Crop Recommendation Predictive Analysis using Ensembling Techniques

Muhammad Umar Abdullahi¹*, Morufu Olalere², Gilbert I. O. Aimufua², Kene Tochukwu Anyachebelu² and Bako Halilu Egga²

¹Department of Computer Science, Federal University of Technology, Owerri, Nigeria
²Department of Computer Science, Nasarawa State University, Keffi, Nigeria
*Corresponding Author Email: umarfaruq54@gmail.com

ABSTRACT
Crop recommendation systems play a crucial role in modern agriculture by aiding farmers in making well-informed choices to optimize crop yield and resource utilization. Ensemble learning approaches can significantly improve the effectiveness of crop recommendation systems. To achieve this, multiple forecasts are combined from various models. In this paper, a complete Machine Learning Pipeline is used to evaluate the performance of ensemble learning models in crop recommendation tasks. A diverse dataset is used to select and train four ensemble learning methods, Bagging, Voting, Stacking, and One-Vs-Rest (OVR), as separate classifiers. The dataset includes various agricultural factors such as soil characteristics, meteorological conditions, and past crop productivity. Various metrics, including accuracy, precision, recall, F1-score, and support, are utilized for each model. Bagging is considered the most effective ensemble learning technique, demonstrating excellent levels of accuracy and overall performance. The bagging algorithm achieves a high level of accuracy, reaching 99.32%. It also achieves perfect precision, recall, and F1-score metrics, with values of 0.99, 1.00, and 1.00 respectively. The support value, which represents the number of instances used for evaluation, is 141. This study provides valuable perspectives on the choice of appropriate ensemble learning models for crop recommendation tasks. Consequently, it enables farmers and other individuals involved in agriculture to make well-informed choices using data, resulting in enhanced agricultural output and sustainability.

Keywords:
Comparative Analysis, Ensembling Learning Models, Crop recommendation, Evaluation Metrics, Model Assessment.

INTRODUCTION
Agriculture serves as the fundamental support for numerous economies, supplying sustenance, livelihoods, and basic resources for diverse sectors. By 2050, the world population is estimated to reach around 10 billion, leading to a substantial rise in the need for food (Tripathi, Mishra, Maurya, Singh & Wilson, 2019). This would need advancements in agricultural production and efficiency (Serraj, Krishnan & Pingali, 2019; Falcon, Naylor & Shankar, 2022). Technology plays a significant role in modern agriculture, particularly with the progress in data science and machine learning. These innovations provide new opportunities for improving agricultural methods.

Historically, farmers have traditionally depended on practical knowledge and regional experience to make informed choices on crop selection, timing of planting, and allocation of resources (Chakauya, Matererechera, Jiri, Chakauya & Machete, 2023). Nevertheless, these approaches are frequently constrained by subjectivity, fluctuations in environmental circumstances, and the incapacity to efficiently handle and interpret substantial amounts of data (Limpoto et al., 2022; Kom, Nethengwe, Mpandeli & Chikoore, 2022). Consequently, there has been an increasing interest in utilizing technology to provide data-driven methods for making decisions in agriculture.

Machine learning, also known as artificial intelligence, has become a powerful tool for analysing agricultural data and generating valuable insights (Redhu, Thakur, Yashveer & Mor, 2022). Machine learning models may utilize algorithms to analyse various datasets, such as soil composition, weather patterns, satellite images, and historical agricultural production data. Through this analysis, these models can find intricate patterns and linkages that may not be easily observable by humans (Shaikh, Rasool & Lochan, 2022). This feature allows for the creation of crop recommendation systems that can forecast the optimal crops for a certain region by considering many parameters.

Ensemble learning is the process of amalgamating predictions from numerous base models to provide a solitary and more precise forecast (Mohammed & Kora, 2023). Ensembling approaches can enhance overall performance by harnessing the variety of various models, therefore mitigating the limitations of individual classifiers (Ganaie, Hu, Malik, Tanveer & Suganthan, 2022). Stacking, Bagging, Voting, and One-Vs-Rest (OVR) are often employed ensemble learning techniques, each employing a distinct strategy for aggregating predictions (Zounemat-Kermani, Batelaan, Fadaee & Hinkelmann, 2021).

