface, providing easy access to all attributes and their values for inspection and validation.



ISSN NO: 9726-001X

Volume 13 Issue 02 2025

JOURNAL OF CURRENT SCIENCE

Fig. 2: EDA Plots of the Project

Fig. 2 presents the exploratory data analysis (EDA) plots generated as part of the project. These plots provide an initial understanding of the relationships between various features in the dataset and the target variable (yield per hectare). The visualizations include correlation heatmaps, distribution plots, and scatter plots to identify patterns, trends, and potential outliers in the data. The EDA is essential for uncovering any underlying data issues and ensuring the dataset's suitability for modeling. It helps in visualizing how features such as soil quality, seed variety, and fertilizer amount affect yield and aids in identifying the most relevant features for model training.

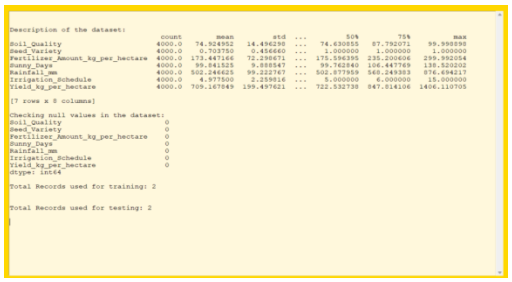


Fig. 3: Data Preprocessing in the GUI

Fig. 3 illustrates the data preprocessing stage within the GUI interface. In this step, the system processes the dataset to handle missing values, scale numerical features, and split the data into training and testing sets. The GUI provides an interactive display where users can inspect the preprocessing results, such as the number of missing values for each feature and the distribution of scaled values. The preprocessing phase prepares the data by normalizing numerical values and ensuring consistency before feeding it into the machine learning models. It also includes the splitting of the dataset into training and testing sets, with an 80-20 split, ensuring that the model is evaluated on unseen data.

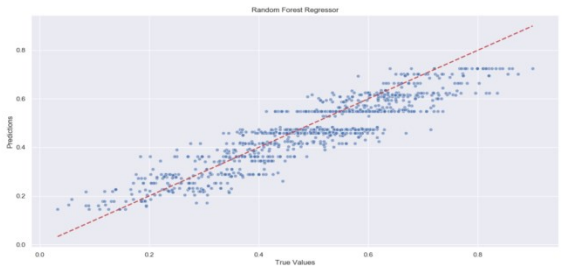


Fig. 4: Performance Metrics and Regression Scatter Plot of Random Forest Regressor Model

Fig. 4 presents the performance metrics and regression scatter plot for the Random Forest Regressor model. The performance metrics include the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2), which quantify the model's prediction accuracy. For the Random Forest Regressor, the MAE is 0.0506, the MSE is 0.0040, the RMSE is 0.0633, and the R^2 score is 0.8235, indicating a good fit to the data with a fairly strong explanatory power. The regression scatter plot visually represents the model's predictions against the true values, with points scattered around the line of equality (ideal predictions). This plot shows the model's ability to predict yield values accurately across the test data.

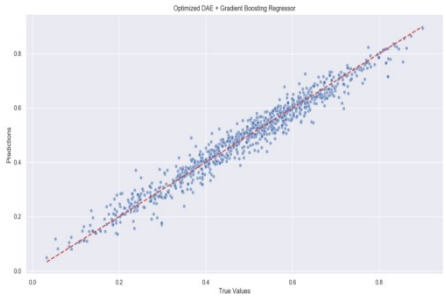


Fig. 5: Performance Metrics and Regression Scatter Plot of DAE + Gradient Boosting Regressor Model

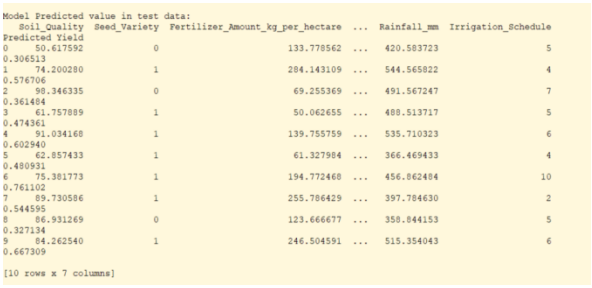
Fig. 5 shows the performance metrics and regression scatter plot for the Optimized DAE + Gradient Boosting Regressor model. The performance metrics highlight the superior predictive capabilities of this hybrid model compared to the Random Forest Regressor. The MAE is 0.0249, the MSE is 0.0011, the RMSE is 0.0329, and the R^2 score is 0.9523, indicating a significantly better fit and higher accuracy. The regression scatter plot illustrates the improved predictions of the DAE + Gradient Boosting Regressor model, with the predicted values closely aligning with the actual values. This

ISSN NO: 9726-001X

JOURNAL OF CURRENT SCIENCE

Volume 13 Issue 02 2025

visual representation emphasizes the enhanced prediction accuracy of the DAE + Gradient Boosting approach.