While there is a growing interest in employing ensemble learning for agricultural purposes, there is a lack of comprehensive comparative assessments to determine the most effective approach for crop recommendation tasks (Agrawal, Govil, & Kumar, 2024). This paper aims to address this gap by conducting a thorough analysis of the efficacy of different ensembling procedures using real agricultural data. This research tends to assess the precision, resilience, and computational efficiency of ensemble learning techniques, including Stacking, Bagging, Voting, and OVR. The aim is to get valuable information regarding the efficacy of these methodologies for crop recommendation (Sajitha, Andrushia, Anand & Naser, 2024). Furthermore, the paper will examine the impact of various dataset factors, such as size, dimensionality, and class distribution, on the efficacy of ensembling algorithms. A thorough understanding of how these factors influence the effectiveness of ensemble learning might offer useful insights for the development and implementation of more robust crop recommendation systems tailored to different agricultural contexts (Shams, Gamel & Talaat, 2024).

The paper enhances agricultural technology by offering empirical data on the effectiveness of ensemble learning approaches for crop recommendation. The results of this study can provide valuable insights for improving the accuracy and dependability of crop recommendation systems. This, in turn, can enable farmers to make informed decisions based on data, leading to increased productivity, sustainability, and resilience in agriculture.

**MATERIALS AND METHODS**

The paper utilizes the Machine Learning Pipeline or Data Science Workflow methodology (Biswas, Wardat & Rajan, 2022). The methodology has eight sequential steps. The procedure illustrated in Figure 1 comprises many stages: data collection, data exploration, pre-processing, dataset partitioning, model training, model testing, performance evaluation, performance comparison, and conclusion (Biswas, Wardat & Rajan, 2022). This figure was modified from the original work of Rana et al. (2015) to align with the research approach utilized in this paper. The attainment of this target was achieved by the systematic implementation of a well-structured series of procedures.

![Figure 1: Data Science Workflow](source)

**Dataset Collection**
The initial step involved the collection of the dataset in CSV format, followed by its importation into Jupyter Notebook for further analysis.

**Data Exploration and Pre-processing**
This phase encompassed four key stages. First, a thorough exploration of the data was conducted to understand its characteristics. Subsequently, features with negative correlations to the target variable were identified and removed to enhance model performance. Finally, feature scaling was applied to ensure that all features were on a consistent scale.

**Splitting the Dataset**
Following the importation of the dataset, it was divided into two distinct subsets. The first subset was utilized for training the machine learning models, while the second
subset was employed to evaluate the performance of these models. This step was undertaken to assess the performance of the models on an independent dataset, therefore mitigating the risk of overfitting.

**Training the Models**

During the training phase, seven separate ensembling learning models were utilized, specifically Stacking, Bagging, Voting, and One-Vs-Rest (OVR). The aforementioned models were employed to construct prediction algorithms.

**Validation of Models**

To improve the prediction performance of the machine learning models, a validation set was utilized. This particular collection played a crucial role in the process of adjusting model parameters to maximize their performance.

**Testing the Models**

The models, having undergone training and validation, were rigorously tested using the testing dataset to assess their accuracy and predictive power.

**Performance Evaluation**

To gauge the effectiveness of the models, a thorough performance evaluation was conducted. This evaluation employed metrics such as the scikit-learn accuracy score and confusion matrix.

**Performance Comparison**

The next phase was a thorough comparison of the seven machine learning models. The goal of this comparison research was to determine which model outperformed the others in the essential duty of predicting crops, therefore adding vital insights to the area of medical diagnostics.

**Source of Dataset**


**Ensemble Learning Techniques**

Ensemble methods encompass various techniques, with common approaches including Bagging (Bootstrap Aggregating), Boosting, Stacking, and OVR (Jia, Liang & Liang, 2023).

**Bagging (Bootstrap Aggregating)**

Bagging is a technique where several models are trained separately on various subsets of the dataset (Breiman, 1996; González, García, Del-Ser, Rokach & Herrera, 2020; Malek et al.,2023). The Bagging method involves the creation of subsets via bootstrapping, which is a resampling strategy where instances are randomly picked with replacements from the original dataset (Gul, Mashwani, Aamir, Aldahmani & Khan, 2023). As a consequence, this leads to a variety of subsets that are used to train separate models inside the ensemble. Bagging, a technique introduced by Nosrati and Rahmani in 2023, aims to reduce the influence of noise or outliers in the data by training models independently on different subsets, thereby introducing variety. An exemplary instance of Bagging is the Random Forest method. In a Random Forest, several decision trees are trained using distinct bootstrap samples of the dataset (Sun et al.,2023). During the prediction process, each tree in the forest provides its own prognosis individually. The final prediction is then derived by combining these individual predictions (Becker, Geubbelmans, Rousseau, Valkenberg & Burzykowski, 2024).

The utilization of an ensemble technique frequently results in a model that is more robust and precise, with the ability to effectively apply its knowledge to unfamiliar data (Jitpakdeebodin & Sinapiromsaran, 2023). Bagging improves the predictive performance of an ensemble by decreasing overfitting and variance. It is an important approach in machine learning, especially when dealing with complicated and noisy data that may affect the performance of individual models.

**Boosting**

Boosting is a technique that involves training models in a sequential manner to correct the errors made by previous models (Emami & Martínez-Muñoz, 2023; Kunapuli, 2023). The core concept of boosting is to progressively enhance the predicted performance of the model by assigning greater importance to difficult data points (Attri, Awasthi, Sharma & Rathee, 2023). AdaBoost and Gradient Boosting are prominent algorithms in the boosting family. AdaBoost stands for Adaptive Boosting, while Gradient Boosting was developed by Thotad, Bharamagoudar, and Kallur in 2023.
During the boosting process, weak learners, which are often basic decision trees known as "stumps," are taught one after another in a sequential manner (Avramopoulos & Vasiloglou, 2023). Following each iteration, greater emphasis is placed on misclassified instances, encouraging the succeeding learner to concentrate on the previously difficult aspects of the material (Miller et al. 2023). This iterative process of adjusting weights allows the model to continuously improve its capacity to identify intricate patterns and connections within the dataset.

In boosting, the ultimate forecast is usually obtained by combining the weak learners' predictions using weights that reflect their accuracy during the training phase (Demir & Sahin, 2023). Boosting is renowned for its ability to enhance model performance, particularly in scenarios where individual models may encounter difficulties.

Stacking
Stacking, which is also known as Stacked Generalization, is a technique that combines predictions from many models using a meta-model (Seireg, Omar & Elmahalawy, 2023). During the stacking process, a set of base models is trained individually. Each of these models captures certain elements or makes unique mistakes on the training data (de-Zarzà, de-Curtò, Hernández-Orallo & Calafate, 2023). The forecasts produced by these foundational models are subsequently combined, and a meta-model, often a less complex algorithm such as linear regression, is trained using these forecasts (Chen, Zeb, Nanehkaran, & Zhang, 2023). The meta-model is designed to optimize the combination of outputs from the basis models, resulting in a more precise and accurate prediction during the testing or validation phase (Liang & Liu, 2023).

The fundamental principle of stacking is its capacity to use the synergistic advantages of different models, therefore exceeding the predictive capabilities of any one model (Yang et al., 2023). Although the concept of stacking shows potential for enhancing performance, it is important to carefully evaluate the variety of models used and select the most suitable meta-model in order to achieve the best possible outcomes in practical applications. Stacking is a powerful technique in ensemble learning that excels in complicated tasks, providing better abilities for subtle pattern detection and improved overall accuracy.

One-Vs-Rest (OVR)
The one-versus-rest (OVR) approach, also known as one-versus-all, is a popular technique used in multi-class classification tasks. It involves training a binary classifier for each class, where the positive class is the one being trained, and all other classes are the negative class (Hsu & Lin, 2002; Rifkin & Klautau, 2004).

The OVR approach involves training a binary classifier for each class in the dataset distinctly. During the training process, examples that belong to the positive class are labeled as such, while instances from all other classes are gathered together and labeled as the negative class. Each binary classifier is trained to differentiate its assigned class from all other classes, resulting in a collection of classifiers specifically designed to identify each individual class (Crammer & Singer, 2002).

During the prediction process, each binary classifier is provided with the instance, and the predicted class for that instance is determined by assigning it the class with the highest confidence score or probability. The final prediction is determined by the classifier that has the highest confidence for its allocated class (Frenay & Verleysen, 2014).

An inherent benefit of the OVR method is its straightforwardness and flexibility to be easily expanded. OVR, or one-vs-rest, is a technique that breaks down the multi-class classification issue into several binary classification subproblems. This approach enables the use of common binary classifiers like Support Vector Machines (SVMs) or Logistic Regression without the need for any changes to the underlying methods (Hsu & Lin, 2002).