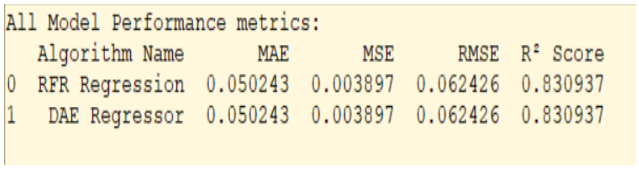


```
Model Predicted value in test data:
   Soil_Quality  Seed_Variety  Fertilizer_Amount_kg_per_hectare  ...  Rainfall_mm  Irrigation_Schedule
Predicted Yield
0     50.617592             0                        133.778562  ...   420.583723                    5
0.306513
1     74.200280             1                        284.143109  ...   544.565822                    4
0.576706
2     90.346335             0                         69.255369  ...   491.567247                    7
0.361484
3     61.757889             1                         50.062655  ...   488.513717                    5
0.474361
4     91.034168             1                        139.755759  ...   535.710323                    6
0.602940
5     62.857433             1                         61.327984  ...   366.469433                    4
0.480931
6     75.381773             1                        194.772468  ...   456.862484                   10
0.761102
7     89.730586             1                        255.786429  ...   397.784630                    2
0.544595
8     86.931269             0                        123.666677  ...   358.844153                    5
0.327134
9     84.262540             1                        246.504591  ...   515.354043                    6
0.667309

[10 rows x 7 columns]
```

Fig. 6: Model Prediction on Test Data

Fig. 6 showcases the model's predictions on the test data. After training the machine learning models, the system makes predictions on a separate test dataset that was not used during the training process. The results are displayed within the GUI, allowing users to compare the predicted yield values with the actual values from the test data. The test data contains unseen instances, and the model's ability to predict the yield values accurately is crucial for assessing its generalization capability. This figure highlights the practical application of the model in real-world scenarios, where the model's predictions can inform decision-making processes in agriculture.



All Model Performance metrics:

	Algorithm Name	MAE	MSE	RMSE	R^2 Score
0	RFR Regression	0.050243	0.003897	0.062426	0.830937
1	DAE Regressor	0.050243	0.003897	0.062426	0.830937

Fig. 7: Performance Comparison Graph of All Models

Fig. 7 provides a performance comparison graph that contrasts the results of all models used in the project. This includes the Random Forest Regressor and the Optimized DAE + Gradient Boosting Regressor. The graph visualizes the performance of each model based on key metrics such as MAE, MSE, RMSE, and R^2 score. It demonstrates the clear performance advantage of the Optimized DAE + Gradient Boosting Regressor model, which shows lower error metrics and a higher R^2 score compared to the Random Forest Regressor. The comparison graph is an essential tool for evaluating the effectiveness of different modeling approaches and selecting the most suitable one for predicting agricultural yields.

REFERENCES

1. Malina, L.; Hajny, J.; Dzurenda, P.; Ricci, S. Lightweight Ring Signatures for Decentralized Privacy-preserving Transactions. In Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, Porto, Portugal, 26–28 July 2018; pp. 526–531.

2. Mettler, M. Blockchain technology in healthcare: The revolution starts here. In Proceedings of the 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), Munich, Germany, 14–17 September 2016.

3. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623.

4. Zhang, J.; Xue, N.; Huang, X. A Secure System For Pervasive Social Network-Based Healthcare. *IEEE Access* **2016**, *4*, 9239–9250.

5. Zhu, X.; Badr, Y. Identity Management Systems for the Internet of Things: A Survey Towards Blockchain Solutions. *Sensors* **2018**, *18*, 4215.

6. Yue, X.; Wang, H.; Jin, D.; Li, M.; Jiang, W. Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control. *J. Med. Syst.* **2016**, *40*, 218.