Furthermore, OVR can be highly efficient in addressing unbalanced class distributions by prioritizing the differentiation of each class separately rather than taking into account the overall class distribution as a whole (Rifkin & Klautau, 2004). In addition, OVR (One-vs-Rest) classification tends to get good results in situations when classes are not mutually exclusive or when there is a substantial overlap across class boundaries (Hsu & Lin, 2002).

Performance Metrics for Classification
The evaluation criteria utilized for gauging the effectiveness of this analysis are as follows:

Accuracy
The effectiveness of a model is evaluated based on the ratio of correct predictions generated for various types of forecasts (Ricciardi, Ramankutty, Mehrabi, Jarvis & Choookolingo, 2018). The assessment technique entails evaluating the precision of categorization by comparing the number of accurately classified cases to the total number of occurrences (Petropoulos & Siemsen, 2023). The assessment of accuracy is especially useful in situations when the distribution of classes in the target variable is evenly distributed across the dataset. Equation 1 represents this concept.

\[
\text{ACCURACY} = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}
\]
**Sensitivity or Recall**
Sensitivity is a quantitative measure used to determine the percentage of positive scenarios that were accurately recognized by the model, but mistakenly classified as negative. Occasionally referred to as recall or true positive rate, as stated by Hutter (2012). Mathematically, it is defined as the quotient obtained by dividing the number of true positive (TP) occurrences by the total of true positive and false negative (FN) cases. The mathematical expression is as follows:

\[
Sensitivity = \frac{TP}{TP + FN}
\]  

(2)

**Specificity**
Specificity, sometimes referred to as the true negative rate, is an important concept in the field of software defect analysis (Everitt, Goertzel & Potapov, 2017). Equation 3 quantifies the proportion of defect-free occurrences in the software system that are accurately identified as such by the model.

\[
Specificity = \frac{TN}{TN + FP}
\]  

(3)

**Detection Rate**
The detection rate is the ratio of successfully detected events to the total sample size (Flasiński, 2016). This statistic measures the efficiency of accurately identifying instances within the dataset.

**F1 score rate**
The F1 score is a metric that calculates the weighted average of accuracy and recall (Bach, 2020). Consequently, this score considers the equilibrium between incorrect positive results and incorrect negative results.

\[
F1 Score = 2 \times \frac{precision \times recall}{precision + recall}
\]  

(6)

**Precision**
Precision is a quantitative measure that evaluates the correctness of positive predictions generated by a model (Davis, 2015). Precision is the quotient obtained by dividing the number of accurately anticipated positive samples by the total number of samples predicted as positive.

\[
Precision = \frac{TP}{TP + FP}
\]  

(5)

**Area Under Curve (AUC)**
The AUC, or Area Under the Curve, is a measure of a parameter's capacity to differentiate between two diagnostic classifications, such as normal and sick (Hajian-Tilaki, 2013). The AUC, which ranges from 0 to 1, measures the degree of discrimination provided by the parameter (Varoquaux & Colliot, 2023). A number close to 1 suggests a very reliable diagnostic outcome, indicating a great capacity to distinguish between the two groups.

**RESULTS AND DISCUSSION**
**Data preprocessing**
The section includes Importing the Libraries, Loading the Dataset, Printing the Head and Tail of the Dataset, Dataset Structure and Dimensions, Handling Missing Values, Dataset Information Overview, Descriptive Statistics of the Dataset, Crop Distribution, Crops Distribution Chart, and Outliers Detection.

**Importing the Libraries**
The Jupyter Notebook was set up with the necessary Python libraries, including Numpy, Pandas, Matplotlib, and Seaborn. Numpy is a powerful package that allows for efficient manipulation and concurrent operations on multi-dimensional arrays (Stančin & Jović, 2019). Pandas is a data analysis and manipulation tool that is open-source and based on the Python programming language (Subasi, 2020). Matplotlib and Seaborn are extensively used software tools for data visualization. The program provides a user-friendly interface that simplifies the process of creating visually appealing and informative graphs. Seaborn, a data visualization software, is a subset of Matplotlib that offers a more limited set of functionalities (Pintor et al., 2019). Figure 2 depicts the process of importing the Numpy, Pandas, Matplotlib, and Seaborn Python libraries into the Jupyter Notebook.

```python
import pandas as pd  # For data manipulation
import numpy as np   # For scientific computing
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import joblib
```

Figure 2: Importing Python Libraries
Loading the Dataset
The dataset was loaded into Jupyter Notebook using `pd.read_csv`. Figure 3 shows the loading of the dataset into Jupyter Notebook.

```python
data = pd.read_csv('Crop_recommendation.csv')
data.head()
```

Figure 3: Loading the Dataset

The Head and Tail of the Dataset
The `head()` function displays the top records in the dataset depicted in Table 1. By default, python displays only the top 5 records while the `tail()` functions display the last five records of the dataset as illustrated in Table 2.

<table>
<thead>
<tr>
<th>Table 1: First Five Rows of the Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Last Five Records of the Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>2195</td>
</tr>
<tr>
<td>2196</td>
</tr>
<tr>
<td>2197</td>
</tr>
<tr>
<td>2198</td>
</tr>
<tr>
<td>2199</td>
</tr>
</tbody>
</table>

[2200 rows x 8 columns]

Dataset Structure and Dimensions
The `shape()` function displays the number of rows and columns of the dataset. Figure 4 shows there are 2200 number of rows and 8 number of columns.

```python
data.shape
```

(2200, 8)

Figure 4: Number of Rows and Columns in the Dataset

Handling Missing Values
The presence of missing values was evaluated for each attribute in the dataset, and no instances of missing values were found. Figure 5 shows that there are no missing values (NaN) in any of the columns. Each column has 0 missing values, as indicated by the values in the right column (0 under each column name).
Crop Recommendation Predictive… Abdullahi et al., JOBASR2024 2(1): 162-176

Figure 5: Checking for Missing Values in the dataset

**Dataset Information Overview**
The dataset was loaded into the Pandas data frame for easy analysis, model development, and prediction. Table 3 depicts the Pandas Data Frame comprising 2200 rows and 8 columns representing agricultural parameters. The columns include Nitrogen, Phosphorus, Potassium, Temperature, Humidity, pH, Rainfall, and Label. The data types vary, with 3 integer columns (Nitrogen, Phosphorus, Potassium), 4 float columns (Temperature, Humidity, pH, Rainfall), and 1 object column (Label).

**Table 3: Info of the Dataset**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
# Column      Non-Null Count Dtype
0 Nitrogen    2200 non-null int64
1 phosphorus  2200 non-null int64
2 potassium   2200 non-null int64
3 temperature 2200 non-null float64
4 humidity    2200 non-null float64
5 ph          2200 non-null float64
6 rainfall    2200 non-null float64
7 label       2200 non-null object
```

**Descriptive Statistics of Dataset**
The `describe()` method provides a concise summary of a dataset containing 2200 records and 7 columns, all of which include numerical values. Table 4 offers statistical data for several factors, including Nitrogen, Phosphorus, Potassium, Temperature, Humidity, pH, and Rainfall. The dataset consists of statistical metrics, including the mean, standard deviation, minimum, maximum, and quartiles. These measurements provide valuable insights into the average values and variability of agricultural indicators, making it easier to analyze and identify any abnormal data points.

**Table 4: Dataset Descriptive Statistics**

<table>
<thead>
<tr>
<th></th>
<th>Nitrogen</th>
<th>phosphorus</th>
<th>potassium</th>
<th>temperature</th>
<th>humidity</th>
<th>ph</th>
<th>rainfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2200.000000</td>
<td>2200.000000</td>
<td>2200.000000</td>
<td>2200.000000</td>
<td>2200.000000</td>
<td>2200.000000</td>
<td>2200.000000</td>
</tr>
<tr>
<td>mean</td>
<td>50.551818</td>
<td>53.362727</td>
<td>48.149091</td>
<td>25.616244</td>
<td>71.481779</td>
<td>6.469480</td>
<td>103.463655</td>
</tr>
<tr>
<td>std</td>
<td>36.917334</td>
<td>32.985883</td>
<td>50.647931</td>
<td>5.063749</td>
<td>22.263812</td>
<td>0.773936</td>
<td>54.958389</td>
</tr>
<tr>
<td>min</td>
<td>0.000000</td>
<td>5.000000</td>
<td>5.000000</td>
<td>8.825675</td>
<td>14.258040</td>
<td>3.504752</td>
<td>20.211267</td>
</tr>
<tr>
<td>25%</td>
<td>21.000000</td>
<td>28.000000</td>
<td>20.000000</td>
<td>22.769375</td>
<td>60.261953</td>
<td>5.971693</td>
<td>64.551686</td>
</tr>
<tr>
<td>50%</td>
<td>37.000000</td>
<td>51.000000</td>
<td>32.000000</td>
<td>25.598693</td>
<td>80.473146</td>
<td>6.425045</td>
<td>94.867624</td>
</tr>
<tr>
<td>75%</td>
<td>84.250000</td>
<td>68.000000</td>
<td>49.000000</td>
<td>28.561654</td>
<td>89.948771</td>
<td>6.923643</td>
<td>124.267508</td>
</tr>
<tr>
<td>max</td>
<td>140.000000</td>
<td>145.000000</td>
<td>205.000000</td>
<td>43.675493</td>
<td>99.981676</td>
<td>9.935091</td>
<td>298.560117</td>
</tr>
</tbody>
</table>

**Crop Distribution**
The frequency of rows matching each crop type in the "label" column is shown in Table 5. The `value_counts()` function computes the occurrence rate of each unique value in the given column. The dataset has 100 rows for each of the 22 crop varieties, including rice, maize, jute, cotton, coconut, papaya, orange, apple, muskmelon, watermelon, grapes, mango, banana, pomegranate, lentil, black gram, mungbean, mothbeans, pigeonpeas, kidney beans, chickpea, and coffee.
### Table 3: Crop Distribution

<table>
<thead>
<tr>
<th>Crop</th>
<th>Number of Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>rice</td>
<td>100</td>
</tr>
<tr>
<td>maize</td>
<td>100</td>
</tr>
<tr>
<td>jute</td>
<td>100</td>
</tr>
<tr>
<td>cotton</td>
<td>100</td>
</tr>
<tr>
<td>coconut</td>
<td>100</td>
</tr>
<tr>
<td>papaya</td>
<td>100</td>
</tr>
<tr>
<td>orange</td>
<td>100</td>
</tr>
<tr>
<td>apple</td>
<td>100</td>
</tr>
<tr>
<td>muskmelon</td>
<td>100</td>
</tr>
<tr>
<td>watermelon</td>
<td>100</td>
</tr>
</tbody>
</table>

**Crops Distribution Chart**

The pie chart was created using the Matplotlib library, illustrating the occurrence of each unique value in the "label" column of the Data Frame. The pie chart depicted in Figure 6 visually illustrates the distribution of different crop types in the dataset. Each segment of the chart represents a specific crop, and the size of each segment indicates the proportion of that crop in the dataset.

![Crops Distribution Chart](image)

**Outliers Detection**

The crop outlier distribution is shown using a boxplot illustrated in Figure 7. This allows for the comparison of quantitative data values across various groups. From the plots, it can be observed that there are many data points indicating outliers for all factors except Nitrogen. This is due to the different optimal soil conditions for the different plants.
Results
The results extracted from the findings of the data analysis are presented here. The results are presented using figures and tables.

Encoding of Target Variable
This is the conversion of letters in the dataset into figures. Figure 8 depicts how the Label Encoder object is created.

```
# # Encode the target variable into numeric values
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Getting the Correlation
This shows the correlation of each of the features to one another. Figure 9 calculates the correlation matrix (corr) for the features in the dataset (data) excluding the 'label' column. The drop method is used to exclude the 'label' column from the dataset.

```
corr= data.drop(['label'],axis=1).corr()
# sns heatmap(corr,annot=True,cbar=True,cmap='coolwarm')

plt.figure(figsize = (10,10))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

The resulting data frame is then used to calculate the correlation matrix using the corr() method depicted in Figure 10.
Feature Scaling
The dataset was standardized using the Standard Scaler to improve model performance. Figure 11 illustrates the scaling and standardization of both the training and test data through the use of a standard scaler.

Splitting the Dataset into two
The dataset is split into input variables (x) and the output variable (y). The (x) contains the features (independent variables) for each data point. The (y) containing the target label (dependent variable) corresponds to each data point illustrated in Figure 12.

Discussion
Table 6 shows the comparative performance analysis of ensembling learning models. Four ensembling learning models—namely, Stacking, Bagging, Voting, and One-Vs-Rest (OVR)—were selected for this study. The models were trained and evaluated as individual classifiers for crop recommendation. The result of the models was analyzed and compared to derive insights to determine the model performance based on standard evaluation metrics such as accuracy, precision, recall, f1-score, and support.

Table 1: Accuracy Table of the Models

<table>
<thead>
<tr>
<th>S/N</th>
<th>Model</th>
<th>Accuracy (%)</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bagging</td>
<td>99.32</td>
<td>0.99</td>
<td>1.00</td>
<td>1.00</td>
<td>141</td>
</tr>
<tr>
<td>2</td>
<td>OVR</td>
<td>91.81</td>
<td>1.00</td>
<td>0.96</td>
<td>0.98</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>Stacking</td>
<td>96.14</td>
<td>0.96</td>
<td>0.99</td>
<td>0.97</td>
<td>141</td>
</tr>
<tr>
<td>4</td>
<td>Voting</td>
<td>99.09</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>141</td>
</tr>
</tbody>
</table>

In comparing the performance of various machine learning models, it is evident that Bagging and Voting outperform other models in terms of accuracy. Bagging achieves the highest accuracy of 99.32% (Nti et al., 2023), closely followed by Voting with an accuracy of 99.09%. These two techniques, Bagging and Voting, consistently demonstrate exceptional accuracy across
different datasets, suggesting their robustness and reliability. OVR (One vs Rest) approach achieves a relatively lower accuracy of 91.81% (Nti et al., 2023) compared to Bagging and Voting. However, it still demonstrates remarkable performance in efficiently differentiating between classes, as indicated by its high recall and F1-score values. Despite its lower accuracy, OVR remains a viable approach for tasks requiring class differentiation. Stacking, although not reaching the same level of accuracy as Bagging or Voting, still achieves a respectable accuracy of 96.14%. It demonstrates balanced performance across various evaluation metrics, indicating its capability to make accurate predictions while minimizing false positives and false negatives. Comparatively, other models such as Deep Reinforcement Learning (Khaki & Wang, 2019), traditional classifiers like KNN, NB, MLR, ANN, RF (Palanivel & Surianarayana, 2019), linear regression, LASSO, Light GBM, and ensemble methods like AdaBoost GB, and Stacked TBEL (Kalimuthu, Vaishnavi & Kishore, 2020), exhibit varying levels of performance, with accuracies ranging from 72.33% to 99.32%. These results highlight the importance of selecting appropriate machine learning techniques based on the specific requirements and characteristics of the dataset. The findings are displayed in a bar chart seen in Figure 13.

**CONCLUSION**
The performed comparison analysis in this paper provides insights into the performance features of several ensemble learning models in the specific context of crop recommendation. By assessing Bagging, Voting, Stacking, and One-Vs-Rest (OVR), we have acquired a significant understanding of the merits and limitations of each method. Bagging and Voting are the most accurate and high-performing techniques. These models exhibit resilience in successfully forecasting crop recommendations, as seen by their high accuracy scores and excellent precision, recall, and F1-score values. Their capacity to combine forecasts from numerous foundational models significantly enhances their exceptional performance. Farmers and agricultural stakeholders in search of precise and dependable crop recommendation systems may find Bagging and Voting methods especially well-suited for their requirements. Nevertheless, Stacking and OVR emerge as feasible alternatives, showcasing robust accuracy and recall skills. Although Bagging and Voting may have somewhat higher accuracy compared to Stacking and OVR, the latter two techniques provide more subtle methods for ensemble learning. Stacking utilizes meta-modeling to merge predictions from many base models, whereas OVR adopts a one-vs-all technique to tackle challenges involving multi-class categorization. These models are more suitable in situations when achieving high levels of accuracy and completeness are crucial, or where the capacity to understand and explain the model's decisions is a top concern.

**REFERENCES**

---

**Figure 13: Comparative Analysis of All Models**

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1-score (%)</th>
<th>Support (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging</td>
<td>90</td>
<td>92</td>
<td>90</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>Stacking</td>
<td>96</td>
<td>98</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>OVR</td>
<td>86</td>
<td>88</td>
<td>90</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>Voting</td>
<td>98</td>
<td>100</td>
<td>98</td>
<td>98</td>
<td>100</td>
</tr>
</tbody>
</table>

---


perspectives and opportunities. Information Fusion, 64, 205-237. https://doi.org/10.1016/j.inffus.2020.07.007


